

# **MELSEC A/Q series**

Programmable Logic Controllers

Programming Manual



**Programming Manual for the  
MELSEC A and Q series and the MELSEC System Q  
Order-No.: 87431**

<b>Revision</b>			<b>Changes / Additions / Correktions</b>
A	09/1998	pdp	Programming manual for the MELSEC A and Q series based on Melsec Medoc <i>plus</i>
B	04/2001	pdp	<p>Ch. 5.1.2: Changed bitmap (ladder diagram in GX IEC Developer) for the ORP instruction            Changed bitmap for the LDP instruction</p> <p>Ch. 7.6.3: Note for the usage of the CALL instruction</p> <p>Ch. 7.14: Addition of the RSET_K_MD and RSET_K_P_MD instruction</p> <p>Ch. 7.11.13: Note for the usage of the ASC(P) instruction</p> <p>Additional information for System Q CPUs (Q02, Q02H, Q06H, Q12H and Q25H).</p>
C	08/2002	pdp-dk	<p>Additional information for Q00JCPU, Q00CPU and Q01CPU of the System Q.Q25H).            New instructions S.TO and FROM for use in a multi-CPU-System.            In chapter 9 now the representation format of the instruction in the GX IEC Developer is shown.            Additional special relays and registers for System Q CPUs with function Version B or later.            Separate tables for processing times for A series and Q series/System Q.            Addition of an example in Ch. 7.6.10, showing the program modification.</p> <p>Corrections:</p> <p>Ch. 6.5.1: Operating errors</p> <p>Ch. 6.5.2: Operating errors</p> <p>Ch. 6.7.3: Additional information for the COM instruction when used in a multi-CPU system</p> <p>Ch. 6.8.9: Time values for n1</p> <p>Ch. 7.1.1: Devices MELSEC Q</p> <p>Ch. 7.1.3: Devices MELSEC Q</p> <p>Ch. 7.1.5: Devices MELSEC Q</p> <p>Ch. 7.1.7: Devices MELSEC Q</p> <p>Ch. 7.5.12: Operating errors</p> <p>Ch. 9.5.1: Processing times for the RBMOV and the BMOV instruction</p>
D	09/2004	pdp-dk	<p>New chapter 10: Instructions for Q4ARCPU</p> <p>New chapter 11: Dedicated instructions for intelligent function modules</p> <p>Ch. 2.8: Summary of the instructions for Q4ARCPU</p> <p>Ch. 2.9: Summary of the dedicated instructions</p> <p>New CPU modules Q12PHCPU and Q25PHCPU</p>



# About this Manual

The texts, illustrations, diagrams, and examples contained in this manual are intended exclusively as support material for the explanation, handling, programming, and operation of the programmable logic controllers of the MELSEC A and Q series and the MELSEC System Q.

If you have any questions concerning the programming and operation of the equipment described in this manual, please contact your relevant sales office or department (refer to back of cover).

Current information and answers to frequently asked questions are also available through the Internet ([www.mitsubishi-automation.com](http://www.mitsubishi-automation.com))

mitsubishi electric europe b.v. reserves the right for technical changes and changes to this manual at any time without prior notice.

© 09/2004



# Contents

<b>1</b>	<b>Introduction</b>	
1.1	Further manuals .....	1-1
1.2	CPU types .....	1-2
1.3	Software .....	1-2
1.4	Finding an instruction .....	1-3
1.5	PLC parameters .....	1-3
1.6	Comparison between GX IEC Developer and GX Developer .....	1-4
<b>2</b>	<b>Instruction Tables</b>	
2.1	Subdivision of instructions .....	2-1
2.2	Overview of instructions .....	2-4
2.2.1	Description of the overview tables .....	2-4
2.3	Sequence instructions .....	2-6
2.3.1	Input instructions .....	2-6
2.3.2	Connection instructions .....	2-7
2.3.3	Output instruction .....	2-8
2.3.4	Shift instructions .....	2-8
2.3.5	Master control instructions .....	2-9
2.3.6	Program termination instructions .....	2-9
2.3.7	Miscellaneous instructions .....	2-9
2.4	Application instructions, Part 1 .....	2-10
2.4.1	Comparison operation instructions .....	2-10
2.4.2	Arithmetic operation instructions .....	2-15
2.4.3	Data conversion instructions .....	2-22
2.4.4	Data transfer instructions .....	2-25
2.4.5	Program branch instructions .....	2-27
2.4.6	Interrupt program execution control instructions .....	2-27
2.4.7	Data refresh instructions .....	2-28
2.4.8	Other convenient instructions .....	2-29

2.5	Application instructions, Part 2	2-31
2.5.1	Logical operation instructions	2-31
2.5.2	Rotation instructions	2-35
2.5.3	Shift instructions	2-36
2.5.4	Bit processing instructions	2-37
2.5.5	Data processing instructions	2-38
2.5.6	Structured program instructions	2-41
2.5.7	Data table operation instructions	2-43
2.5.8	Buffer memory access instructions	2-44
2.5.9	Display instructions	2-45
2.5.10	Debugging and failure diagnosis instructions	2-46
2.5.11	Character string processing instructions	2-47
2.5.12	Special function instructions	2-51
2.5.13	Data control instructions	2-54
2.5.14	File register switching instructions	2-55
2.5.15	Clock instructions	2-56
2.5.16	Peripheral device instructions	2-57
2.5.17	Program instructions	2-57
2.5.18	Other instructions	2-58
2.6	Data link instructions	2-60
2.6.1	Network refresh instructions	2-60
2.6.2	Dedicated data link instructions	2-60
2.6.3	A series compatible data link instructions	2-61
2.6.4	Read/Write routing information	2-61
2.7	Instructions for System Q CPUs	2-62
2.7.1	Reading of module informations	2-62
2.7.2	Debugging and failure diagnosis instructions	2-62
2.7.3	Writing to and reading from a file	2-63
2.7.4	Program instructions	2-63
2.7.5	Data transfer instructions	2-64
2.7.6	Instructions for data exchange in a multi-CPU system	2-64
2.8	Dedicated instructions for Q4ARCPU	2-65
2.8.1	Instructions for mode setting	2-65
2.8.2	Data transfer instructions	2-65



2.9	Instructions for special function modules	2-66
2.9.1	Instructions for serial communication modules	2-66
2.9.2	Instructions for PROFIBUS/DP interface modules	2-67
2.9.3	Instructions for ETHERNET interface modules	2-68
2.9.4	Instruction for MELSECNET/10	2-68
2.9.5	Instructions for CC-Link	2-69

### **3 Configuration of Instructions**

3.1	The structure of an instruction	3-1
3.1.1	Source of data (s)	3-1
3.1.2	Destination of data (d)	3-2
3.1.3	Number (n)	3-2
3.2	Notation of instructions	3-3
3.2.1	16/ 32-bit and pulse	3-3
3.2.2	MELSEC and IEC	3-3
3.2.3	Further characteristics of the instruction notation	3-5
3.2.4	Specification of the notation	3-5
3.3	Programming of dedicated instructions	3-6
3.4	Programming of variables	3-7
3.4.1	Programming with the GX IEC Developer	3-7
3.4.2	Programming with the GX Developer	3-8
3.5	Data types	3-9
3.5.1	Processing of data	3-11
3.5.2	Addressing of arrays and registers in the GX IEC Developer	3-19
3.5.3	Usage of character string data (STRING)	3-22
3.6	Index qualification	3-24
3.6.1	Index qualification	3-24
3.6.2	Special characteristics of the System Q and QnA CPUs	3-26
3.6.3	Special characteristics of the AnA, AnAS, and AnU CPUs	3-28
3.7	Indirect Designation (GX Developer only)	3-29
3.8	Operation errors	3-31
3.8.1	Verification of the device range	3-31
3.8.2	Verification of the device data	3-33

3.9	Execution conditions of the instructions	3-34
3.9.1	Execution condition	3-34
3.9.2	EN input and ENO output	3-35
3.10	Number of program steps	3-37
3.10.1	For a System Q and QnA CPU	3-37
3.10.2	For an AnA, AnAS, and AnU CPU	3-38

#### **4 Layout and Structure of the Chapters**

4.1	Overview of the instructions	4-2
4.2	The CPU table	4-2
4.3	Devices MELSEC A	4-3
4.4	Devices MELSEC Q	4-4
4.5	Representation format of the instruction	4-4
4.5.1	Representation in the GX IEC Developer	4-4
4.5.2	Representation in the GX Developer	4-5
4.6	Variables	4-5
4.7	Functions	4-6
4.8	Notes	4-6
4.9	Operation Errors	4-6
4.10	Program Examples	4-7

#### **5 Sequence Instructions**

5.1	Input Instructions	5-4
5.1.1	LD, LDI, AND, ANI, OR, ORI	5-4
5.1.2	LDP, LDF, ANDP, ANDF, ORP, ORF	5-8
5.2	Connection Instructions	5-11
5.2.1	ANB, ORB	5-11
5.2.2	MPS, MRD, MPP	5-14
5.2.3	INV	5-17
5.2.4	MEP, MEF	5-19
5.2.5	EGP, EGF	5-21

---

5.3	Output Instructions	5-23
5.3.1	OUT	5-23
5.3.2	OUT T, OUTH T	5-25
5.3.3	OUT C	5-28
5.3.4	OUT F	5-31
5.3.5	SET	5-34
5.3.6	RST	5-36
5.3.7	SET F, RST F	5-39
5.3.8	PLS, PLF	5-42
5.3.9	FF	5-46
5.3.10	CHK	5-48
5.3.11	DELTA, DELTAP	5-50
5.4	Shift Instructions	5-52
5.4.1	SFT, SFTP	5-52
5.5	Master Control Instructions	5-55
5.5.1	MC, MCR	5-55
5.6	Termination Instructions	5-61
5.6.1	FEND	5-61
5.6.2	END	5-64
5.7	Miscellaneous Instructions	5-67
5.7.1	STOP	5-67
5.7.2	NOP	5-70

<b>6</b>	<b>Application Instructions, Part 1</b>	
6.1	Comparison Operation Instructions	6-2
6.1.1	=, <>, >, <=, <, >=	6-5
6.1.2	D=, D<>, D>, D<=, D<, D>=	6-8
6.1.3	E=, E<>, E>, E<=, E<, E>=	6-11
6.1.4	\$=, \$<>, \$>, \$<=, \$<, \$>=	6-15
6.1.5	BKCMP, BKCMPP	6-20
6.2	Arithmetic Operation Instructions	6-25
6.2.1	+, +P, -, -P	6-28
6.2.2	D+, D+P, D-, D-P	6-32
6.2.3	x, xP, /, /P	6-36
6.2.4	Dx, DxP, D/, D/P	6-40
6.2.5	B+, B+P, B-, B-P	6-43
6.2.6	DB+, DB+P, DB-, DB-P	6-48
6.2.7	Bx, BxP, B/, B/P	6-53
6.2.8	DBx, DBxP, DB/, DB/P	6-56
6.2.9	E+, E+P, E-, E-P	6-60
6.2.10	Ex, ExP, E/, E/P	6-65
6.2.11	BK+, BK+P, BK-, BK-P	6-68
6.2.12	\$+, \$+P	6-72
6.2.13	INC, INCP, DEC, DECP	6-75
6.2.14	DINC, DINCP, DDEC, DDECP	6-78
6.3	Data Conversion Instructions	6-81
6.3.1	BCD, BCDP, DBCD, DBCDP	6-83
6.3.2	BIN, BINP, DBIN, DBINP	6-86
6.3.3	FLT, FLTP, DFLT, DFLTP	6-90
6.3.4	INT, INTP, DINT, DINTP	6-93
6.3.5	DBL, DBLP	6-96
6.3.6	WORD, WORDP	6-98
6.3.7	GRY, GRYP, DGRY, DGRYP	6-100
6.3.8	GBIN, GBINP, DGBIN, DGBINP	6-103
6.3.9	NEG, NEGP, DNEG, DNEGP	6-106
6.3.10	ENEG, ENEGP	6-109
6.3.11	BKBCD, BKBCDP	6-111
6.3.12	BKBIN, BKBINP	6-114

---

6.4	Data transfer instructions	6-117
6.4.1		6-117
6.4.2	MOV, MOVP, DMOV, DMOVP	6-118
6.4.3	EMOV, EMOVP	6-121
6.4.4	\$MOV, \$MOVP	6-124
6.4.5	CML, CMLP, DCML, DCMLP	6-127
6.4.6	BMOV, BMOVP	6-132
6.4.7	FMOV, FMOVP	6-135
6.4.8	XCH, XCHP, DXCH, DXCHP	6-138
6.4.9	BXCH, BXCHP	6-141
6.4.10	SWAP, SWAPP	6-144
6.5	Program Branch Instructions	6-147
6.5.1	CJ, SCJ, JMP	6-148
6.5.2	GOEND	6-153
6.6	Program Execution Control Instructions	6-155
6.6.1		6-155
6.6.2	DI, EI, IMASK	6-156
6.6.3	IRET	6-163
6.7	Link Refresh Instructions	6-165
6.7.1		6-165
6.7.2	RFS, RFSP	6-166
6.7.3	SEG	6-168
6.7.4	COM	6-172
6.7.5	EI, DI	6-175
6.8	Other convenient Instructions	6-178
6.8.1		6-178
6.8.2	UDCNT1	6-179
6.8.3	UDCNT2	6-182
6.8.4	TTMR	6-185
6.8.5	STMR, STMRH	6-187
6.8.6	ROTC	6-191
6.8.7	RAMP	6-195
6.8.8	SPD	6-197
6.8.9	PLSY	6-199
6.8.10	PWM	6-201
6.8.11	MTR	6-203

<b>7</b>	<b>Application Instructions, Part 2</b>	
7.1	Logical operation instructions	7-2
7.1.1	WAND, WANDP, DAND, DANDP	7-4
7.1.2	BKAND, BKANDP	7-11
7.1.3	WOR, WORP, DOR, DORP	7-14
7.1.4	BKOR, BKORP	7-20
7.1.5	WXOR, WXORP, DXOR, DXORP	7-23
7.1.6	BKXOR, BKXORP	7-29
7.1.7	WXNR, WXNRP, DXNR, DXNRP	7-32
7.1.8	BKXNR, BKXNRP	7-39
7.2	Data rotation instructions	7-42
7.2.1	ROR, RORP, RCR, RCRP	7-43
7.2.2	ROL, ROLP, RCL, RCLP	7-46
7.2.3	DROR, DRORP, DROR, DRORP	7-49
7.2.4	DROR, DRORP, DROR, DRORP	7-52
7.3	Data shift instructions	7-55
7.3.1	SFR, SFRP, SFL, SFLP	7-56
7.3.2	BSFR, BSFRP, BSFL, BSFLP	7-59
7.3.3	DSFR, DSFRP, DSFL, DSFLP	7-62
7.4	Bit processing instructions	7-65
7.4.1	BSET, BSETP, BRST, BRSTP	7-66
7.4.2	TEST, TESTP, DTEST, DTESTP	7-69
7.4.3	BKRST, BKRSTP	7-73
7.5	Data processing instructions	7-76
7.5.1	SER, SERP, DSER, DSERP	7-78
7.5.2	SUM, SUMP, DSUM, DSUMP	7-84
7.5.3	DECO, DECOP	7-87
7.5.4	ENCO, ENCOP	7-89
7.5.5	SEG, SEGP	7-91
7.5.6	DIS, DISP	7-95
7.5.7	UNI, UNIP	7-98
7.5.8	NDIS, NDISP, NUNI, NUNIP	7-101
7.5.9	WTOB, WTOBP, BTOW, BTOWP	7-106
7.5.10	MAX, MAXP, DMAX, DMAXP	7-111
7.5.11	MIN, MINP, DMIN, DMINP	7-114
7.5.12	SORT, SORTP, DSORT, DSORTP	7-117
7.5.13	WSUM, WSUMP	7-121
7.5.14	DWSUM, DWSUMP	7-123

---

7.6	Structured program instructions	7-125
7.6.1	FOR, NEXT	7-126
7.6.2	BREAK, BREAKP	7-129
7.6.3	CALL, CALLP	7-132
7.6.4	RET	7-135
7.6.5	FCALL, FCALLP	7-137
7.6.6	ECALL, ECALLP	7-141
7.6.7	EFCALL, EFCALLP	7-144
7.6.8	CHG	7-147
7.6.9	SUB, SUBP	7-156
7.6.10	IX, IXEND	7-159
7.6.11	IXDEV, IXSET	7-164
7.7	Data table operation instructions	7-167
7.7.1	FIFW, FIFWP	7-168
7.7.2	FIFR, FIFRP	7-172
7.7.3	FPOP, FPOPP	7-176
7.7.4	FDEL, FDELP, FINS, FINSP	7-180
7.8	Buffer Memory Access Instructions	7-185
7.8.1	FROM, DFRO	7-186
7.8.2	TO, DTO, DTO, DTO	7-190
7.9	Display Instructions	7-194
7.9.1		7-194
7.9.2	PR	7-196
7.9.3	PRC	7-201
7.9.4	LED	7-205
7.9.5	LEDC	7-208
7.9.6	LEDA, LEDB	7-211
7.9.7	LEDR	7-213
7.10	Failure diagnosis and debugging	7-217
7.10.1	CHKST, CHK (Q series and System Q only)	7-218
7.10.2	CHK (A series only)	7-226
7.10.3	CHKCIR, CHKEND	7-234
7.10.4	SLT, SLTR	7-239
7.10.5	STRA, STRAR	7-241
7.10.6	PTRA, PTRAR, PTRAEEXE, PTRAEEXEP	7-243

7.11	Character string processing instructions	7-245
7.11.1	BINDA, BINDAP, DBINDA, DBINDAP	7-248
7.11.2	BINHA, BINHAP, DBINHA, DBINHAP	7-253
7.11.3	BCDDA, BCDDAP, DBCDDA, DBCDDAP	7-258
7.11.4	DABIN, DABINP, DDABIN, DDABINP	7-263
7.11.5	HABIN, HABINP, DHABIN, DHABINP	7-268
7.11.6	DABCD, DABCDP, DDABCD, DDABCDP	7-272
7.11.7	COMRD, COMRDP	7-277
7.11.8	LEN, LENP	7-281
7.11.9	STR, STRP, DSTR, DSTRP	7-284
7.11.10	VAL, VALP, DVAL, DVALP	7-292
7.11.11	ESTR, ESTRP	7-298
7.11.12	EVAL, EVALP	7-307
7.11.13	ASC, ASCP (Q series and System Q)	7-313
7.11.14	ASC (A series)	7-316
7.11.15	HEX, HEXP	7-318
7.11.16	RIGHT, RIGHTP, LEFT, LEFTP	7-322
7.11.17	MIDR, MIDRP, MIDW, MIDWP	7-326
7.11.18	INSTR, INSTRP	7-332
7.11.19	EMOD, EMODP	7-336
7.11.20	EREXP, EREXPP	7-339
7.12	Special functions	7-342
7.12.1	SIN, SINP	7-344
7.12.2	COS, COSP	7-347
7.12.3	TAN, TANP	7-350
7.12.4	ASIN, ASINP	7-353
7.12.5	ACOS, ACOSP	7-356
7.12.6	ATAN, ATANP	7-359
7.12.7	RAD, RADP	7-362
7.12.8	DEG, DEGP	7-365
7.12.9	SQR, SQRP	7-368
7.12.10	EXP, EXPP	7-371
7.12.11	LOG, LOGP	7-374
7.12.12	RND, RNDP, SRND, SRNDP	7-377
7.12.13	BSQR, BSQRP, BDSQR, BDSQRP	7-379
7.12.14	BSIN, BSINP	7-383
7.12.15	BCOS, BCOSP	7-386
7.12.16	BTAN, BTANP	7-389
7.12.17	BASIN, BASINP	7-392



---

7.12.18	BACOS, BACOSP	7-395
7.12.19	BATAN, BATANP	7-398
7.13	Data control instructions	7-401
7.13.1	LIMIT, LIMITP, DLIMIT, DLIMITP	7-402
7.13.2	BAND, BANDP, DBAND, DBANDP	7-406
7.13.3	ZONE, ZONEP, DZONE, DZONEP	7-410
7.14	File register switching instructions	7-414
7.14.1	RSET, RSETP	7-415
7.14.2	QDRSET, QDRSETP	7-418
7.14.3	QCDSET, QCDSETP	7-421
7.15	Clock instructions	7-424
7.15.1	DATERD, DATERDP	7-425
7.15.2	DATEWR, DATEWRP	7-429
7.15.3	DATE+, DATE+P	7-433
7.15.4	DATE-, DATE-P	7-438
7.15.5	SECOND, SECONDP, HOUR, HOURP	7-443
7.16	Peripheral device instructions	7-449
7.16.1	MSG	7-450
7.16.2	PKEY	7-453
7.17	Program control instructions	7-456
7.17.1	PSTOP, PSTOPP	7-457
7.17.2	POFF, POFFP	7-459
7.17.3	PSCAN, PSCANP	7-461
7.17.4	PLOW, PLOWP	7-463
7.18	Other convenient instructions	7-465
7.18.1	WDT, WDTP	7-466
7.18.2	STC, CLC	7-468
7.18.3	DUTY	7-470
7.18.4	ZRRDB, ZRRDBP	7-473
7.18.5	ZRWRB, ZRWRBP	7-477
7.18.6	ADRESET, ADRSETP	7-481
7.18.7	KEY	7-482
7.18.8	ZPUSH, ZPUSHP, ZPOP, ZPOPP	7-488
7.18.9	EROMWR, EROMWRP	7-491

<b>8</b>	<b>Data Link Instructions</b>	
8.1	Fundamentals	8-1
8.2	Categories of instructions	8-1
8.3	Data read/write ranges	8-3
8.3.1	MELSECNET/10	8-3
8.3.2	MELSECNET	8-4
8.4	Dedicated data link instructions	8-4
8.4.1	Simultaneous execution	8-4
8.4.2	Transmission completion	8-4
8.5	Data refresh instructions	8-6
8.5.1	ZCOM	8-7
8.6	Dedicated data link instructions for the QnA series	8-11
8.6.1	READ	8-12
8.6.2	SREAD	8-17
8.6.3	WRITE	8-23
8.6.4	SWRITE	8-30
8.6.5	SEND	8-37
8.6.6	RECV	8-45
8.6.7	REQ	8-50
8.6.8	ZNFR	8-60
8.6.9	ZNTO	8-66
8.7	A series compatible data link instructions	8-72
8.7.1	ZNRD	8-73
8.7.2	ZNWR	8-77
8.7.3	LRDP	8-81
8.7.4	LWTP	8-85
8.7.5	RFRP	8-89
8.7.6	RTOP	8-95
8.8	Reading and writing routing information	8-101
8.8.1	RTREAD	8-102
8.8.2	RTWRITE	8-104

---

<b>9</b>	<b>Instructions for System Q CPUs</b>	
9.1	Reading Module Information .....	9-2
9.1.1	UNIRD, UNIRD P .....	9-2
9.2	Debugging and failure diagnosis instructions .....	9-7
9.2.1	TRACE, TRACER .....	9-7
9.3	Writing to and reading from files .....	9-9
9.3.1	FWRITE .....	9-9
9.3.2	FREAD .....	9-20
9.4	Program instructions .....	9-33
9.4.1	PLOADP .....	9-33
9.4.2	PUNLOADP .....	9-36
9.4.3	PSWAPP .....	9-38
9.5	Data transfer instructions .....	9-41
9.5.1	RBMOV, RBMOV P .....	9-41
9.6	Instructions for use in a Multi-CPU System .....	9-46
9.6.1	S.TO, SP.TO .....	9-46
9.6.2	FROM, FROM P .....	9-49
<b>10</b>	<b>Instructions for Q4ARCPU</b>	
10.1	Mode setting instructions .....	10-2
10.1.1	STMODE .....	10-2
10.1.2	CGMODE .....	10-4
10.2	Instructions for data transfer .....	10-6
10.2.1	TRUCK .....	10-6
10.2.2	SPREF .....	10-11

<b>11</b>	<b>Instructions for Special Function Modules</b>	
11.1	Instructions for Serial Communication Modules	11-2
11.1.1	BUFRCVS	11-3
11.1.2	GETE, GETEP	11-6
11.1.3	PUTE, PUTEP	11-11
11.1.4	PRR, PRRP	11-18
11.2	Instructions for PROFIBUS/DP interface modules	11-26
11.2.1	BBLKRD, BBLKRDP	11-27
11.2.2	BBLKWR, BBLKWRP	11-30
11.3	Instructions for ETHERNET interface modules	11-33
11.3.1	BUFRCV	11-34
11.3.2	BUFRCVS	11-39
11.3.3	BUFSND	11-42
11.3.4	OPEN	11-47
11.3.5	CLOSE	11-56
11.3.6	ERRCLR	11-61
11.3.7	ERRRD	11-67
11.3.8	UINI	11-72
11.4	Instructions for MELSECNET/10	11-78
11.4.1	PAIRSET	11-79
11.5	Instructions for CC-Link	11-82
11.5.1	RLPA (A series)	11-83
11.5.2	RLPASET (System Q)	11-89
11.5.3	RRPA (A series)	11-101
11.5.4	RIRD (A series)	11-108
11.5.5	RIRD (QnA series and System Q)	11-114
11.5.6	RIWT (A series)	11-122
11.5.7	RIWT (QnA series and System Q)	11-128
11.5.8	RIRCV (A series)	11-136
11.5.9	RIRCV (QnA series and System Q)	11-142
11.5.10	RISEND (A series)	11-148
11.5.11	RISEND (QnA series and System Q)	11-154
11.5.12	RITO (A series)	11-160
11.5.13	RITO (QnA series and System Q)	11-164
11.5.14	RIFR (A series)	11-168
11.5.15	RIFR (QnA series and System Q)	11-172

<b>12</b>	<b>Microcomputer Mode (AnN(S))</b>	
12.1	Storage capacities and memory areas .....	12-1
12.2	Applying user-created microcomputer programs .....	12-2
12.2.1	Memory map .....	12-3
12.2.2	Address configuration of the data storage area .....	12-3
12.2.3	Configuration of data memory area .....	12-4
<b>13</b>	<b>Error Codes</b>	
13.1	Table of error codes; Q00J, Q00 and Q01CPU .....	13-2
13.2	Table of Error Codes; QnA CPUs and System Q .....	13-12
13.3	Table of error codes; A series (except AnA and AnAS) .....	13-39
13.4	Table of error codes; AnA and AnAS CPUs .....	13-43
<b>A</b>	<b>Appendix A</b>	
A.1	Definition of the Processing Times .....	A-1
A.2	Processing times .....	A-2
A.2.1	Table of Processing Times (QnA series and System Q) .....	A-3
A.2.2	Processing times of the A series CPUs .....	A-23
A.3	Comparison of the CPUs .....	A-32
A.3.1	Available devices .....	A-32
A.3.2	I/O control modes .....	A-34
A.3.3	Data types .....	A-34
A.3.4	Timer-Vergleich .....	A-35
A.3.5	Comparison of counters .....	A-38
A.3.6	Comparison of display instructions .....	A-39
A.3.7	Q series and System Q instructions equivalent to A series instructions .....	A-40
A.3.8	Comparison between QnA/Q2AS CPU and MELSEC System Q CPU .....	A-41
A.4	Overview of special relays .....	A-43
A.4.1	Table of diagnostic special relays (Q series and System Q) .....	A-43
A.4.2	Table of special relays (M) (A series) .....	A-64
A.4.3	Table of link relays (A series only) .....	A-70

A.5 Table of Special Registers ..... A-73  
A.5.1 Table of special registers (Q series and System Q) ..... A-73  
A.5.2 Table of special registers (D) (A series only) ..... A-110  
A.5.3 Table of link registers (A series only) ..... A-120

**Index**







# 1 Introduction

This manual describes the programming and processing of the sequence and application instructions that are provided by the CPUs of the MELSEC A and Q series.

## 1.1 Further manuals

QCPU/QnACPU Programming Manual (PID Instructions)

- Description of the PID control instructions

QnPHCPU Programming Manual (Process Control Instructions)

- Description of the PID control instructions

Programming Manual (AD57/58)

- Description of specific instructions for the special function modules AD57/58

QCPU/QnACPU Programming Manual (SFC)

- Description of the instructions for sequential function charts

GX Developer Operation Manuals

- Fundamentals of programming in GX Developer

GX IEC Developer Beginner's Manual

- Fundamentals of programming in GX IEC Developer

GX IEC Developer Reference Manual

- Detailed description of programming in GX IEC Developer

- Description of the IEC instructions (IEC standard library)

### NOTE

*All manuals are listed in our current PLC price list.*

*You can also download all manuals as PDF from the MITSUBISHI ELETRIC homepage ([www.mitsubishi-automation.com](http://www.mitsubishi-automation.com)).*

## 1.2 CPU types

The functions described in this manual can be transferred to all CPU types by the current versions of the GX Developer and the GX IEC Developer provided that the according CPU supports the instructions.

The different PLC types with their specific CPU are listed below in detail:

PLC Type		CPU Type
A Series	AnA/AnU	A2A, A2A-S1, A2U, A2U-S1, A3A, A3U
	AnAS/AnUS	A2AS, A2AS-S1, A2AS-S30, A2AS-S60, A2US, A2US-S1
	AnN	A1, A2, A2C A3M, A3N
	AnS	A1S, A1S-S1, A2S, A2S-S1
Q Series	QnA	Q2A/Q2AS, Q2A-S1/Q2AS-S Q3A Q4A, Q4AR
System Q	Q (single processor types)	Q00J
	Q (multi processor types)	Q00, Q01 (restricted use in a multi-CPU System) Q02, Q02H, Q06H, Q12H, Q12PH, Q25H, Q25PH PC-CPU-Module: PPC-CPU686(MS)-64 PPC-CPU686(MS)-128  Up to 4 multi processor type PLC CPUs can be used in a multi-CPU system, thus sharing control and communication tasks.

If, e.g. in tables, A and Q is mentioned, all CPU types of the A series and the Q series/ System Q are included. Exceptions are marked separately.

## 1.3 Software

All the described instructions, with few exceptions, can be applied with the available software packages:

- GX Developer
- GX IEC Developer

The program examples contained in this manual were created with the GX IEC Developer. The representation of the MELSEC Instruction List (IL) generally corresponds to that of the GX Developer.

All the instructions described in this manual are included within the standard library of the GX IEC Developer.

Corresponding to the selected CPU only those instructions are available within the GX IEC Developer dialog box that can actually be processed by the CPU.

## 1.4 Finding an instruction

### Advanced

If you are already familiar with the programming of instructions for the MELSEC A and Q series as well as the System Q, look up the instruction chapters 5 through 9. The header line contains the name of the instruction as it is applied within GX Developer and the MELSEC editor of the GX IEC Developer.

### Beginners

If you are not really familiar with the handling of the instructions, proceed as follows:

- Read through chapter 3 regarding the differing representation of instructions within the MELSEC and the IEC editor.
- Read through chapter 4 regarding the consistent layout and structure of each description of instruction.
- Use
  - the tabular overview of instruction categories with brief descriptions in chapter 2
  - the index containing the entire instructions

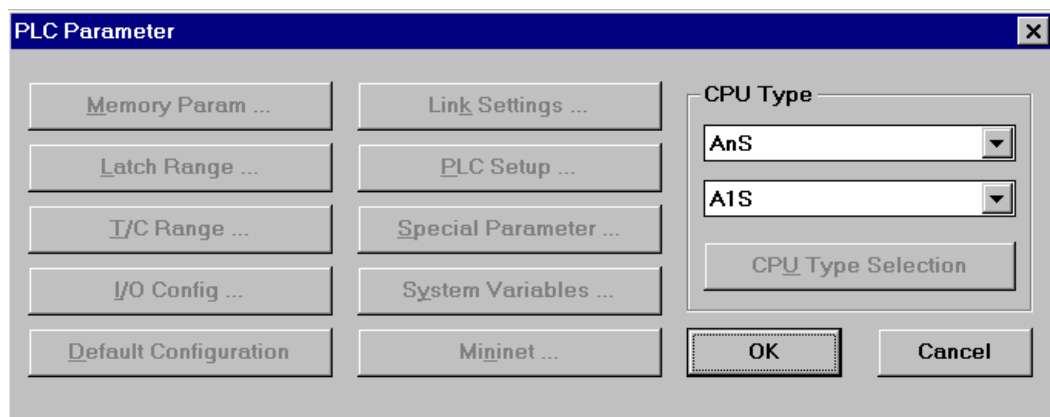
### NOTE

*All the instructions contained in this manual are also included within the online help of the GX IEC Developer as detailed as here.*

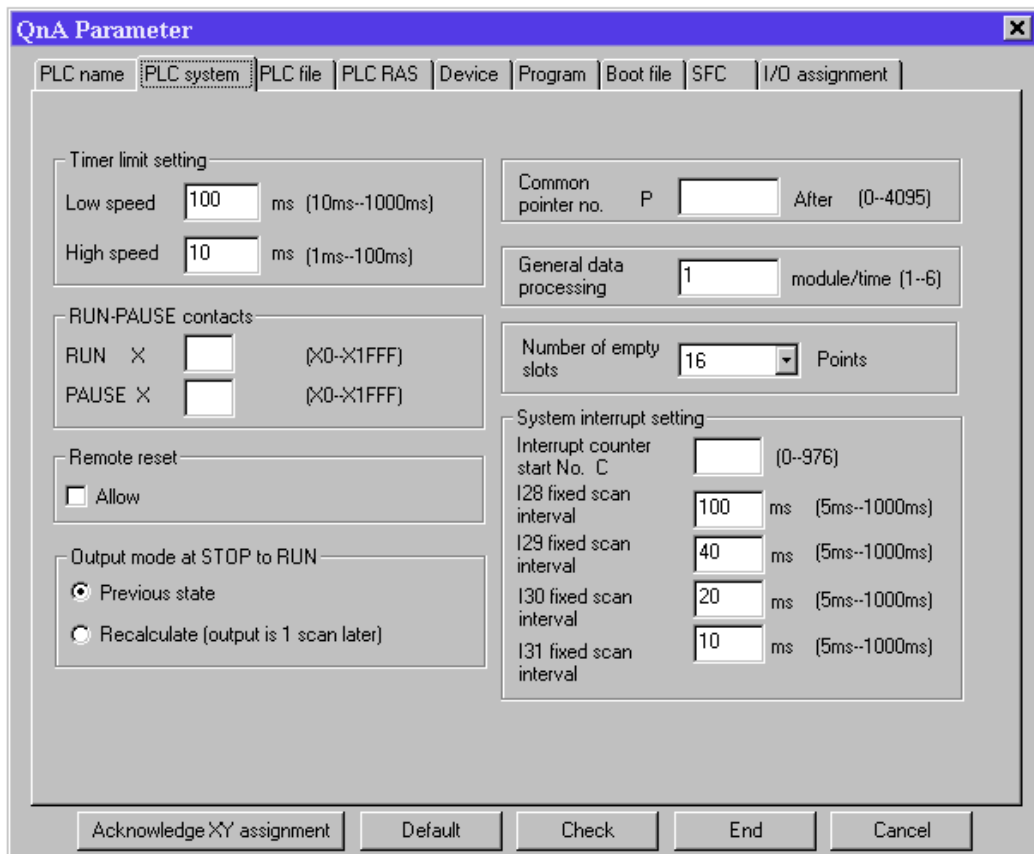
## 1.5 PLC parameters

Via parameters several functions, device ranges, etc. are set up. For the programming of the functions described in this manual, the parameter settings can remain preset or customised to the user's needs. Refer to the according hardware manuals of the CPUs and programming manuals for detailed descriptions of the PLC parameter settings.

Example: GX IEC Developer



Example: GX Developer, GX IEC Developer 6.0



## 1.6 Comparison between GX IEC Developer and GX Developer

The most important features of the GX IEC Developer and the GX Developer are listed in the following table:

GX IEC Developer	GX Developer
Structured use	Simple to use
Programming in comply with IEC 1131	—
Editors: Instruction List, Ladder Diagram, Structured Text, SFC, FUB	Editors: Instruction List, Ladder Diagram, SFC
Functions und Funktion Blocks	Funktion Blocks (V 7.0 or later)
Program modifications in online mode	Program modifications in online mode Program change in online mode
Diagnostic functions for the PLC	Diagnostic functions for the PLC
Diagnostic functions for network systems	Diagnostic functions for network systems

# 2 Instruction Tables

## 2.1 Subdivision of instructions

The instructions are subdivided into four major categories:

- Sequence instructions
- Application instructions, Part 1
- Application instructions, Part 2
- Data link instructions

The categories of instructions are described more detailed in the following table:

Category of Instruction		Description	Reference
Sequence instructions	Input instruction	Operation start, series and parallel connection of contacts.	Ch. 5.1
	Connection instruction	Series and parallel block connection, storage and processing of operation results, inversion of operation results, conversion of operation results into pulses, setting of edge relays.	Ch. 5.2
	Output instruction	Bit devices, counter and timer contacts, output, setting, and resetting of annunciators, setting and resetting of devices, leading edge and trailing edge output, bit device output inversion, generating pulses.	Ch. 5.3
	Shift instruction	Shifting bit devices.	Ch. 5.4
	Master control instruction	Setting and resetting single parts of a program.	Ch. 5.5
	Termination instruction	End of a part of program, end of sequence and routine programs.	Ch. 5.6
	Miscellaneous instructions	Sequence program stop, no operation.	Ch. 5.7
Application instructions Part 1	Comparison operation instruction	Compares data to data (e.g. =, >, ≥)	Ch. 6.1
	Arithmetic operation instruction	Adds, subtracts, multiplies, divides, increments, and decrements BIN and BCD data, floating point data, and BIN block data, links character strings	Ch. 6.2
	Data conversion instruction	Converts data types, e.g. BCD → BIN, BIN → BCD	Ch. 6.3
	Data transfer instruction	Transmits designated data	Ch. 6.4
	Program branch instruction	Program jump commands	Ch. 6.5
	Program execution control instruction	Enables and disables program interrupts	Ch. 6.6
	Refresh instruction	Refreshes bit devices, links, and I/O interfaces	Abs 6.7
	Other convenient instructions	Count 1- or 2-phase input up or down, teaching timer, special function timer, rotary table near path rotation control, ramp signal, pulse density measurement, fixed cycle pulse output, pulse width modulation, matrix input	Ch. 6.8

Category of Instruction		Description	Reference
Application instructions Part 2	Logical operation instructions	Logical AND / OR, logical exclusive OR / exclusive NOR	Ch. 7.1
	Rotation instructions	16-bit and 32-bit data right / left rotation	Ch. 7.2
	Shift instructions	Shift data by bit or word	Ch. 7.3
	Bit processing instructions	Set, reset, and test bits	Ch. 7.4
	Data processing instructions	Search, encode, and decode data at specified devices Disunite and unite data	Ch. 7.5
	Structured program instructions	Repeated operation, subroutine program calls, subroutine calls between program files, switching between main and subprogram parts, micro computer program calls, index qualification of entire ladders, store index qualification values in data tables	Ch. 7.6
	Data table operation instructions	Write to and read data from a data table, delete and insert data blocks in a data table	Ch. 7.7
	Buffer memory access instructions	Buffer memory access of special function modules or remote modules	Ch. 7.8
	Display instructions	Output ASCII characters to the outputs of a module or to an LED display	Ch. 7.9
	Debugging and failure diagnosis instructions	Failure checks, setting and resetting status latch, sampling trace, program trace	Ch. 7.10
	Character string processing instructions	Character string (ASCII code) processing	Ch. 7.11
	Special function instructions	Trigonometrical functions, square root and exponential calculation with BCD data and floating point data	Ch. 7.12
	Data control instructions	Upper and lower limit control and storage of checked data	Ch. 7.13
	File register switching instructions	Switching between file register blocks and files	Ch. 7.14
	Clock instructions	Writing and reading clock data	Ch. 7.15
	Peripheral device instructions	Message output and key input on peripheral units	Ch. 7.16
	Program instructions	Select different program execution modes	Ch. 7.17
Other instructions	Reset watchdog timer (WDT), set and reset carry, pulse generation, direct read from indirect access file registers, numerical key input from keyboard, batch save or recovery of index registers, write to EEPROM file registers	Ch. 7.18	
Data link instructions	Network refresh instructions	Instructions for data refresh operations in network modules.	Ch. 8.5
	Dedicated data link instructions	Read and write data from and to object stations in object networks, Send data to network modules in object stations in object networks, Read data sent via SEND instruction, Data requests to different stations (write/read operations with clock data, RUN/STOP operations), Read and write data from and to special function modules in remote I/O stations.	Ch. 8.6
	A series compatible data link instructions	Read and write data from and to object stations in different networks, Read and write data from and to local stations (at master stations only), Read and write data from and to special function modules in remote I/O stations.	Ch. 8.7
	Read/Write routing information	Read and write routing parameters (network number and station number of relay station, station number of routing station).	Ch. 8.8

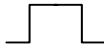



Category of Instruction		Description	Reference
Instructions for a CPU of the System Q	Reading module information	Reads module information from the designated head I/O number	Ch. 9.1
	Trace set/Trace reset	Stores trace data in the trace file in a memory card	Ch. 9.2
	Writing to and reading from files	Writes data to the designated file. Reads data from the designated file	Ch. 9.3
	Programm instructions	Load, unload, load + unload program from memory card	Ch. 9.4
	Data transfer	High-speed block transfer of file register	Ch. 9.5
	Data exchange in a multi-CPU system	Writing to the CPU shared memory Reading from the CPU shared memory of another CPU	Ch. 9.6
Instructions for a Q4ARCPU	Mode settings	Operation mode setting for CPU start up and for switching from the control system to the standby system	Ch. 10.1
	Data transfer	Transfer of data from the control system CPU to the CPU of the standby system Batch transfer of data to and from the buffer memory of special function modules	Ch. 10.2

## 2.2 Overview of instructions

### 2.2.1 Description of the overview tables

The following sections 2.3 through 2.6 include an overview of all instructions described in this manual.

In the following the layout of the overview table is described in detail:

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Addition and subtraction of 16-bit binary data	+	s, d	$(d)+(s) \rightarrow (d)$		3	5	6.2.1
	+P						6.2.1
	+	s1, s2, d1	$(s1)+(s2) \rightarrow (d1)$		4	7	6.2.1
	+P						6.2.1

(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)

Explanation of the different columns:

(1) Category of instruction

(2) Specification of instruction name ("command") for the programming

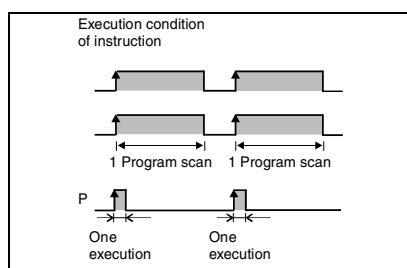
The instruction names are represented in MELSEC notation (refer to section 3.2 for explanation of the notation).

In general, 16-bit instructions are represented. All 32-bit instructions are indicated by a leading "D".

Example: - 16-bit instruction: +  
 - 32-bit instruction: D+

Pulse instructions, i.e. instructions that are only executed at leading edge of a signal are indicated by an appended "P".

Example: - Execution when ON: +  
 - Execution at leading edge: +P





Instructions, processing character strings are indicated by a leading "\$"

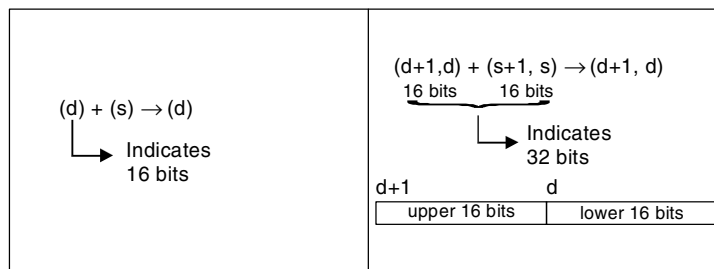
Example: - standard instructions: +  
 - character string instruction: +P

(3) Specification of variables

Here, the variables to be used are specified. The data source is represented by an "s", the data destination is represented by a "d".

Example: s = if there is only one data source  
 s1, s2 = if there are several data sources  
 s+0, s+1, (s1)+0, (s1)+1 = for 32-bit instructions  
 e.g. s1 = data register D0, (s1)+1 = data register D1  
 s+0, s+1, s+2, s+3 = 4 successive devices, e.g. for an array

(4) Meaning and processing of the entire control instruction



(5) Indication of the execution condition according to the following table

Symbol	Execution condition
no indication	The instruction is executed continuously and independent from the prior execution condition. If the precondition is not set, the instruction is not executed.
	The instruction is executed as long as the precondition is ON. If the precondition is OFF, the instruction is not executed and no processing is conducted.
	This instruction is a pulsed instruction. It is only executed once and at leading edge of the input signal (e.g. if the precondition alters from OFF to ON). Afterwards, the instruction will not be executed any longer even if the input signal is still ON.
	This instruction is a pulsed instruction as well. It is only executed once and at trailing edge of the input signal (e.g. if the precondition alters from OFF to ON). Afterwards, the instruction will not be executed any longer even if the input signal is still ON.

(6+7) Indication of the number of program steps

Indicated is the number of steps that are required for the entire execution of the instruction. A distinction is drawn between the MELSEC A and Q series/System Q. Refer to section 3.9 for details.

(8) Indication of the reference chapter

Indicates the chapter and section of this manual where the instruction is described in detail.

## 2.3 Sequence instructions

### 2.3.1 Input instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Input instruction	LD		Operation start (Load (normally open contact))		*	1	5.1.1
	LDI		Operation start (Load (normally closed contact))				
	AND		Series connection (of NO contacts)				
	ANI		Series connection (of NC contacts)				
	OR		Parallel connection (of NO contacts)				
	ORI		Parallel connection (of NC contacts)				
	LDP		Pulse operation start (leading edge)		*	2	5.1.2
	LDF		Pulse operation start (trailing edge)				
	ANDP	s	Pulse series connection (leading edge)				
	ANDF	s	Pulse series connection (trailing edge)				
	ORP	s	Pulse parallel connection (leading edge)				
	ORF	s	Pulse parallel connection (trailing edge)				

- \*: The number of program steps depends on the devices used.
- For the use of internal devices or file registers (R0 through R32767) : 1
  - For the use of a direct access input (DX) : 2
  - For the use of other devices : 3

NOTE: The number of program steps can double if file registers 2R on a memory card are used.

2.3.2 Connection instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Connection instruction	ANB	—	Block series connection (Ladder block series connection)		1	1	5.2.1
	ORB		Block parallel connection (Ladder block parallel connection)				
	MPS	—	Operation result processing (Store operation result (memory push))		1	1	5.2.2
	MRD		Operation result processing (Read operation result (memory read))				
	MPP		Operation result processing (Read and clear operation result (memory pop))				
	INV	—	Operation result inversion (Inversion instruction)		1		5.2.3
	MEP	—	Operation result into pulse conversion (Pulse generation at leading edge of operation result)		1		5.2.4
	MEF		Operation result into pulse conversion (Pulse generation at trailing edge of operation result)				
	EGP	d	Setting of edge relays (Setting an edge relay with leading edge of an operation result)		1		5.2.5
	EGF		Setting of edge relays (Setting an edge relay with trailing edge of an operation result)				

2.3.3 Output instruction

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Output instruction	OUT	d	Setting instructions for outputs		* 1	* 1	5.3.1
	SET	d	Setting of devices	** 	* 1		5.3.5
	RST	d	Resetting devices	** 	2	* 1	5.3.6
	PLS	d	Output at leading edge		2	* 3	5.3.8
	PLF		Output at trailing edge				
	FF	s	Inversion of bit output device		2		5.3.9
	DELTA	d	Generating pulses at direct access outputs		2		5.3.11
	DELTAP						

\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

\*\* : This execution condition is only applied, if the annunciator (F) is used.

2.3.4 Shift instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Verschiebe- anweisungen	SFT	d	Shifting bit devices		2	* 3	5.4.1
	SFTP						

\*: Refer to chapter 3.9.2 "For an AnA, AnAS, and AnU CPU" for the according number of steps.

**2.3.5 Master control instructions**


Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Master control instruction	MC	n, d	Activating indicated program parts		2	* 3/5	5.5.1
	MCR	n	Deactivating indicated program parts		1		

\*: The according number of steps is 5 for all MC instructions and 3 for the MCR instruction. Refer to chapter 3.9.2 "For an AnA, AnAS, and AnU CPU" for the according number of steps.

**2.3.6 Program termination instructions**

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Termination instruction	FEND	—	End of program branches		1		5.6.1
	END		End of sequence program				5.6.2

**2.3.7 Miscellaneous instructions**

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Sonstige Anweisungen	STOP	—	Stop instruction		1		5.7.1
	NOP	—	No operation program step				5.7.2

## 2.4 Application instructions, Part 1

### 2.4.1 Comparison operation instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BIN 16-bit data comparison	LD=	s1, s2	Sets the output, if s1 = s2		3	* 5/7	6.1.1
	AND=						
	OR=						
	LD<>	s1, s2	Sets the output, if s1 ≠ s2		3	* 5/7	6.1.1
	AND<>						
	OR<>						
	LD>	s1, s2	Sets the output, if s1 > s2		3	* 5/7	6.1.1
	AND>						
	OR>						
	LD<=	s1, s2	Sets the output, if s1 ≤ s2		3	* 5/7	6.1.1
	AND<=						
	OR<=						
	LD<	s1, s2	Sets the output, if s1 < s2		3	* 5/7	6.1.1
	AND<						
	OR<						
LD>=	s1, s2	Sets the output, if s1 ≥ s2		3	* 5/7	6.1.1	
AND>=							
OR>=							

\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BIN 32-bit data comparison	LDD=	s1, s2	Sets the output, if s1 = s2		*	**	6.1.2
	ANDD=				3	11	
	ORD=						
	LDD<>	s1, s2	Sets the output, if s1 ≠ s2		*	**	6.1.2
	ANDD<>				3	11	
	ORD<>						
	LDD>	s1, s2	Sets the output, if s1 > s2		*	**	6.1.2
	ANDD>				3	11	
	ORD>						
	LDD<=	s1, s2	Sets the output, if s1 ≤ s2		*	**	6.1.2
	ANDD<=				3	11	
	ORD<=						
	LDD<	s1, s2	Sets the output, if s1 < s2		*	**	6.1.2
	ANDD<				3	11	
	ORD<						
LDD>=	s1, s2	Sets the output, if s1 ≥ s2		*	**	6.1.2	
ANDD>=				3	11		
ORD>=							

\*: The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU or a single processor CPU of the System Q is used: 3
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 5  
 constants: 5  
 Bit Devices, whose device numbers are multiplies of 16,  
 whose digit designation is K8, and which use no index qualification: 5
- If a System Q multi processor CPU is used with devices other than above mentioned: 5

The processing speed is faster with a System Q CPU although the number of steps is increased.

\*\* : The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

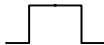

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Floating point data comparison	LDE=	s1, s2	Sets the output, if $s1 = s2$		3		6.1.3
	ANDE=						
	ORE=						
	LDE<>	s1, s2	Sets the output, if $s1 \neq s2$		3		6.1.3
	ANDE<>						
	ORE<>						
	LDE>	s1, s2	Sets the output, if $s1 > s2$		3		6.1.3
	ANDE>						
	ORE>						
	LDE<=	s1, s2	Sets the output, if $s1 \leq s2$		3		6.1.3
	ANDE<=						
	ORE<=						
	LDE<	s1, s2	Sets the output, if $s1 < s2$		3		6.1.3
	ANDE<						
	ORE<						
LDE>=	s1, s2	Sets the output, if $s1 \geq s2$		3		6.1.3	
ANDE>=							
ORE>=							





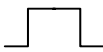

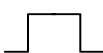

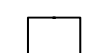

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Character string data comparison	LD\$=	s1, s2	* Compares the character strings in s1 and s2 character by character. Sets the output, if s1 = s2		3		6.1.4
	AND\$=						
	OR\$=						
	LD\$<>	s1, s2	* Compares the character strings in s1 and s2 character by character. Sets the output, if s1 ≠ s2		3		6.1.4
	AND\$<>						
	OR\$<>						
	LD\$>	s1, s2	* Compares the character strings in s1 and s2 character by character. Sets the output, if s1 > s2		3		6.1.4
	AND\$>						
	OR\$>						
	LD\$<=	s1, s2	* Compares the character strings in s1 and s2 character by character. Sets the output, if s1 <= s2		3		6.1.4
	AND\$<=						
	OR\$<=						
	LD\$<	s1, s2	* Compares the character strings in s1 and s2 character by character. Sets the output, if s1 < s2		3		6.1.4
	AND\$<						
	OR\$<						
LD\$>=	s1, s2	* Compares the character strings in s1 and s2 character by character. Sets the output, if s1 >= s2		3		6.1.4	
AND\$>=							
OR\$>=							

\*: Conditions under which the character string comparison is processed:

- Match: All characters in the string must match.
- Larger string: If the character strings differ, the larger string is determined.
- Smaller string: If the character strings differ, the smaller string is determined.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BIN block data comparison	BKCMP=	s1, s2, n, d1	It compares the nth BIN 16-bit block in s1 to the nth BIN 16-bit block in s2, beginning with the first number of device. The result of each block comparison is stored from d1 onwards.		5		6.1.5
	BKCMP<>	s1, s2, n, d1					
	BKCMP>	s1, s2, n, d1					
	BKCMP<=	s1, s2, n, d1					
	BKCMP<	s1, s2, n, d1					
	BKCMP>=	s1, s2, n, d1					
	BKCMP=P	s1, s2, n, d1					
	BKCMP<>P	s1, s2, n, d1					
	BKCMP>P	s1, s2, n, d1					
	BKCMP<=P	s1, s2, n, d1					
	BKCMP<P	s1, s2, n, d1					
	BKCMP>=P	s1, s2, n, d1					

2.4.2 Arithmetic operation instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BIN 16-bit addition and subtraction operations	+	s, d	$(d)+(s) \rightarrow (d)$		3	5	6.2.1
	+P						6.2.1
	+	s1, s2, d1	$(s1)+(s2) \rightarrow (d1)$		4	7	6.2.1
	+P						6.2.1
	-	s, d	$(d)-(s) \rightarrow (d)$		3	5	6.2.1
	-P						6.2.1
	-	s1, s2, d1	$(s1)-(s2) \rightarrow (d1)$		4	7	6.2.1
	-P						6.2.1







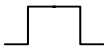

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference	
					Q	A		
BIN 32-bit addition and subtraction operations	D+	s, d	$(d+1, d)+(s+1, s) \rightarrow (d+1, d)$		*	3	9	6.2.2
	D+P							
	D+	s1, s2, d1	$((s1)+1, s1)+((s2)+1, s2) \rightarrow ((d1)+1, d1)$		**	4	11	6.2.2
	D+P							
	D-	s, d	$(d+1, d)-(s+1, s) \rightarrow (d+1, d)$		*	3	9	6.2.2
	D-P							
	D-	s1, s2, d1	$((s1)+1, s1)-((s2)+1, s2) \rightarrow ((d1)+1, d1)$		**	4	11	6.2.2
	D-P							

\*: The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU or single processor CPU of the System Q is used: 3
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 5  
constants: 5  
Bit Devices, whose device numbers are multiplies of 16,  
whose digit designation is K8, and which use no index qualification: 5
- If a System Q multi processor CPU is used with devices other than above mentioned: 3

\*\* : The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU single processor CPU of the System Q is used: 4
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 6  
constants: 6  
Bit Devices, whose device numbers are multiplies of 16,  
whose digit designation is K8, and which use no index qualification: 6
- If a System Q multi processor CPU is used with devices other than above mentioned: 4

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference	
					Q	A		
BIN 16-bit multiplication and division	x	s1, s2, d1	$(s1) \times (s2) \rightarrow ((d1)+1, d1)$		*	4	**	6.2.4
	xP						7	6.2.4
	/	s1, s2, d1	$(s1)/(s2) \rightarrow$ Quotient (d1), remainder $((d1)+1)$		*	4	**	6.2.4
	/P						7	6.2.4
BIN 32-bit multiplication and division	Dx	s1, s2, d1	$((s1)+1, s1) \times ((s2)+1, s2)$ $\rightarrow$ $((d1)+3, (d1)+2,$ $(d1)+1, d1)$		*	4	**	6.2.4
	DxP						11	6.2.4
	D/	s1, s2, d1	$((s1)+1, s1) / ((s2)+1, s2)$ $\rightarrow$ Quotient $((d1)+1, d1),$ remainder $((d1)+3,$ $(d1)+2)$		*	4	**	6.2.4
	D/P						11	6.2.4







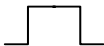

\*: The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU or System Q single processor CPU is used: 4
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 3  
constants: 3  
Bit Devices, whose device numbers are multiples of 16,  
whose digit designation is K8, and which use no index qualification: 3
- If a System Q multi processor CPU is used with devices other than above mentioned: 4

\*\* : The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BCD 4-digit addition and subtraction operations	B+	s, d	$(d)+(s) \rightarrow (d)$		3	* 7	6.2.5
	B+P						6.2.5
	B+	s1, s2, d1	$(s1)+(s2) \rightarrow (d1)$		4	* 9	6.2.5
	B+P						6.2.5
	B-	s, d	$(d)-(s) \rightarrow (d)$		3	* 7	6.2.5
	B-P						6.2.5
	B-	s1, s2, d1	$(s1)-(s2) \rightarrow (d1)$		4	* 9	6.2.5
	B-P						6.2.5
BCD 8-digit addition and subtraction operations	DB+	s, d	$(d+1, d)+(s+1,s) \rightarrow (d+1, d)$		3	* 9	6.2.6
	DB+P						6.2.6
	DB+	s1, s2, d1	$((s1)+1, s1)+((s2)+1,s2) \rightarrow ((d1)+1, d1)$		4	* 11	6.2.6
	DB+P						6.2.6
	DB-	s, d	$(d+1, d)+(s+1,s) \rightarrow (d+1, d)$		3	* 9	6.2.6
	DB-P						6.2.6
	DB-	s1, s2, d1	$((s1)+1, s1)+((s2)+1,s2) \rightarrow ((d1)+1, d1)$		4	* 11	6.2.6
	DB-P						6.2.6





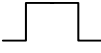
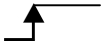


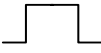


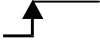
\*\* : The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BCD 4-digit multiplication and division operations	B $\times$	s1, s2, d1	$(s1) \times (s2) \rightarrow ((d1)+1, d1)$		4	* 9	6.2.7
	B $\times$ P						6.2.7
	B/	s1, s2, d1	$(s1)/(s2) \rightarrow$ Quotient (d1), remainder ((d1)+1)		4	* 9	6.2.7
	B/P						6.2.7
BCD 8-digit multiplication and division operations	DB $\times$	s1, s2, d1	$((s1)+1, s1) \times ((s2)+1, s2) \rightarrow$ $((d1)+3, (d1)+2,$ $(d1)+1, d1)$		4	* 11	6.2.8
	DB $\times$ P						6.2.8
	DB/	s1, s2, d1	$((s1)+1, s1)/((s2)+1, s2) \rightarrow$ Quotient ((d1)+1, d1), remainder ((d1)+3, (d1)+2)		4	* 11	6.2.8
	DB/P						6.2.8

\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Floating point data addition and subtraction operations	E+	s, d	$(d+1, d)+(s+1, s) \rightarrow (d+1, d)$		3		6.2.9
	E+P						6.2.9
	E+	s1, s2, d1	$((s1)+1, s1)+((s2)+1, s2) \rightarrow ((d1)+1, d1)$		4		6.2.9
	E+P						6.2.9
	E-	s, d	$(d+1, d)-(s+1, s) \rightarrow (d+1, d)$		3		6.2.9
	E-P						6.2.9
	E-	s1, s2, d1	$((s1)+1, s1)-((s2)+1, s2) \rightarrow ((d1)+1, d1)$		4		6.2.9
	E-P						6.2.9
Floating point data multiplication and division operations	Ex	s1, s2, d1	$((s1)+1, s1) \times ((s2)+1, s2) \rightarrow ((d1)+1, d1)$		4		6.2.10
	ExP						6.2.10
	E/	s1, s2, d1	$((s1)+1, s1) / ((s2)+1, s2) \rightarrow$ Quotient $((d1)+1, d1)$		4		6.2.10
	E/P						6.2.10
BIN block addition and subtraction operations	BK+	s1, s2, d, n	Adds the nth 16-bit block in s1 to the nth 16-bit block in s2.		5		6.2.11
	BK+P						6.2.11
	BK-	s1, s2, d, n	Subtracts the nth 16-bit block in s2 from the nth 16-bit block in s1.		5		6.2.11
	BK-P						6.2.11



Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Character string linking operations	\$+	s, d	Character string data in s is appended to character data in d. The linked character string is stored in d.		3		6.2.12
	\$+P						6.2.12
	\$+	s1, s2, d1	Character string data in s is appended to character data in d. The linked character string is stored in d.		4		6.2.12
	\$+P						6.2.12
BIN increment operations	INC	d	(d)+1 → (d)		2	** 3	6.2.13
	INCP						6.2.13
	DINC	d	(d+1, d)+1 → (d+1, d)		* 2	** 3	6.2.13
	DINCP						6.2.13
BIN decrement operations	DEC	d	(d)-1 → (d)		2	** 3	6.2.14
	DECP						6.2.14
	DDEC	d	(d+1, d)-1 → (d+1, d)		* 2	** 3	6.2.14
	DDECP						6.2.14

\*: The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU or System Q single processor CPU is used: 2
- If a Q multi processor CPU is used with internal word devices (except for file register ZR): 3 constants: 3
- Bit Devices, whose device numbers are multiples of 16, whose digit designation is K8, and which use no index qualification: 3
- If a System Q multi processor CPU is used with devices other than above mentioned: 2

\*\* : The number of program steps depends on the devices used. Refer to the reference chapter for the accurate number of steps.

2.4.3 Data conversion instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Conversion from BIN data into BCD data	BCD	s, d	<p>BCD conversion → (d)                      ↑                      BIN (0 to 9999)</p>		3	* 5	6.3.1
	BCDP			6.3.1			
	DBCD	s, d	<p>BCD conversion                      (s+1, s) → (d+1, d)                      ↑                      BIN (0 to 9999999)</p>		3	* 9	6.3.1
	DBCDP			6.3.1			
Conversion from BCD data into BIN data	BIN	s, d	<p>BIN conversion → (d)                      ↑                      BCD (0 to 9999)</p>		3	* 5	6.3.2
	BINP			6.3.2			
	DBIN	s, d	<p>BIN conversion                      (s+1, s) → (d+1, d)                      ↑                      BCD (0 to 99999999)</p>		3	* 9	6.3.2
	DBINP			6.3.2			
Conversion from BIN data into floating point data	FLT	s, d	<p>Floating point conversion                      (s+1, s) → (d)                      ↑                      Binary value (-32768 to 32767)</p>		3		6.3.3
	FLTP			6.3.3			
	DFLT	s, d	<p>Floating point conversion                      (s+1, s) → (d+1, d)                      ↑                      Binary value (-2147483648 to 2147483647)</p>		3		6.3.3
	DFLTP			6.3.3			
Conversion from floating point data into BIN data	INT	s, d	<p>BIN conversion                      (s+1, s) → (d)                      ↑                      Floating point value (-32768 to 32767)</p>		3		6.3.4
	INTP			6.3.4			
	DINT	s, d	<p>Floating point conversion                      (s+1, s) → (d+1, d)                      ↑                      Binary value (-2147483648 bis 2147483647)</p>		3		6.3.4
	DINTP			6.3.4			

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Conversion from BIN 16-bit data into BIN 32-bit data	DBL	s, d	Conversion $(s) \rightarrow (d+1, d)$ BIN (-32768 to 32767)		3		6.3.5
	DBLP						6.3.5
Conversion from BIN 32-bit data into BIN 16-bit data	WORD	s, d	Conversion $(s+1, s) \rightarrow (d)$ BIN (-32768 to 32767)		3		6.3.6
	WORDP						6.3.6
Conversion from BIN 16-/32-bit data into Gray code data	GRY	s, d	Conversion into Gray code $(s) \rightarrow (d)$ Binary value (-32768 to 32767)		3		6.3.7
	GRYP						6.3.7
	DGRY	s, d	Conversion into Gray code $(s+1, s) \rightarrow (d+1, d)$ Binary value (-2147483648 to 2147483647)		3		6.3.7
	DGRYP						6.3.7
Conversion from Gray code data into BIN 16-/32-bit data	GBIN	s, d	BIN conversion $(s) \rightarrow (d)$ Gray code (-32768 to 32767)		3		6.3.8
	GBINP						6.3.8
	DGBIN	s, d	BIN conversion $(s+1, s) \rightarrow (d+1, d)$ Gray code (-2147483648 to 2147483647)		3		6.3.8
	DGBINP						6.3.8
Sign reversal for BIN 16-/32-bit data (complement of 2)	NEG	d	$(\bar{d}) \rightarrow (d)$ BIN data		2	* 3	6.3.9
	NEGP						6.3.9
	DNEG	d	$(\bar{d+1}, d) \rightarrow (d+1, d)$ BIN data		2	* 3	6.3.9
	DNEGP						6.3.9

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Signal reversal for floating point data	ENEG	d			2		6.3.10
	ENEGP						
Conversion from BIN block data into BCD block data	BKBCD	s, d, n	This instruction converts each nth BIN 16-bit block in s into the nth BCD 4-digit block. Converted data is stored in d.		4		6.3.11
	BKBCDP	s, d, n					
Conversion from BCD block data into BIN block data	BKBIN	s, d, n	This instruction converts each nth BCD 4-digit block in s into the nth BIN 16-bit block. Converted data is stored in d.		4		6.3.12
	BKBINP	s, d, n					

2.4.4 Data transfer instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BIN 16-bit data transfer	MOV	s, d	(s) → (d)		*	***	6.4.1
	MOVP	s, d			3	5	
BIN 32-bit data transfer	DMOV	s, d	+1, s) → (d+1, d)		**	***	6.4.1
	DMOVP	s, d			3	7	
Floating point data transfer	EMOV	s, d	(s+1, s) → (d+1, d)		3		6.4.2
	EMOVP	s, d	└─ Floating point value				
Character string data transfer	\$MOV	s, d	Transfers character string data in s to d.		3		6.4.3
	\$MOVP	s, d					
BIN 16-bit data inversion	CML	s, d	(s) → (d)		*	***	6.4.4
	CMLP	s, d			3	5	
BIN 32-bit data inversion	DCML	s, d	(s+1, s) → (d1+1, d1)		**	***	6.4.4
	DCMLP	s, d			3	7	

- \*: The number of program steps depends on the devices used and the type of CPU.
- If a QnA CPU or System Q single processor CPU is used: 3
  - If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 2
  - constants: 2
  - Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 2
  - If a System Q multi processor CPU is used with devices other than above mentioned: 3


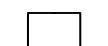
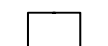
- \*: The number of program steps depends on the devices used and the type of CPU.
- If a System Q single processor CPU is used: 2
  - If a QnA CPU or a System Q multi processor CPU is used: 3

\*\*\*: The number of program steps depends on the devices used. Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
BIN block data transfer	BMOV	s, n, d			4	* 9	6.4.5
	BMOVP	s, n, d					
Identical BIN block data transfer	FMOV	s, n, d			4	* 9	6.4.6
	FMOVP	s, n, d					
BIN 16-bit data exchange	XCH	d1, d2			3	* 5	6.4.7
	XCHP	d1, d2					
BIN 32-bit data exchange	DXCH	d1, d2			3	* 7	6.4.7
	DXCHP	d1, d2					
BIN block data exchange	BXCH	n, d1, d2			4		6.4.8
	BXCHP	n, d1, d2					
Upper and lower byte exchanges	SWAP	s			3		6.4.9
	SWAPP	s					

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

2.4.5 Program branch instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Jump instructions	CJ	p	Conditional jump (p = jump destination)		2	* 3	6.5.1
	SCJ	p	Conditional jump from next program scan (p = jump destination)				
	JMP	p	Jump instruction (p = jump destination)		2	* 3	6.5.1
	GOEND		Jump to the end of a program		1		6.5.2


\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

2.4.6 Interrupt program execution control instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Interrupt disabled	DI		Disables the execution of an interrupt program		1	* 1	6.6.1
Interrupt enabled	EI		Enables invoking an interrupt program		1	* 1	6.6.1
Bit pattern of execution conditions of interrupt programs	IMASK	s	In the bit pattern designated by s a particular interrupt address is allocated to each bit.		2	* 1	6.6.1
Return from an interrupt program to the main program	IRET		End of an interrupt program		1	* 1	6.6.2

\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

2.4.7 Data refresh instructions

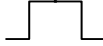
Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
I/O partial refresh	RFS	s, n	The RFS instruction refreshes the inputs and outputs of the designates range of I/O devices during one program scan.		3		6.7.1
I/O partial refresh	SEG	s, d	The SEG instruction enables refreshing a determined range of I/O devices, if the input condition is set.			* 9	6.7.2
Refresh instruction for link and interface data and CPU shared memory	COM		If SM775 (Q series and System Q only) is not set (0), the link and interface data are refreshed (link refresh) and general data processing is performed (END processing). Used also for automatic refresh of the multi-CPU shared memory		1	* 3	6.7.3
Disable link refresh execution	DI		The DI instruction disables the execution of a link refresh until an EI instruction is executed.		1		6.7.4
Enable link refresh execution	EI		The execution of a link refresh is enabled after setting an EI instruction.		1		6.7.4

\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.



2.4.8 Other convenient instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
1-Phase Input count-up/-down Counter	UDCNT1	s, n, d			4		6.8.1
2-Phase Input count-up/-down Counter	UDCNT2	s, n, d			4		6.8.2
Programmable (teaching) Timer	TTMR	d, n	(Time, the timer is set) x n → (d) n=0:1, n=1:10, n=2:100		3		6.8.3
Special Function Timer (Timer instruction for low speed timers)	STMR	s,n, d	The STMR instruction uses outputs designated by d+0 through d+3 to perform four different timer functions: d+0:OFF delay timer output d+1:One shot timer output after OFF (Set by trailing edge) d+2:One shot timer output after ON (Set by leading edge) d+3:ON delay timer output		3		6.8.4
Special Function Timer (Timer instruction for high speed timers)	STMRH	s,n, d	see above		3		6.8.4
Positioning instruction for rotary tables	ROTC	s, n1, n2, d	The ROTC instruction rotates a sector designated by s+2 on a table with a specified number of sectors (divisions) designated by n1 to a specified position designated by s+1.		5		6.8.5
Ramp Signal	RAMP	n1, n2, n3, d1, d2	A RAMP instruction changes the content in (d1)+0 gradually from the initial value designated by n1 to the final value designated by n2.		6		6.8.6
Pulse density measurement	SPD	s, n, d	The SPD instruction counts pulses at the input designated by s for a period of time specified by n. The result of the measurement is stored in d.		4		6.8.7
Pulse output with adjustable number of pulses	PLSY	s1, s2, d	The PLSY instruction outputs a number of pulses specified by s2 at a frequency specified by s1 to an output designated by d.		4		6.8.8
Pulse width modulation	PWM	n1, n2, d			4		6.8.9

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Building an input matrix	MTR	s, n , d1, d2	The MTR instruction reads the information of 16 bits beginning from the device designated by s. The number of repetitions (rows) is designated by n. The conditions of read data are stored in the device designated by d2 onwards.		5		6.8.10

## 2.5 Application instructions, Part 2

### 2.5.1 Logical operation instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Logical product	WAND	s, d	$(d) \wedge (s) \longrightarrow (d)$		3	*** 5	7.1.1
	WANDP						
	WAND	s1, s2, d1	$(s1) \wedge (s2) \longrightarrow (d1)$		4	*** 7	7.1.1
	WANDP						
	DAND	s, d	$(d+1, d) \wedge (s+1, s) \longrightarrow (d+1, d)$		* 4	*** 9	7.1.1
	DANDP						
	DAND	s1, s2, d	$((s1)+1, s1) \wedge ((s2)+1, s2) \longrightarrow (d+1, d)$		** 4		7.1.1
	DANDP						
	BKAND	s1, s2, n, d			5		7.1.2
BKANDP							

\*: The number of program steps depends on the devices used and the type of CPU.

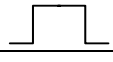
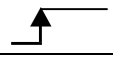
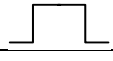
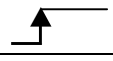
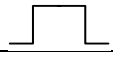
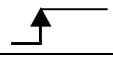
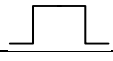
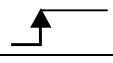
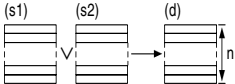
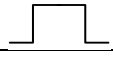
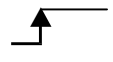
- If a QnA CPU is used: 4
- If a System Q single processor CPU is used: 3
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 6 constants: 6  
Bit Devices, whose device numbers are multiples of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q multi processor CPU is used with devices other than above mentioned: 4

\*\* : The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU is used: 4
- If a System Q CPU is used with internal word devices (except for file register ZR): 6 constants: 6  
Bit Devices, whose device numbers are multiples of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q CPU is used with devices other than above mentioned: 4

\*\*\*: The number of program steps depends on the devices used.

Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Logical sum	WOR	s, d	$(d) \vee (s) \longrightarrow (d)$		3	*** 5	7.1.3
	WORP						
	WOR	s1, s2, d1	$(s1) \vee (s2) \longrightarrow (d1)$		4	*** 7	7.1.3
	WORP						
	DOR	s, d	$(d+1, d) \vee (s+1, s) \longrightarrow (d+1, d)$		*	*** 9	7.1.3
	DORP						
	DOR	s1, s2, d	$((s1)+1, s1) \vee ((s2)+1, s2) \longrightarrow (d+1, d)$		**	4	7.1.3
	DORP						
	BKOR	s1, s2, n, d			5		7.1.4
	BKORP						

\*: The number of program steps depends on the devices used and the type of CPU.





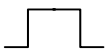
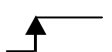


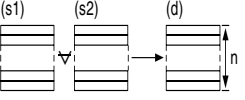
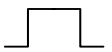
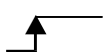
- If a QnA CPU is used: 4
- If a System Q single processor CPU is used: 3
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 6  
constants: 6  
Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q multi processor CPU is used with devices other than above mentioned: 4

\*\* : The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU is used: 4
- If a System Q CPU is used with internal word devices (except for file register ZR): 6  
constants: 6  
Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q CPU is used with devices other than above mentioned: 4

\*\*\*: The number of program steps depends on the devices used.

Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Logical exclusive OR	WXOR	s, d	$(d) \vee (s) \longrightarrow (d)$		3	*** 5	7.1.5
	WXORP						
	WXOR	s1, s2, d1	$(s1) \vee (s2) \longrightarrow (d1)$		4	*** 7	7.1.5
	WXORP						
	DXOR	s, d	$(d+1, d) \vee (s+1, s) \longrightarrow (d+1, d)$		* 3	*** 9	7.1.5
	DXORP						
	DXOR	s1, s2, d	$((s1)+1, s1) \vee ((s2)+1, s2) \longrightarrow (d+1, d)$		** 4		7.1.5
	DXORP						
	BKXOR	s1, s2, n, d			5		7.1.6
	BKXORP						

\*: The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU is used: 4
- If a System Q single processor CPU is used: 3
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 6  
constants: 6  
Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q multi processor CPU is used with devices other than above mentioned: 4

\*\* : The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU is used: 4
- If a System Q CPU is used with internal word devices (except for file register ZR): 6  
constants: 6  
Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q CPU is used with devices other than above mentioned: 4

\*\*\*: The number of program steps depends on the devices used.

Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Logical exclusive NOR	WNXR	s, d	$(d) \vee (s) \longrightarrow (d)$		3	*** 5	7.1.7
	WNXRP						
	WNXR	s1, s2, d1	$\overline{(s1)} \vee \overline{(s2)} \overset{\text{d1}}{\text{d}}$		4	*** 7	7.1.7
	WNXRP						
	DNXR	s, d	$\overline{(d+1, d)} \vee \overline{(s+1, s)} \overset{\text{d}}{\text{d}}$		*	*** 9	7.1.7
	DNXRP						
	DNXR	s1, s2, d	$\overline{((s1)+1, s1)} \vee \overline{((s2)+1, s2)} \overset{\text{d}}{\text{d}}$		**	4	7.1.7
	DNXRP						
	BKXNR	s1, s2, n, d			5		7.1.8
BKXNRP							

\*: The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU is used: 4
- If a System Q single processor CPU is used: 3
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 6  
constants: 6  
Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q multi processor CPU is used with devices other than above mentioned: 4

\*\* : The number of program steps depends on the devices used and the type of CPU.

- If a QnA CPU is used: 4
- If a System Q CPU is used with internal word devices (except for file register ZR): 6  
constants: 6  
Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 6
- If a System Q CPU is used with devices other than above mentioned: 4

\*\*\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

2.5.2 Rotation instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Data rotation to the right (16-bit)	ROR	n, d			3	*	7.2.1
	RORP		rotates by n bits to the right				
	RRCR	n, d			3	*	7.2.1
	RRCRP		rotates by n bits to the right				
Data rotation to the left (16-bit)	ROL	n, d			3	*	7.2.2
	ROLP		rotates by n bits to the left				
	RCL	n, d			3	*	7.2.2
	RCLP		rotates by n bits to the left				
Data rotation to the right (32-bit)	DROR	n, d			3	*	7.2.3
	DRORP		rotates by n bits to the right				
	DRRCR	n, d			3	*	7.2.3
	DRRCRP		rotates by n bits to the right				
Data rotation to the left (32-bit)	DROL	n, d			3	*	7.2.4
	DROLP		rotates by n bits to the left				
	DRCL	n, d			3	*	7.2.4
	DRCLP		rotates by n bits to the left				

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

2.5.3 Shift instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Shift a 16-bit data word by n bits	SFR	n, d			3	* 3	7.3.1
	SFRP						
	SFL	n, d			3	* 3	7.3.1
	SFLP						
Shift n bit devices by 1 bit	BSFR	n, d			3	* 7	7.3.2
	BSFRP						
	BSFL	n, d			3	* 7	7.3.2
	BSFLP						
Shift n word devices by one digit	DSFR	n, d			3	* 7	7.3.3
	DSFRP						
	DSFL	n, d			3	* 7	7.3.3
	DSFLP						

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.



2.5.4 Bit processing instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Set / reset single bits	BSET	n, d			3	* 3	7.4.1
	BSETP						
	BRST	n, d			3	* 7	
	BRSTP						
Test condition of single bits in 16-/32-bit data words	TEST	s1, s2, d			4		7.4.2
	TESTP						
	DTEST	s1, s2, d			4		
	DTESTP						
Reset sections of bits in a batch	BKRST	s, n			3		7.4.3
BKRSTP							

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

2.5.5 Data processing instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference	
					Q	A		
Search 16-bit data	SER	s1, s2, n (A) s1, s2, n, d (Q)			5	* 9	7.5.1	
	SERP							
	DSER	s1, s2, n (A) s1, s2, n, d (Q)			5	* 9	7.5.1	
	DSERP							
Check data bits (16-/32-bit)	SUM	s (A) s, d (Q)			3	* 3	7.5.2	
	SUMP							
	DSUM	s (A) s, d (Q)			3	* 3	7.5.2	
	DSUMP							
Decoding data	DECO	s, d, n	Decoding from 8 to 256 bits			4	* 9	7.5.3
	DECOP							
Encoding data	ENCO	s, d, n	Encoding from 256 to 8 bits			4	* 9	7.5.4
	ENCOP							
7-segment decoding	SEG	s, d			3	7	7.5.5	
	SEGP							

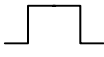

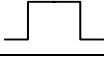
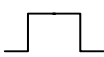
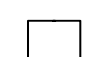
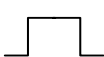
\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Disunite/unite 16-bit data words	DIS	s, n, d	The DIS instruction disunites a 16-bit data value to groupings of 4 bits. The data value to be disunited in s, the number of 4-bit groupings in n, and the first number of destination device in d must be specified.		4	* 9	7.5.6
	DISP						
	UNI	s, n, d	The UNI instruction separates each 4 lowest bits of up to four 16-bit data values and unites their conditions in one 16-bit data value.		4	* 9	7.5.7
	UNIP						
	NDIS	s1, s2, d	The NDIS instruction disunites data in devices specified from s1 on to bit groupings with a number of bits specified by s2. The disunited bit groupings are stored separately in the device specified by d onwards.		4		7.5.8
	NDISP						
	NUNI	s1, s2, d	The NUNI instruction separates bit groupings of a size specified by s2 from devices specified by s1 and unites these bit groupings in one data value. The bit groupings are stored successively from the device specified by d onwards.		4		7.5.8
	NUNIP						
	WTOB	s, n, d	For this instruction the data values in s to be disunited, the number of byte units in n, and the first number of destination device in d must be specified.		4		7.5.9
	WTOBP						
	BTOW	s, n, d	The initial number of data value in s to be united, the number of byte units n, and destination device in d must be specified.		4		7.5.9
	BTOWP						
Search maximum values in 16-/32-bit data	MAX	s, n, d	The MAX instruction searches for maximum values in 16-bit data blocks. The number of data blocks to be searched through is specified by n. The greatest value found in s through s+(n-1) is stored in d.		4		7.5.10
	MAXP						
	DMAX	s, n, d	The DMAX instruction searches for maximum values in 32-bit data blocks. The number of data blocks to be searched through is specified by n. The greatest value found in s through s+(n-1) is stored in d.		4		7.5.10
	DMAXP						

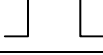


\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference		
					Q	A			
Searching minimum values in 16-/32-bit data	MIN	s, n, d	The MIN instruction searches for minimum values in 16-bit data blocks. The number of data blocks to be searched through is specified by n. The smallest value found in s through s+(n-1) is stored in d.		4		7.5.11		
	MINP								
	DMIN	s, n, d	The DMIN instruction searches for minimum values in 32-bit data blocks. The number of data blocks to be searched through is specified by n. The smallest value found in s through s+(n-1) is stored in d.						
	DMINP								
Sorting 16-/32-bit data	SORT	s1, n, s2, d1, d2	The SORT instruction sorts 16-bit data specified by s1 in ascending or descending order. The number of data to be sorted is specified by n.		6		7.5.12		
	SORTP								
	DSORT		The DSORT instruction sorts 32-data specified by s1 in ascending or descending order. The number of data to be sorted is specified by n.						
	DSORTP								
Calculating totals of 16-/32-bit BIN data blocks	WSUM	s, n, d	The WSUM instruction calculates the total of 16-bit data blocks in the device specified by s. The result is stored in the device specified by d and d+1.		4		7.5.13		
	WSUMP								
	DWSUM	s, n, d	The DWSUM instruction calculates the total of 32-bit data blocks in the device specified by s and s+1. The result is stored in d through d+3.				4		7.5.14
	DWSUMP								

2.5.6 Structured program instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Repetition instructions	FOR	n	The FOR/NEXT loop repeats single program sequences without setting an input condition. The program sequence located between the FOR and the NEXT command is repeated for n times.		2	*	7.6.1
	NEXT				1	1	
	BREAK	p, d	The BREAK instruction terminates a FOR/NEXT loop execution and jumps to the pointer/label specified by p.		3		7.6.2
	BREAKP						
Subroutine program calls	CALL	p	The CALL instruction calls a subroutine program specified by a pointer Pxx in GX Developer or GX IEC Developer..		2	*	7.6.3
	CALLP						
	RET		The RET instruction marks the end of a subroutine program.		1	*	7.6.4
	FCALL	p	On resetting the execution condition for the FCALL instruction, the contacts and coils in the subroutine program specified in p (pointer/label) are treated as if the execution condition of the according instruction was not set.		2		7.6.5
	FCALLP						
Subroutine program calls between program files	ECALL	file name, p	The ECALL instruction calls a subroutine program specified by a pointer address (label) in a program file specified by a file name.		3		7.6.6
	ECALLP						
Subroutine program calls between program files	EFCALL	file name, p	On resetting the execution condition for the EFCALL instruction, the contacts and coils in the subroutine program specified in p (pointer/label) are treated as if the execution condition of the according instruction was not set.		3		7.6.7
	EFCALLP						
Main/subprogram switching	CHG		With the input condition set, the CHG instruction enables switching between MAIN and SUB programs.			1	7.6.8

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Microcomputer program call	SUB	n	If the input condition is set, the SUB instruction calls the microcomputer program located at the address "n".		3		7.6.9
	SUBP						
Index qualification of entire ladders	IX	s	The IX and IXEND instructions perform index qualification on those devices in the program part located between the IX and IXEND instructions.		2		7.6.10
	IXEND				1		
Designation of qualification values in index qualification of entire ladders	IXDEV	p, d	The IXDEV and IXSET instructions read the addresses of the devices in the offset designation area and write these offset numbers to an index table in the device designated by d.		1		7.6.11
	IXSET						

2.5.7 Data table operation instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Write data to a data table	FIFW	s, d			3	* 7	7.7.1
	FIFWP						
Read data entered first from data table	FIFR	s, d			3	* 7	7.7.2
	FIFRP						
Read data entered last from data table	FPOP	s, d			3		7.7.3
	FPOPP						
Delete specified data blocks from data table	FDEL	s, n, d			4		7.7.4
	FDELP						
Insert specified data blocks in data table	FINS						
	FINSP						

\*: The number of program steps depends on the devices used.  
 Refer to the reference chapter for the accurate number of steps.





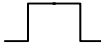


2.5.8 Buffer memory access instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Reading data from a special function module	FROM	n1, n2, n3, d	The FROM instruction reads 1-word data (16-bit) from the buffer memory of a special function module.		5	* 9	7.8.1
	FROMP						
	DFRO		The DFRO instruction reads 2-word data (32-bit) from the buffer memory of a special function module.				
	DFROP						
Writing data to a special function module	TO	s, n1, n2, n3	The TO instruction writes 1-word data (16-bit) from the memory of the CPU to the buffer memory of a special function module.		5	* 9	7.8.2
	TOP						
	DTO		The DTO instruction writes 2-word data (32-bit) from the memory of the CPU to the buffer memory of a special function module.				
	DTOP						
Reading data from a remote station	FROM, PRC	n1, n2, n3, d (FROM(P)/DFRO(P)) s, d PRC	The FROM/PRC instruction reads 1-word data (16-bit) from the buffer memory of a remote module.		7/9	* 7/9	7.8.3
	FROMP, PRC						
	DFRO, PRC		The DFRO/PRC instruction reads 2-word data (32-bit) from the buffer memory of a remote module.				
	DFROP, PRC						
Writing data to a remote station	TO, PRC	s, n1, n2, n3 (TO(P)/DTO(P)) s, d (PRC)	The TO/PRC instruction writes 1-word data (16-bit) from the memory of the CPU to the buffer memory of a remote module.		7/9	* 7/9	7.8.4
	TOP, PRC						
	DTO, PRC		The DTO/PRC instruction writes 2-word (32-bit) data from the memory of the CPU to the buffer memory of a remote module.				
	DTOP, PRC						

\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.








2.5.9 Display instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
ASCII character output	PR	s, d	SM701 (Q series/ System Q) set (1): Output of an ASCII character string of 16 characters to an output module. The character string, divided into twice 8 characters, is read from the address area s and output to the outputs specified by d. SM701 (Q series/ System Q) not set (0): Output of ASCII character string data up to the character code "00H" in hexadecimal format from the address area s to the outputs specified by d.		3	* 7	7.9.1
	PRC	s, d	The PRC instruction outputs a comment of a device (in ASCII code) to an output module. If SM701 (Q series/ System Q) is set (1), 16 characters are output; if SM701 is not set (0), 32 characters are output.		3	* 7	7.9.2
Display of ASCII character and comments	LED	s	The LED instruction reads ASCII data (16 characters) from a specified address area and displays it on a suitable CPU display.		2	* 3	7.9.3
	LEDC	s (Q)	The LEDC instruction reads comment data (16 characters) from a specified address area and displays it on a suitable CPU display.		2	* 3	7.9.4
	LEDA	n	These instructions display an ASCII character string in the LED display of a suitable CPU.			* 13	7.9.5
	LEDB						
Clear display	LEDR		Resetting annunciators and error displays		1	* 1	7.9.6

\*: The number of program steps depends on the devices used. Refer to the reference chapter for the accurate number of steps.

2.5.10 Debugging and failure diagnosis instructions











Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Failure check	CHKST		The CHKST instruction starts the execution of the CHK instruction. If the execution condition for the CHKST instruction is not set (0), the program step following the CHK instruction will be executed.		1		7.10.1
	CHK (Q)		The CHK instruction supports failure check operations for contact circuits. Once an error occurs within such a circuit, the device in d1 is set and the corresponding error code is stored in d2.				
	CHK (A)	d1, d2	The CHK instruction supports a failure check in a contact circuit with limit switches. Once an error occurs within such a circuit the device in d1 is set and the corresponding error code is stored in d2.		5		7.10.2
	CHKCIR		The CHKCIR instruction generates error check circuits for the CHK instruction and starts the program section with the generated error check circuits.		1		7.10.3
	CHKEND		End instructions for a program part with generated check circuits.				
Set / reset status latch	SLT		The SLT instruction executes the temporary storage of specified device data. The data are stored in the status latch memory and can be checked and displayed.		1	1	7.10.4
	SLTR		The SLTR instruction clears the data temporarily stored in the status latch area, and resets (re-enables) the SLT instruction.				
Set / reset sampling trace	STRA		Set sampling trace		1	1	7.10.5
	STRAR		Reset sampling trace				
Execute/ set/ reset program trace	PTRA		Set program trace		1		7.10.6
	PTRAR		Reset program trace				
	PTRAEXE		Execute program trace				
	PTRAEXEP						

2.5.11 Character string processing instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Conversion of 16-/32-bit binary data into decimal values in ASCII code	BINDA	s, d	The BINDA instruction converts a 16-bit binary value specified by s into a 5-digit decimal value in ASCII code and stores it in the device specified in d.		3		7.11.1
	BINDAP						
	DBINDA		The DBINDA instruction converts 32-bit binary data specified by s into a 10-digit decimal value in ASCII code and stores it in the device specified in d.				
	DBINDAP						
Conversion of 16-/32-bit binary data into hexadecimal values in ASCII code	BINHA	s, d	The BINHA instruction converts 16-bit binary data specified by s into a 4-digit hexadecimal value in ASCII code and stores it in the devices specified by d.		3		7.11.2
	BINHAP						
	DBINHA		The DBINHA instruction converts 32-bit binary data specified by s into a 8-digit hexadecimal value in ASCII code and stores it in the devices specified by d.				
	DBINHAP						
Conversion of 4-/8-digit BCD data into ASCII code	BCDDA	s, d	The BCDDA instruction converts 4-digit BCD data specified by s into the ASCII format and stores it in the devices specified by d.		3		7.11.3
	BCDDAP						
	DBCDDA		The DBCDDA instruction converts 8-digit BCD data specified by s into the ASCII format and stores it in the devices specified by d.				
	DBCDDAP						
Conversion of decimal ASCII data into BIN 16-/32-bit binary data	DABIN	s, d	The DABIN instruction converts the 5-digit decimal ASCII data specified by s into the BIN 16-bit format and stores it in the devices specified by d.		3		7.11.4
	DABINP						
	DDABIN		The DDABIN instruction converts the 10-digit decimal ASCII data specified by s into the BIN 32-bit format and stores it in the devices specified by d.				
	DDABINP						

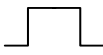

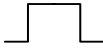


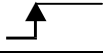
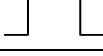



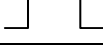

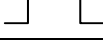

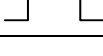
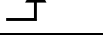

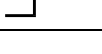
Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Conversion of hexadecimal ASCII data into 16-/32-bit binary data	HABIN	s, d	The HABIN instruction converts the 4-digit hexadecimal ASCII data in the device specified by s into the BIN 16-bit binary format and stores it in the devices specified by d.		3		7.11.5
	HABINP						
	DHABIN		The DHABIN instruction converts the 8-digit hexadecimal ASCII data specified in the area s into the BIN 32-bit format and stores it in the devices specified by d.				
	DHABINP						
Conversion of decimal ASCII data into 4-/8-digit BCD data	DABCD	s, d	The DABCD instruction converts the decimal ASCII data in s into the 4-digit BCD data format and stores it in the devices specified by d.		3		7.11.6
	DABCDP						
	DDABCD		The DDABCD instruction converts the decimal ASCII data specified by s into the 8-digit BCD format and stores it in the devices specified in d.				
	DDABCDP						
Read-out of comment data	COMRD	s, d	The COMRD instruction reads comment data from the device specified by s and stores it as ASCII code in the area d.		3		7.11.7
	COMRDP						
Detection of character string length	LEN	s, d	The length instruction detects the length of a character string specified in s and stores the result in the device specified by d.		3		7.11.8
	LENP						
Conversion of BIN 16-/32-bit binary data into character string data	STR	s1, s2, d	The STR instruction adds a decimal point to the BIN 16-bit binary value in the device specified by s2 to the digit specified by the devices s1, converts the data into a character string, and stores it in the area of the devices specified by d.		4		7.11.9
	STRP						
	DSTR		The DSTR instruction adds a decimal point to the BIN 32-bit binary value in the device specified by s2 to the digit specified by the devices s1, converts the data into a character string, and stores it in the area of the devices specified by d.				
	DSTRP						

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Conversion of character string data into BIN 16-/32-bit binary data	VAL	s, d1, d2	The VAL instruction converts the character strings stored in the area s into BIN 16-bit data. The number of digits and the binary value are stored in d1 and d2.		4		7.11.10
	VALP						
	DVAL		The DVAL instruction converts the character strings stored in s into BIN 32-bit data. The number of digits and the binary value are stored in d1 and d2.				
	DVALP						
Conversion of floating point data into character string data	ESTR	s1, s2, d	The ESTR instruction converts the floating point data (real numbers) in s1 into character string data. The data format of the character string is specified in s2. The result is stored in d.		4		7.11.11
	ESTRP						
Conversion of character string data into decimal floating point data	EVAL	s, d	The EVAL instruction converts the character string in s into a decimal floating point number (real number). The result is stored in d.		3		7.11.12
	EVALP						
Conversion of 16-bit data into ASCII code (Q)	ASC	s, n, d	The ASCII instruction converts the 16-bit binary data stored from s onwards into the hexadecimal ASCII format and stores the result considering the number of characters specified by n from d onwards.		4		7.11.13
	ASCP						
Conversion of alphanumerical character strings into ASCII code (A)	ASC	d	The ASC instruction converts alphanumerical character strings with up to 8 characters into the ASCII code. The result is stored from d onwards.			* 13	7.11.14
Conversion of hexadecimal ASCII values into binary values	HEX	s, n, d	The HEX instruction converts the hexadecimal ASCII characters from s onwards into binary values. The number of characters to be converted is specified by n. The result is stored from d onwards.		4		7.11.15
	HEXP						
Extraction of character string data (right part of character string)	RIGHT	s, n, d	The RIGHT instruction stores n characters from the right side of the character string (end of character string) in s. The characters are stored in d.		4		7.11.16
	RIGHTP						
Extraction of character string data (left part of character string)	LEFT	s, n, d	The LEFT instruction stores n characters from the left side of the character string (beginning of character string) in s. The characters are stored in d.		4		7.11.16
	LEFTP						

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Selecting and moving parts of character strings into a character string	MIDR	s1, s2, d	The MIDR instruction stores a specified part of the character string stored in s. The first character of the part to be stored is specified in s2.		4		7.11.17
	MIDRP						
	MIDW	s1, s2, d	The MIDW instruction stores a part of specified length of the character string stored in s1 in the area specified in d. The first address of the storage area in d is specified in s2.				
	MIDWP						
Search for character strings	INSTR	s1, s2, n, d	The INSTR instruction searches the character string specified in s1 within the character string data specified by s2. The search begins with the character specified in n.		5		7.11.18
	INSTRP						
Floating point data conversion with BCD representation	EMOD	s1, s2, d1	The EMOD instruction calculates the BCD format from the floating point number (real number) in s1 considering the decimal point shift to the right specified in s2. The result is stored in d1.		4		7.11.19
	EMODP						
BCD data conversion with decimal floating point format	EREXP	s1, s2, d1	The EREXP instruction calculates the decimal format of the floating point data (real number) from the floating point data in BCD format in s1, considering the decimal places specified in s2. The result is stored in d1.		3		7.11.20
	EREXPP						

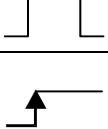
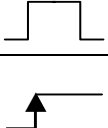
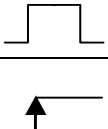
\*: The number of program steps depends on the devices used.  
Refer to the reference chapter for the accurate number of steps.

2.5.12 Special function instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Sine calculation	SIN	s, d	$\text{SIN}(s+1, s) \rightarrow (d+1, d)$		3		7.12.1
	SINP						
Cosine calculation	COS	s, d	$\text{COS}(s+1, s) \rightarrow (d+1, d)$		3		7.12.2
	COSP						
Tangent calculation	TAN	s, d	$\text{TAN}(s+1, s) \rightarrow (d+1, d)$		3		7.12.3
	TANP						
Arcus sine calculation	ASIN	s, d	$\text{SIN}^{-1}(s+1, s) \rightarrow (d+1, d)$		3		7.12.4
	ASINP						
Arcus cosine calculation	ACOS	s, d	$\text{COS}^{-1}(s+1, s) \rightarrow (d+1, d)$		3		7.12.5
	ACOSP						
Arcus tangent calculation	ATAN	s, d	$\text{TAN}^{-1}(s+1, s) \rightarrow (d+1, d)$		3		7.12.6
	ATANP						
Conversion from degrees into radian	RAD	s, d	$(s+1, s) \rightarrow (d+1, d)$ Conversion from degrees into radian		3		7.12.7
	RADP						
Conversion from radian into degree	DEG	s, d	$(s+1, s) \rightarrow (d+1, d)$ Conversion from radian into degree		3		7.12.8
	DEGP						
Square root calculation	SQR	s, d	$\sqrt{(s+1, s)} \rightarrow (d+1, d)$		3		7.12.9
	SQRP						

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Floating point value as exponent of e	EXP	s, d	$e^{(s+1, s)} \rightarrow (d+1, d)$		3		7.12.10
	EXPP						
Logarithm (natural) calculation	LOG	s, d	$\text{LOG } e^{(s+1, s)} \rightarrow (d+1, d)$		3		7.12.11
	LOGP						
Randomize value	RND	d	Stores the generated random value in d.		3		7.12.12
	RNDP						
Update random values	SRND	s	Updates the series of random values stored in s.		3		
	SRNDP						
Square root calculation from 4-digit BCD data	BSQR	s, d	$\sqrt{(s)} \rightarrow (d)+0$ Integer +1 Decimal place		3		7.12.13
	BSQRP						
Square root calculation from 8-digit BCD data	BDSQR	s, d	$\sqrt{(s+1, s)} \rightarrow (d)+0$ Integer +1 Decimal place		3		
	BDSQRP						
Sine calculation from BCD data	BSIN	s, d	$\sin(s) \rightarrow (d)+0$ Sign character +1 Integer +2 Decimal place		3		7.12.14
	BSINP						
Cosine calculation from BCD data	BCOS	s, d	$\cos(s) \rightarrow (d)+0$ Sign character +1 Integer +2 Decimal place		3		7.12.15
	BCOSP						
Tangent calculation from BCD data	BTAN	s, d	$\tan(s) \rightarrow (d)+0$ Sign character +1 Integer +2 Decimal place		3		7.12.16
	BTANP						

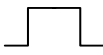

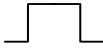


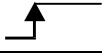


Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference			
					Q	A				
Arcus sine calculation from BCD data	BASIN	s, d	$\sin^{-1}(s) \rightarrow$ (d) +0 +1 +2 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Character sign</td></tr> <tr><td>Integer</td></tr> <tr><td>Decimal place</td></tr> </table>	Character sign	Integer	Decimal place		3		7.12.17
	Character sign									
Integer										
Decimal place										
BASINP										
Arcus cosine calculation from BCD data	BACOS	s, d	$\cos^{-1}(s) \rightarrow$ (d) +0 +1 +2 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Character sign</td></tr> <tr><td>Integer</td></tr> <tr><td>Decimal place</td></tr> </table>	Character sign	Integer	Decimal place		3		7.12.18
	Character sign									
Integer										
Decimal place										
BACOSP										
Arcus tangent calculation from BCD data	BATAN	s, d	$\tan^{-1}(s) \rightarrow$ (d) +0 +1 +2 <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Character sign</td></tr> <tr><td>Integer</td></tr> <tr><td>Decimal place</td></tr> </table>	Character sign	Integer	Decimal place		3		7.12.19
	Character sign									
Integer										
Decimal place										
BATANP										

2.5.13 Data control instructions

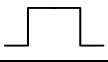

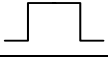
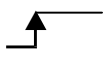
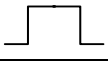

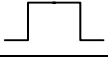

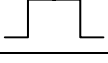
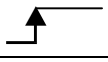

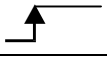
Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Upper and lower limit controls for BIN 16-/32-bit data	LIMIT	s1, s2, s3, d	If (s3)<(s1) the data value in s1 is stored in d.		5		7.13.1
	LIMITP		If (s1)≤(s3)≤(s2) the data value in s3 is stored in d.  If (s2)<(s3) the data value in s2 is stored in d.				
	DLIMIT	s1, s2, s3, d	If ((s3)+1, s3)<((s1)+1, s1) the data value in ((s1)+1, s1) is stored in (d+1, d).				
	DLIMITP		If ((s1)+1, s1)≤((s3)+1, s3) <((s2)+1, s2) the data value in ((s3)+1, s3) is stored in (d+1, d).  If ((s2)+1, s2)<((s3)+1, s3)<((s2)+1, s2) the data value in ((s2)+1, s2) is stored in (d+1, d).				
Dead band controls for BIN 16-/32-bit data	BAND	s1, s2, s3, d	If (s1)≤(s3)≤(s2) 0 → (d)		5		7.13.2
	BANDP		If (s3)<(s1) (s3)→(s1)→(d)  If (s2)<(s3) (s3)→(s2)→(d)				
	DBAND	s1, s2, s3, d	If ((s1)+1, s1)≤((s3)+1, s3) ≤((s2)+1, s2) 0 → (d+1, d)				
	DBANDP		If ((s3)+1, s3)<(s1+1, s1) ((s3)+1, s3)-((s1)+1, s1) →(d+1, d)  If ((s2)+1, s2)<((s3)+1, s3) ((s3)+1, s3)-((s2)+1, s2) →(d+1, d)				
Zone control for BIN 16-/32-bit data	ZONE	s1, s2, s3, d	If s3=0: 0 → (d)  If s3>0: s3 + s2 → (d)		5		7.13.3
	ZONEP		If s3<0: s3 → s1 → (d)				
	DZONE	s1, s2, s3, d	If ((s3)+1, s3)=0 0 → (d+1, d)  If ((s3)+1, s3)>0 ((s3)+1, s3)+((s2)+1, s2) →(d+1, d)				
	DZONEP		If ((s3)+1, s3)<0 ((s3)+1, s3)+((s1)+1, s1) →(d+1, d)				

2.5.14 File register switching instructions



Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Switch instruction for file register blocks	RSET	s	The RSET instruction switches from a file register block being in use by a program to a file register block with the number specified by s.		2		7.14.1
	RSETP						
Switch instruction for file register files	QDRSET	s	The QDRSET instruction switches from a file register file being in use by a program to a file register file specified by s.		* 2 + n		7.14.2
	QDRSETP						
Switch instruction for comment files	QCDSET	s	The QCDSET instruction switches from a comment file being in use by a program to a comment file specified by s.		* 2 + n		7.14.3
	QCDSETP						

\*:  $n = (\text{number of program name characters})/2 = \text{Number of additional steps}$  (Decimal fractions are rounded up)


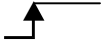




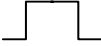
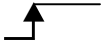
2.5.15 Clock instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference																		
					Q	A																			
Reading clock data	DATERD	d	QnA CPU clock → <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>d+0</td><td>Year</td></tr> <tr><td>d+1</td><td>Month</td></tr> <tr><td>d+2</td><td>Day</td></tr> <tr><td>d+3</td><td>Hour</td></tr> <tr><td>d+4</td><td>Minute</td></tr> <tr><td>d+5</td><td>Second</td></tr> <tr><td>d+6</td><td>Weekday</td></tr> </table>	d+0	Year	d+1	Month	d+2	Day	d+3	Hour	d+4	Minute	d+5	Second	d+6	Weekday	 	2		7.15.1				
	d+0			Year																					
d+1	Month																								
d+2	Day																								
d+3	Hour																								
d+4	Minute																								
d+5	Second																								
d+6	Weekday																								
DATERDP																									
Writing clock data	DATEWR	s	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>s+0</td><td>Year</td></tr> <tr><td>s+1</td><td>Month</td></tr> <tr><td>s+2</td><td>Day</td></tr> <tr><td>s+3</td><td>Hour</td></tr> <tr><td>s+4</td><td>Minute</td></tr> <tr><td>s+5</td><td>Second</td></tr> <tr><td>s+6</td><td>Weekday</td></tr> </table> → QnA CPU clock	s+0	Year	s+1	Month	s+2	Day	s+3	Hour	s+4	Minute	s+5	Second	s+6	Weekday	 	2		7.15.2				
	s+0			Year																					
s+1	Month																								
s+2	Day																								
s+3	Hour																								
s+4	Minute																								
s+5	Second																								
s+6	Weekday																								
DATEWRP																									
Adding clock data	DATE+	s1, s2, d	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>s1</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table> + <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>s2</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table> → <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>d</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table>	s1	Hour		Minute		Second	s2	Hour		Minute		Second	d	Hour		Minute		Second	 	4		7.15.3
	s1			Hour																					
	Minute																								
	Second																								
s2	Hour																								
	Minute																								
	Second																								
d	Hour																								
	Minute																								
	Second																								
DATE+P																									
Subtracting clock data	DATE-	s1, s2, d	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>s1</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table> - <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>s2</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table> → <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>d</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table>	s1	Hour		Minute		Second	s2	Hour		Minute		Second	d	Hour		Minute		Second	 	4		7.15.4
	s1			Hour																					
	Minute																								
	Second																								
s2	Hour																								
	Minute																								
	Second																								
d	Hour																								
	Minute																								
	Second																								
DATE-P																									
Changing clock data format from hh:mm:ss to seconds	SECOND	s, d	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>s</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table> → <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>d</td><td>Second</td></tr> </table>	s	Hour		Minute		Second	d	Second	 	3		7.15.5										
	s			Hour																					
	Minute																								
	Second																								
d	Second																								
SECONDP																									
Changing clock data format from seconds to hh:mm:ss	HOUR	s, d	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>s</td><td>Second</td></tr> </table> → <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>d</td><td>Hour</td></tr> <tr><td></td><td>Minute</td></tr> <tr><td></td><td>Second</td></tr> </table>	s	Second	d	Hour		Minute		Second	 													
	s			Second																					
d	Hour																								
	Minute																								
	Second																								
HOURP																									

2.5.16 Peripheral device instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Output of messages to peripheral devices	MSG	s	The MSG instruction outputs a character string stored in a device specified from s to a peripheral device specified in terminal mode.		2		7.16.1
Key input of data from peripheral devices	PKEY	d	The key input data (characters) are read from the peripheral device specified in terminal mode and written in ASCII format to the devices specified in d.		2		7.16.2





2.5.17 Program instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Switching programs into stand-by mode	PSTOP	s	The PSTOP instruction sets the program specified by the device in s into the stand-by mode.		2		7.17.1
	PSTOPP						
Switching programs into stand-by mode and reset of outputs	POFF	s	The POFF instruction sets the program specified by the device in s into the stand-by mode and resets the outputs addressed by the program.		2		7.17.2
	POFFP						
Switching programs into scan execution mode	PSCAN	s	The PSCAN instruction sets the program specified by the device in s into the scan execution mode. In this mode the program is only executed once during one program scan.		3		7.17.3
	PSCANP						
Switching programs into low-speed execution mode	PLow	s	The PLOW instruction sets the program specified by the device in s into the low-speed execution mode.		3		7.17.4
	PLowP						

\*: n = (number of program name characters)/2 = Number of additional steps (Decimal fractions are rounded up)


2.5.18 Other instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Reset watchdog timer	WDT		The WDT instruction resets the watchdog timer (WDT) during execution of a sequence program.		1	1	7.18.1
	WDTP						
Set and reset carry flag	STC		The carry flag stores the carry (0 or 1) of rotation and shift operations.			1	7.18.2
	CLC	The carry flag is reset after the execution of the CLC instruction.					
Preset number of execution scans	DUTY	n1, n2, d	<p>SM420 to SM424, SM 430 to SM434</p>		4	7	7.18.3
Direct read of one byte	ZRRDB	n, d			3		7.18.4
	ZRRDBP						
Direct write of one byte	ZRWRB	n, s			3		7.18.5
	ZRWRBP						
Storing of an indirect address	ADRSET	s, d	Stores the indirect address of the device designated by s at d and d+1. This address is used when an indirect device read is performed.		3		7.18.6
	ADRSETP						
Numerical key input from keyboard	KEY	s, n, d1, d2	The KEY instruction supports the key input of 8 ASCII characters at the inputs specified by s (X). The values entered at the inputs are encoded in hexadecimal format and stored in the devices specified by d1.		5		7.18.7
Batch save of index register contents	ZPUSH	d	The ZPUSH instruction saves the contents of the index registers Z0 through Z15 in d.		3		7.18.8
	ZPUSHP						

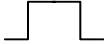
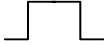
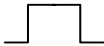
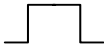
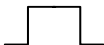



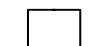
Batch recovery of index register contents	ZPOP	d	The ZPOP instruction recovers the contents of the index registers Z0 through Z15 in d.		3		7.18.9
	ZPOPP						
Batch write of data to EEPROM register	EROMWR	s, n, d1,d2	The EROMWR instruction writes the number specified by n of data words stored in the device specified by s to an EEPROM file register specified by d1.		6		7.18.10
	EROMWRP						

## 2.6 Data link instructions

### 2.6.1 Network refresh instructions



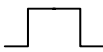

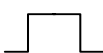

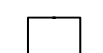

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Network refresh instructions	ZCOM	Jn	Data refresh in network modules		6		8.5.1
		Un					

### 2.6.2 Dedicated data link instructions

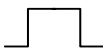

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Dedicated data link instructions	READ	Jn, s1, s2, d1, d2	Reading word device data from another station		11		8.6.1
		Un, s1, s2, d1, d2					
	SREAD	Jn, s1, s2, d1, d2, d3	Reading word device data from another station		13		8.6.2
		Un, s1, s2, d1, d2, d3					
	WRITE	Jn, s1, s2, d1, d2	Writing word device data to another station		12		8.6.3
		Un, s1, s2, d1, d2					
	SWRITE	Jn, s1, s2, d1, d2, d3	Writing word device data to another station		13		8.6.4
		Un, s1, s2, d1, d2, d3					
	SEND	Jn, s1, s2, d	Sending data to other stations		10		8.6.5
		Un, s1, s2, d					
	RECV	Jn, s, d1, d2	Receiving sent data from other stations or receives the data sent via the SEND instruction		9		8.6.6
		Un, s, d1, d2					
	REQ	Jn, s1, s2, d1, d2	Request data from other stations		10		8.6.7
		Un, s1, s2, d1, d2					
ZNFR	Jn, s1, s2, d	Reading data from special function modules in remote I/O stations		9		8.6.8	
	Un, s1, s2, d						
ZNT0	Jn, s1, s2, d	Writing data to special function modules in remote I/O stations		9		8.6.9	
	Un, s1, s2, d						



2.6.3 A series compatible data link instructions



Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
A series compatible data link instructions	J.ZNRD	Jn, n1, s, n2, d1, d2	Read QnA data from object stations in object networks		12		8.7.1
	JP.ZNRD		Read data from local stations				
	J.ZNWR	Jn, n1, s, n2, d1, d2	Write QnA data to object stations in object networks		12		8.7.2
	JP.ZNWR		Write data to local stations				
	LRDP	s, n1, n2, d	A series only: Read data from local stations			11	8.7.3
	LWTP	Jn, s, d1, d2	A series only: Write data to local stations			11	8.7.4
	RFRP	n1, n2, n3, d	Reading data from a special function module in a remote station		9	11	8.7.5
	G.RFRP	Un, n1, n2, d1, d2					
	RTOP	s, n1, n2, n3	Writing data to a special function module in a remote station		9	11	8.7.6
G.RTOP	Un, n1, s, n2, d1						

2.6.4 Read/Write routing information



Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Read/Write routing information	Z.RTREAD	n, d	The RTREAD instruction reads the routing information from the destination network specified by n. The routing information is stored in routing parameters. The read routing information is stored from d onwards.		7		8.8.1
	ZP.RTREAD						
	Z.RTWRITE	s, n	The RTWRITE instruction writes the routing information to the destination network specified by n. The read routing information is stored from s on.		8		8.8.2
	ZP.RTWRITE						

## 2.7 Instructions for System Q CPUs

### 2.7.1 Reading of module informations

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Reading module information	UNIRD	n1, d, n2	Reads the module information stored in the area starting from the I/O No. designated by n1 and stores it in the area starting from the device designated by d. The number of points is designated by n2.		4		9.1.1
	UNIRDP						

### 2.7.2 Debugging and failure diagnosis instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Trace set	TRACE		Stores trace data set at a peripheral device to trace file in IC memory card by the designated number when SM800, SM801, and SM802 turns ON.		1		9.2.1
Trace reset	TRACER		Resets the data set by the TRACE instruction		1		9.2.1

**2.7.3 Writing to and reading from a file**

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Writing data to a designated file	SP.FWRITE	u0, s0, d0, s1, s2, d1	Writes data to a designated file		11		9.3.1
Reading data from a designated file	SP.FREAD	u0, s0, d0, s1, d1, d2	Reads data from a designated file		11		9.3.2

**2.7.4 Program instructions**

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Loading program from memory	PLOADP	s, d	Transfers the program stored in a memory card or standard memory card (other than drive 0) to drive 0 and places the program in standby status.		3		9.4.1
Unloading program from program memory	PUNLOADP	s, d	Deletes the standby program stored in standard memory (drive 0)		3		9.4.2
Load and unload	PSWAPP	s1, s2, d	Deletes standby program stored in standard memory (drive 0) designated by s1. Then the program (s2) stored in a memory card or standard memory (other than drive 0) is transferred to drive 0 and placed in standby status.		4		9.4.3

The instructions are only available within the GX Developer. The GX IEC Developer does not support the file system.

2.7.5 Data transfer instructions

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Highspeed block transfer of file register	RBMOV	s, d, n			4		9.5.1
	RBMOVP	s, d, n					

2.7.6 Instructions for data exchange in a multi-CPU system

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Write to CPU shared memory	S.TO	s1, s2, s3, s4, d	Writes data from device memory to the shared memory of the same CPU (which is executing the S.TO instruction).		5		9.6.1
	S.TOP				5		
Read from the shared memory of another CPU	FROM	n1, n2, n3, d	Reads data from the shared memory of another CPU and stores the data in the device memory of the CPU performing the FROM instruction.		5		9.6.2
	FROMP				5		
Automatic refresh of CPU shared memory	COM	—	Performs the automatic refresh of the intelligent function module, general data processing and the multi-CPU shared memory.		1		6.7.3

## 2.8 Dedicated instructions for Q4ARCPU

### 2.8.1 Instructions for mode setting

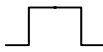
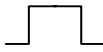

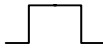
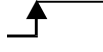


Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Setting of the start up mode	SGMODE	s1, s2	Selection between initial start mode and hot start mode				10.1.1
Setting of the operation mode when the CPU is changed	CGMODE	s	Selection of the action during switching from the control system to the standby system				10.1.2

### 2.8.2 Data transfer instructions





Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Data tracking instruction	TRUCK	s	Transfer of device memory from the CPU of the control system to the CPU of the standby system				10.2.1
Buffer memory batch refresh instruction	SPREF	s	Batch transfer of data in and out of the buffer memory of special function modules				10.2.2

## 2.9 Instructions for special function modules







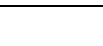

### 2.9.1 Instructions for serial communication modules

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Reading of data from a serial communication module	BUFRCVS	„Un“, n1, d1	Reading of received data from a serial communication module QJ71C24 to the PLC CPU in an interrupt program.				11.1.1
Reading of user registered frames	GETE	Un, s1, s2, d	User registered frames are read from a serial communication module				11.1.2
	GETEP						
Registration or deletion of user frames	PUTE	Un, s1, s2, d	User frames are registered to or deleted from a serial communication module				11.1.3
	PUTEP						
Transmission of data	PRR	Un, s, d	Sending of data via the serial communication module using user frames				11.1.4
	PRRP						

2.9.2 Instructions for PROFIBUS/DP interface modules

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Reading of data	BBLKRD	„Un“, n1, n2, d	Data is read from the buffer memory of a PROFIBUS/DP interface module and stored in the PLC CPU				11.2.1
	BBLKRD						
Writing of data	BBLKWR	„Un“, n1, n2, s	Data stored in the PLC CPU is written to the buffer memory of a PROFIBUS/DP interface module				11.2.2
	BBLKWR						

2.9.3 Instructions for ETHERNET interface modules

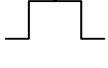
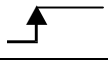
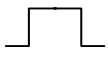
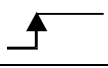
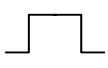
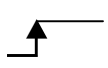
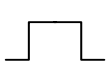
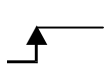
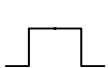

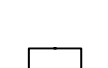

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Reading from fixed buffer	BUFRCV	„Un“, s1, s2, d1, d2	Data received during fixed buffer communication is read from the ETHERNET interface module				11.3.1
	BUFRCVS	„Un“, s1, d1					11.3.2
Writing to fixed buffer	BUFSND	„Un“, s1, s2, s3, d1	Data stored in the PLC CPU is moved to a fixed buffer of an ETHERNET interface module				11.3.3
Open connection	OPEN	„Un“, s1, s2, d1	Open processing for a connection				11.3.4
Close connection	CLOSE	„Un“, s1, s2, d1	Close processing for a connection				11.3.5
Error clear	ERRCLR	„Un“, s1, d1	Error codes stored in the buffer memory of the ETHERNET interface module are cleared and the "ERR." LED is switched off.				11.3.6
Reading of an error code	ERRRD	„Un“, s1, d1	Error codes stored in the buffer memory of the ETHERNET interface module are read to the PLC CPU				11.3.7
Re-initialization	UINI	„Un“, s1, d1	Re-initial processing of an ETHERNET interface module				11.3.8

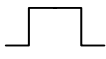

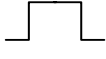

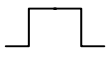

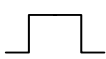

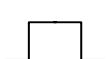

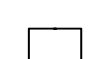

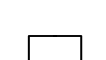
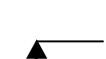
2.9.4 Instruction for MELSECNET/10

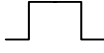

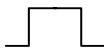

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Pairing setting	PAIRSET	Jn, s1	Setting of stations for duplex network				11.4.1



2.9.5 Instructions for CC-Link

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Parameter setting (A series)	RLPA	n, d1, d2	Transfer of the parameter settings to the master station of CC-Link		—	23	11.5.1
	RLPA_P						
Parameter setting (System Q)	RLPASET	Un, s1 to s5, d1			—	—	11.5.2
	RLPASET_P						
Setting of automatic refresh parameter (A series)	RRPA	n, d	Setting of the devices on which automatic refresh will be performed between the master/local module and the PLC CPU		—	20	11.5.3
	RRPA_P						
Reading from the buffer memory or from the device memory of a CPU (A series)	RIRD	n1, n2, d1, d2	Data is read from the buffer memory of another stations CC-Link module or from the device memory of that stations PLC CPU		—	26	11.5.4
	RIRD_P						
Reading from the buffer memory or from the device memory of a CPU (QnA series and System Q)	RIRD	Un, s, d1, d2			8	—	11.5.5
	RIRD_P						
Writing to the buffer memory or to the device memory of a CPU (A series)	RIWT	n1, n2, d1, d2	Data is written to the buffer memory of another stations CC-Link module or to the device memory of that stations PLC CPU		—	26	11.5.6
	RIWT_P						

Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference
					Q	A	
Writing to the buffer memory or to the device memory of a CPU (QnA series, System Q)	RIWT	Un, s, d1, d2	Data is written to the buffer memory of another stations CC-Link module or to the device memory of that stations PLC CPU		8	—	11.5.7
	RIWT_P						
Reading from an intelligent device station (A series)	RICV	n1, n2, d1, d2, d3	Data is read with handshake from the buffer memory of an intelligent device station connected to CC-Link		—	29	11.5.8
	RICV_P						
Reading from an intelligent device station (QnA series, System Q)	RICV	Un, s1, s2, d1, d2			10	—	11.5.9
	RICV_P						
Writing to an intelligent device station (A-Serie)	RISEND	n1, n2, d1, d2, d3	Data is written with handshake to the buffer memory of an intelligent device station connected to CC-Link		—	29	11.5.10
	RISEND_P						
Writing to an intelligent device station (QnA-Serie, System Q)	RISEND	Un, s1, s2, d1, d2			10	—	11.5.11
	RISEND_P						
Writing to automatic updating buffer memory (A series)	RITO	n1, n2, n3, n4, d1	Data is moved from the device memory of the PLC CPU to the automatic updating buffer memory of the master station. This data is then transferred to another station connected to CC-Link.		—	29	11.5.12
	RITO_P						
Writing to automatic updating buffer memory (QnA series, System Q)	RITO	Un, n1, n2, n3, d			9	—	11.5.13
	RITO_P						

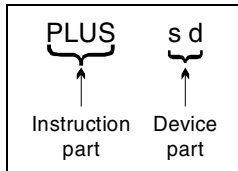
Category	Instruction	Variables	Meaning	Execution Condition	Number of steps		Reference	
					Q	A		
Reading from automatic updating buffer memory (A series)	RIFR	n1, n2, n3, n4, d1	Data transmitted from another station to the automatic updating buffer memory of the master station is moved to the device memory of the PLC CPU.		—	29	11.5.14	
	RIFR_P							
Reading from automatic updating buffer memory (QnA series, System Q)	RIFR	Un, n1, n2, n3, d			9	—		11.5.15
	RIFR_P							



## 3 Configuration of Instructions

### 3.1 The structure of an instruction

Most of the instructions consist of an instruction part and a device part. Other instructions do not require a device part and thus only consist of the instruction part.



#### Instruction part

The instruction part describes the functions of the instruction.

PLUS  $\hat{=}$  Addition

#### Device part

The device part describes the constants or variables to be specified. The device part can comprise three items: the source of data (s), the destination of data (d), and the number (n).

#### 3.1.1 Source of data (s)

- The data source designates the devices to be processed by the instruction.  
For 16-bit instructions the notation of the data source is s.  
For 32-bit instructions its notation is s+1 and s.
- Within the data source constants or variables can be specified.

#### Constants

Constants specify a constant numerical value to be processed by the instruction. This value is constantly set by the user written program and cannot be altered during program execution. It is recommended to index qualify each variable to be used as constant.

#### Variables

Variables specify a device storing data to be processed by the instruction (also refer to chapter 3.4).

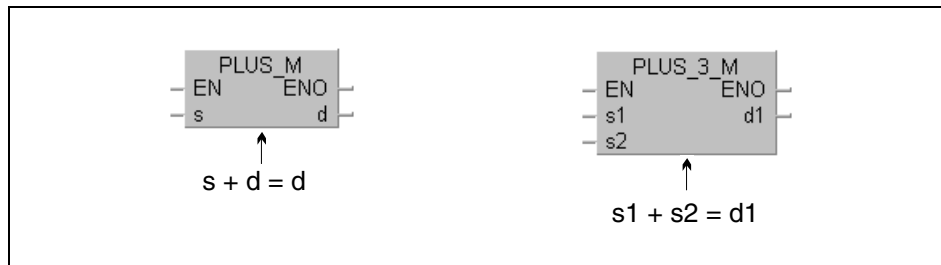
Before an instruction is executed, the data must be stored in the device. The data stored in variables can be altered during program execution.

3.1.2 Destination of data (d)

- The data destination designates the devices to store the data after being processed by the instruction.  
 For 16-bit instructions the notation of the data destination is d.  
 For 32-bit instructions its notation is d+1 and d.  
 However, some instructions with 2 devices require a value to be processed stored in the data destination d before the instruction is executed. In this case, the result of the operation will be stored in the same device as well.

Example:

The addition instruction for BIN 16-bit data. Here, d first stores data for the operation and then the operation result:



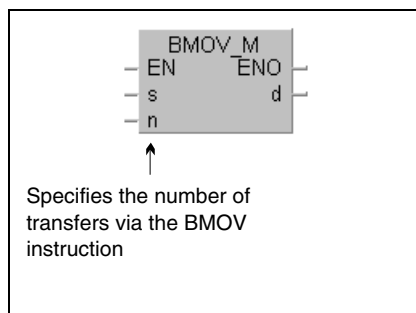
- A device for the storage of data has always to be set as data destination.

3.1.3 Number (n)

- The number n specifies how many devices are to be used or how often an instruction is to be executed.

Example:

The BMOV instruction for block data transfer:



- The value n may range from 0 to 32767. If n is specified 0, the instruction will not be executed.

### 3.2 Notation of instructions

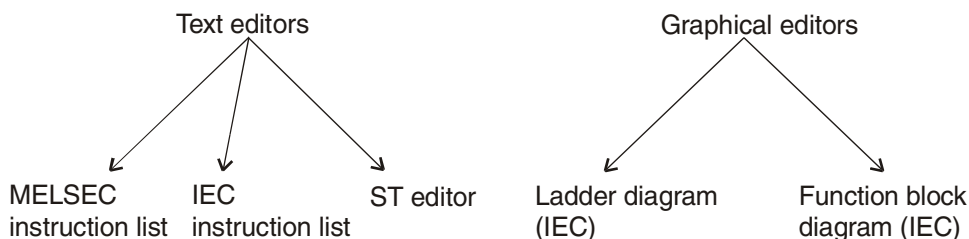
From the notation certain characteristics of the instructions can be derived.

#### 3.2.1 16/ 32-bit and pulse

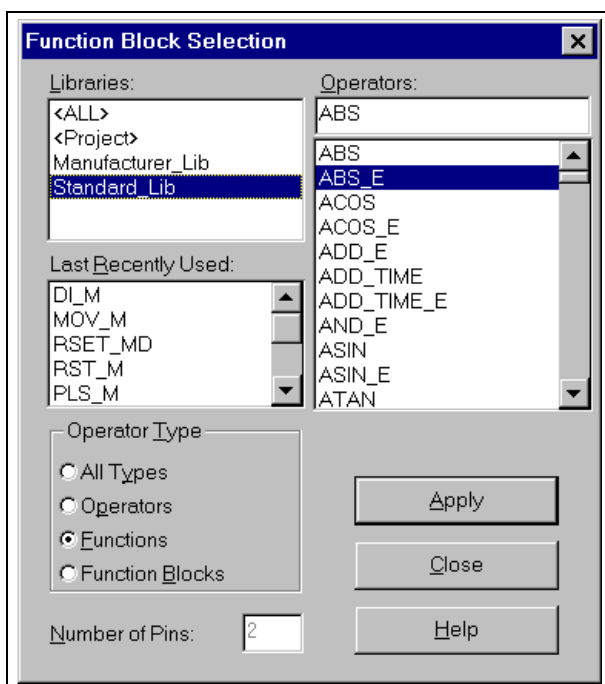
- SORT 16 bit processing
- SORTP 16 bit processing with pulse
- DSORT 32 bit processing
- DSORTP 32 bit processing with pulse

#### 3.2.2 MELSEC and IEC

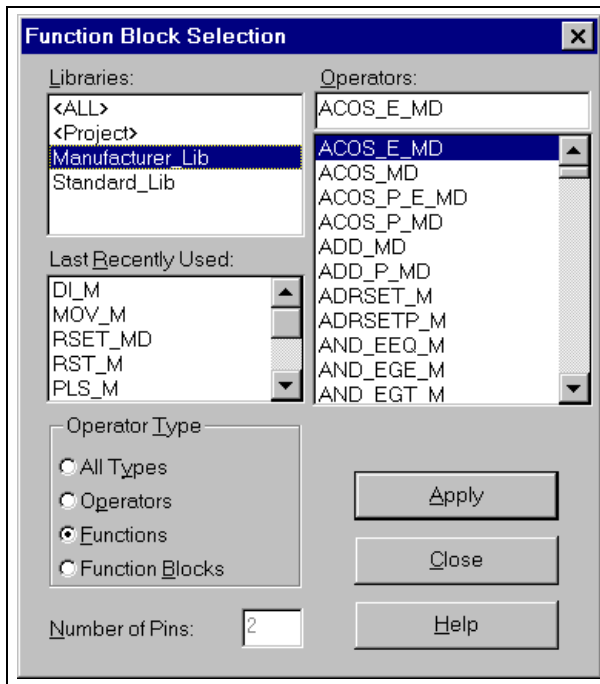
The GX IEC Developer includes several editors for the instructions:



Within these editors the instructions are represented in different notations.



For the selection of an instruction in the GX IEC Developer this dialog box will appear. Depending on the selected library different instructions can be chosen:  
 ALL: MELSEC and IEC instructions  
 Project: Functions and Function Blocks created by the user  
 Manufacturer: MELSEC instructions  
 Standard: IEC instructions



For example, this dialog box will appear when the the manufacturer library is selected. This listing contains the "adapted" MELSEC instructions.

The functions of the "pure" and "adapted" instructions are identical. Only their notation differs.

**Legend of the extensions within the IEC editor:**

Extension in IEC Editor	Meaning
_M	MELSEC instruction
_P_M	Pulse execution of an instruction
_MD	Dedicated MELSEC instruction (also refer to chapter 3.3)
_P_MD	Pulse execution of a dedicated instruction
_K_MD	Use of a constant in a dedicated instruction
_K_P_MD	Use of a constant and pulse execution in a dedicated instruction.
_S_MD	Dedicated MELSEC instruction for System Q CPUs
_P_S_MD	Pulse execution of a dedicated MELSEC instruction for System Q CPUs



### 3.2.3 Further characteristics of the instruction notation

The table below contains the symbols that represent several functions within the MELSEC editor. The column on the right shows the according instruction names within the IEC editor.

Example:

MELSEC editor IEC editor

LD\$>LD\_STRING\_GT\_M

MELSEC Editor	IEC Editor
\$	STRING
=	EQ
<>	NE
<=	LE
<	LT
>=	GE
>	GT
+	PLUS
-	MINUS
x	MULTI
/	DIVID

### 3.2.4 Specification of the notation

The chapters 5 through 8 that give a detailed description of the instructions contain illustrations of both editors, i.e. both notations. The header line contains the "pure" MELSEC instruction as it occurs in the MELSEC instruction list.

#### NOTE

*The tabular overview at the beginning of each instruction category always represents both notations.*

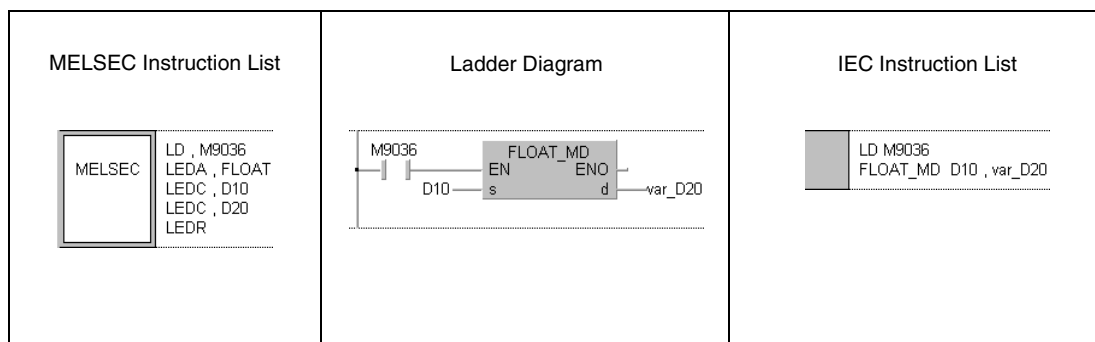
### 3.3 Programming of dedicated instructions

The dedicated instructions are customised instructions that do not only differ in notation from the pure MELSEC instructions. They also require a particular programming technique for the different CPUs.

In order to obtain the functions of the FLOAT\_MD instruction as well in the MELSEC editor of an A series CPU a certain procedure is required. In the MELSEC editor the FLOAT\_MD instruction has to be programmed in combination with the LEDA, LEDC, LEDR instructions. In the IEC editors the dedicated instructions can be programmed as usual.

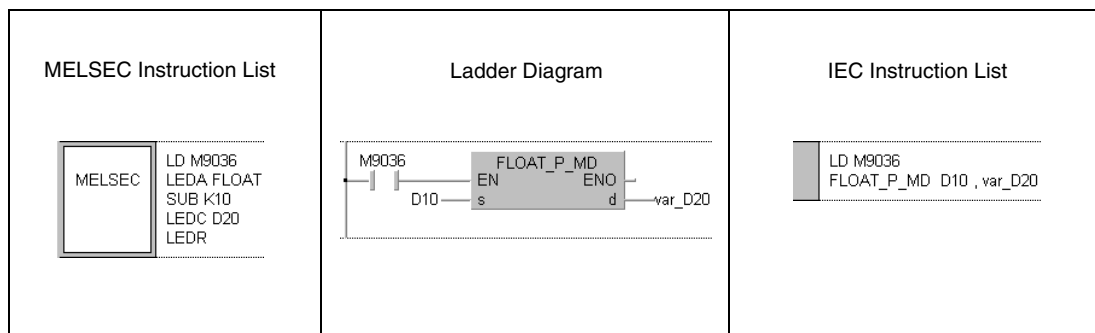
Example:

Programming of the FLOAT\_MD instruction (common execution 16-bit)



Example:

Programming of the FLOAT\_P\_MD instruction  
(pulse execution 16-bit, use of a constant in device s)



Refer to the following manuals for further information on the programming of dedicated instructions:

- GX IEC Developer Reference Manual
- Programming Manual (Dedicated Instructions)

### 3.4 Programming of variables

#### 3.4.1 Programming with the GX IEC Developer

The majority of instructions besides the instruction part also require a device part with specified variables. These variables contain the values for the execution of the instruction.

According to the selected editor in the GX IEC Developer a different method of programming of the variables is required.

**In the MELSEC editor:**

The data registers D100 and D10 can be assigned directly to the variable designation D100 and D10.

The connected PLC automatically detects that the following devices are designated:

D100=D100 and D101  
 D10=D10, D11, D12, D13

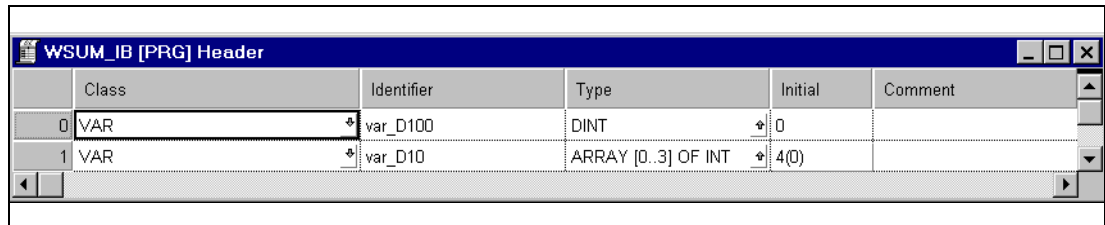
**In the IEC editor:**

In the IEC editor direct devices can only be entered, if actually only this device is to be designated.

Example: AND D10

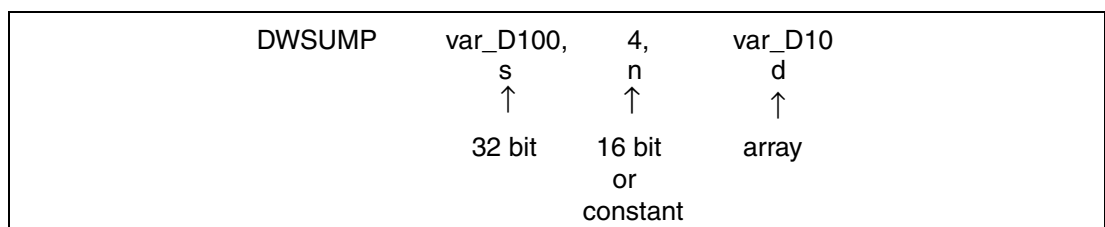
Before a DWSUMP\_M instruction can be processed, the variables have to be defined in the header of the program organisation unit (POU).

Example: Header of the IEC AWL



var\_D100 and var\_D10 are entered here as identifiers. The PLC actually does not assign the devices D100 and D10 but internally allocates free register areas for the variables.

Example: DWSUMP



The variable var\_D100 is of type DINT (32-bit). The variable var\_D10 is of type ARRAY. The array contains four 16-bit registers of type INT (also refer to chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer").

**Specification of the notation**

The designation var\_D100 or var\_D10 in the screenshots indicate that not direct devices are designated but identifiers. In these cases the variable definition is compulsory! If an instruction can only be programmed over a variable definition this is explicitly noted.

**NOTE**

As identifier any name can be entered (e.g. Motor 1, Indicator). The names var\_D100 or var\_D10 were selected here for a clear comparison to the programming in the MELSEC editor.

The table of variables at the beginning of any instruction gives an overview of the data types of the devices for each instruction (the example shows the DWSUM instruction 7.5.14).

**Variables**

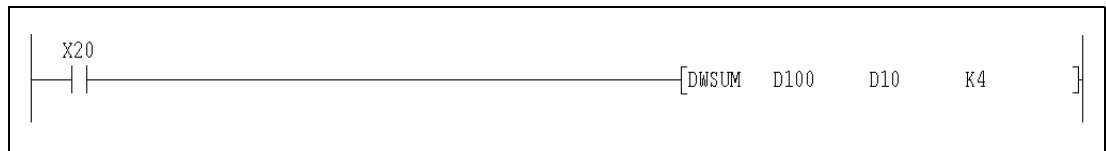
Set Data	Meaning	Data Type	
		MELSEC	IEC
s	First number of device storing data to be added.	BIN 32-bit	ANY32
d	First number of device storing result.	BIN 64-bit	Array [1..4] of ANY16
n	Number of data blocks to be added.	BIN 16-bit	ANY16

**3.4.2 Programming with the GX Developer**

The data registers D100 and D10 can be assigned directly to the variable designation D100 and D10.

The connected PLC automatically detects that the following devices are designated:

- D100=D100 and D101
- D10 = D10, D11, D12, D13



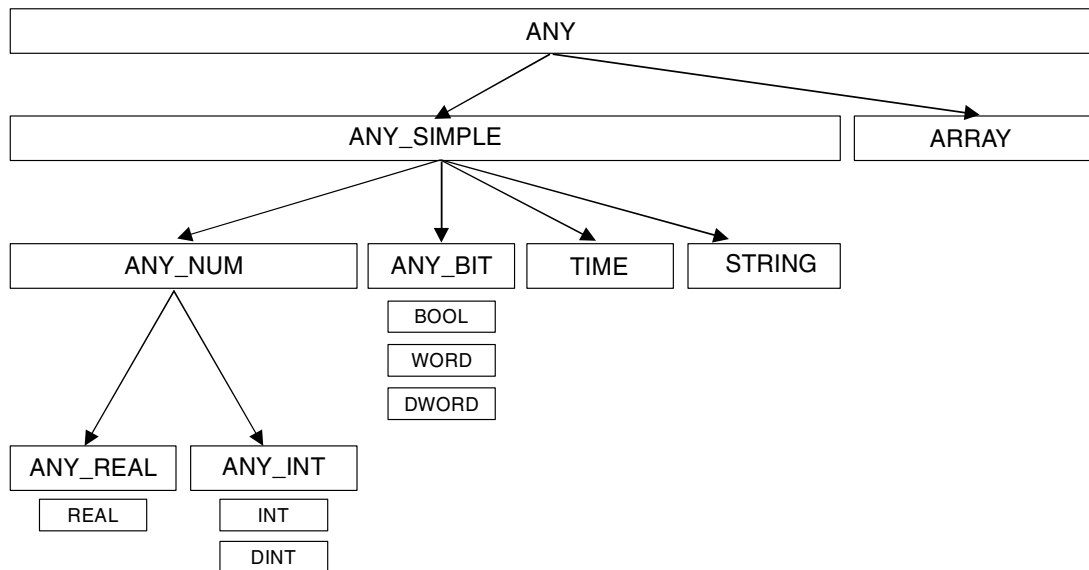
### 3.5 Data types

The data type determines the number and processing of bits as well as the value range of the variables.

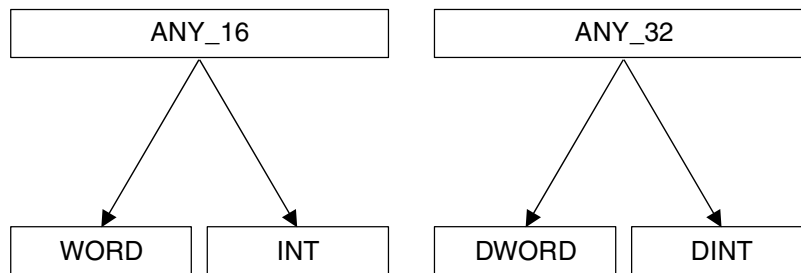
The following data types exist:

Data Type		Value Range	Number of bits	Applicable type of CPU
BOOL	Boolean	0 (FALSE), 1 (TRUE)	1 bit	A series Q series System Q
INT	INTEGER	-32.768 through 32.767	16 bits	
DINT	Double INTEGER	-2.147.483.648 through 2.147.483.647	32 bits	
WORD	Bit string 16	0 through 65.535	16 bits	
DWORD	Bit string 32	0 through 4.294.967.295	32 bits	
REAL	Floating point number	3.4 +/- 38 (7 digits)	32 bits	
TIME	Time value	T#-24d-0h31m23s648.00ms through T#24d20h31m23s647.00ms	32 bits	Q series System Q
STRING	Character string	max. 50 characters		

#### Hierarchy of data types ANY



## Hierarchy of data types ANY16 and ANY32



Data type	Meaning
ANY	Any data type
ANY_SIMPLE	Simple data type
ANY_NUM	Numeric data type
ANY_REAL	Floating point number
ANY_INT	Integer data type
ANY_BIT	Bit processing data type
ANY_16	Any 16-bit data type
ANY_32	Any 32-bit data type
TIME	Time
STRING	Character string
REAL	Floating point number
INT	Integer value
DINT	Double integer value
BOOL	Boolean value
WORD	Word (16 bits)
DWORD	Double word (32 bits)
ARRAY	Array

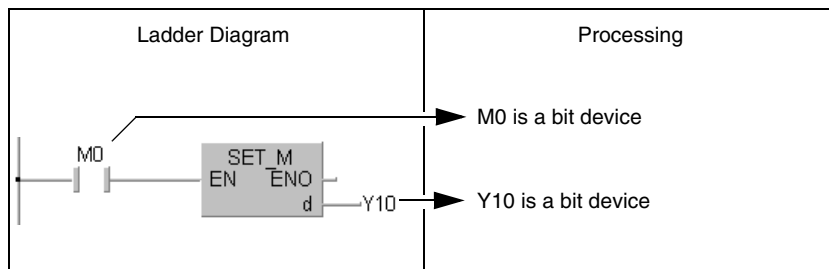
### 3.5.1 Processing of data

#### Processing of bit data

A bit device (X, Y, M, K, S, B or F) can obtain two states (ON=1 or OFF=0). Its status therefore can be represented by one bit (1 or 0). Bit processing is always performed, if a specified bit device is addressed by the program. For the processing of 16-bit or 32-bit instructions several bit devices are grouped in blocks of 16 or 32 device numbers (i.e. 16 or 32 addresses).

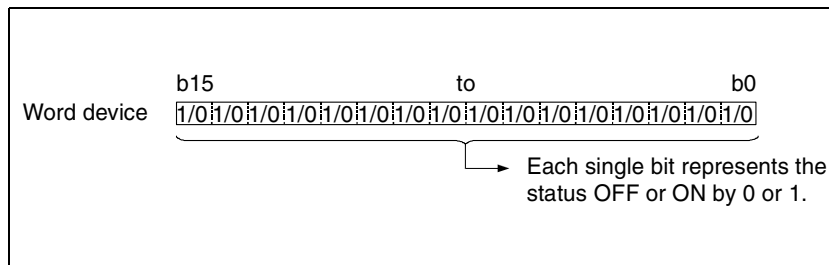
- Usage of bit devices

A bit device (e.g. inputs, outputs, relays) consists of one bit.



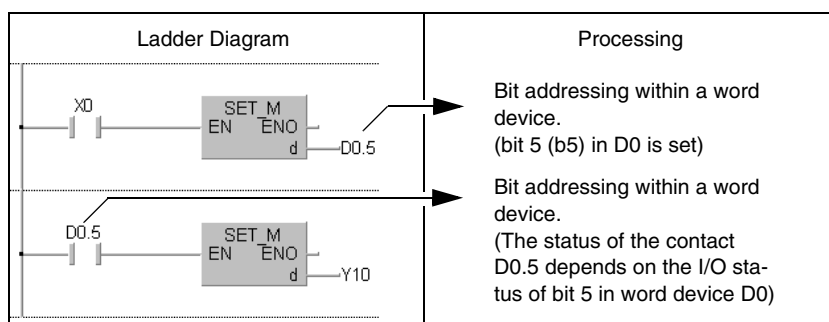
- Usage of word devices

The CPUs of the MELSEC A and Q series as well as the system Q support the addressing of each single bit in a word device.



The bits have to be addressed in hexadecimal format. For example, the bit 5 (b5) in D0 is addressed D0.5. Bit 10 in D0 is addressed D0.A.

Single bits of timers, counters, and retentive timers can not be addressed.



- Usage of bit blocks

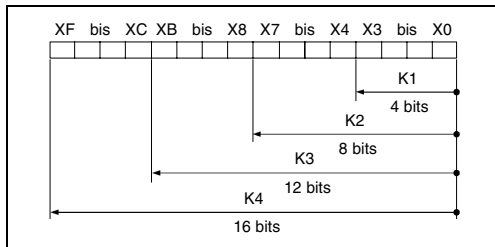
Single bits can be grouped in blocks of four and thus process word data. The detailed description is given in the following sections, "Processing of word data (16/ 32 bits)".

**Processing of word data (16 bits)**

- Usage of bit devices

Bit devices are capable of processing word data provided that the number of bit devices (addresses) is determined. Up to 16 bits can be processed in blocks of 4 bits each. The length of each block (i.e. the digit designation) is determined by K1 to K4.

- K1X0 4 addresses from X0 through X3
- K2X0 8 addresses from X0 through X7
- K3X0 12 addresses from X0 through XB
- K4X0 16 addresses from X0 through XF

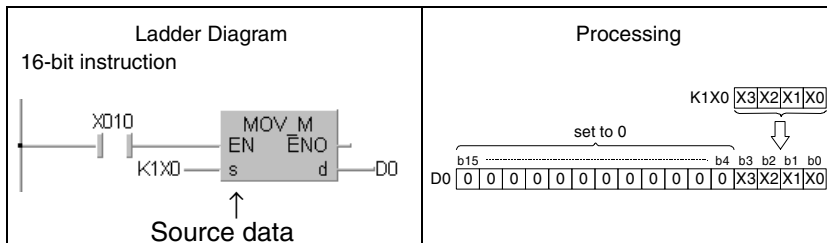


- Designation of bit blocks for s

The table below shows the range of values processed as source data for the digit designation of source data (s)

Digit Designation	16-bit instruction
K1 (4 digits)	0 to 15
K2 (8 digits)	0 to 255
K3 (12 digits)	0 to 4095
K4 (16 digits)	-32768 to 32767

The bit addresses not used are set to 0.



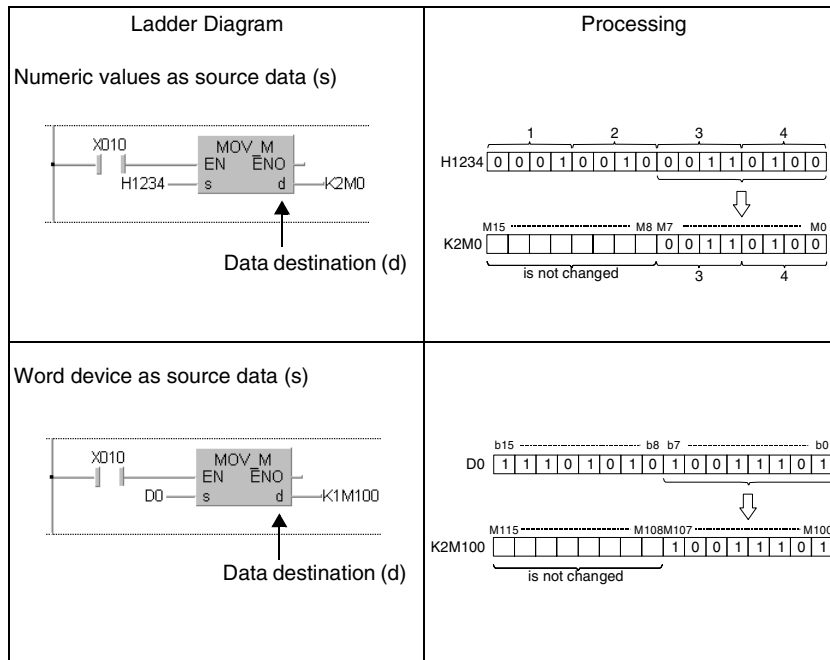
**NOTE**

For the block by block addressing of bit devices the number of the first bit device (initial device number) can be designated at any random value.



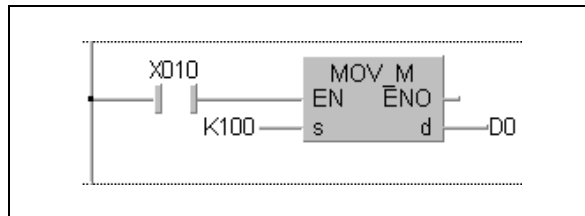
- Designation of bit blocks for d

The digit designation for the destination data (d) determines the address range the data is to be written to. The bit addresses exceeding the determined address range remain ignored.



- Usage of word devices

Word devices are determined by an address. This address comprises 16 bits.

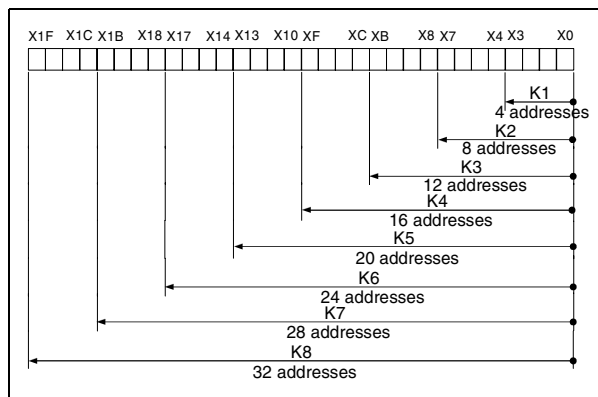


**Processing of double word data (32 bits)**

● Usage of bit devices

Bit devices are capable of processing word data provided that the number of bit devices (addresses) is determined. Up to 32 bits can be processed in blocks of 4 bits each. The length of each block (i.e. the digit designation) is determined by K1 to K8.

- K1X0 4 addresses from X0 through X3
- K2X0 8 addresses from X0 through X7
- K3X0 12 addresses from X0 through XB
- K4X0 16 addresses from X0 through XF
- K5X0 20 addresses from X0 through X13
- K6X0 24 addresses from X0 through X17
- K7X0 28 addresses from X0 through X1B
- K8X0 32 addresses from X0 through X1F

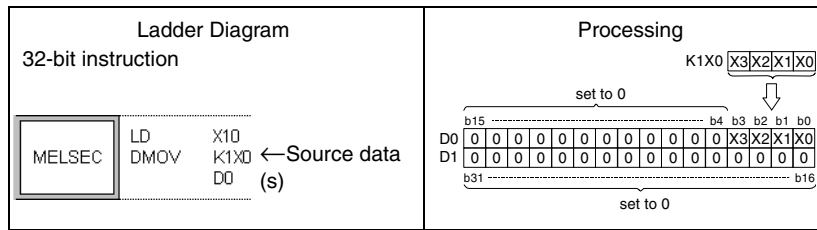


● Designation of bit blocks for s

For a specification of the digit designation the range of the values processed as source data is listed in the table below:

Digit Designation	32-bit Instruction
K1 (4 digits)	0 to 15
K2 (8 digits)	0 to 255
K3 (12 digits)	0 to 4095
K4 (16 digits)	-32768 to 32767
K5 (20 digits)	0 to 1048575
K6 (24 digits)	0 to 16777215
K7 (28 digits)	0 to 268435455
K8 (32 digits)	-2147483648 to 2147483647

The bit addresses not used are set to 0.

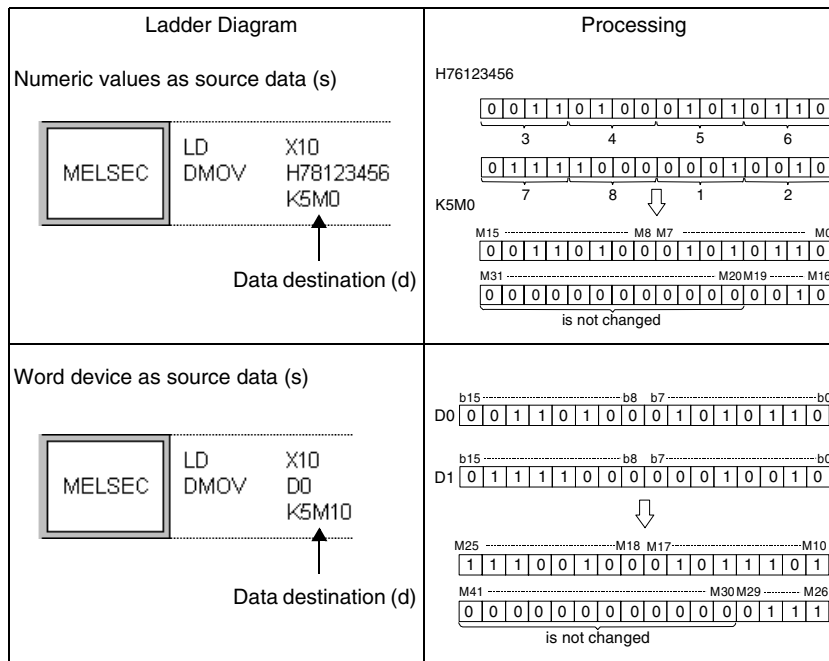


**NOTE**

For the block by block addressing of bit devices the number of the first bit device (initial device number) can be designated at any random value.

- Designation of bit blocks for d

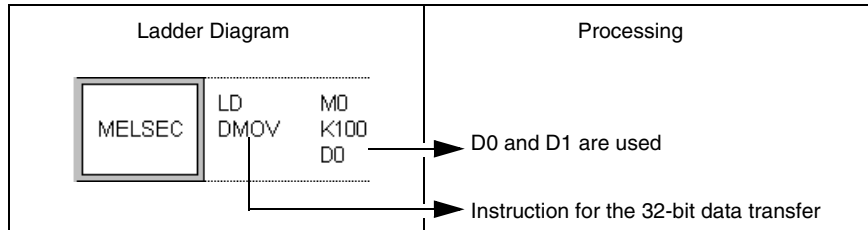
The digit designation for the destination data (d) determines the address range the data is to be written to. The bit addresses exceeding the determined address range remain ignored.



- Usage of word devices

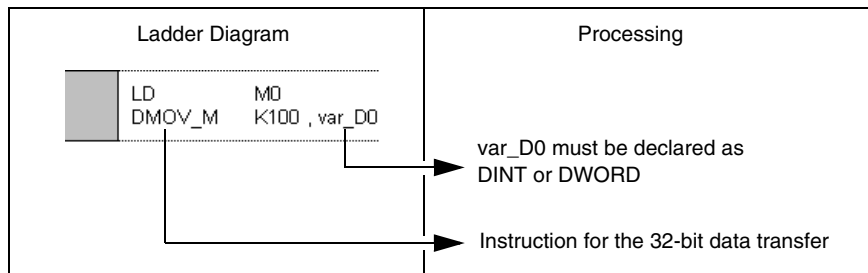
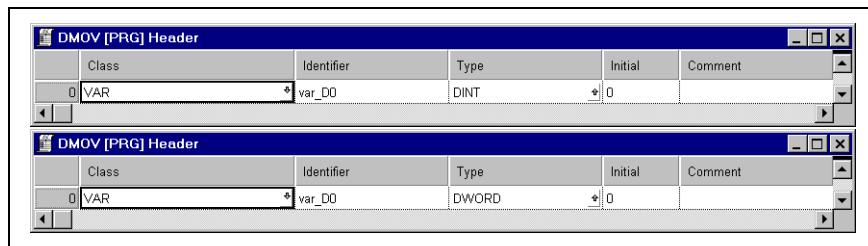
Double word devices comprise two 16-bit devices.  
According to the programming software and selected editor double word devices are programmed differently.

- In the MM, and MELSEC editor of the GX IEC Developer

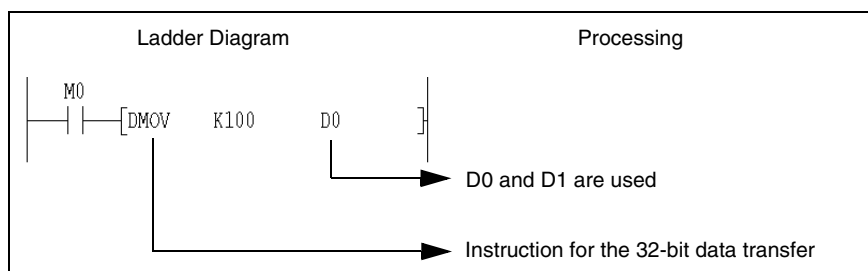


- In the IEC editor of the GX IEC Developer

Before a 32-bit device can be programmed in the IEC editor of the GX IEC Developer, the variables have to be defined in the header of the program organisation unit (POU). The data types DWORD and DINT are of the 32-bit type.

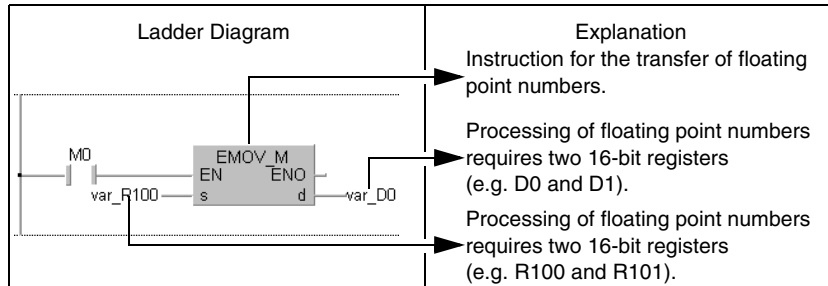


- In the editor of the GX Developer



**Processing of data of the data type REAL**

Data of the REAL type are 32-bit floating point numbers.  
 Only word devices are capable of storing floating point numbers.  
 Devices that process floating point numbers in instructions are addressed by the lower 16 bits.  
 The 32-bit floating point number is stored in two successive 16-bit registers.  
 If an AnA/AnU CPU is intended to process the data type REAL, the corresponding dedicated instructions must be applied (refer to chapter 3.3, "Programming of dedicated instructions").



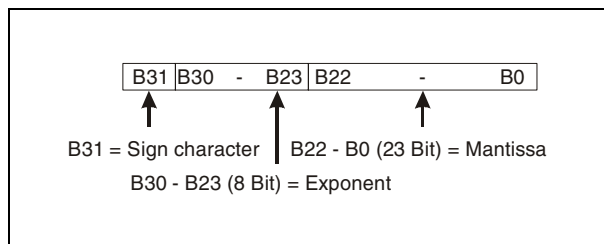
**NOTE**

The GX IEC Developer designates the floating point number  $E \square$ . Instructions processing floating point numbers begin with an E.

Two word devices are required for storing a floating point number. Therefore, it is divided into the following components:

Sign character;  $2^{[Exponent]}$ ; [Mantissa]

The bit configuration of the registers and their contents are shown in the figure below:



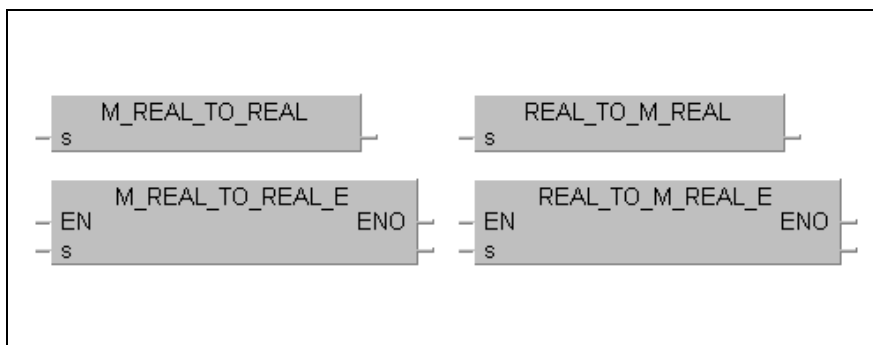
- Sign of the floating point number: The sign is stored in b31.  
 0 = Positive  
 1 = Negative
- Exponent: The n from  $2^n$  is binary stored from b23 through b30.  
 The meaning of the binary value n is shown in the following figure.

b23 to b30	FFH	FEH	FDH		81H	80H	7FH	7EH		02H	01H	00H
n	free	127	126		2	1	0	-1		-125	-126	free

Example: If the binary coded value 81H is stored in b23 to b30, then n=2.

- Mantissa: With the 23 bits from b0 through b22 7 digits can be represented binary (XXXXXX or 1,XXXXXX).

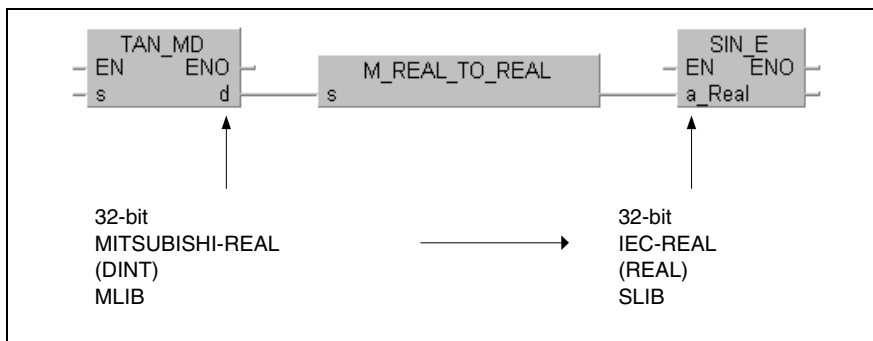
Since the REAL IEC function uses the data type REAL as input/output but the MELSEC instructions use the data type DINT, the following functions are provided to compensate this difference:



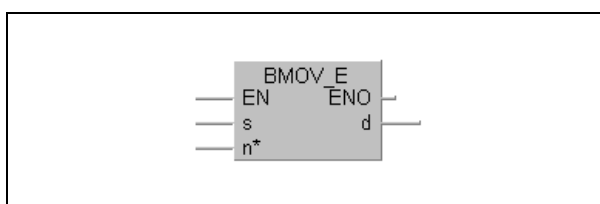
The conversion from the IEC data type REAL into the MELSEC data type is performed by the instruction REAL\_TO\_M\_REAL (REAL\_TO\_M\_REAL\_E).

The conversion from the MELSEC data type into the IEC data type is performed by the instruction M\_REAL\_TO\_REAL (M\_REAL\_TO\_REAL\_E).

Example: For the application of dedicated instructions that process the data type REAL and for IEC instructions the REAL to REAL conversion is required.



When programming in GX IEC Developer the BMOV\_E instruction can be used to switch off the variable check. No additional code is created.



Any type of data can be specified in s, even arrays are possible. n holds the number of 16 bit data to copy.

### 3.5.2 Addressing of arrays and registers in the GX IEC Developer

#### Addressing of 32-bit registers

The addressing of 32-bit registers (data type DINT, DWORD) requires a variable definition in the header of the program organisation unit (POU).

In the following example the DMOV instruction requires two 16-bit registers for moving one 32-bit data word. For the addressing in the MELSEC editor of the GX IEC Developer only the initial registers (here D10, D20) are designated. Each required second 16-bit register (D11, D21) is addressed automatically by the compiler.

In the IEC editor of the GX IEC Developer instead of the initial register a variable (here var\_D10, var\_D20) with a specific data type (here DINT (32 bits)) has to be defined in the header of the program organisation unit according to the header of the instruction. For these variables the compiler assigns corresponding addresses internally.

<p style="text-align: center;"><b>MELSEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: center;">MELSEC</td> <td style="width: 15%; text-align: center;">LD</td> <td style="width: 15%; text-align: center;">DMOV</td> <td style="width: 15%; text-align: center;">X0</td> <td style="width: 15%; text-align: center;">D10</td> <td style="width: 15%; text-align: center;">D20</td> </tr> </table>	MELSEC	LD	DMOV	X0	D10	D20	<p style="text-align: center;"><b>Ladder Diagram</b></p>	<p style="text-align: center;"><b>IEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"></td> <td style="width: 15%; text-align: center;">LD</td> <td style="width: 15%; text-align: center;">DMOV_M</td> <td style="width: 15%; text-align: center;">X0</td> <td style="width: 15%; text-align: center;">var_D10 , var_D20</td> </tr> </table>		LD	DMOV_M	X0	var_D10 , var_D20							
MELSEC	LD	DMOV	X0	D10	D20															
	LD	DMOV_M	X0	var_D10 , var_D20																
<b>Header of the DMOV instruction</b>																				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 5%;"></th> <th style="width: 25%;">Class</th> <th style="width: 15%;">Identifier</th> <th style="width: 15%;">Type</th> <th style="width: 10%;">Initial</th> <th style="width: 20%;">Comment</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>VAR_INPUT</td> <td style="text-align: center;">s</td> <td>ANY32</td> <td style="text-align: center;">0</td> <td></td> </tr> <tr> <td style="text-align: center;">1</td> <td>VAR_OUTPUT</td> <td style="text-align: center;">d</td> <td>ANY32</td> <td style="text-align: center;">0</td> <td></td> </tr> </tbody> </table>				Class	Identifier	Type	Initial	Comment	0	VAR_INPUT	s	ANY32	0		1	VAR_OUTPUT	d	ANY32	0	
	Class	Identifier	Type	Initial	Comment															
0	VAR_INPUT	s	ANY32	0																
1	VAR_OUTPUT	d	ANY32	0																
<b>Header of the program organisation unit (POU)</b>																				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 5%;"></th> <th style="width: 25%;">Class</th> <th style="width: 15%;">Identifier</th> <th style="width: 15%;">Type</th> <th style="width: 10%;">Initial</th> <th style="width: 20%;">Comment</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>VAR</td> <td style="text-align: center;">var_D10</td> <td>DINT</td> <td style="text-align: center;">0</td> <td></td> </tr> <tr> <td style="text-align: center;">1</td> <td>VAR</td> <td style="text-align: center;">var_D20</td> <td>DINT</td> <td style="text-align: center;">0</td> <td></td> </tr> </tbody> </table>				Class	Identifier	Type	Initial	Comment	0	VAR	var_D10	DINT	0		1	VAR	var_D20	DINT	0	
	Class	Identifier	Type	Initial	Comment															
0	VAR	var_D10	DINT	0																
1	VAR	var_D20	DINT	0																

**Addressing of arrays**

For the programming of instructions that use an array with array elements as input or output devices (16-bit registers) the variables in the header of the program organisation unit have to be defined according to the header of the instruction.

The individual array elements are addressed by specifying the array and the array element in square parentheses (var\_xx[x]).

The figures below show the addressing via arrays for the positioning instruction for rotary tables (ROTC):

<p style="text-align: center;"><b>MELSEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td></td><td>OUT</td><td>M0</td></tr> <tr><td></td><td>LD</td><td>X1</td></tr> <tr><td></td><td>OUT</td><td>M1</td></tr> <tr><td></td><td>LD</td><td>X2</td></tr> <tr><td></td><td>OUT</td><td>M2</td></tr> <tr><td></td><td>LD</td><td>X10</td></tr> <tr><td></td><td>ROTC</td><td>D200</td></tr> <tr><td></td><td></td><td>K10</td></tr> <tr><td></td><td></td><td>K2</td></tr> <tr><td></td><td></td><td>M0</td></tr> </table>		LD	X0		OUT	M0		LD	X1		OUT	M1		LD	X2		OUT	M2		LD	X10		ROTC	D200			K10			K2			M0	<p style="text-align: center;"><b>Ladder Diagram</b></p>	<p style="text-align: center;"><b>IEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td></td><td>ST</td><td>var_M0[0]</td></tr> <tr><td></td><td>LD</td><td>X1</td></tr> <tr><td></td><td>ST</td><td>var_M0[1]</td></tr> <tr><td></td><td>LD</td><td>X2</td></tr> <tr><td></td><td>ST</td><td>var_M0[2]</td></tr> <tr><td></td><td>LD</td><td>X10</td></tr> <tr><td></td><td>ROTC_M</td><td>var_D200, 10, 2, var_M0</td></tr> </table>		LD	X0		ST	var_M0[0]		LD	X1		ST	var_M0[1]		LD	X2		ST	var_M0[2]		LD	X10		ROTC_M	var_D200, 10, 2, var_M0
	LD	X0																																																									
	OUT	M0																																																									
	LD	X1																																																									
	OUT	M1																																																									
	LD	X2																																																									
	OUT	M2																																																									
	LD	X10																																																									
	ROTC	D200																																																									
		K10																																																									
		K2																																																									
		M0																																																									
	LD	X0																																																									
	ST	var_M0[0]																																																									
	LD	X1																																																									
	ST	var_M0[1]																																																									
	LD	X2																																																									
	ST	var_M0[2]																																																									
	LD	X10																																																									
	ROTC_M	var_D200, 10, 2, var_M0																																																									
<b>Header of the ROTC instruction</b>																																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 5%;"></th> <th style="width: 25%;">Class</th> <th style="width: 15%;">Identifier</th> <th style="width: 25%;">Type</th> <th style="width: 10%;">Initial</th> <th style="width: 20%;">Comment</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>VAR_INPUT</td> <td>s</td> <td>ARRAY [ 1.. 3 ] OF ANY16</td> <td>0,0,0</td> <td></td> </tr> <tr> <td>1</td> <td>VAR_INPUT</td> <td>n1</td> <td>ANY16</td> <td>0</td> <td></td> </tr> <tr> <td>2</td> <td>VAR_INPUT</td> <td>n2</td> <td>ANY16</td> <td>0</td> <td></td> </tr> <tr> <td>3</td> <td>VAR_OUTPUT</td> <td>d</td> <td>ARRAY [ 1.. 8 ] OF BOOL</td> <td>8(FALSE)</td> <td></td> </tr> </tbody> </table>				Class	Identifier	Type	Initial	Comment	0	VAR_INPUT	s	ARRAY [ 1.. 3 ] OF ANY16	0,0,0		1	VAR_INPUT	n1	ANY16	0		2	VAR_INPUT	n2	ANY16	0		3	VAR_OUTPUT	d	ARRAY [ 1.. 8 ] OF BOOL	8(FALSE)																												
	Class	Identifier	Type	Initial	Comment																																																						
0	VAR_INPUT	s	ARRAY [ 1.. 3 ] OF ANY16	0,0,0																																																							
1	VAR_INPUT	n1	ANY16	0																																																							
2	VAR_INPUT	n2	ANY16	0																																																							
3	VAR_OUTPUT	d	ARRAY [ 1.. 8 ] OF BOOL	8(FALSE)																																																							
<b>Header of the program organisation unit (POU)</b>																																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 5%;"></th> <th style="width: 25%;">Class</th> <th style="width: 15%;">Identifier</th> <th style="width: 25%;">Type</th> <th style="width: 10%;">Initial</th> <th style="width: 20%;">Comment</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>VAR</td> <td>var_D200</td> <td>ARRAY [0..2] OF INT</td> <td>3(0)</td> <td></td> </tr> <tr> <td>1</td> <td>VAR</td> <td>var_M0</td> <td>ARRAY [0..7] OF BOOL</td> <td>8(FALSE)</td> <td></td> </tr> </tbody> </table>				Class	Identifier	Type	Initial	Comment	0	VAR	var_D200	ARRAY [0..2] OF INT	3(0)		1	VAR	var_M0	ARRAY [0..7] OF BOOL	8(FALSE)																																								
	Class	Identifier	Type	Initial	Comment																																																						
0	VAR	var_D200	ARRAY [0..2] OF INT	3(0)																																																							
1	VAR	var_M0	ARRAY [0..7] OF BOOL	8(FALSE)																																																							

You can infer from the header of the ROTC instruction that the input device range s consists of 3 array elements of the type ANY16 and the output device range consists of 8 array elements of the type BOOL.

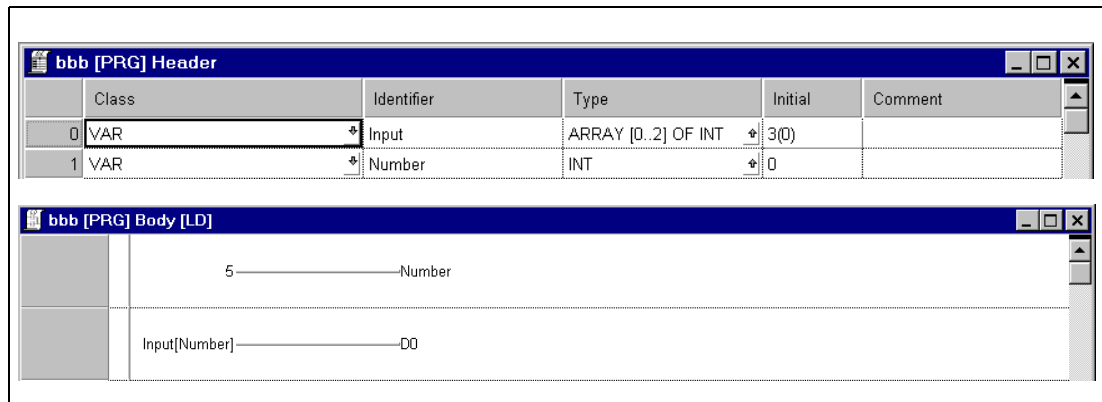
In the GX Developer and in the MELSEC editor of the GX IEC Developer for the input/output device ranges s and d only each of the initial devices D200 and M0 is specified. The compiler addresses the registers D200 through D202 for s and M0 through M7 for d.

In the IEC editors arrays must be defined for s and d. The input array s is defined as var\_D200. It consists of 3 array elements (var\_D200[0] – var\_D200[2]) of the type INT (16-bit integer). The output array d is defined as var\_M0. It consists of 8 array elements (var\_M0[0] – var\_M0[7]) of the type BOOL (bit). For these variables the compiler assigns corresponding addresses internally.



**NOTE**

Arrays can also be addressed variably. In this case instead of the array element number in square brackets any identifier for example [Number] is entered. "Number" must be declared in the header of the program organisation unit. Then a value corresponding to the according array element can be moved to the register "Number".



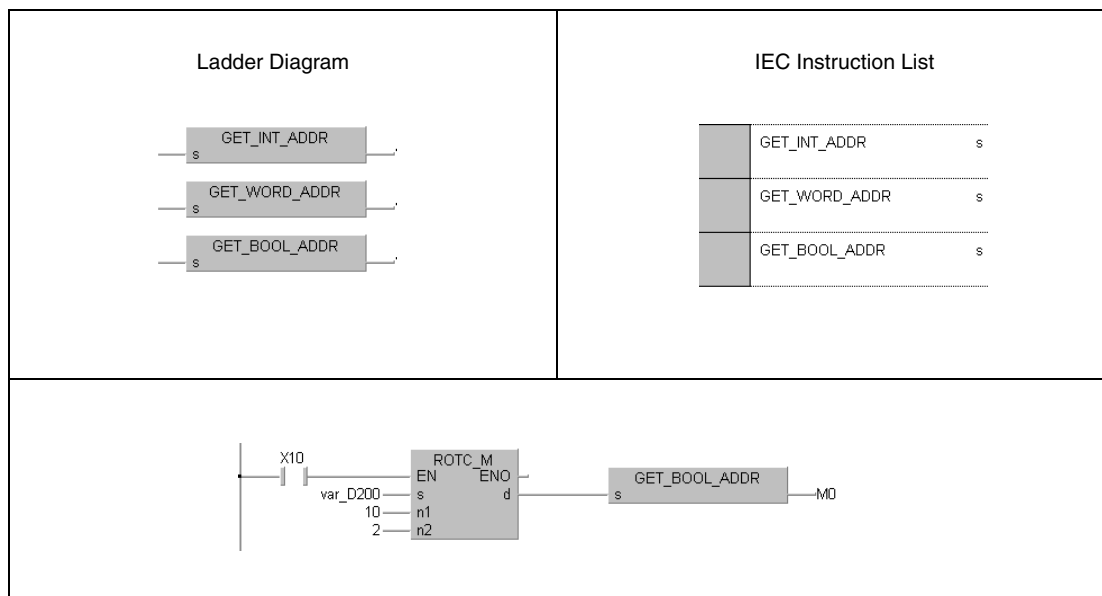
**Instructions for the array address/ initial address conversion**

The instruction set for the conversion of an output array into an initial address of a device range comprises three instructions.

The instruction GET\_INT\_ADDR converts an output array with array elements of the type INT (16-bit integer) into an initial address of a device range.

The instruction GET\_WORD\_ADDR converts an output array with array elements of the type WORD (16-bit word) into an initial address of a device range.

The instruction GET\_BOOL\_ADDR converts an output array with array elements of the type BOOL (bit) into an initial address of a device range.



After the conversion the array elements can be processed as individual devices. Therefore, the variable definition in the header of the program organisation unit is not required.

In the program with the ROTC instruction shown above instead of the array elements var\_M0[0] – var\_M0[7] the relays M0 through M7 can be used.

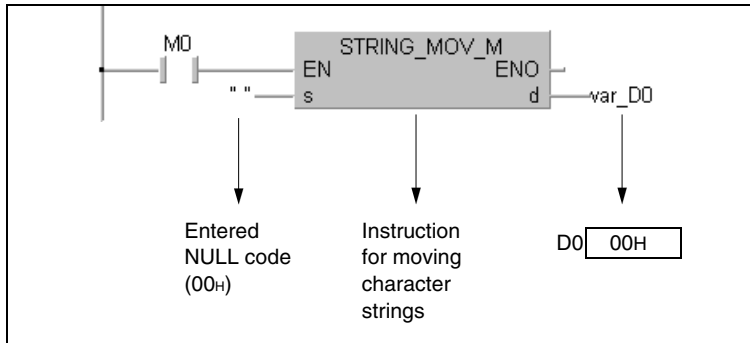
The methods of addressing devices in GX Developer and the GX IEC Developer are identical.

These instructions only convert output arrays. Input arrays must be addressed and declared as previously described.

### 3.5.3 Usage of character string data (STRING)

The data string STRING (\$) processes character strings. Character strings are all entered characters (max. 50 characters) up to the NULL code (00H).

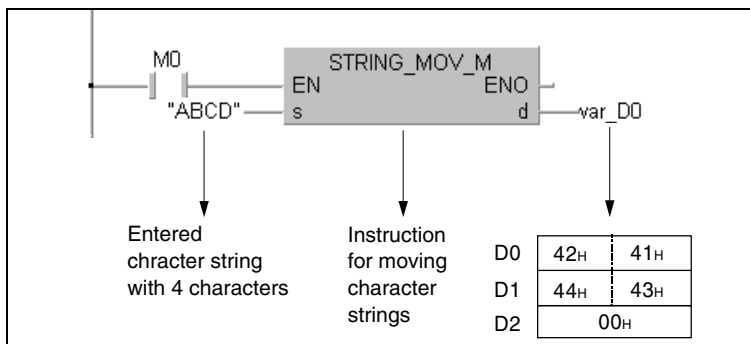
- If the entered character is the NULL code (00H)  
For the storage of the NULL code a data word (register) is required.



- If the number of characters contained in the string is even  
The storage of character strings with an even number of characters requires a number of data words calculated by the following formula:

$$(Number\ of\ characters / 2) + 1$$

If for example the character string "ABCD" is to be moved to D0, the registers D0 through D1 are required for the string and the register D2 is required for the NULL code indicating the end of string.

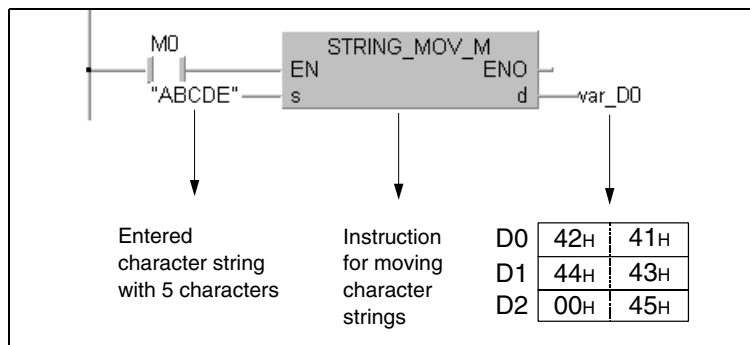


- If the number of characters contained in the character string is odd

The storage of character strings with an uneven number of characters requires a number of data words calculated by the following formula:

$$(Number\ of\ characters / 2)$$

If for example the character string "ABCDE" is to be moved to D0, the registers D0 through D2 are required for the character string. The NULL code indicating the end of string is written to the upper byte of D2.



### 3.6 Index qualification

Since the System Q and the Q series differ in index qualification from the A series, the characteristics of the CPU types are described separately in chapters 3.6.1 and 3.6.2.

Index qualification is an indirect addressing method of a device through an index register. For the index qualification within a program the device obtains the directly entered device number plus the contents of the index register as address.

#### Usage of the index qualification in the program

The program shown below gives an example of the index qualification. In the first program line the value 1 is assigned to the index register Z0. This register serves as index for D10 in the second program line. Therefore, D0 stores the value of D11 ( $D10Z = D(10+1) = D11$ ).

Ladder Diagram	Explanation
	The constant 1 is stored in the index register Z0.
	The data from the index register designated Z0 ( $D10+Z0(1)=D11$ ) are stored under D0.

The following diagram shows another example for the index qualification clarifying the processing of devices ( $Z0=20, Z1=5$ ).

Ladder Diagram	Explanation
	The constant 20 is stored in the index register Z0.
	The constant 5 is stored in the register Z1.
	The constant 100 is indexed Z0 ( $100+Z0(20)=120$ ) and stored in the register W53 ( $W53 + Z1(5)=W58$ ) indexed Z1.

**Devices that can be designated by index qualification.**

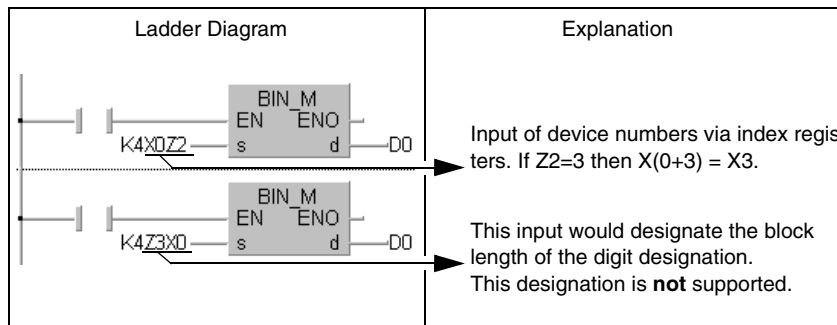
The index qualification can be applied to devices, contacts, and coils. The index registers serve for the indirect addressing of a device and contain a numeric value from -32768 to 32767.

**Devices that can not be designated by index qualification.**

Device	Meaning
E	Floating point number
\$	Character string
□.□	Bit addressing of word devices
FX, FY, FD	Function devices
P	Pointers used as label
I	Interrupt pointers used as label
Z	Index registers
S	Step relays
TV, STV	Setting values of timers
CV	Setting values of counters
N	Nesting levels
A0	AKKU
A1	AKKU

**Bit data (except AnN)**

Devices can as well be index qualified for the digit designation. The block length of the digit designation can not be affected.



### 3.6.1 Special characteristics of the System Q and QnA CPUs

A CPU of the System Q and CPU of the QnA series provides 16 index registers (Z0 – Z15). The following table shows the value ranges of timers and counters that can be designated by index qualification:

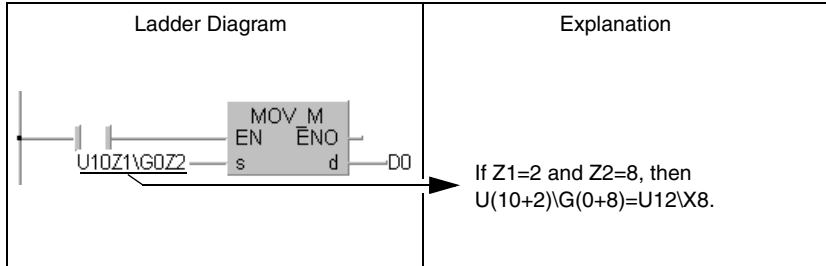
Device	Meaning	Application Example
TC	Only the registers Z0 and Z1 can be used for addressing timer contacts and coils.	
CC	Only the registers Z0 and Z1 can be used for addressing counter contacts and coils.	

**NOTE** *There are no restrictions on the addressing of current values of timers and counters.*

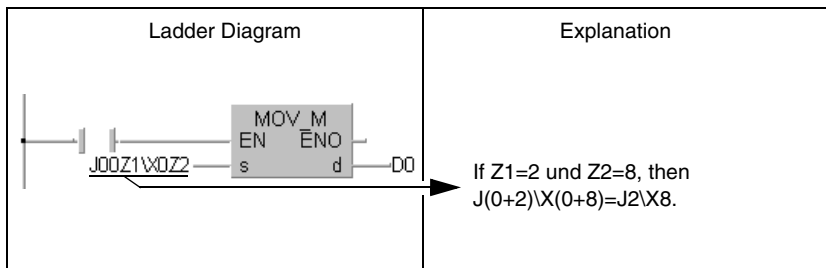
Ladder Diagram	Explanation
	Setting value of timer (TV). Index qualification <b>not</b> supported.
	Current value of timer (TN). Index qualification supported.
	Setting value of counter (CV). Index qualification <b>not</b> supported.
	Current value of counter (CN). Index qualification supported.

Another difference to the A CPUs is the support of index qualification for I/O numbers, buffer memory addresses, network numbers, and device numbers of network modules.

The diagram below shows the designation of I/O numbers and buffer memory addresses in special function modules.



The diagram below shows the designation of network numbers and device numbers of network modules.



**NOTE**

*Refer to the "QnA CPU Programming Manual (Fundamentals)", the "Q CPU (Q mode) User's Manual (Functions/programming fundamentals) and the manuals of the corresponding modules for further information on special function modules or network modules.*

### 3.6.2 Special characteristics of the AnA, AnAS, and AnU CPUs

Device numbers within a program can be designated by an index (Z or V).

In the following cases an operation error occurs when processing instructions.

- The address range of the devices is exceeded during index qualification.  
The constants K and H in this case are omitted.
- The initial address of a device range exceeds the relevant device range during index qualification.

#### NOTE

*In order to reduce the processing times, the AnA, AnAS, and AnU CPUs do not verify the device numbers during index qualification. For this reason errors occurring due to index qualification are not acknowledged as processing errors.*

*If an error occurs due to index qualification device data might be changed unintendedly.*

*Programs that contain an index qualification therefore must be written with the greatest care!*

In combination with an AnA, AnAS or AnU CPU index qualification can also be performed with bit devices used with an LD, OUT or similar instruction.

#### Storage of 32-bit data in index registers

32-bit data can be stored in the extended index registers (Z1 through Z6 and V1 through V6) of an AnA or AnU CPU. The following index registers then must be used in pairs of two:

Z1 and V1

Z2 and V2

Z3 and V3

Z4 and V4

Z5 and V5

Z6 and V6

Zn contains the lower 16 bits, Vn contains the higher 16 bits. In a 32-bit instruction only the device Z must be designated. If the device V is specified, the program cannot be processed.

32-bit instructions can only be stored in the register pairs listed above. Other combinations are not allowed. If a device in a register pair is used for the index qualification of an instruction, the data in this register are processed as 16-bit data for the index qualification.



### 3.7 Indirect Designation (GX Developer only)

With indirect designation, a device address is stored in a word device. In the sequence program the device address is not directly designated. For operations concerning this device address the word device is used instead.

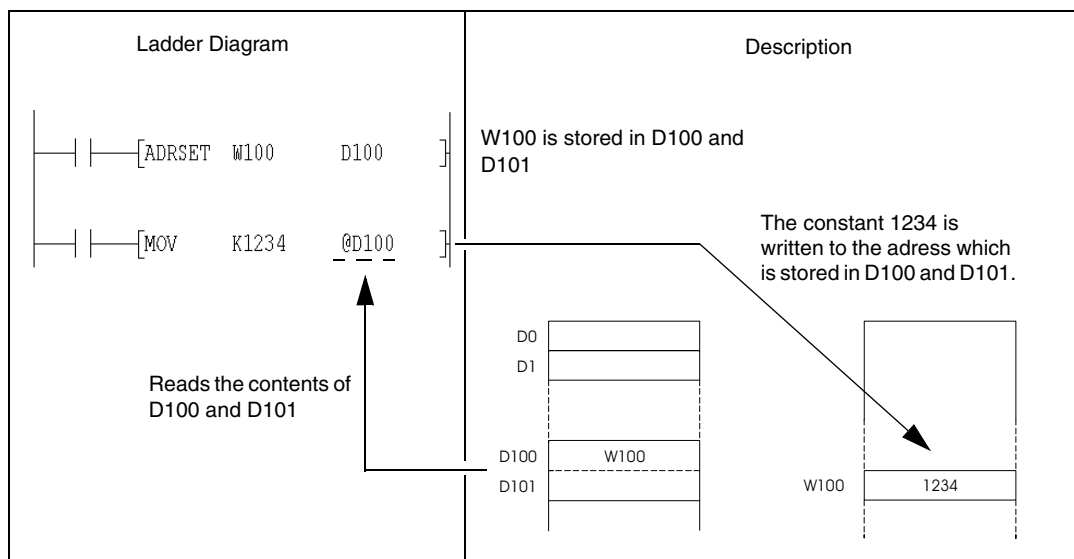
This method can be used when the index register is insufficient.

The device which contains the device address for indirect designation has the prefix "@". For example, designation of @D100 will make the contents of D100 and D101 the device Address

The address of the device performing indirect designation can be stored in the word device with the ADRSET instruction.

**NOTE**

*The ADRSET instruction is not supported by the GX IEC Developer.*



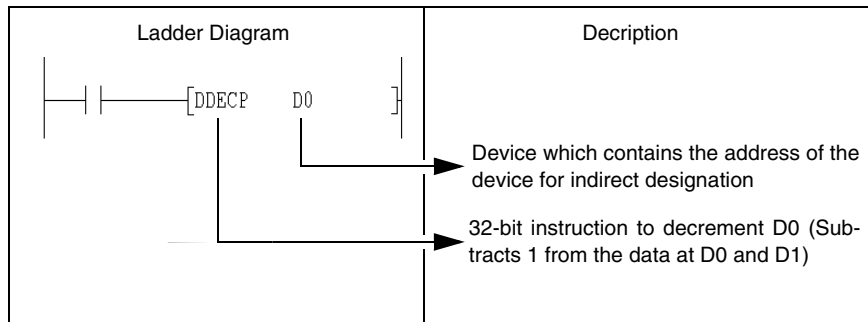
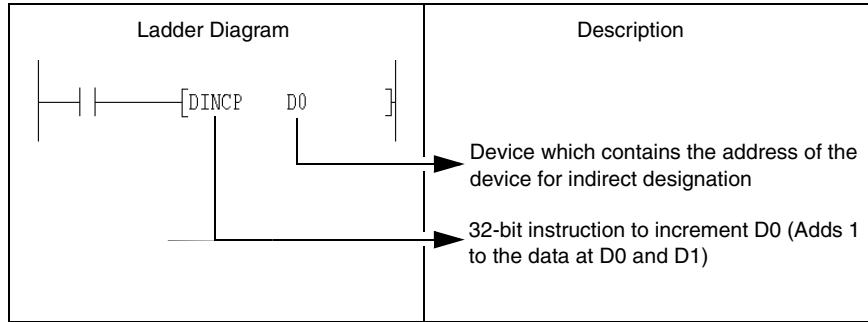
A list of devices which are capable of indirect designation is shown below.

Device Type		Indirekte Adressierung	Beispiel zur indirekten Adressierung
Internal devices (System, user)	Bit devices	Incapable	—
	Word devices	Capable	<ul style="list-style-type: none"> <li>• @D100</li> <li>• @D100Z2 (Index qualification)</li> </ul>
MELSECNET/10	Bit devices	Incapable	—
	Word devices	Capable (The ADRSET instruction cannot be used to write the indirect address)	<ul style="list-style-type: none"> <li>• @J1\W10</li> <li>• @J1Z1\W10Z2 (Index qualification)</li> </ul>
Special function module			<ul style="list-style-type: none"> <li>• @U10\G0</li> <li>• @U10Z1\G0Z2 (Index qualification)</li> </ul>
Index register Zn		Incapable	—
File register		Capable	<ul style="list-style-type: none"> <li>• @R0, @ZR20000</li> <li>• @R0Z1, @ZR20000Z1 (Index qualification)</li> </ul>
Nesting		Incapable	—
Pointer			—
Constants			—
Other			—

**NOTE** Refer to the "QnA CPU Programming Manual (Fundamentals)" or the "Q CPU (Q mode) User's Manual (Functions/programming fundamentals) for further information on device names.

**NOTE** To store an address for indirect designation, two words are used. Therefore, to decrease or increase a stored address for indirect designation by arithmetic instructions, the addition or subtraction of 32-Bit data is required.

In the following program examples the device which stores the device for indirect designation is incremented and decremented by 32-Bit instructions. By doing so, the address of the device for indirect designation is increased resp. decreased by 1.



### 3.8 Operation errors

In the following cases operation errors occur:

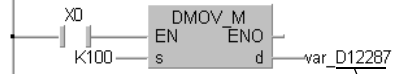
- If the error conditions described under the topic "Operation Errors" for the individual instructions match, an error code is returned.
- If a buffer register is used, but there is no special function module connected to the specified I/O number.
- If a link device is used, but the corresponding network does not exist.
- If a link device is used, but there is no network module connected to the specified I/O number.

**NOTE**

*If a file register is specified in the parameters but no memory card (System Q and Q series CPUs only) installed, an error code is returned (2401 = File Set Error).  
If a file register is accessed although there are no file registers specified in the parameters, an error code is returned. If the file register is read out, the code "FFFFH" is returned.*

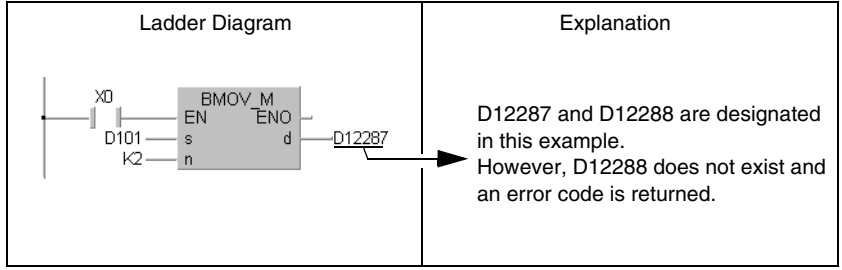
#### 3.8.1 Verification of the device range

- If instructions use devices with fixed length (MOV, DMOV, etc.), the device range will not be verified.  
In those cases where the relevant address range is exceeded the data to be written is written to a vacant register.  
If for example, 12k addresses are designated, there will no error code be returned until the register address D12287 is exceeded.

Ladder Diagram	Explanation
 <p>The diagram shows a normally open contact labeled 'X0' in series with a coil labeled 'K100'. This coil is connected to the 'EN' (Enable) terminal of a rectangular instruction block labeled 'DMOV M'. The 'ENO' (Enable Out) terminal of the block is connected to a variable labeled 'var_D12287'.</p>	<p>D12287 and D12288 are designated in this example but D12288 does not exist. A vacant register will be overwritten with the contents of D12287.</p>

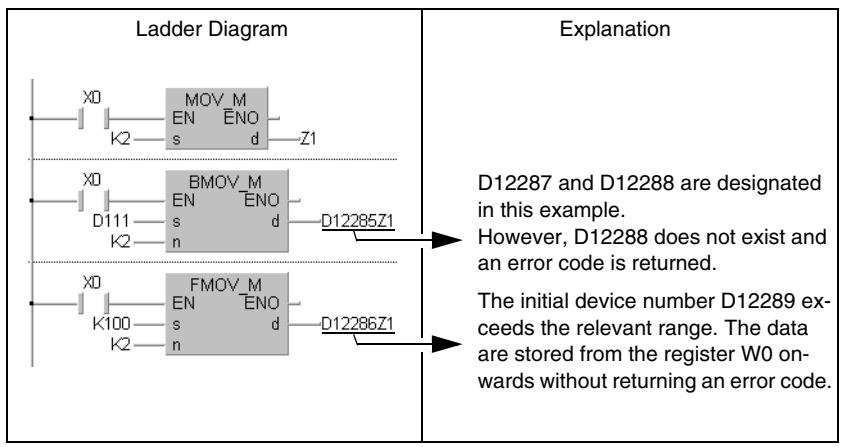
For an index qualification the device range is not verified either.

- If instructions use devices with variable length, the device range is verified (BMOV, FMOV, and other instructions that designate initial addresses). In those cases where the relevant address range is exceeded an error code is returned. If for example, 12k addresses are designated, the error code is only returned after the register address D12287 is exceeded.

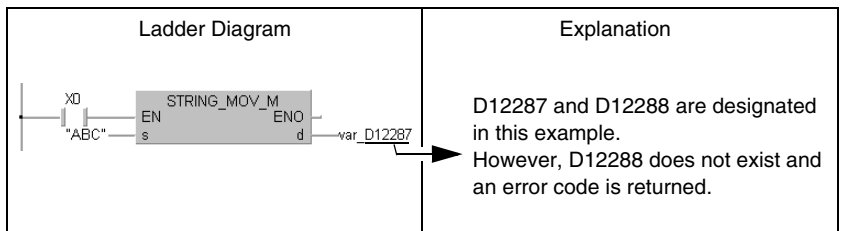


The device range is verified for an index qualification too.

There is no error code returned, if the initial device number exceeds the address range.



Since character strings are of variable lengths the device range is verified. In cases where the corresponding device range is exceeded, an error code is returned. If for example, 12k addresses are designated, there will no error code be returned until the register address D12287 is exceeded.



- The device range is verified for an index qualification of the direct output (DY).

### 3.8.2 Verification of the device data

#### Verification of binary data

- If the operation result exceeds the value range, no error code is returned. The carry flag in this case is not set.

#### Verification of BCD data

- Each digit of the BCD values (0 to 9) is verified.  
If one individual digit exceeds the range of 0 to 9 (A to F), an error code is returned.
- If the operation result exceeds the value range, no error code is returned. The carry flag in this case is not set.

#### Verification of floating point numbers

Operation errors occur in the following cases:

- The value of the floating point number becomes 0.
- The absolute value of the floating point number falls below the value  $1.0 \times 2^{-127}$
- The absolute value of the floating point number exceeds the value  $1.0 \times 2^{129}$

#### Verification of character strings

The device data are not verified.

### 3.9 Execution conditions of the instructions

#### 3.9.1 Execution condition

There are 4 different types of execution conditions for the instructions:

- Non-conditional execution

The instructions are executed regardless of the signal status of the devices.  
Example: LD X0, OUT Y10

- Execution at ON

The instructions are executed as long as the execution instruction is set.  
Example: MOV, FROM

- Execution at leading edge

The instructions are executed at leading edge (signal status changes from 0 to 1) from the execution condition.  
Example: PLS, MOVP

- Execution at trailing edge

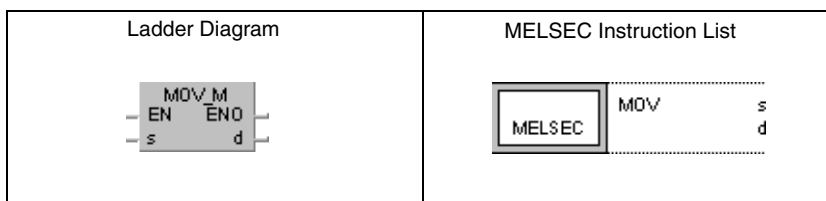
The instructions are executed at trailing edge (signal status changes from 1 to 0) from the execution condition.  
Example: PLF

The vast majority of instructions are of the following two types:

- Execution at ON
- Execution at leading edge from the execution condition

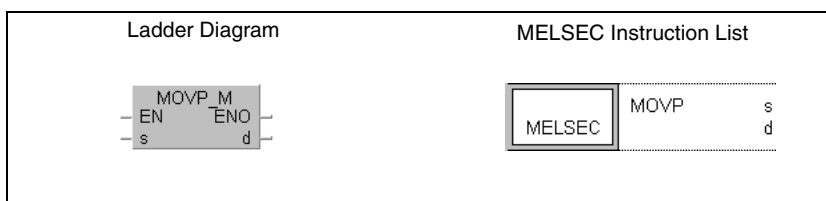
The instruction is executed as long as the execution instruction is set. Such instructions are not particularly indicated.

Example: MOV\_M/ MOV

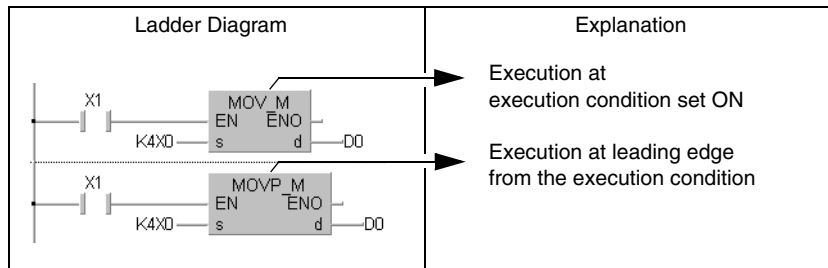


When judging the leading edge from the execution condition the instruction is executed only if the signal state changes from 0 to 1.

Example: MOVP\_M/ MOVP



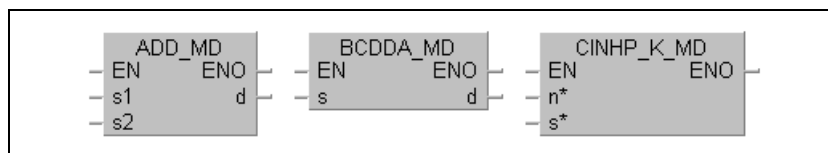
The following example shows the execution of the MOV instruction with the execution condition set ON and the execution at leading edge from the execution condition:



### 3.9.2 EN input and ENO output

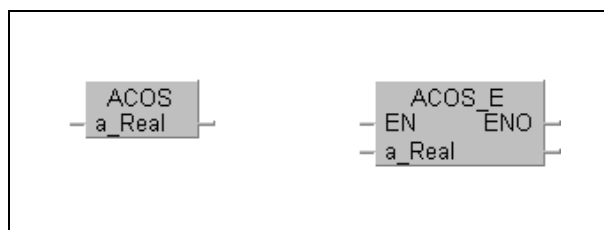
All instructions described in this manual are provided in the manufacturer library of the GX IEC Developer. These instructions in addition to the input and output variables provide an EN input and an ENO output.

The figure below shows several MELSEC instructions from the GX IEC Developer manufacturer library:



In the IEC standard library nearly all instructions appear twice. They just differ in the suffix "\_E". These instructions provide an EN input and an ENO output.

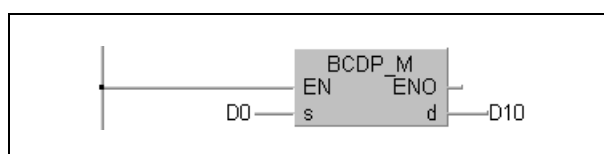
The figure below shows two IEC instructions from the standard library of the GX IEC Developer:



The following examples show the differing execution of the instruction with and without EN inputs and ENO outputs.

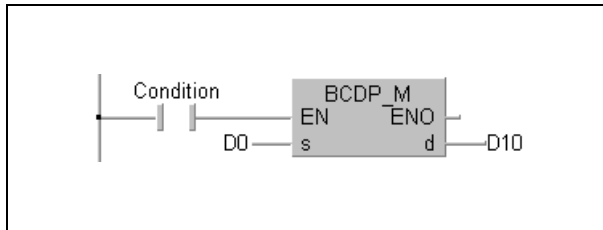
Example 1: Without additional connection

Without additional connection the execution condition of the instruction is permanently set.



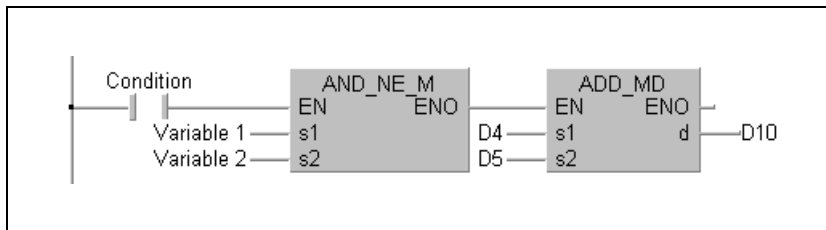
Example 2: Connection with a contact

If the EN input is connected with a contact, the instruction is executed if the condition is matched.



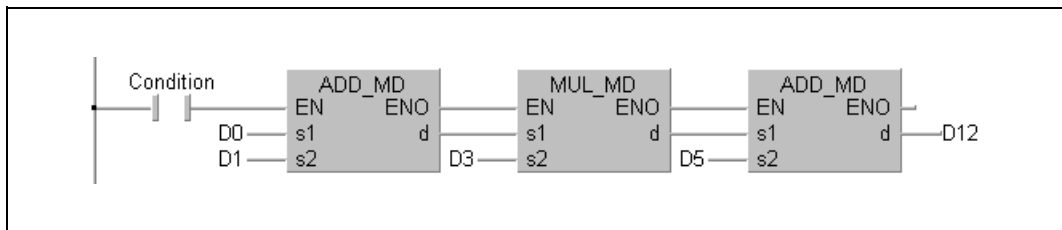
Example 3: Connection with an operation result

If the boolean result of an arithmetic operation is connected to the EN input, the instruction is only executed, if the result of the arithmetic operation is TRUE.



Example 4: Connection with the preceding instruction

If the EN input is connected to the ENO output of the preceding instruction, the instructions are only executed, if the condition is matched.



**NOTE**

*The ENO output must not compulsorily be connected. The signal at the EN input is looped-through to the ENO output. If the EN input is "TRUE", the ENO output is "TRUE" as well.*



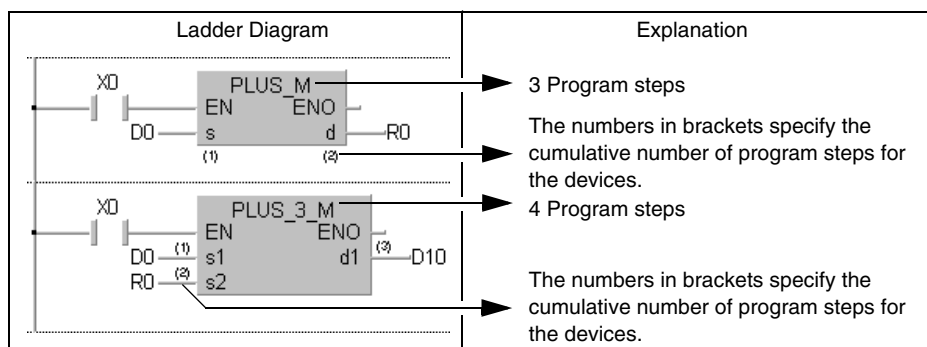
### 3.10 Number of program steps

In order not to exceed the required memory capacity in the internal memory and ROM or RAM memory of the memory cards and memory cartridges a calculation of the total number of steps in a program is required. In the following sections the calculation of steps for the instructions of the System Q, QnA and A CPUs is described.

#### 3.10.1 For a System Q and QnA CPU

The number of steps for an instruction depends on the number of basic steps. Most of the instructions for their execution only require a number of basic steps. The number of basic steps depends on the number of used devices plus 1.

The example below shows the calculation of the number of basic steps for the PLUS instruction:



- The number of program steps for the application of input and output instructions:

The number of program steps for input instructions (LD, LDI, AND, ANI, OR, ORI) depends on the devices used.

If internal devices or file registers (R0 through R32767) are used, the number of steps is 1.

If direct access inputs (DX) are used, the number of steps is 3.

The number of program steps for output instructions (LDP, LDF, ANDP, ANDF, ORP, ORF) depends on the devices used.

If internal devices or file registers (R0 through R32767) are used, the number of steps is 2.

If other devices are used the number of steps is 4.

- The number of program steps for several transfer instructions:

Devices increasing the Number of Steps	Added Steps	Example
Devices of special function modules	1	MOV <u>U4\G10</u> D0
Link devices		MOV <u>J3\B20</u> D0
File registers addressed in series		MOV <u>ZR123</u> D0
32-bit constants		DMOV <u>K123</u> D0
Floating point number as constants		EMOV <u>E0.1</u> D0
Character strings	For an odd number: (number of characters/2)-1 For an even number: Number of characters/2	\$MOV <u>„123“</u> D0

In cases where several of these factors apply the number of steps sums up.  
 If for example, MOV U1\G10 ZR123 is programmed, 1 step is added for the buffer memory and 1 step for the file register addressed in series, resulting in a total of 2 steps.

### 3.10.2 For an AnA, AnAS, and AnU CPU

In combination with an AnA, AnAS or AnU CPU a number of peculiarities has to be considered described in the following section.

The number of steps increases by 1, if one of the following device numbers listed in the table below (extended range of AnA series) is designated by an instruction.

Devices	Device Range
Relay M, L, S	2048 to 8191
Timer T	256 to 2047
Counter C	256 to 1023
Link relay B	400 to FFF
Data register D	1024 to 6143
Link register W	400 to FFF

If any device from the extended address range is index qualified by an extended index register, the number of steps also increases by 1.

The figure below shows several examples for the calculation of program steps. The first example shows the configuration of steps for the programming of instructions from the normal address range.

The succeeding examples show the configuration of program steps for the usage of devices from the extended address area.

Ladder Diagram	Explanation
	LDT01 Step + D0W010 <u>5 Steps</u> <b>6 Steps</b>
	LDT3002 Steps + D0W800 <u>6 Steps</u> <b>8 Steps</b>
	LDT10002 Steps + D2000W010 <u>Z16 Steps</u> <b>9 Steps</b>
	LDT01 Step + D2000 Z1D300 <u>5 Steps</u> <b>6 Steps</b>

For an index qualification in a 1 step instruction (e.g. LD or OUT) the number of steps increases by 1.

The examples below show the difference of the programming with or without index qualification. The number of steps even increases by 1 only, if the index qualification is applied with an extended index register (Z1 through Z6, V1 through V6).

Ladder Diagram	Explanation
	LDX01 Step + OUTY40 <u>1 Step</u> <b>2 Steps</b>
	LDX0 Z2 Steps + OUTY40 <u>1 Step</u> <b>3 Steps</b>  Index qualification



# 4 Layout and Structure of the Chapters

This chapter gives an introduction to the chapters 5 through 9 and describes the layout and structure of the explanations to the instructions for the MELSEC A and Q series and the System Q.

The figure below shows that each of these chapters starts with a table that lists and comments the structure and subdivision of the instructions described in that chapter.

## 6 Application Instructions, Part 1

The application instructions, part 1 comprise instructions that process numerical 16-bit and 32-bit data, floating point data, and character string data. Commonly, these basic instructions perform comparison and arithmetic operations.

Instruction	Meaning
Comparison operation instruction	Compares data to data (e.g. =, >, ≥)
Arithmetic operation instruction	Adds, subtracts, multiplies, divides, increments, and decrements BIN and BCD data, floating point data, and BIN block data Links character strings
Data conversion instruction	Converts data types (e.g. BCD -> BIN, BIN -> BCD)
Data transfer instruction	Transmits designated data
Program branch instruction	Program jump commands
Program execution control instruction	Enables and disables program interrupts
Refresh instruction	Refreshes bit devices, links, and I/O interfaces
Other convenient instructions	Count 1- or 2-phase input up or down, teaching timer, special function timer, rotary table near path rotation control, ramp signal, pulse density measurement, fixed cycle pulse output, pulse width modulation, matrix input

Each subdivided topic is described in the following according chapter and illustrated by program examples.

## 4.1 Overview of the instructions

Each subdivided topic starts with a table that lists all individual instructions described in this section. As the figure below shows, all variations of the instructions are represented in MELSEC and IEC editor notation.

### 6.1 Comparison Operation Instructions

Comparison operation instructions compare data values (e.g. equal to =, greater than >, less than <). Programming the comparison operation instructions is similar to the corresponding basic instructions:

LD, LDI ⇒ LD=, LDD=

AND, ANI ⇒ AND=, ANDD=

OR, ORI ⇒ OR=, ORD=

Function	MELSEC-Instruction in MELSEC Editor	MELSEC-Instruction in IEC Editor	Function	MELSEC-Instruction in MELSEC Editor	MELSEC-Instruction in IEC Editor
= equal	LD=	LD_EQ_M	≤ less equal	LD≤=	LD_LE_M
	AND=	AND_EQ_M		AND≤=	AND_LE_M
	OR=	OR_EQ_M		OR≤=	OR_LE_M
	LDD=	LDD_EQ_M		LDD≤=	LDD_LE_M
	ANDD=	ANDD_EQ_M		ANDD≤=	ANDD_LE_M
	ORD=	ORD_EQ_M		ORD≤=	ORD_LE_M
	LDE=	LD_EEQ_M		LDE≤=	LD_ELE_M
	ANDE=	AND_EEQ_M		ANDE≤=	AND_ELE_M
ORE=	OR_EEQ_M	ORE≤=	OR_ELE_M		

When using the GX IEC Developer, always choose the IEC instruction when different notations are offered.

## 4.2 The CPU table

The sections describing the instructions start with a table that indicates each CPU (AnS, AnN, AnA, AnAS, AnU, QnA, QnAS, Q4AR, System Q) capable of processing the respective instruction. The capable CPUs are indicated by a black spot.

Data Conversion Instructions **INT, INTP, DINT, DINTP**

---

**6.3.4 INT, INTP, DINT, DINTP**

CPU	AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
			● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.  
<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Any particular processing details of a certain CPU are commented in a footnote (e.g. extended instructions, refer to "3.3 Programming of the extended instructions").

### 4.3 Devices MELSEC A

The table "Devices MELSEC A" lists all usable devices that can be used for the internal variables (e.g. s1, s2, d).

Devices MELSEC A	Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011					
	Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level				
	X	Y	M	L	S	B	F	T	C	D	W	R	AO	A1	Z	V						K	H (16#)	P	I	N
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K4	5/7	●		●	
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									

<sup>1</sup> The number of steps is 7, provided the index function was started, the digit designation of a bit device is not K4, and the head address of a bit device is not a multiple of 8 (or 16 for the A3H, A3M, AnA, AnAS and AnU CPU). Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

The usable bit and word devices are listed separately. Only the devices indicated by a black spot can be used.

Whether decimal (K) or hexadecimal (H, 16#) constants can be processed by the instruction is indicated in the column "Constant".

The column "Pointer" indicates whether the instruction can use pointers (P) and/or interrupt pointers (I).

Whether the instruction can be executed in nesting levels is indicated in the column "Level".

The digit designation (block length) for bit devices available for the instruction is listed in the column "Digit designation". The sample above shows that the instruction can address digit designations from (K1 to K4) 4 to 16 bits.

The number of program steps used is listed in the column "Number of steps".

Whether the instruction can apply an index qualification is indicated in the column "Index".

Whether the instruction can set the carry flag is indicated in the column "Carry Flag".

Whether the instruction can set the error flag is indicated in the column "Error Flag".

Any particular details are commented in footnotes below the table.

## 4.4 Devices MELSEC Q

Under the term "MELSEC Q" all CPUs of the System Q and the QnA, QnAS and Q4AR CPUs are grouped together.

The table "Devices MELSEC Q" lists all usable devices that can be used for the internal variables (e.g. s1, s2, d).

The devices are not listed separately; only a distinction is drawn whether the instruction is capable of designating bit and/or word devices.

Devices MELSEC Q	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	—	3
s2	●	●	●	●	●	●	●	●	—	—	

Whether the instruction supports file register access is indicated in the column "File Register". The column "MELSECNET/10 Direct J□□" specifies whether the instruction supports read/write operations of bit and/or word data from/to stations connected to the MELSECNET/10. "J□" specifies the station number and "□" the device number.

The column "Special Function Module U□G□" specifies whether the instruction supports read/write operations of data from/to the buffer memory of an installed special function module. "U□" specifies the head address of the special function module and "G□" the buffer memory address.

Whether the instruction can apply an index qualification is indicated in the column "Index Register Zn".

Whether decimal (K) or hexadecimal (H, 16#) constants can be processed by the instruction is indicated in the column "Constant K, H (16#)".

The column "Other" specifies whether the instruction uses any other devices and constants.

Whether the instruction can set the error flag is indicated in the column "Error Flag".

The number of program steps used is listed in the column "Number of steps".

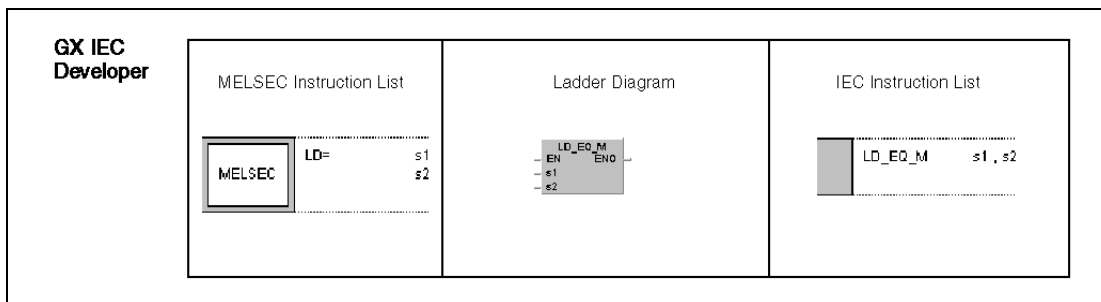
Any particular details are commented in footnotes below the table.

## 4.5 Representation format of the instruction

### 4.5.1 Representation in the GX IEC Developer

The device tables are followed by the representation format of the instruction in the GX IEC Developer.

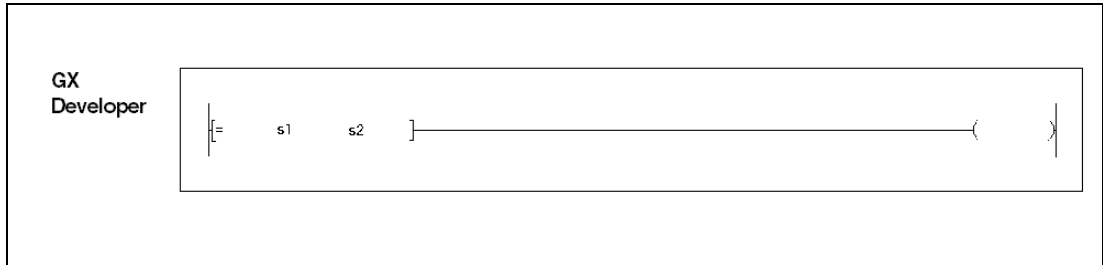
The figure below from the left to the right shows the representation of the instruction LD\_EQ\_M in the MELSEC editor (MELSEC instruction list) and in the IEC editor (ladder diagram and IEC instruction list).





### 4.5.2 Representation in the GX Developer

The representation format for the instruction in the GX IEC Developer is followed by the representation format of the instruction in the GX Developer.



## 4.6 Variables

The table of variables lists all internal variables of the instruction.

Variables	Set Data	Meaning	Data Type	
			MELSEC	IEC
s	s	s+0: Measurement of table rpm (internal use only).	BIN 16-bit	Array [1..3] of ANY16
		s+1: Number of position.		
		s+2: Number of sector.		
	n1	Number of sectors (divisions) on table (2 to 32767).		ANY16
n2	Number of low speed sectors (0 to n1).	ANY16		
d	d	d+0: A-phase input signal.	Bit	Array [1..8] of Bool
		d+1: B-phase input signal.		
		d+2: Zero position detection input signal.		
		d+3: High speed forward output signal (internal use only).		
		d+4: Low speed forward output signal (internal use only).		
		d+5: Stop output signal (internal use only).		
		d+6: High speed reverse output signal (internal use only).		
		d+7: Low speed reverse output signal (internal use only).		

The column "Meaning" describes the functions of the devices and device elements. The column "Data Type" lists the data types of the devices. Provided that there are differences between the data types of the MELSEC and the IEC editor, these are listed as well. Refer to the chapters "3.4 Programming of variables" and "3.5 Data types" for further details on variables.

## 4.7 Functions

The section "Functions" describes the functions of the instruction in detail.

The figure below shows the description of the functions of the LDF/LDP instruction.

<b>Functions</b>	<p><b>Pulse operation start</b></p> <p><b>LDP leading edge</b></p> <p><b>LDF trailing edge</b></p> <p>Similar to the LD and LDI instructions, these instructions designate contacts specified by bit or word devices. The result of the LDP instruction is 1, if the addressed bit of the device changes from 0 to 1 (leading edge). The result of the LDF instruction is 1, if the addressed bit of the device changes from 1 to 0 (trailing edge). As single instruction the LDP instruction executes the same function as a PLS instruction and with the input condition at leading edge generates a pulse output.</p> <p>The program example on the left shows a ladder diagram applying an LDP instruction. The example on the right does not apply an LDP instruction.</p>
------------------	--

## 4.8 Notes

The section "NOTE" points out particular details, errors, and sources of malfunction in the programming of the instruction.

<b>NOTE</b>	<p><i>The MEP and MEF instructions will occasionally not function properly when pulse conversion is applied to contacts that are indexed by a subroutine or by a FOR/NEXT instruction. In this case, the EGP/EGF instruction has to be applied.</i></p> <p><i>The MEP/MEF instruction operates with the operation results immediately prior to the MEP and MEF instructions. For this reason, an AND instruction should be used at the same position. The MEP and MEF instructions cannot be used at the LD or OR position.</i></p>
-------------	---

## 4.9 Operation Errors

The description of the operation errors mainly refers to the error codes of the Q series and the System Q (see "11.1 Table of error codes; Q00J, Q00 and Q01CPU and "11.2 Table of error codes; Q series and System Q"). For information on the error codes of the A series refer to the chapters "11.3 Table of error codes; A series (except AnA and AnAS)" and "11.4 Table of error codes; AnA and AnAS CPUs".

The figure below shows the operation errors of the DELTA-/DELTAP instruction.

<b>Operation Errors</b>	<p>In the following cases an operation error occurs and the error flag is set:</p> <ul style="list-style-type: none"> <li>● The number of output designated by d exceeds the output range (error code: 4101).</li> </ul>
-------------------------	--

## 4.10 Program Examples

The program examples given at the end of each section primarily contain programs for the Q series and the System Q.

The program examples are programmed in the representation format of the MELSEC instruction list, the ladder diagram and the IEC instruction list. For a clearer description in many cases graphical illustrations were added.

The figure below shows a program example of the instructions LD, AND, OR, and ORI.

**Program Example 1** LD, AND, OR, ORI

The following program shows series and parallel connections of contacts. Bit 5 (b5) in D0 is also read as contact.

MELSEC Instruction List			Ladder Diagram			IEC Instruction List		
MELSEC	LD	X3		LD	X3			
	OR	D0.5		OR	D0.5			
	OR	X5		OR	X5			
	ST	Y33		ST	Y33			
MELSEC	LD	X5		LD	X5			
	AND	M11		AND	M11			
	OR	X6		ORN	X6			
	ST	Y34		ST	Y34			

In the following figure a program example for the RBMOV instruction is shown. The representation of the instructions is that of the GX Developer.

**Program Example 1** RBMOV

The following program transfers the lower four bits (b0 through b3) of data in D66 through D69 to the outputs Y30 through Y3F with the rising edge of SM402. The number of data (4 blocks) is specified by n.

The bit patterns show the structure of bits before and after the transfer.

Instruction List		
0	LD	SM402
1	RBMOV	D66 K1Y30 K4
5	END	

Ladder Diagram

	b15	-----	b4	b3	-----	b0		
D66			1	1	0	1	1 1 0 1	Y33 - Y30
D67			0	0	0	0	0 0 0 0	Y37 - Y34
D68			1	0	0	1	0 0 1 1	Y3B - Y38
D69			0	1	1	0	1 1 0 1	Y3F - Y3C

<sup>1</sup> These bits are ignored.



# 5 Sequence Instructions









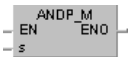
Sequence instructions, besides conventional instructions to program input and output contacts, also include program jump commands, block connection instructions and bit shift instructions, master control, program termination and other instructions. These are the fundamental instructions for programming the MELSEC series.

The following table shows the division of the fundamental instruction set:

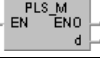
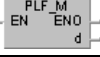
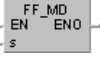
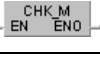
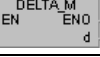
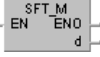
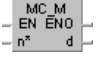
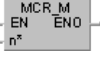
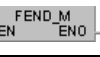
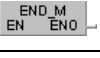
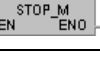
Instruction	Meaning
Input instruction	Operation start, series and parallel connection of contacts.
Connection instruction	Series and parallel block connection, storage and processing of operation results, inversion of operation results, conversion of operation results into pulses, setting of edge relays.
Output instruction	Bit devices, counter and timer contacts, output, setting, and resetting of annunciators, setting and resetting of devices, leading edge and trailing edge output, bit device output inversion, generating pulses.
Shift instruction	Shifting bit devices.
Master control instruction	Setting and resetting single parts of a program.
Termination instruction	End of a part of program, end of sequence and routine programs.
Miscellaneous instructions	Sequence program stop, no operation.

**NOTE**

The following table, besides the MELSEC instructions in the different editors, also contains the according IEC instructions:

in MELSEC Editor	MELSEC Instruction			IEC Instruction in IEC Editor
	in IEC Editor		Ladder Diagram	
	Instruction List			
LD	—		—	LD
LDI	—		—	LDN
AND	—			AND
ANI	—			ANDN
OR	—		—	OR
ORI	—		—	ORN
LDP	LDP_M	—	—	—
LDF	LDF_M	—	—	—
ANDP	ANDP_M	—		—

MELSEC Instruction				IEC Instruction in IEC Editor
in MELSEC Editor	in IEC Editor			
	Instruction List	Ladder Diagram		
ANDF	ANDF_M	—		—
ORP	ORP_M	—		—
ORF	ORF_M	—		—
ANB	—		—	AND ( ... )
ORB	—		—	OR ( ... )
MPS	MPS_M			—
MRD	MRD_M			—
MPP	MPP_M			—
INV	INV_M			NOT
MEP	MEP_M	—		—
MEF	MEF_M	—		—
EGP	EGP_M	—		—
EGF	EGF_M	—		—
OUT	OUT_M			ST
OUT T	TIMER_M	—		—
OUT TH	TIMER_H_M	—		—
OUT C	COUNTER_M	—		—
SET	SET_M			S
RST	RST_M			R

MELSEC Instruction				IEC Instruction in IEC Editor
in MELSEC Editor	in IEC Editor			
	Instruction List	Ladder Diagram		
PLS	PLS_M	—		R_TRIG ● <sup>1</sup>
PLF	PLF_M	—		R_TRIG ● <sup>1</sup>
FF	FF_M	—		—
CHK	CHK_M	—		—
DELTA	DELTA_M	—		—
SFT	SFT_M	—		SHL/SHR
MC	MC_M	—		—
MCR	MCR_M	—		—
FEND	FEND_M	—		● <sup>2</sup>
END	END_M	—		● <sup>2</sup>
STOP	STOP_M	—		—
NOP	—	—	—	—

<sup>1</sup> These are IEC function blocks.

<sup>2</sup> FEND and END are set automatically by the GX Developer and the GX IEC Developer.

## 5.1 Input Instructions

### 5.1.1 LD, LDI, AND, ANI, OR, ORI

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011				
Bit Devices								Word Devices (16-bit)						Constant	Pointer						Level			
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P	I
●	●	●	●	●	●	●	●	●													1	● <sup>1</sup>		

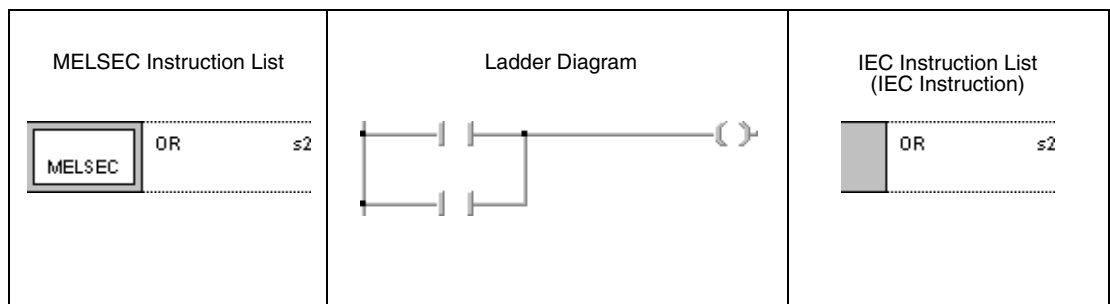
<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

**Devices  
MELSEC Q**

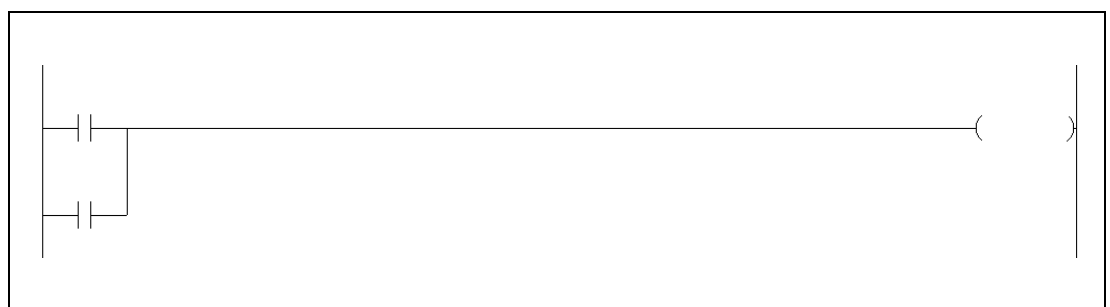
s	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other DX, BL
	Bit	Word		Bit	Word						
●	●	●	●	●	●	—	—	●	—	1 ● <sup>1</sup>	

<sup>1</sup> The number of steps varies:  
 - Using an internal device or using file registers R0 to R32767: 1 step  
 - Using direct access inputs (DX): 2 steps  
 - Using other devices: 3 steps

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	Devices used as connections.	bit



**Functions      Operation start****LD          Load (normally open contact)****LDI        Load inverse (normally closed contact)**

Every operation starts with an LD (**LoaD**) or an LDI (**LoaD Inverse**) instruction. The LD instruction specifies an NO contact (normally open) and the LDI instruction specifies an NC contact (normally closed). The device designated by the instruction is the input condition (operation result) for the following instruction.

**Series connection****AND        of NO contacts****ANI        of NC contacts**

Contacts are connected in series via an AND instruction as NO contact or via an ANI instruction as NC contact.

Both commands are logical connections and must not be programmed at the beginning of an operation.

**Parallel connection****OR         of NO contacts****ORI        of NC contacts**

Parallel connection of contacts is established via an OR instruction as NO contact or via an ORI instruction as NC contact. The device designated by the instruction sets the operation condition for the following instruction.

Both commands are logical connections and must not be programmed at the beginning of an operation.

**NOTE**

*The devices designated by the instructions can also be word devices. In this case, the condition of a specified bit is read as contact (Q series and System Q only).*

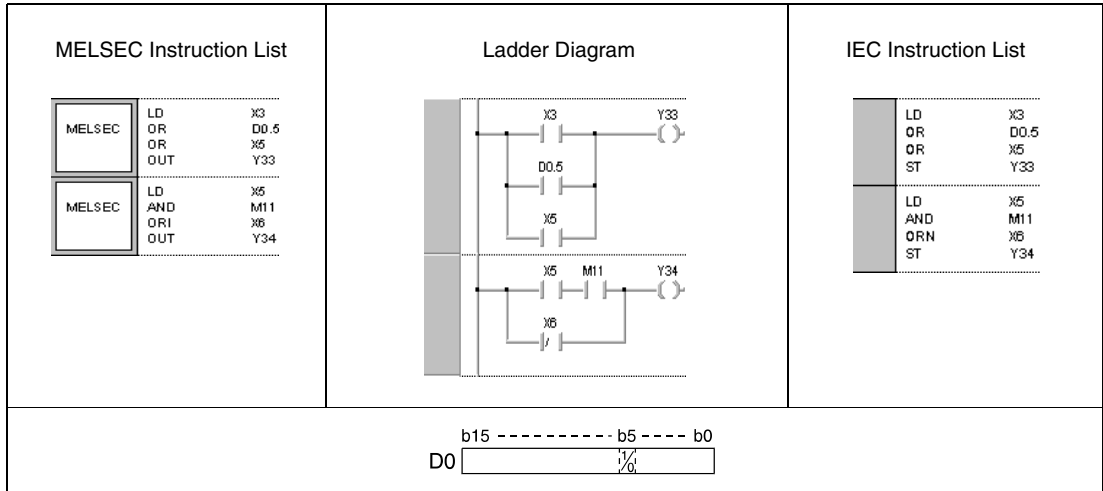
*Word devices are designated in hexadecimal code. Bit b11 in D0 for example is designated as D0.0B (Q series and System Q only).*

*For further information on addressing bits in word devices refer to chapter "Configuration of Instructions" (Q series and System Q only).*

Program Example 1

LD, AND, OR, ORI

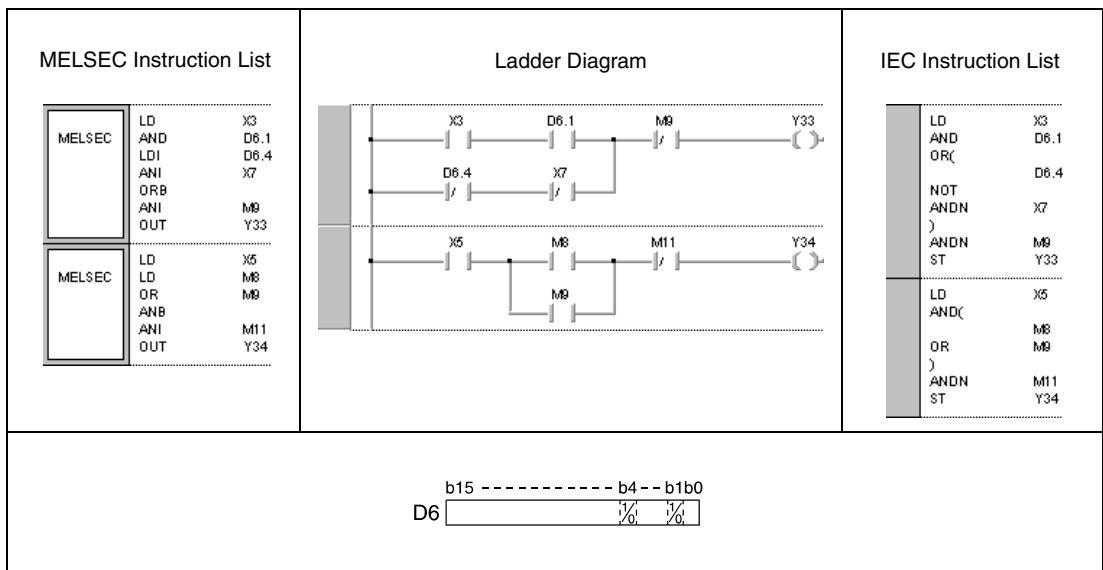
The following program shows series and parallel connections of contacts. Bit 5 (b5) in D0 is also read as contact.



Program Example 2

LD, LDI, AND, ANI, OR

The following program shows combined connections. Some contact points are connected via ORB and ANB instructions. Bits (b1 and b4) in D6 are read as contacts.



**Program** LD, AND, ANI

**Example 3**

The following program outputs operation results of devices at Y35 through Y37.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																																																																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 15%;">MELSEC</td> <td style="width: 15%;">LD</td> <td style="width: 15%;">X5</td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> </tr> <tr> <td></td> <td>OUT</td> <td>Y35</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>AND</td> <td>X8</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>OUT</td> <td>Y36</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>ANI</td> <td>X9</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>OUT</td> <td>Y37</td> <td></td> <td></td> <td></td> </tr> </table>	MELSEC	LD	X5					OUT	Y35					AND	X8					OUT	Y36					ANI	X9					OUT	Y37					<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"></td> <td style="width: 15%;">LD</td> <td style="width: 15%;">X5</td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> </tr> <tr> <td></td> <td>ST</td> <td>Y35</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>AND</td> <td>X8</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>ST</td> <td>Y36</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>ANDN</td> <td>X9</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>ST</td> <td>Y37</td> <td></td> <td></td> <td></td> </tr> </table>		LD	X5					ST	Y35					AND	X8					ST	Y36					ANDN	X9					ST	Y37			
MELSEC	LD	X5																																																																								
	OUT	Y35																																																																								
	AND	X8																																																																								
	OUT	Y36																																																																								
	ANI	X9																																																																								
	OUT	Y37																																																																								
	LD	X5																																																																								
	ST	Y35																																																																								
	AND	X8																																																																								
	ST	Y36																																																																								
	ANDN	X9																																																																								
	ST	Y37																																																																								

5.1.2 LDP, LDF, ANDP, ANDF, ORP, ORF

CPU

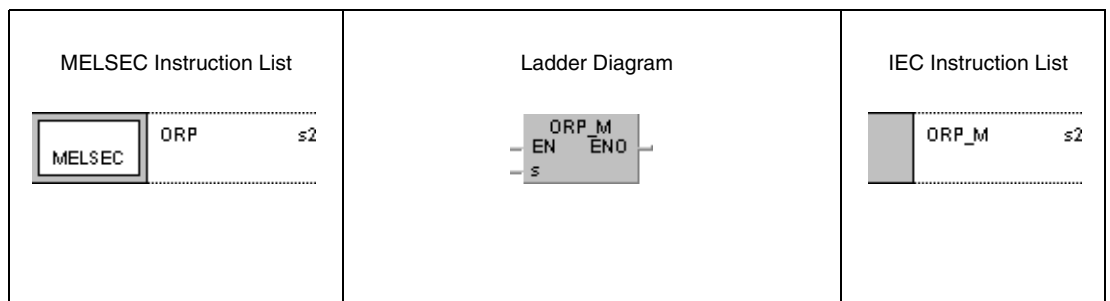
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

Devices  
MELSEC Q

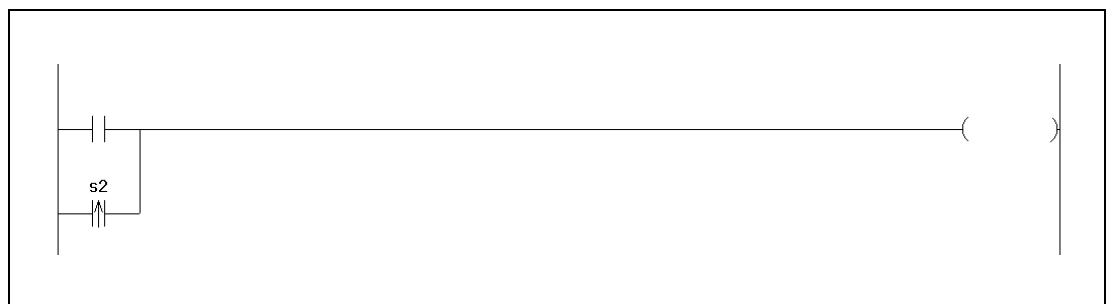
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other DX		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	—	—	●	—	2 <sup>1</sup> ●

- <sup>1</sup> The number of steps varies:
- Using an internal device or using file registers R0 to R32767: 2 steps
  - Using direct access inputs (DX): 3 steps
  - Using other devices: 4 steps

GX IEC Developer



GX Developer



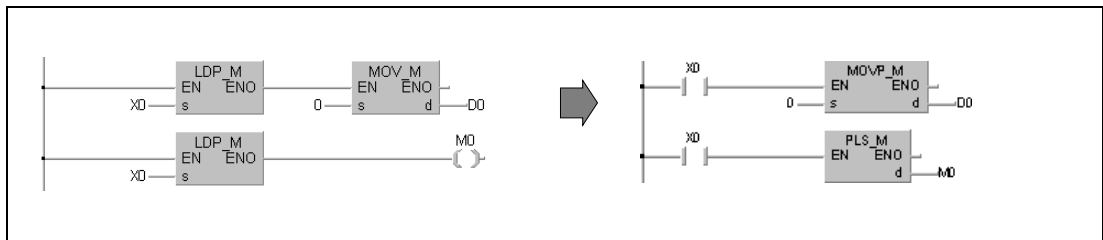
Variables

Set Data	Meaning	Data Type
s	Devices used as connections.	bit

- Functions**
- Pulse operation start**
  - LDP leading edge**
  - LDF trailing edge**

Similar to the LD and LDI instructions, these instructions designate contacts specified by bit or word devices. The result of the LDP instruction is 1, if the addressed bit of the device changes from 0 to 1 (leading edge). The result of the LDF instruction is 1, if the addressed bit of the device changes from 1 to 0 (trailing edge). As single instruction the LDP instruction executes the same function as a PLS instruction and with the input condition at leading edge generates a pulse output.

The program example on the left shows a ladder diagram applying an LDP instruction. The example on the right does not apply an LDP instruction.



**Pulse series connection**

- ANDP leading edge**
- ANDF trailing edge**

The ANDP instruction connects a contact in series with a contact specified by a bit or word device. This contact has the condition 1, if the addressed bit of a device changes from 0 to 1.

Using an ANDF instruction the specified contact has the condition 1, if the addressed bit of a device changes from 1 to 0.

**Pulse parallel connection**

- ORP leading edge**
- ORF trailing edge**

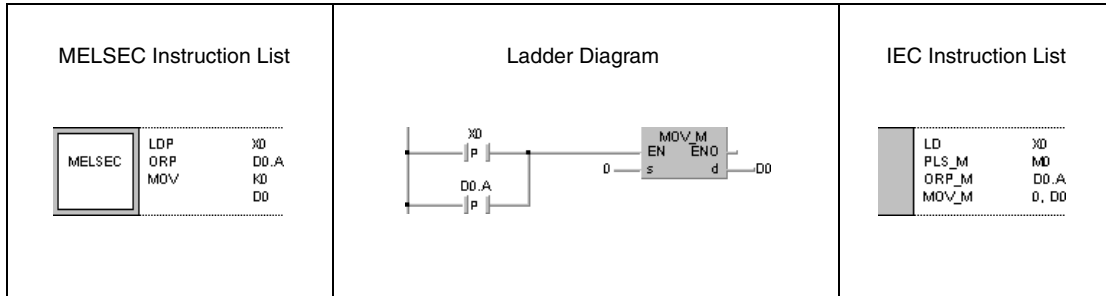
The ORP instruction connects a contact in parallel to a contact specified by a bit or word device. This contact has the condition 1, if the addressed bit of a device changes from 0 to 1.

Using an ORF instruction the specified contact has the condition 1, if the addressed bit of a device changes from 1 to 0.

Device specified by ANDP/ORP Instruction	Result of ANDP/ORP Instruction	Device specified by ANDF/ORF Instruction	Result of ANDF/ORF Instruction
Bit Device/Word Device		Bit Device/Word Device	
0 → 1	1	0 → 1	0
0		0	
1		1	
1 → 0		1 → 0	
	0		1

**NOTE** Word devices are designated in hexadecimal code. Bit b11 in D0 for example is designated as D0.0B.

**Program Example** ORP  
 With leading edge from X0 or by setting (leading edge) bit 10 (b10) in data register D0, the following program executes a MOV instruction.



## 5.2 Connection Instructions

### 5.2.1 ANB, ORB

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●



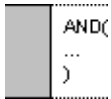
**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
												1										

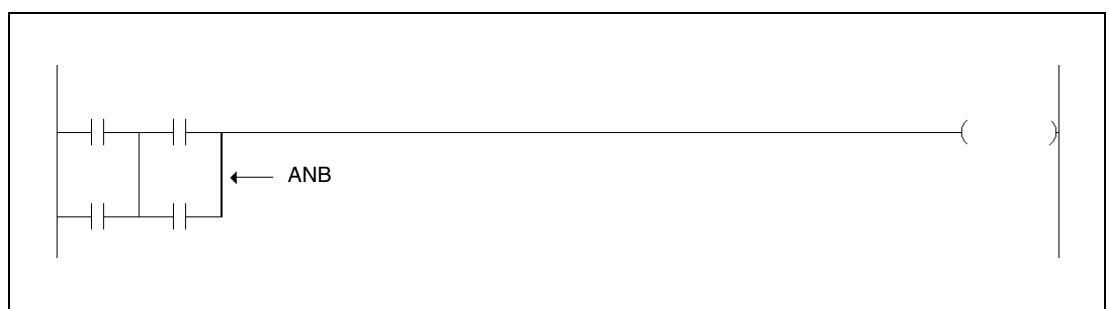
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List (IEC Instruction)</p> 
--	--	---

**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions Ladder block series connection**

**ANB Block series connection**

The ANB instruction (AND block) connects two or more parallel connection blocks in series and supplies an operation result for the following operations.

If more than two blocks are connected in series, after each parallel block an ANB instruction has to be programmed.

The ANB connection is an independent instruction and does not require any device.

Within one program the ANB instruction can be applied any number of times.

If more than two blocks are connected consecutively, the number of ANB instructions is limited to 15 (= 16 blocks) with a QnA, AnA, AnAS or AnU CPU and to 7 (= 8 blocks) with all other CPUs. Exceeding these limits results in malfunction.

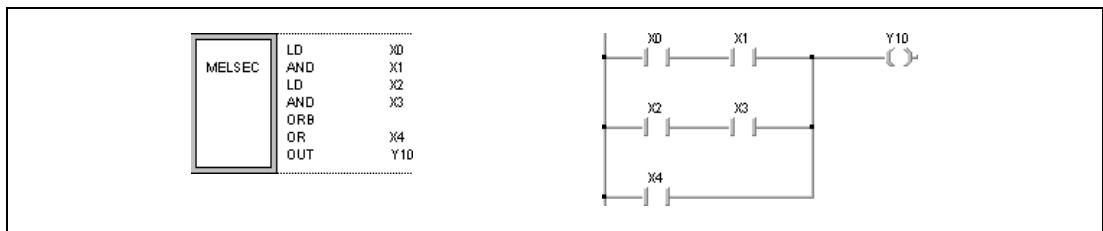
**Ladder block parallel connection**

**ORB Block parallel connection**

The ORB instruction (OR block) connects two or more series connection blocks in parallel and supplies an operation result for the following operations.

If more than two blocks are connected in parallel, after each series block an ORB instruction has to be programmed.

For block parallel connections designating one contact only an OR or ORI instruction has to be set.



The ORB connection is an independent instruction and does not require any device.

Within one program the ORB instruction can be applied any number of times.

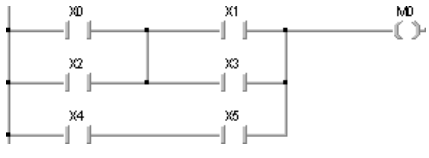
If more than two blocks are connected consecutively, the number of ANB instructions is limited to 15 (= 16 blocks) with a QnA, AnA, AnAS or AnU CPU and to 7 (= 8 blocks) with all other CPUs. Exceeding these limits results in malfunction.



**Program Example**

ANB, ORB

The following program connects the parallel connection block of X0 and X2 in series with the parallel connection block of X1 and X3. The result is connected in parallel with the series connection of X4 and X5.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																																																												
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">MELSEC</td><td></td><td></td></tr> <tr><td>LD</td><td>X0</td><td></td></tr> <tr><td>OR</td><td>X2</td><td></td></tr> <tr><td>LD</td><td>X1</td><td></td></tr> <tr><td>OR</td><td>X3</td><td></td></tr> <tr><td>ANB</td><td></td><td></td></tr> <tr><td>LD</td><td>X4</td><td></td></tr> <tr><td>AND</td><td>X5</td><td></td></tr> <tr><td>ORB</td><td></td><td></td></tr> <tr><td>OUT</td><td>M0</td><td></td></tr> </table>	MELSEC			LD	X0		OR	X2		LD	X1		OR	X3		ANB			LD	X4		AND	X5		ORB			OUT	M0			<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>LD</td><td>X0</td></tr> <tr><td>OR</td><td>X2</td></tr> <tr><td>AND(</td><td></td></tr> <tr><td></td><td>X1</td></tr> <tr><td></td><td>X3</td></tr> <tr><td>)</td><td></td></tr> <tr><td>OR(</td><td></td></tr> <tr><td></td><td>X4</td></tr> <tr><td></td><td>X5</td></tr> <tr><td>)</td><td></td></tr> <tr><td>AND</td><td></td></tr> <tr><td></td><td>X4</td></tr> <tr><td></td><td>X5</td></tr> <tr><td>)</td><td></td></tr> <tr><td>ST</td><td>M0</td></tr> </table>	LD	X0	OR	X2	AND(			X1		X3	)		OR(			X4		X5	)		AND			X4		X5	)		ST	M0
MELSEC																																																														
LD	X0																																																													
OR	X2																																																													
LD	X1																																																													
OR	X3																																																													
ANB																																																														
LD	X4																																																													
AND	X5																																																													
ORB																																																														
OUT	M0																																																													
LD	X0																																																													
OR	X2																																																													
AND(																																																														
	X1																																																													
	X3																																																													
)																																																														
OR(																																																														
	X4																																																													
	X5																																																													
)																																																														
AND																																																														
	X4																																																													
	X5																																																													
)																																																														
ST	M0																																																													

5.2.2 MPS, MRD, MPP

**NOTE** *These instructions should not be used within the IEC editors.*

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

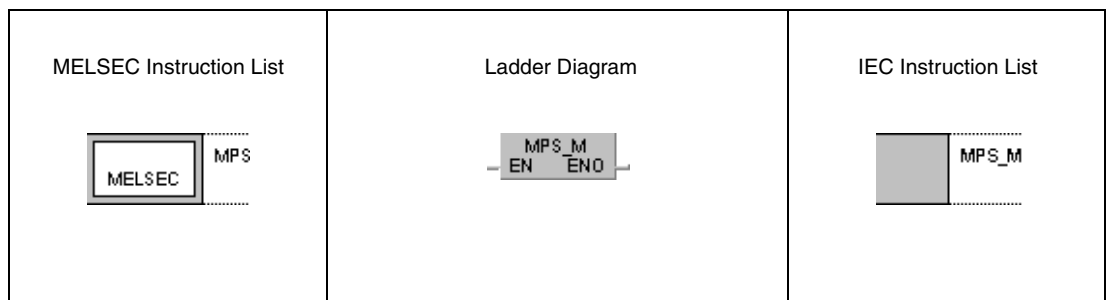
**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
																	1					

**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions    Operation result processing**

**MPS    Store operation result (memory push)**

The MPS instruction stores the operation result preceding the MPS instruction.

Using a QnA, AnA, AnAS or AnU CPU, up to 16 consecutive MPS instructions per network can be programmed. With all other CPUs this limit is 12 instructions. If an MPP instruction is set between two MPS instructions, this limit is reduced by one.

**MRD    Read operation result (memory read)**

The MRD instruction reads stored operation results via an MPS instruction. The following operation executed depends on the reading result.

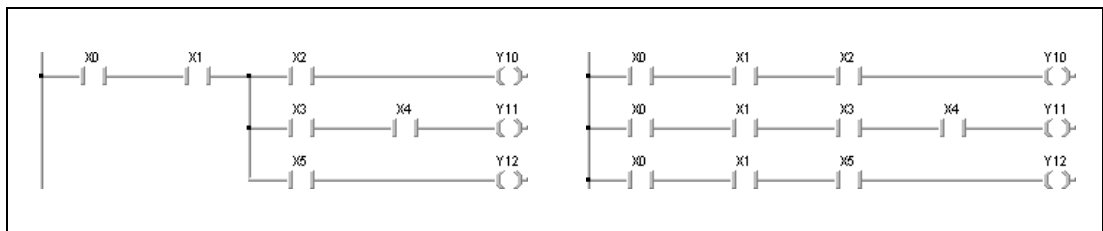
**MPP    Read and clear operation result (memory pop)**

The MRD instruction reads stored operation results via an MPS instruction. The following operation executed depends on the reading result. Then the result is cleared.

The MPS, MRP and MPP instructions are independent instructions and do not require any device.

In ladder programming mode the MPS, MRD and MPP instructions are not displayed explicitly. Whether connections are of the MPS, MRD or MPP type depends on the structure of the ladder diagram.

The example on the left shows a ladder diagram applying MPS, MRD or MPP instructions. The example on the right shows a ladder diagram without MPS, MRD or MPP instructions.



The number of MPS instructions in a program must equal the number of MPP instructions.

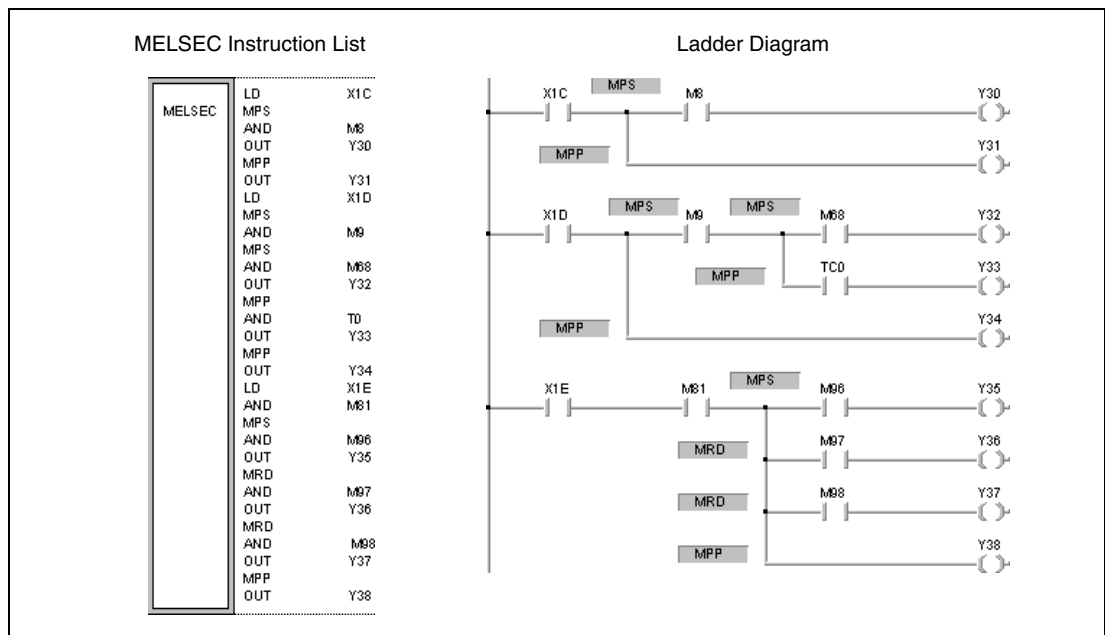
If the number of MPS instructions exceeds the number of MPP instruction a NOP instruction is set instead of the MPP instruction and the course of the program is changed accordingly.

If the number of MPP instructions exceeds the number of MPS instructions the logical sequence of the program is suspended. In this case, the program execution is not proceeded and the CPU returns an error message.

**Program Example 1**

MPS, MRD, MPP

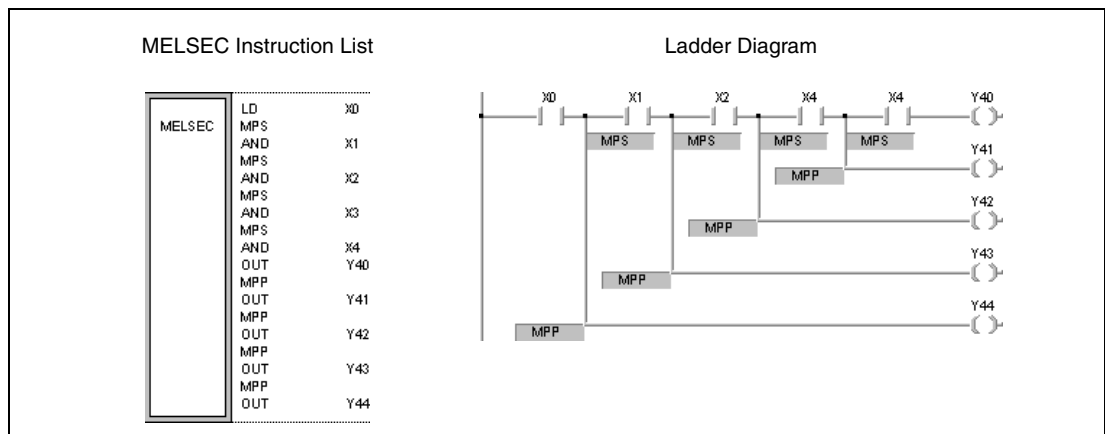
The following program illustrates the use of instructions for programming combined connections.



**Program Example 2**

MPS, MRD, MPP

The following program illustrates the programming of instructions that output interim results in a series connection.



5.2.3 INV

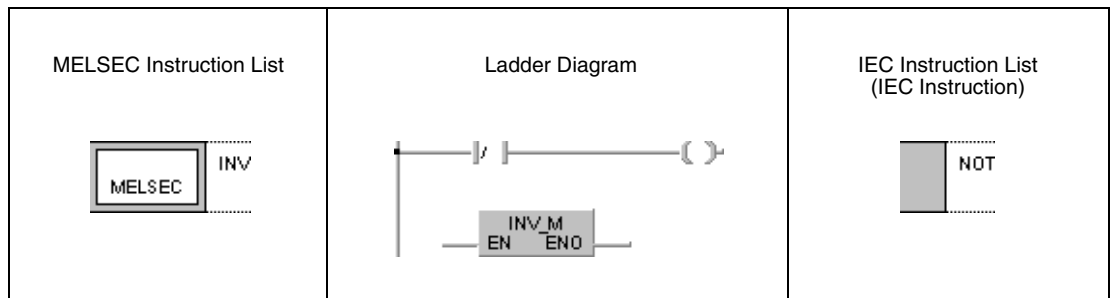
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

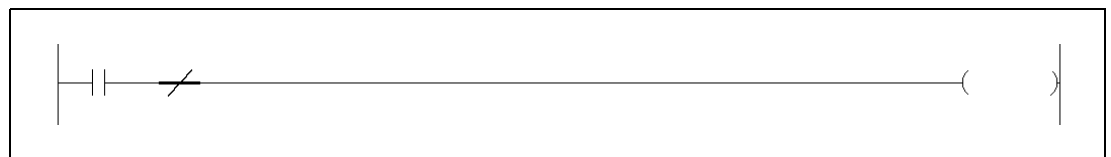
Devices  
MELSEC Q

Usable Devices									Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
Bit	Word		Bit	Word				U		
—	—	—	—	—	—	—	—	—	—	1

GX IEC Developer



GX Developer



Variables

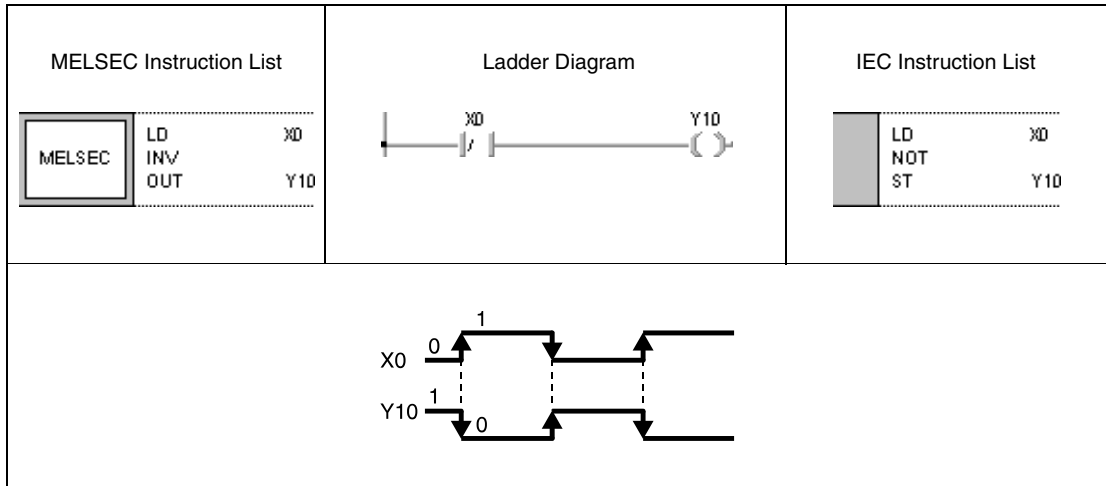
Set Data	Meaning	Data Type
—	—	—

**Functions**      **Operation result inversion**  
**INV**    **Inversion instruction**

The INV instruction inverts the operation result preceding the INV instruction.  
 If the result is 1 before the operation it will be 0 afterwards.  
 If the result is 0 before the operation it will be 1 afterwards.

**Program Example**

The following program inverts the status of X0 and outputs the inverted signal at Y10.



5.2.4 MEP, MEF

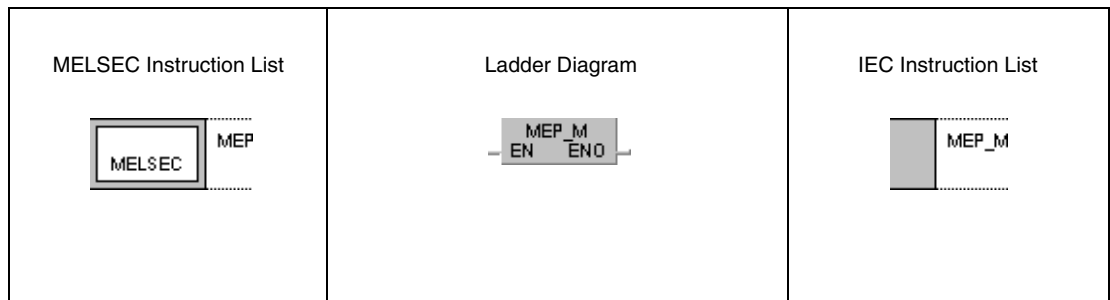
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

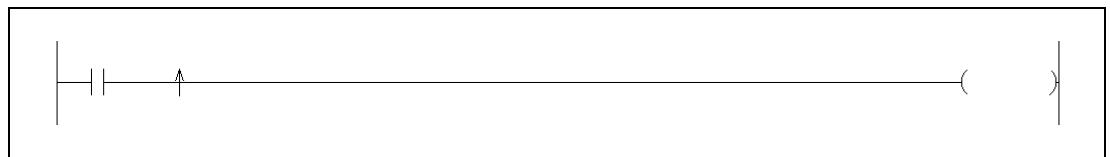
Devices  
MELSEC Q

Usable Devices									Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other U		
Bit	Word		Bit	Word						
—	—	—	—	—	—	—	—	—	—	1

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
—	—	—

**Functions**      **Operation result into pulse conversion**

**MEP Pulse generation at leading edge of operation result**

The MEP instruction is used in cases where the applied instructions cannot output operation results as specified pulse output. The MEP instruction is set after the according instruction and generates one output pulse, when the input signal changes from 0 to 1 (at leading edge). The next pulse is generated when the input is at leading edge once again.

**MEF Pulse generation at trailing edge of operation result**

The MEF instruction is used in cases where the applied instructions cannot output operation results as specified pulse output. The MEF instruction is set after the according instruction and generates one output pulse, when the input signal changes from 1 to 0 (at trailing edge). The next pulse is generated when the input is at trailing edge once again.

These two instructions are especially suitable for multiple contacts connections. For example, multiple NO contacts (normally open contacts) connected in series would maintain the operation result 1 if they were all closed. If a relay was set by this operation result, it could not be reset. With a MEP instruction connected in series with these NO contacts the relay could be reset because the instruction outputs one pulse only, if the series connection result of all contacts changes from 0 to 1.



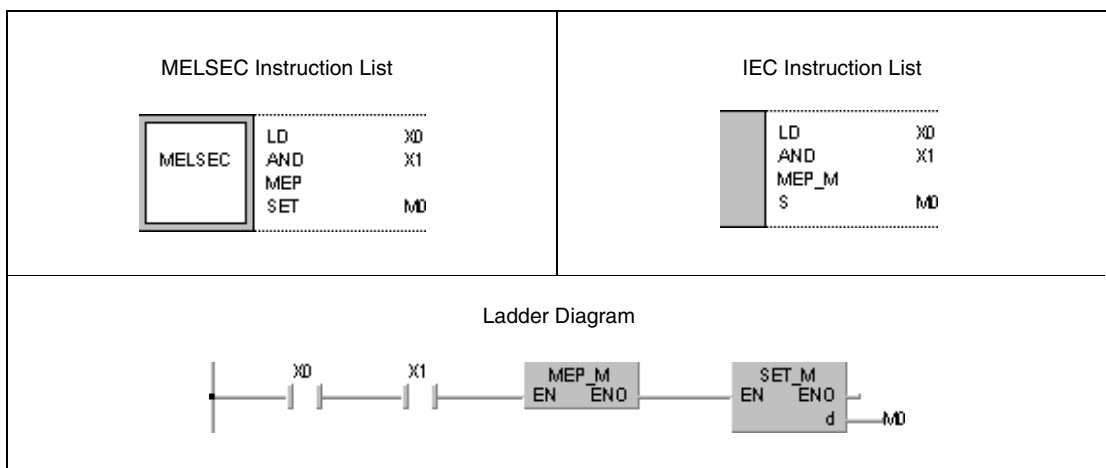
*The MEP and MEF instructions will occasionally not function properly when pulse conversion is applied to contacts that are indexed by a subroutine or by a FOR/NEXT instruction. In this case, the EGP/EGF instruction has to be applied.*

*The MEP/MEF instruction operates with the operation results immediately prior to the MEP and MEF instructions. For this reason, an AND instruction should be used at the same position. The MEP and MEF instructions cannot be used at the LD or OR position.*

**Program Example**

**MEP**

With leading edge from the series connection result at X0 and X1, the following program sets the relay M0.





### 5.2.5 EGP, EGF

**CPU**

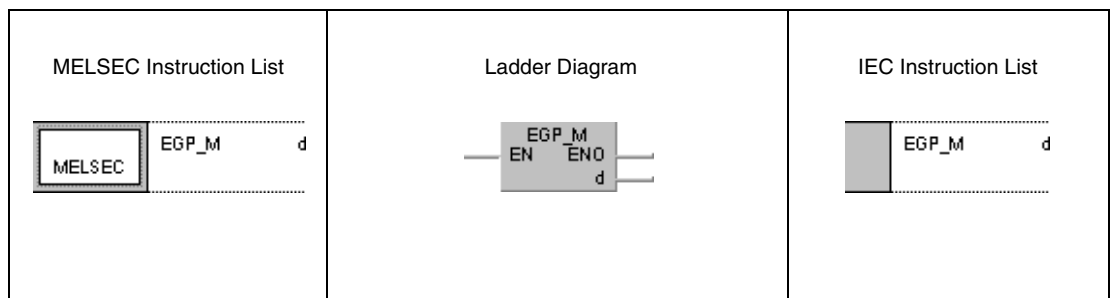
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

**Devices  
MELSEC Q**

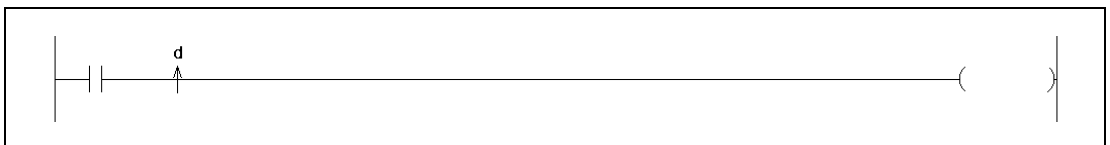
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other U
	Bit	Word		Bit	Word						
d	● <sup>1</sup>	—	—	—	—	—	—	—	—	1	

<sup>1</sup> V only

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
d	Edge relay, storing the operation result.	bit (V only)

**Functions**

**Setting of edge relays**

**EGP Setting an edge relay with leading edge of an operation result**

The EGP instruction sets the edge relay (V) depending on the operation result of the preceding instruction. If the result changes from 0 to 1, the edge relay is set. On all other conditions of the EGP instruction, for example, changing from 1 to 0 or remaining at condition 1 or 0 the edge relay is not set.

**EGF Setting an edge relay with trailing edge of an operation result**

The EGF instruction sets the edge relay (V) depending on the operation result of the preceding instruction. If the result changes from 1 to 0, the edge relay is set. On all other conditions of the EGF instruction, for example, changing from 0 to 1 or remaining at condition 0 or 1 the edge relay is not set.

The EGP and EGF instructions are applied in subroutines or programs placed within FOR/NEXT instructions and operating with addressing via index registers (index qualification).

The EGP and EGF instructions can be used like an AND instruction.

**Program Example**

EGP

The following program first resets the index register Z0 to 0 and then calls the subroutine UP1 (1). With leading edge X0Z0 is set to X0 and V0Z0 is set to V0. Further, D0Z0 is set to D0 and incremented by 1.

After returning, the index register Z0 stores 1, and the subroutine is called again (2). With leading edge from X1, V1 is set and D1 is incremented.

MELSEC Instruction List	IEC Instruction List																																																																								
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; width: 10%; text-align: center;">MELSEC</td> <td style="width: 10%; border: none;">LD</td> <td style="width: 10%; border: none;">SM400</td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">MOV</td> <td style="border: none;">K0</td> <td style="border: none;">Z0</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">CALL</td> <td style="border: none;">UP1</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">MOV</td> <td style="border: none;">K1</td> <td style="border: none;">Z0</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">CALL</td> <td style="border: none;">UP1</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">FEND</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> </table>	MELSEC	LD	SM400					MOV	K0	Z0				CALL	UP1					MOV	K1	Z0				CALL	UP1					FEND					<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%; border: none;">LD</td> <td style="width: 10%; border: none;">SM400</td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">MOVE_E</td> <td style="border: none;">0</td> <td style="border: none;">Z0</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">JMP</td> <td style="border: none;">UP1</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">LD</td> <td style="border: none;">SM400</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">MOVE_E</td> <td style="border: none;">1</td> <td style="border: none;">Z0</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">JMP</td> <td style="border: none;">UP1</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> </table>		LD	SM400					MOVE_E	0	Z0				JMP	UP1					LD	SM400					MOVE_E	1	Z0				JMP	UP1			
MELSEC	LD	SM400																																																																							
	MOV	K0	Z0																																																																						
	CALL	UP1																																																																							
	MOV	K1	Z0																																																																						
	CALL	UP1																																																																							
	FEND																																																																								
	LD	SM400																																																																							
	MOVE_E	0	Z0																																																																						
	JMP	UP1																																																																							
	LD	SM400																																																																							
	MOVE_E	1	Z0																																																																						
	JMP	UP1																																																																							
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%; border: none;">LD</td> <td style="width: 10%; border: none;">X0Z0</td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">EGP</td> <td style="border: none;">M</td> <td style="border: none;">V0Z0</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">INC</td> <td style="border: none;">M</td> <td style="border: none;">D0Z0</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">RET</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> </table>		LD	X0Z0					EGP	M	V0Z0				INC	M	D0Z0				RET					<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%; border: none;">LD</td> <td style="width: 10%; border: none;">X0Z0</td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> <td style="width: 10%; border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">EGP_M</td> <td style="border: none;">EN</td> <td style="border: none;">ENO</td> <td style="border: none;">d</td> <td style="border: none;">Z0</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">INC_M</td> <td style="border: none;">EN</td> <td style="border: none;">ENO</td> <td style="border: none;">d</td> <td style="border: none;">D0Z0</td> </tr> </table>		LD	X0Z0					EGP_M	EN	ENO	d	Z0		INC_M	EN	ENO	d	D0Z0																														
	LD	X0Z0																																																																							
	EGP	M	V0Z0																																																																						
	INC	M	D0Z0																																																																						
	RET																																																																								
	LD	X0Z0																																																																							
	EGP_M	EN	ENO	d	Z0																																																																				
	INC_M	EN	ENO	d	D0Z0																																																																				
<p>Ladder Diagram</p>																																																																									

### 5.3 Output Instructions

#### 5.3.1 OUT

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

d	Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011				
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level										
	X	Y	M	L	S	B	F	T	C	D	W	R				A0						A1	Z	V	K
●	●	●	●	●	●																	● <sup>1</sup>	● <sup>2</sup>		

<sup>1</sup> In general, 1 step. Exception: 3 steps for programming internal relays or annunciators as a device for the OUT instruction. Refer to section "Programming an AnA, AnAS and AnU CPU" in the Programming Manual for the according number of steps.

<sup>2</sup> Index qualification only supplied with AnA, AnAS or AnU CPUs.

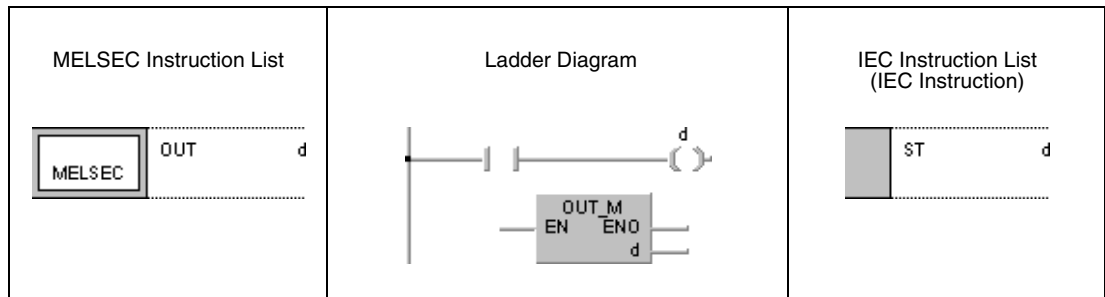
**Devices  
MELSEC Q**

d	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
● <sup>1</sup>	●	●	●	●	●	—	—	●	—	1 ● <sup>2</sup>	

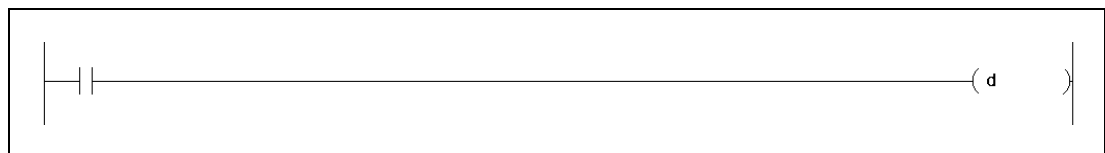
<sup>1</sup> Except T,C,F

<sup>2</sup> 1 step using internal devices, 2 steps using direct access outputs DY, 3 steps using any other devices (incl. serial number access file registers).

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
d	Number of device to be set (1) or reset (0).	bit

## Functions Output instruction

### OUT Setting instructions for outputs

An output is set depending on the preceding input condition.

Several OUT instructions can be programmed in parallel following an input condition.

The operation result of an OUT contact can be used as input condition for the following program steps as NO contact (normally open) or NC contact (normally closed).

Input Condition	OUT Instruction			If Bit of Word Device is designated
	Output Contact	Contact Type		Designated Bit
		NO Contact	NC Contact	
0	OFF	Non-continuity	Continuity	0
1	ON	Continuity	Non-continuity	1

**Operation Errors** See Programming Manual, part 1.

### Program Example 1

OUT

The following program shows the programming of an OUT instruction using bit devices as outputs (Y33 through Y35).

<p>MELSEC Instruction List</p> <table border="1"><tr><td>MELSEC</td><td>LD</td><td>X5</td></tr><tr><td></td><td>OUT</td><td>Y33</td></tr><tr><td></td><td>LD</td><td>X6</td></tr><tr><td></td><td>OUT</td><td>Y34</td></tr><tr><td></td><td>OUT</td><td>Y35</td></tr></table>	MELSEC	LD	X5		OUT	Y33		LD	X6		OUT	Y34		OUT	Y35	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1"><tr><td></td><td>LD</td><td>X5</td></tr><tr><td></td><td>ST</td><td>Y33</td></tr><tr><td></td><td>LD</td><td>X6</td></tr><tr><td></td><td>ST</td><td>Y34</td></tr><tr><td></td><td>ST</td><td>Y35</td></tr></table>		LD	X5		ST	Y33		LD	X6		ST	Y34		ST	Y35
MELSEC	LD	X5																														
	OUT	Y33																														
	LD	X6																														
	OUT	Y34																														
	OUT	Y35																														
	LD	X5																														
	ST	Y33																														
	LD	X6																														
	ST	Y34																														
	ST	Y35																														

### Program Example 2

OUT

The following program shows the programming of an OUT instruction using bits of the word device D0 as outputs (bits b5 through b7).

<p>MELSEC Instruction List</p> <table border="1"><tr><td>MELSEC</td><td>LD</td><td>X5</td></tr><tr><td></td><td>OUT</td><td>D0.5</td></tr><tr><td></td><td>LD</td><td>X6</td></tr><tr><td></td><td>OUT</td><td>D0.6</td></tr><tr><td></td><td>OUT</td><td>D0.7</td></tr></table>	MELSEC	LD	X5		OUT	D0.5		LD	X6		OUT	D0.6		OUT	D0.7	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1"><tr><td></td><td>LD</td><td>X5</td></tr><tr><td></td><td>ST</td><td>D0.5</td></tr><tr><td></td><td>LD</td><td>X6</td></tr><tr><td></td><td>ST</td><td>D0.6</td></tr><tr><td></td><td>ST</td><td>D0.7</td></tr></table>		LD	X5		ST	D0.5		LD	X6		ST	D0.6		ST	D0.7
MELSEC	LD	X5																														
	OUT	D0.5																														
	LD	X6																														
	OUT	D0.6																														
	OUT	D0.7																														
	LD	X5																														
	ST	D0.5																														
	LD	X6																														
	ST	D0.6																														
	ST	D0.7																														
<p>D0</p> <table border="1"><tr><td>b15</td><td>-----</td><td>b7</td><td>b6</td><td>b5</td><td>-----</td><td>b0</td></tr><tr><td></td><td></td><td>1</td><td>1</td><td>1</td><td></td><td></td></tr></table>			b15	-----	b7	b6	b5	-----	b0			1	1	1																		
b15	-----	b7	b6	b5	-----	b0																										
		1	1	1																												

5.3.2 OUT T, OUTH T

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag	
	Bit Devices								Word Devices (16-bit)													Constant
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P
d							●									●	●					
● <sup>1</sup>									● <sup>2</sup>							● <sup>2</sup>						

<sup>1</sup> Time setting

<sup>2</sup> Refer to section "Setting values of extension timers and counters" in the Programming Manual for an AnA, AnAS or AnU CPU.

Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□□		Special Function Module U□NG□	Index Register Zn	Constant K	Other		
	Bit	Word		Bit	Word						
d	● <sup>1</sup>	—	—	—	—	—	—	—	—	—	4
● <sup>2</sup>	—	● <sup>3</sup>	●	—	●	●	—	● <sup>4</sup>	—	—	4

<sup>1</sup> T only

<sup>2</sup> Time setting

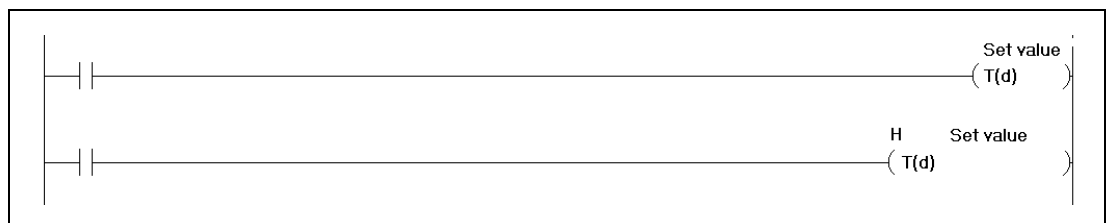
<sup>3</sup> Except T and C

<sup>4</sup> Specification of time settings by decimal constants (K) only. Hexadecimal constants cannot be read.

GX IEC  
Developer

MELSEC Instruction List	Ladder Diagram				
<table border="0" style="width: 100%;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding-left: 10px;">OUT      T(d)      (* Low Speed Timer *)                   Set Value</td> </tr> <tr> <td></td> <td style="padding-left: 10px;">OUTH      T(d)      (* High Speed Timer *)                   Set Value</td> </tr> </table>	MELSEC	OUT      T(d)      (* Low Speed Timer *) Set Value		OUTH      T(d)      (* High Speed Timer *) Set Value	
MELSEC	OUT      T(d)      (* Low Speed Timer *) Set Value				
	OUTH      T(d)      (* High Speed Timer *) Set Value				
IEC Instruction List					
<table border="0" style="width: 100%;"> <tr> <td style="border: 1px solid black; padding: 2px;">TIMER_M</td> <td style="padding-left: 10px;">TC(d) , Set Value      (* Low Speed Timer *)</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">TIMER_H_M</td> <td style="padding-left: 10px;">TC(d) , Set Value      (* High Speed Timer *)</td> </tr> </table>		TIMER_M	TC(d) , Set Value      (* Low Speed Timer *)	TIMER_H_M	TC(d) , Set Value      (* High Speed Timer *)
TIMER_M	TC(d) , Set Value      (* Low Speed Timer *)				
TIMER_H_M	TC(d) , Set Value      (* High Speed Timer *)				

GX  
Developer



**Variables**

Set Data	Meaning	Data Type
d	Number of timer.	bit
Set value	Time setting.	BIN 16 bit

**Functions**

**Setting timers**

**OUT T Low speed timer (100 ms)**

**OUTH T High speed timer (10 ms)**

If the input condition of an OUT(H) T instruction is set, the timer contact is being set (1) and remains set for a specified time. This time is designated directly by a constant or variably by the value in a data register.

The operation result of the OUT(H) T contact is programmed as input condition in one (or several) following program step(s) like a common NO (normally open) or NC (normally closed) contact.

After the specified time has passed (actual value = setting value) the succeeding input contact is set.

Several OUT(H) T instructions can be programmed succeeding one single input condition.

Timer as Output Contact			Timer as Input Condition			
Type	Contact Condition	Actual Value	Contact Condition before time setting passed		Contact Condition after time setting passed	
			NO contact	NC contact	NO contact	NC contact
100 ms	OFF	0	Non-continuity	Continuity	Non-continuity	Continuity
10 ms						
100 ms retentive	OFF	Actual value maintained	Non-continuity	Continuity	Continuity	Non-continuity
10 ms retentive						

The operation result of a retentive timer is maintained until it is reset via an RST instruction.

A timer cannot process negative time settings (-32768 to -1). A time setting of 0 would be processed as 1.

The execution of the OUT(H) T instruction performs as follows:

The timer coil designated by d is set or reset.

The according timer contact is set or reset.

The time settings are refreshed.

If a program jumps to an OUT(H) T instruction while it is executed, the contact conditions and timer settings are maintained.

If one instruction is executed repeatedly within one cycle, the value of the repetitions is refreshed.

Designation of counter coils and contacts via index registers (index qualification) can only be achieved with the index registers Z0 and Z1.

**NOTE**

*The register for the timer setting must not be designated indirectly!*

*Please refer to chapter A.3.4 for more informations about timers.*

**Program Example 1**

**OUT T**

10 seconds after setting X0, the following program sets the outputs Y10 and Y14. A low speed timer (100 ms) is used.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td rowspan="4" style="text-align: center; vertical-align: middle;">MELSEC</td><td>OUT</td><td>T1</td></tr> <tr><td></td><td>K100</td></tr> <tr><td>LD</td><td>T1</td></tr> <tr><td>OUT</td><td>Y10</td></tr> <tr><td></td><td></td><td>Y14</td></tr> </table>		LD	X0	MELSEC	OUT	T1		K100	LD	T1	OUT	Y10			Y14	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td rowspan="4" style="text-align: center; vertical-align: middle;">IEC</td><td>TIMER_M</td><td>TC1, 100</td></tr> <tr><td>LD</td><td>TS1</td></tr> <tr><td>ST</td><td>Y10</td></tr> <tr><td>ST</td><td>Y14</td></tr> </table>		LD	X0	IEC	TIMER_M	TC1, 100	LD	TS1	ST	Y10	ST	Y14
	LD	X0																											
MELSEC	OUT	T1																											
		K100																											
	LD	T1																											
	OUT	Y10																											
		Y14																											
	LD	X0																											
IEC	TIMER_M	TC1, 100																											
	LD	TS1																											
	ST	Y10																											
	ST	Y14																											

**Program Example 2**

**OUT T**

The following program reads the time setting via the inputs X10 to X1F in BCD data format. With leading edge from X0 BCD data is converted into BIN data first and stored in D10. After setting X2 the time setting is read. After the set time has passed Y15 is set. A low speed timer (100 ms) is used.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td rowspan="4" style="text-align: center; vertical-align: middle;">MELSEC</td><td>BINP</td><td>K4X10</td></tr> <tr><td></td><td>D10</td></tr> <tr><td>LD</td><td>X2</td></tr> <tr><td>OUT</td><td>T2</td></tr> <tr><td></td><td></td><td>D10</td></tr> <tr><td>LD</td><td>T2</td><td></td></tr> <tr><td>OUT</td><td>Y15</td><td></td></tr> </table>		LD	X0	MELSEC	BINP	K4X10		D10	LD	X2	OUT	T2			D10	LD	T2		OUT	Y15		<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td rowspan="4" style="text-align: center; vertical-align: middle;">IEC</td><td>BINP_M</td><td>K4X10, D10</td></tr> <tr><td>LD</td><td>X2</td></tr> <tr><td>TIMER_M</td><td>TC2, D10</td></tr> <tr><td>ST</td><td>Y15</td></tr> </table>		LD	X0	IEC	BINP_M	K4X10, D10	LD	X2	TIMER_M	TC2, D10	ST	Y15
	LD	X0																																	
MELSEC	BINP	K4X10																																	
		D10																																	
	LD	X2																																	
	OUT	T2																																	
		D10																																	
LD	T2																																		
OUT	Y15																																		
	LD	X0																																	
IEC	BINP_M	K4X10, D10																																	
	LD	X2																																	
	TIMER_M	TC2, D10																																	
	ST	Y15																																	

**Program Example 3**

**OUTH T**

250 ms after setting X10 the following program sets the output Y10. A high speed timer (10 ms) is used.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td rowspan="4" style="text-align: center; vertical-align: middle;">MELSEC</td><td>OUTH</td><td>T0</td></tr> <tr><td></td><td>K25</td></tr> <tr><td>LD</td><td>T0</td></tr> <tr><td>OUT</td><td>Y10</td></tr> </table>		LD	X0	MELSEC	OUTH	T0		K25	LD	T0	OUT	Y10	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;"></td><td style="width: 10%;">LD</td><td style="width: 10%;">X0</td></tr> <tr><td rowspan="3" style="text-align: center; vertical-align: middle;">IEC</td><td>TIMER_H_M</td><td>TC0, 25</td></tr> <tr><td>LD</td><td>T0</td></tr> <tr><td>ST</td><td>Y10</td></tr> </table>		LD	X0	IEC	TIMER_H_M	TC0, 25	LD	T0	ST	Y10
	LD	X0																						
MELSEC	OUTH	T0																						
		K25																						
	LD	T0																						
	OUT	Y10																						
	LD	X0																						
IEC	TIMER_H_M	TC0, 25																						
	LD	T0																						
	ST	Y10																						

5.3.3 OUT C

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Usable Devices															Digit designation	Number of steps	Index	Carry Flag	Error Flag								
	Bit Devices					Word Devices (16-bit)					Constant	Pointer	Level															
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z				V	K	H (16#)	P	I	N	M9012	M9010 M9011		
d								●									●	●								1		
● <sup>1</sup>									● <sup>2</sup>								● <sup>2</sup>	●										

<sup>1</sup> Count setting

<sup>2</sup> Refer to section "Setting values of extension timers and counters" in the Programming Manual for an AnA, AnAS or AnU CPU.

Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K	Other U		
	Bit	Word		Bit	Word						
d	● <sup>1</sup>	—	—	—	—	—	—	—	—	—	4
● <sup>2</sup>	—	● <sup>3</sup>	●	—	●	●	—	● <sup>4</sup>	—	—	4

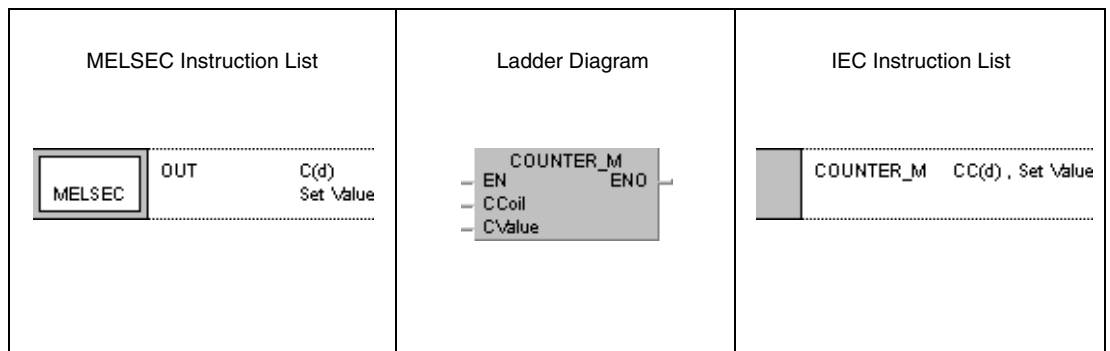
<sup>1</sup> C only

<sup>2</sup> Count setting

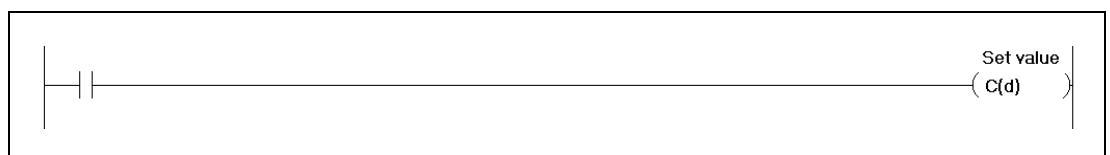
<sup>3</sup> Except T and C

<sup>4</sup> Specification of count settings by decimal constants (K) only. Hexadecimal constants cannot be read.

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
d	Number of counter.	bit
Set value	Count setting.	BIN 16-bit



**Functions**     **Setting counters**  
**OUT C Counter**

If the input condition for an OUT C instruction is set, the actual value of the counter is increased by 1.

The operation result of the OUT C contact is programmed as input condition in one (or several) following program step(s) like a common NO (normally open) or NC (normally closed) contact.

After the counter has reached the setting value the succeeding input contact is set.

If the input condition of the OUT C instruction remains set, the counting operation is not proceeded. Therefore, the counter does not require pulse input.

After completion of the counter operation the count setting and operation result can only be reset via an RST instruction.

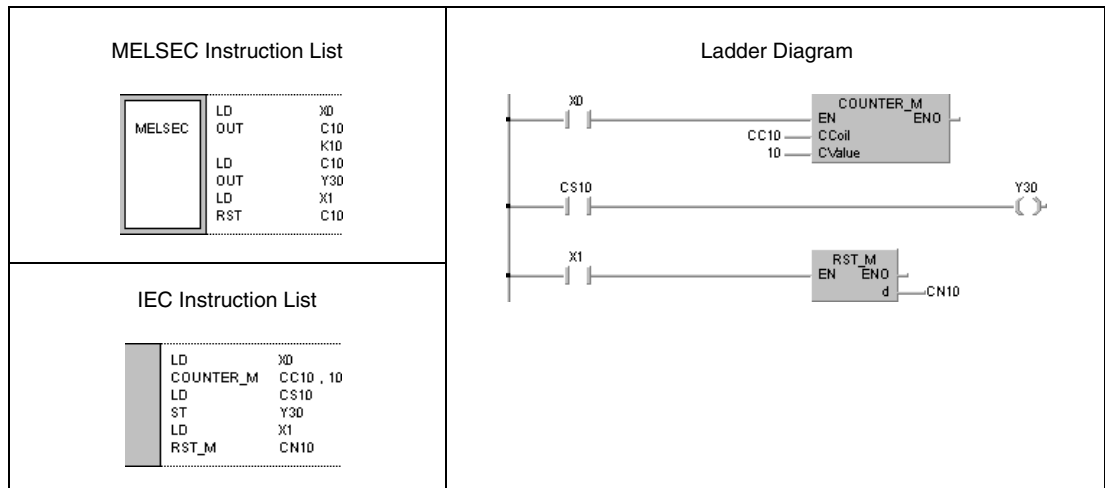
If the extension counters C256 to C1023 are used with an AnA, AnAS or AnU CPU, refer to the section "Setting values of extension timers and counters" in this Programming Manual.

A counter cannot process negative count settings (-32768 to -1). A count setting of 0 would be processed as 1.

Designation of counter coils and contacts via index registers (index qualification) can only be achieved with the index registers Z0 and Z1.

**NOTE**     *The register for the count setting must not be designated indirectly!*  
*Please refer to chapter A.3.5 of this manual for more information about counters.*

**Program Example 1**     **OUT C**  
 After X0 has been set for 10 times, the following program sets Y30 and if X1 is set resets Y30.



**Program Example 2**

OUT C

The following program sets the setting value in C10 to 10 (D0 =10) with leading edge from X0, and to 20 (D0 =20) with leading edge from X1. If X3 is set, the counter starts counting and sets Y30 when it reaches the setting value in D0.

MELSEC Instruction List	Ladder Diagram																																																																																																																																		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 10%;">MELSEC</td> <td style="width: 10%; border-top: 1px dotted black;">LD</td> <td style="width: 10%;">X0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td></td> <td>ANI</td> <td>X1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>MOV_P</td> <td>K10</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td>D0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>X1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>ANI</td> <td>X0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>MOV_P</td> <td>K20</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td>D0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>X3</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>OUT</td> <td>C10</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td>D0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>C10</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>OUT</td> <td>Y30</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>	MELSEC	LD	X0									ANI	X1									MOV_P	K10										D0									LD	X1									ANI	X0									MOV_P	K20										D0									LD	X3									OUT	C10										D0									LD	C10									OUT	Y30								
MELSEC	LD	X0																																																																																																																																	
	ANI	X1																																																																																																																																	
	MOV_P	K10																																																																																																																																	
		D0																																																																																																																																	
	LD	X1																																																																																																																																	
	ANI	X0																																																																																																																																	
	MOV_P	K20																																																																																																																																	
		D0																																																																																																																																	
	LD	X3																																																																																																																																	
	OUT	C10																																																																																																																																	
		D0																																																																																																																																	
	LD	C10																																																																																																																																	
	OUT	Y30																																																																																																																																	
<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%; border-top: 1px dotted black;">LD</td> <td style="width: 10%;">X0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td></td> <td>ANDN</td> <td>X1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>MOV_P_M</td> <td>10 ,</td> <td>D0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>X1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>ANDN</td> <td>X0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>MOV_P_M</td> <td>20 ,</td> <td>D0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>X3</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>COUNTER_M</td> <td>CC10 ,</td> <td>D0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>CS10</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>ST</td> <td>Y30</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>		LD	X0									ANDN	X1									MOV_P_M	10 ,	D0								LD	X1									ANDN	X0									MOV_P_M	20 ,	D0								LD	X3									COUNTER_M	CC10 ,	D0								LD	CS10									ST	Y30																																						
	LD	X0																																																																																																																																	
	ANDN	X1																																																																																																																																	
	MOV_P_M	10 ,	D0																																																																																																																																
	LD	X1																																																																																																																																	
	ANDN	X0																																																																																																																																	
	MOV_P_M	20 ,	D0																																																																																																																																
	LD	X3																																																																																																																																	
	COUNTER_M	CC10 ,	D0																																																																																																																																
	LD	CS10																																																																																																																																	
	ST	Y30																																																																																																																																	

5.3.4 OUT F

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

d	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag				
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level										
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N	M9012
						●																● <sup>1</sup>	● <sup>2</sup>		

<sup>1</sup> In general, 1 step. Exception: 3 steps for programming internal relays or annunciators as device for the OUT instruction. Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

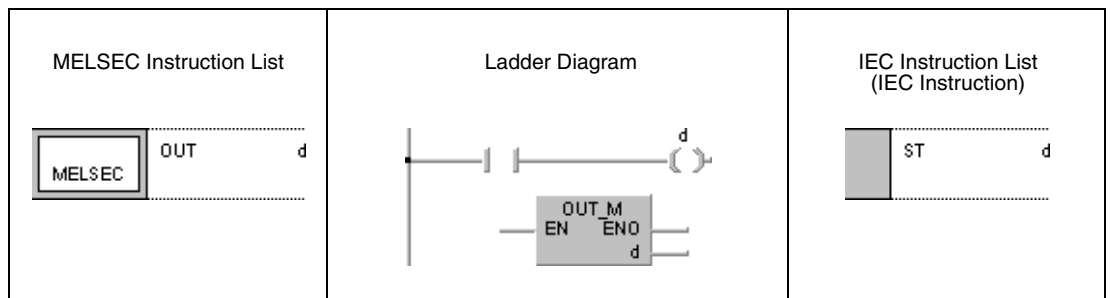
<sup>2</sup> Index qualification only supplied with AnA, AnAS or AnU CPUs.

Devices  
MELSEC Q

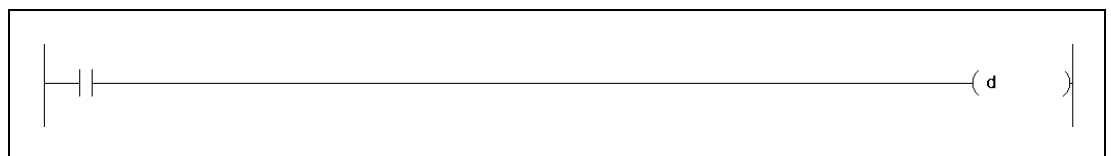
d	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
	● <sup>1</sup>	—	—	—	—	—	—	—	—	—	4

<sup>1</sup> F only

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
d	Number of annunciator to be set.	bit (F only)

**Functions     Output of annunciators****OUT F    Annunciator (Q series and System Q)**

If the input condition of an OUT F instruction is set, the annunciator is set and the following operations are performed:

The number of annunciator is displayed on the LED display of the CPU (Q3A and Q4AR), and the "USER" LED lights up.

The numbers of set annunciators are stored in the special registers SD64 through SD79.

The value in SD63 is incremented by 1.

If special register SD63 stores the value 16, i.e. 16 numbers of set annunciators are stored, no further numbers are stored in the range of SD64 through SD79.

If an annunciator is reset via an OUT instruction, the reading on the LED display, the condition of the "USER" LED, and the content of the special registers SD63 through SD79 are maintained.

Annunciators, registers, and displays are cleared via the RST F instruction.

**OUT F    Annunciator (A series)**

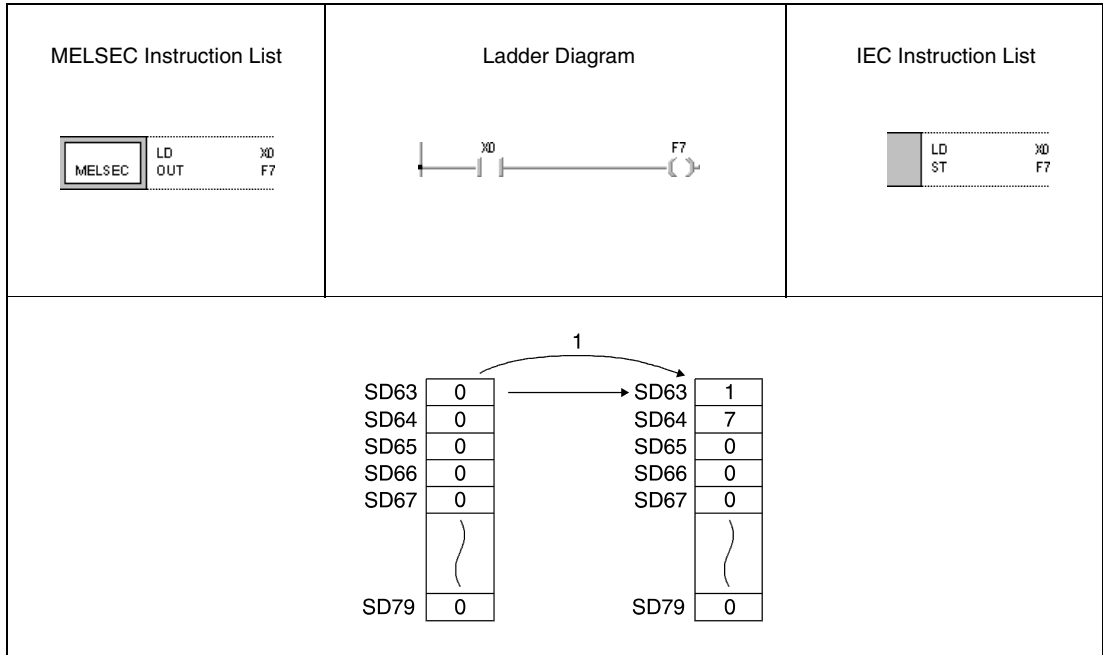
If a program sets an annunciator (F), the ERROR LED and the according LED displays on the CPU module light up. The number of set annunciators is stored in a special register. Refer to the Programming Manual, part 1 for further details.

Annunciators must not be set via an OUT instruction, because in that case the LED error display does not correspond to the contact condition of the output instruction. To avoid this, an annunciator should be set via the SET instruction. Setting an annunciator via an OUT instruction also leads to a reset of the annunciator if the input condition is reset. The LED displays the condition of the ERROR LED, and the content of the special registers are maintained.

**Program Example (Q series)**

**OUT F**

If X0 is set, the following program sets the annunciator F7. The number 7 is stored in the registers SD64 through SD79. The value in register SD63 is incremented by 1 (i.e. 1 number of annunciator stored).



<sup>1</sup> X0 is set

5.3.5 SET

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

d	Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011					
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level											
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P	I	N
	●	●	●	●	●																	1	1	2		

<sup>1</sup> The number of steps is 3, if internal relays, link relays, or annunciators (M, B, F) are set via the SET instruction, or if an internal relay or any word device is reset.

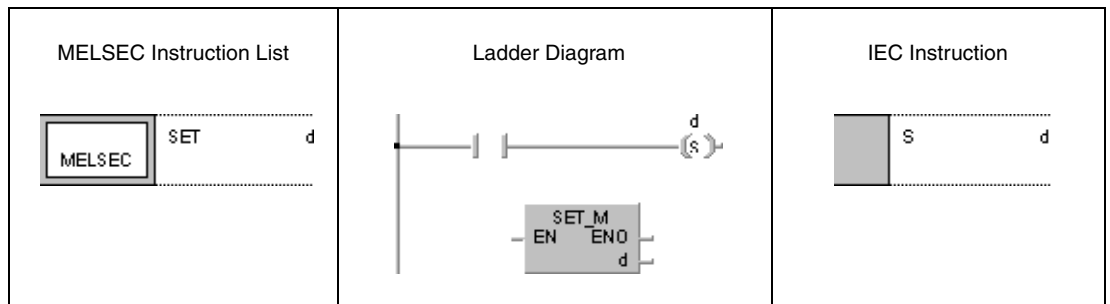
<sup>2</sup> Index qualification only supplied with AnA, AnAS, or AnU CPUs.

Devices  
MELSEC Q

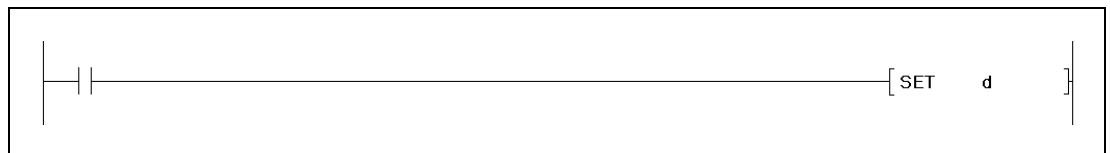
d	Usable Devices										Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
	Bit	Word		Bit	Word				BL	DY		
	●	●	●	●	●	●	—	—	●	●	—	1

<sup>1</sup> 1 step using internal devices, 2 steps using direct access outputs DY or SFC blocks (BL), 3 steps using any other devices (incl. serial number access file registers), 4 steps using timers (T) or counters (C).

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
d	Number of bit device (output contact) to be set / word device bit designation.	bit

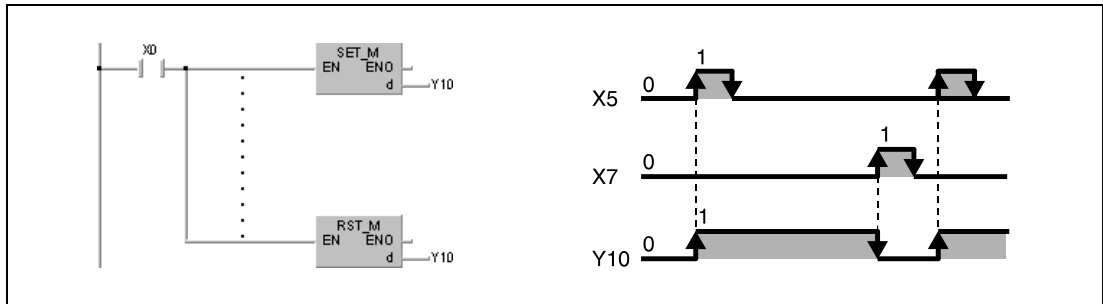
**Functions Setting of devices**

**SET Set instruction**

The SET instruction consists of a SET command followed by a number (address) of device d to be set.

After execution of the input condition the SET instruction and the number of device d are set or the designated bit of a word device is set to 1.

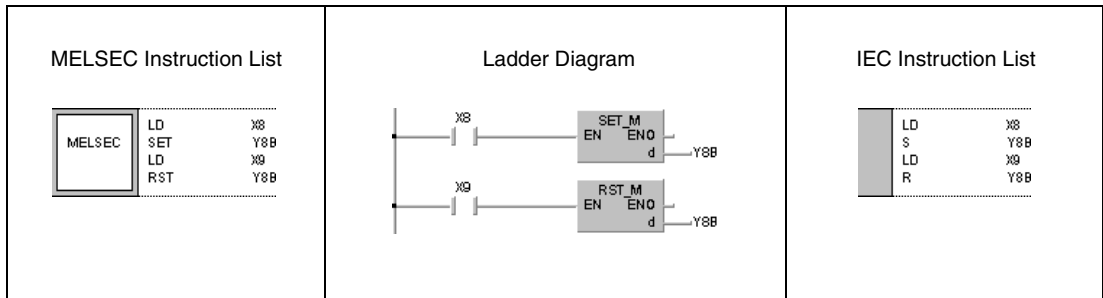
If the input condition is reset once again, the set device remains set. A device can be reset via the RST instruction.



**Program Example 1**

**SET**

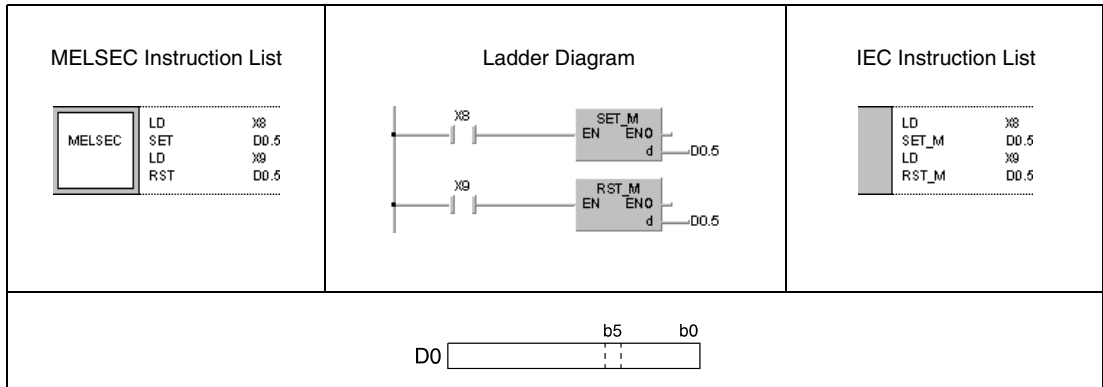
If X8 is set, the following program sets the output Y8B. If X9 is set, Y8B is reset.



**Program Example 2**

**SET**

If X8 is set, the following program sets bit 5 (b5) in D0 from 0 to 1. If X9 is set, this bit is reset.



5.3.6 RST

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag				
	Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level									
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I	N	M9012	M9010 M9011
d	●	●	●	●	●		●	●	●	●	●	●	●	●	●								1 ● <sup>1</sup>	● <sup>2</sup>		

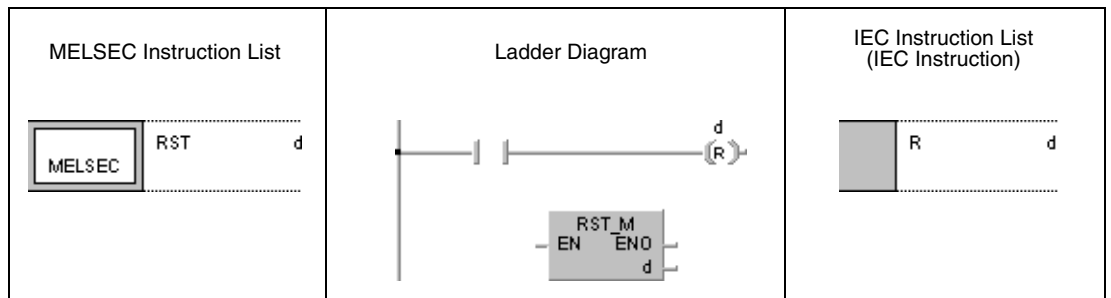
<sup>1</sup> The number of steps is 3, if internal relays, link relays, or annunciators (M, B, F) are set via the SET instruction, or if an internal relay or any word device is reset.

<sup>2</sup> Index qualification only supplied with AnA, AnAS, or AnU CPUs.

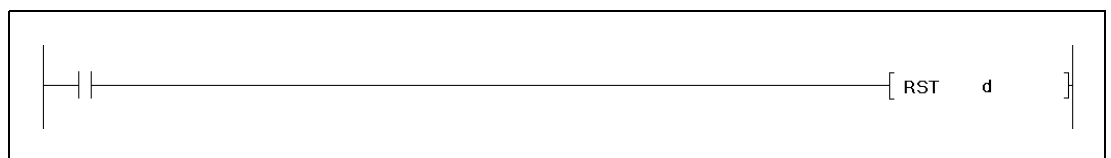
Devices  
MELSEC Q

	Usable Devices										Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
	Bit	Word		Bit	Word				BL	DY		
d	●	●	●	●	●	●	●	—	—	●	—	2

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
d	Number of bit device (output contact) to be reset / word device bit designation.	bit ● <sup>1</sup>

<sup>1</sup> A special function of the RST\_M instruction is the capability to reset entire word devices. Thus, less steps are required than using a MOV instruction with the constant K0.



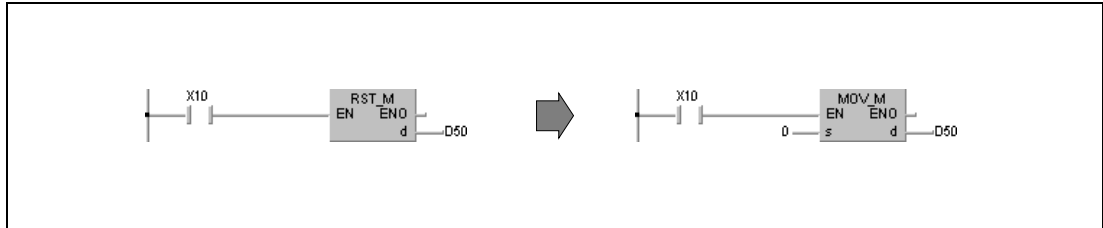
**Functions     Resetting devices**

**RST     Reset instruction**

The RST instruction consists of an RST command followed by a number (address) of device d to be reset.

After execution of the RST instruction input and output contacts of bit devices are switched off (0), actual values of timers and counters (T, C) are reset to 0 and the according contacts are switched off, the designated bit of a word device is reset to 0, and the content of word devices is reset to 0.

In the following diagram the function of the RST instruction is identical to that of the MOV instruction on the right. X10 serves as RST input.



**Program  
Example 1**

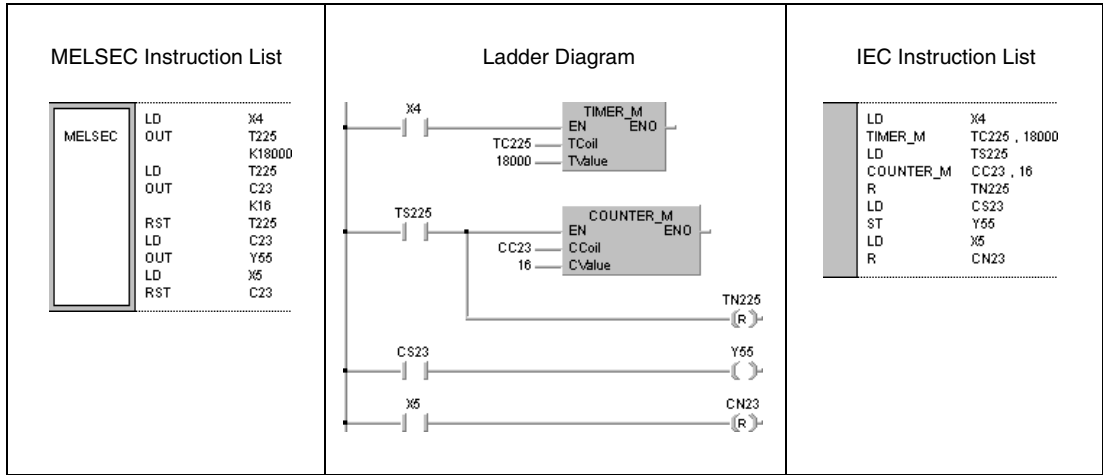
**RST**

With leading edge from X0, the following program stores the content at X10 through X1F in the data register D8. If X5 is set, the content of D8 is reset to 0.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; border: 2px solid black;">MELSEC</td> <td style="border: none;">LD     X0</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">MOV    K4X10</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">D8</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">LD     X5</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">RST    D8</td> </tr> </table>	MELSEC	LD     X0		MOV    K4X10		D8		LD     X5		RST    D8		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: none;"></td> <td style="border: none;">LD     X0</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">MOVE_E    K4X10 , D8</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">LD     X5</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">RST_M     D8</td> </tr> </table>		LD     X0		MOVE_E    K4X10 , D8		LD     X5		RST_M     D8
MELSEC	LD     X0																			
	MOV    K4X10																			
	D8																			
	LD     X5																			
	RST    D8																			
	LD     X0																			
	MOVE_E    K4X10 , D8																			
	LD     X5																			
	RST_M     D8																			

**Program Example 2** RST T, C

The following program illustrates resetting of retentive timers and counters. In the first program step T225 is set, if X4 has been set for 30 minutes (18000 seconds). In the second program step C23 counts the number of times T225 is set. If this timer is set for 16 times (setting value of C23 = 16) the output Y55 is set. If X5 is set, the counter will be reset to 0.



5.3.7 SET F, RST F

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

d	Usable Devices																	Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011		
	Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level							
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K						H (16#)	P
						●															3	● <sup>1</sup>		

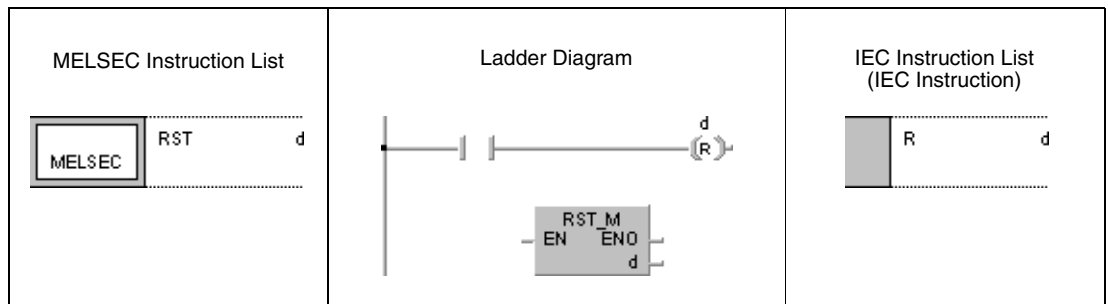
<sup>1</sup> Index qualification only supplied with AnA, AnAS, or AnU CPUs.

Devices  
MELSEC Q

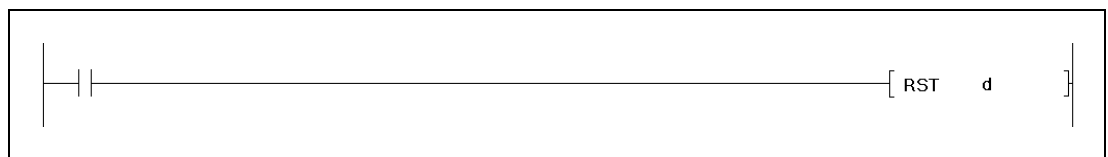
d	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						U
	● <sup>1</sup>	—	—	—	—	—	—	—	—	3	

<sup>1</sup> F only.

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
d (SET)	Number of annunciator to be set.	bit (F only)
d (RST)	Number of annunciator to be reset.	bit (F only)

**Functions     Setting and resetting of annunciators (Q series and System Q)**

**SET F    Set instruction**

The SET F instruction consists of a SET command followed by a number (address) of device d to be set. After execution of the input condition, the SET instruction and the designated device number d are set. The SET instruction outputs a pulse to set an annunciator.

The following procedures are executed:

The number (address) of the annunciator is displayed on the LED display of the CPU (Q3A and Q4AR), and the "USER" LED lights up.

The numbers (addresses) of set annunciators are stored in the registers SD64 through SD79.

The value in SD63 is incremented by 1.

If special register SD63 stores the value 16, i.e. 16 numbers of set annunciators are stored, no further numbers are stored in the range of SD64 through SD79.

**RST F    Reset instruction**

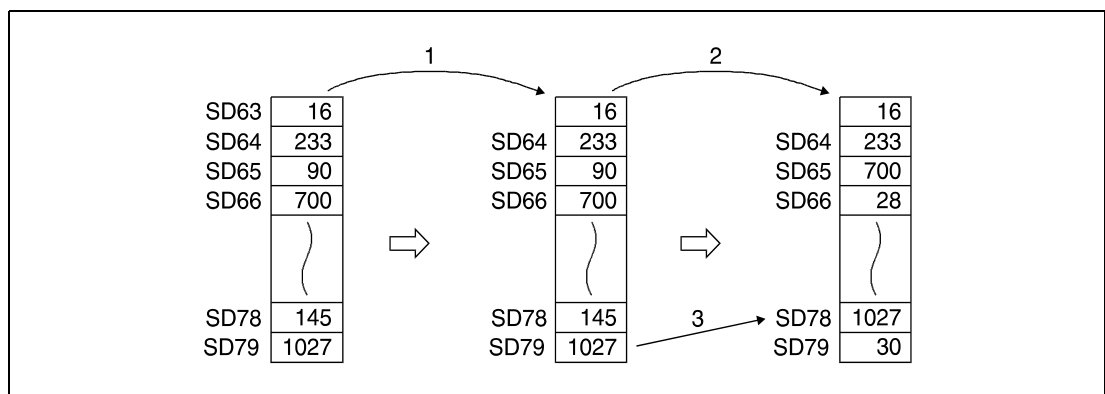
The RST F instruction consists of an RST command followed by a number (address) of device d to be reset.

After execution of the input condition the RST instruction is set and the designated device number is reset. The output signal resetting an annunciator is a pulse.

The number of a reset annunciator is cleared from the registers SD64 through SD79 and the value in register SD63 is decremented by 1. If the value in the register SD63 was 16 and annunciators are cleared from this register via an RST F instruction then those annunciator numbers are stored that could not be stored before. These annunciator numbers are stored in the cleared registers within SD64 through SD79.

If the value in special register SD63 is decremented to 0 and all annunciators are reset, LED display and "USER" LED turn off.

In the diagram below F30 is set in a first step (1) but cannot be registered because there are 16 numbers already stored. In a second step (2) F90 is reset. Thus, in a third step (3) F30 can be stored in SD79 because the other stored annunciators are shifted up by one cleared register (SD65).



**Setting and resetting of annunciators (A series)**

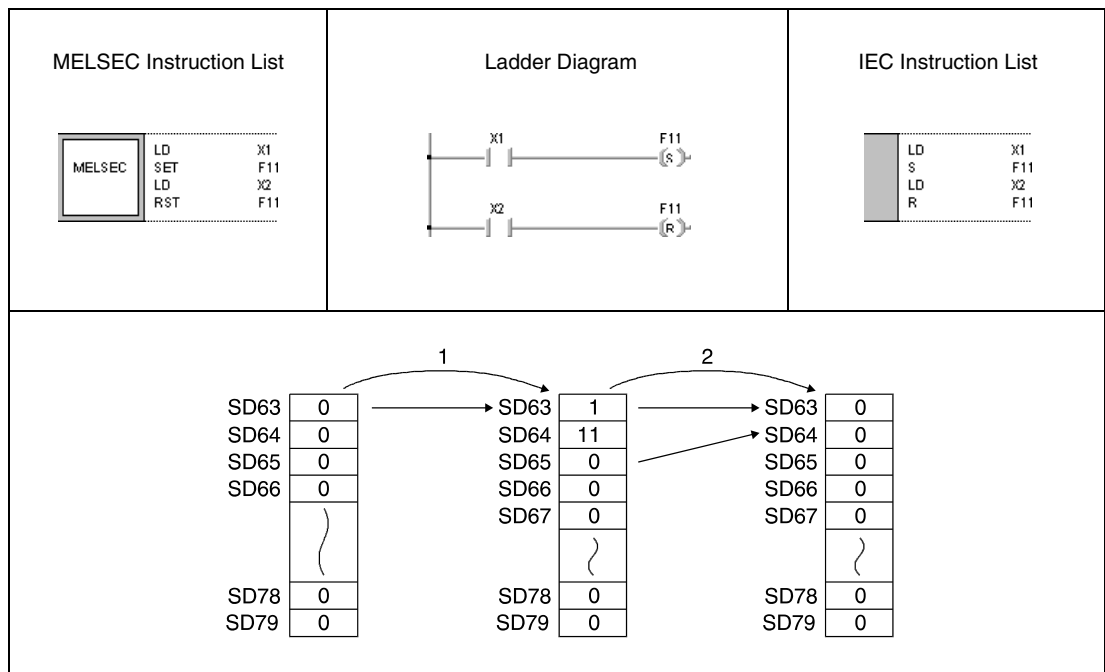
**SET F/ RST F Set / reset instruction**

If an annunciator F is set or reset via the SET/RST instruction, the according LED displays, the condition of the error LED on the CPU, and the content of the according special register change. Annunciators are set or reset by pulse signals.

**Program Example**

**SET F/ RST F (Q series and System Q)**

If X1 is set, the following program sets the annunciator F11. The number 11 is stored in the registers SD64 through SD79 and the value in SD63 is incremented by 1 (1). Then, if X2 is set, the annunciator F11 is reset. The number 11 is cleared from the special registers SD64 through SD79 and the value in SD63 is decremented by 1 (2).



5.3.8 PLS, PLF

CPU

AnS	AnN	AnA (S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag					
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level											
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N	M9012	M9010 M9011
d	●	●	●	●	●	●																	3 <sup>1</sup>	2 <sup>2</sup>		

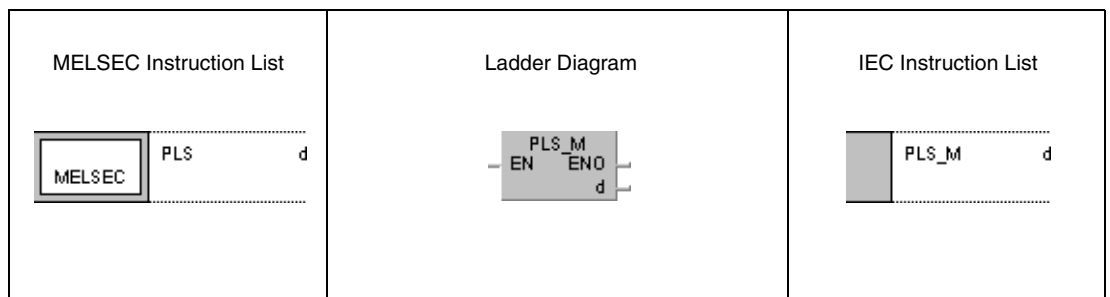
<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

<sup>2</sup> Index qualification only supplied with AnA, AnAS, or AnU CPUs.

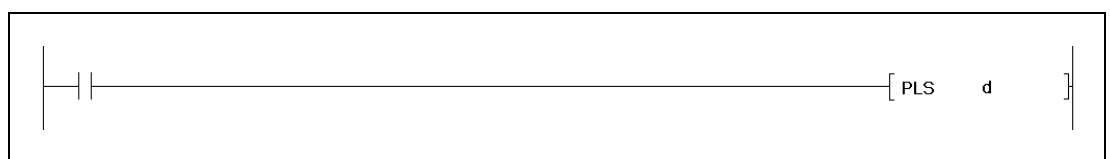
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
d	●	●	●	●	●	●	—	—	●	—	2

GX IEC Developer



GX Developer



Variables

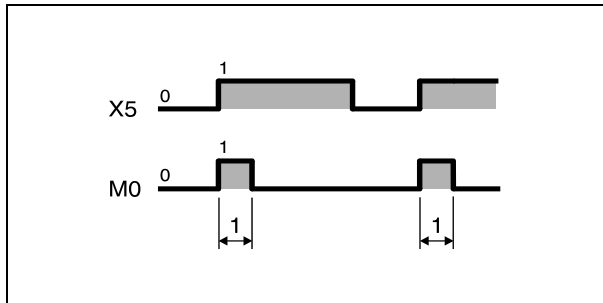
Set Data	Meaning	Data Type
d	Device of which the output signal is converted into a pulse.	bit

**Functions**    **Leading edge and trailing edge output**

**PLS    Output at leading edge**

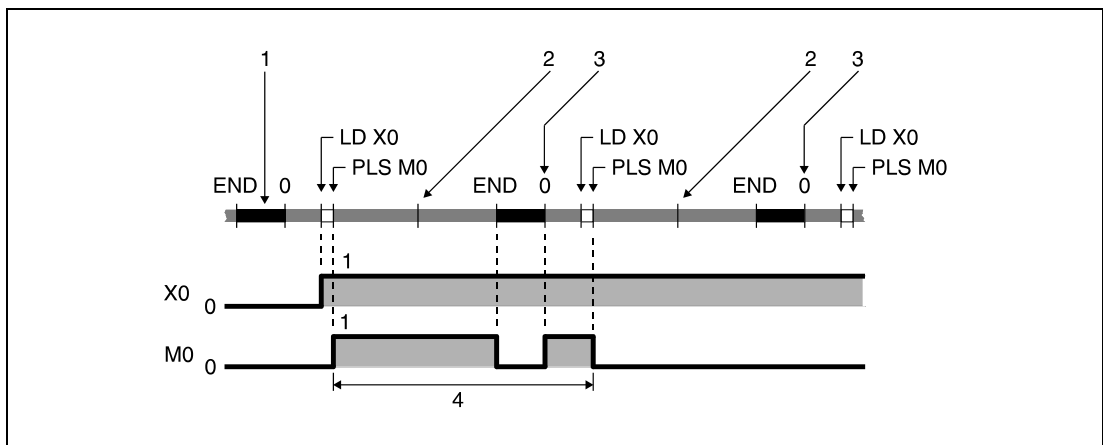
The PLS instruction consists of the PLS command followed by the number of device d to be set.

The PLS instruction (pulse) with leading edge from the input condition sets a device for one program scan. If the designated device is already set, this device will be reset for one program scan.



<sup>1</sup> One scan

If the RUN/STOP key switch on the CPU unit is set to STOP while a PLS instruction is executed, the PLS instruction will not be executed further on after the switch is set back to RUN even if the input condition is still set.



<sup>1</sup> END processing

<sup>2</sup> RUN/STOP switch of the CPU switched from RUN to STOP

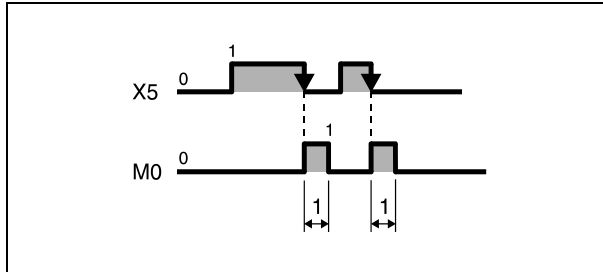
<sup>3</sup> RUN/STOP switch of the CPU switched from STOP to RUN

<sup>4</sup> One scan of PLS M0

If a latch relay is designated by a PLS instruction, and the power is turned OFF while a latch relay is set, after turning ON the power again the designated latch relay is set for one scan.

**PLF Output at trailing edge**

The PLF instruction consists of the PLF command followed by the number of device *d* to be set. The PLF instruction with trailing edge from the input condition sets a device for one program scan. If the designated device is already set, this device will be reset for one program scan.



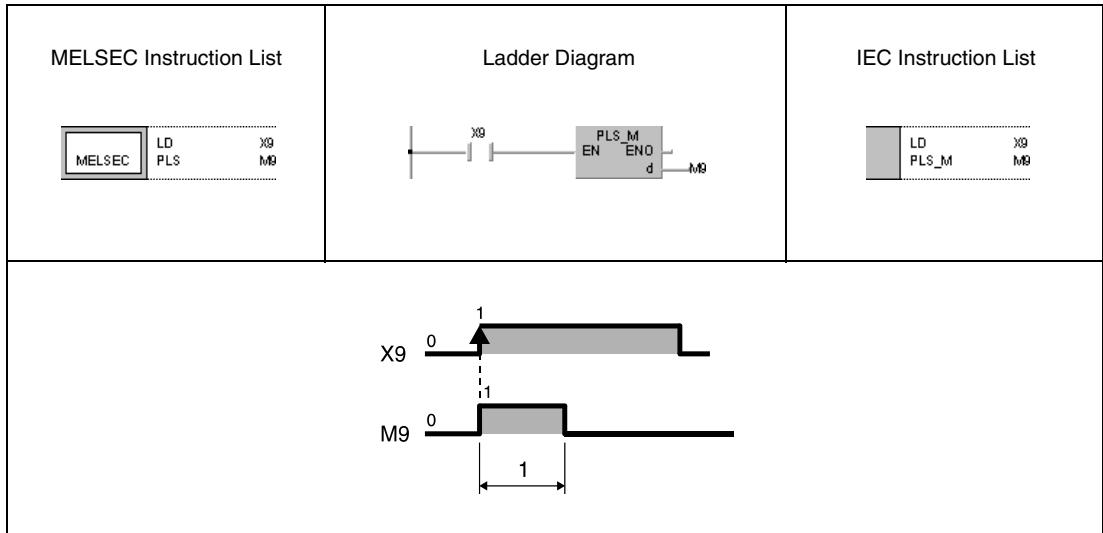
<sup>1</sup> One scan

If the RUN/STOP switch of the CPU unit is set to STOP while a PLS instruction is executed, the PLS instruction will not be executed further on after the switch is set back to RUN even if the input condition is still set.

**NOTE** *The device *d* designated by a PLS or PLF instruction remains set for more than one program scan if a CJ or similar instruction was applied to jump to the PLS or PLF instruction and the part of program was not executed.*

**Program Example 1**

PLS  
With leading edge from X9, the following program sets the internal relay M9 for one program scan.



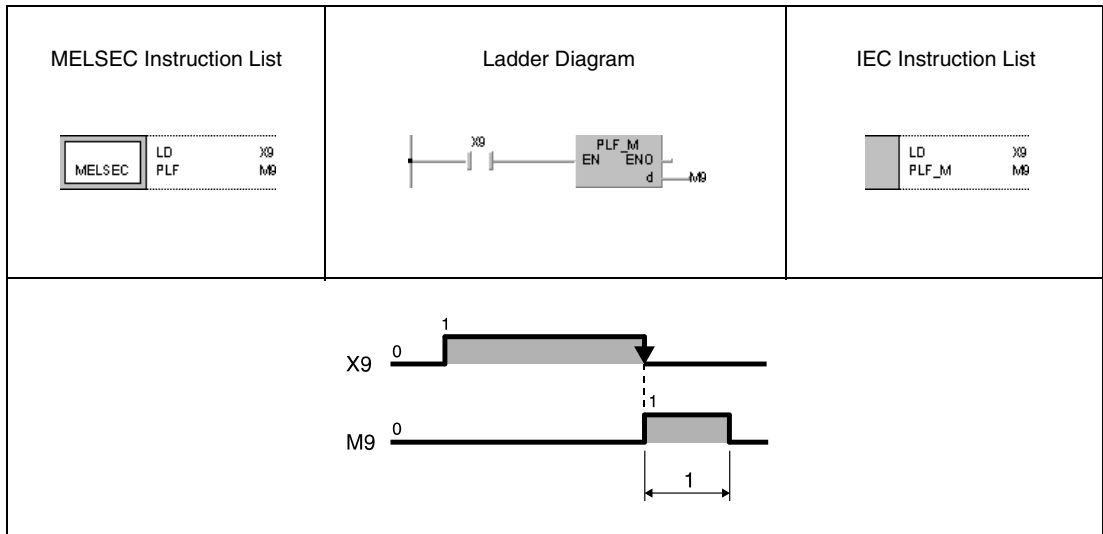
<sup>1</sup> One scan



**Program Example2**

PLF

With trailing edge from X9, the following program sets the internal relay M9 for one program scan.



<sup>1</sup> One scan

5.3.9 FF

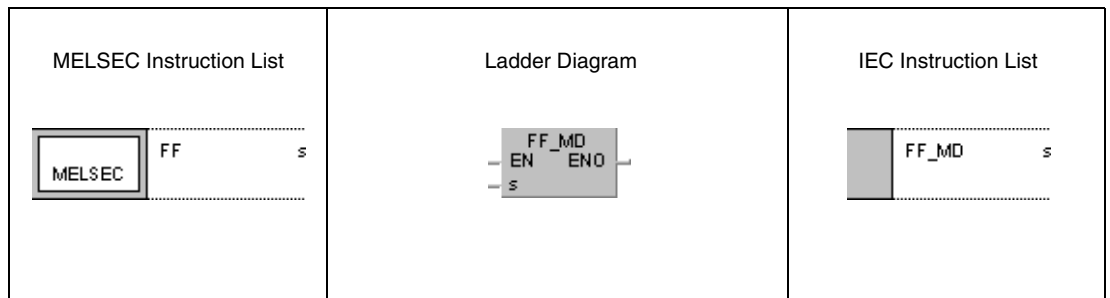
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

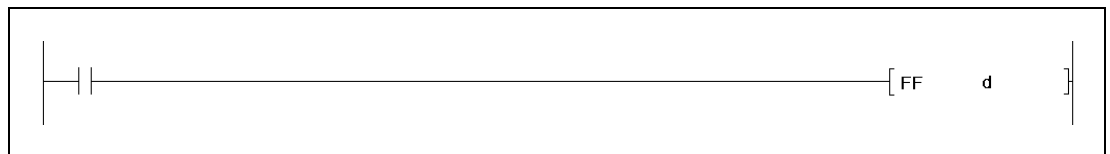
Devices  
MELSEC Q

d	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word				DY		
●	●	●	●	●	●	—	—	●	—	2	

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
d	Number of bit device or designated bit of word device to be inverted.	bit

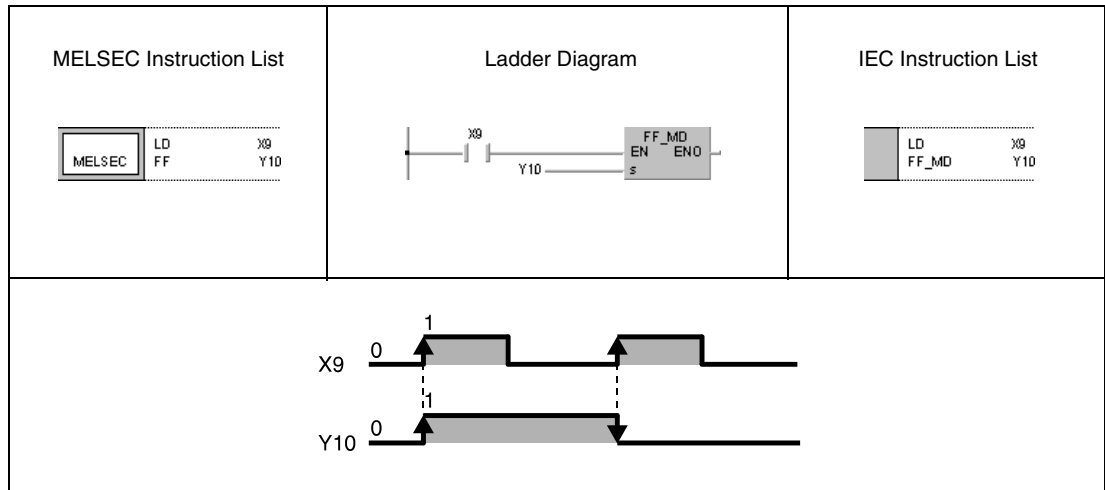
**Functions Bit device output inversion**

**FF Inversion of bit output device**

The FF instruction inverts the operation condition of the device designated by d with leading edge at the input of the FF instruction. The device can be a bit device or a specified bit of a word device. If the condition of the output device is set (1) it will be reset (0) after inversion. If the condition of the output device is reset (0), it will be set (1) after inversion.

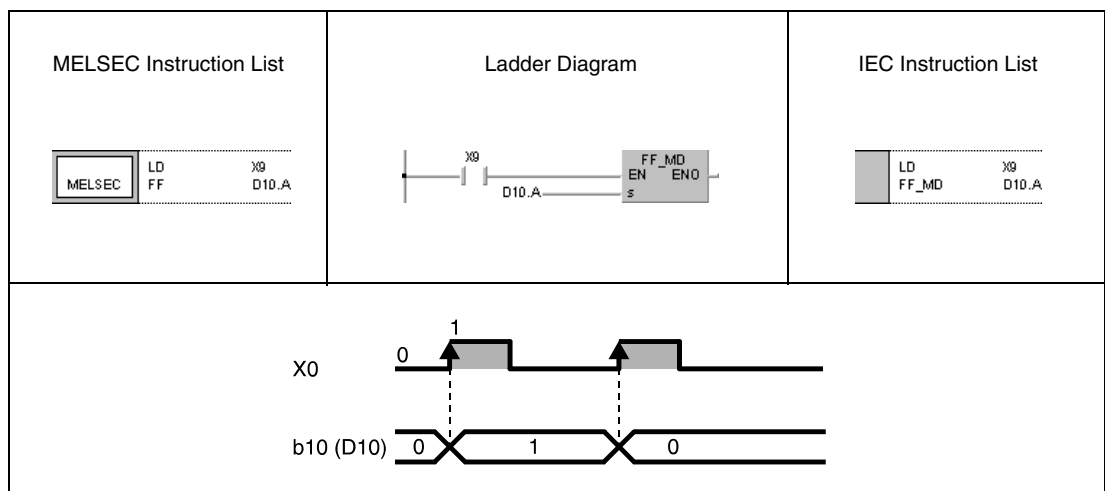
**Program Example 1**

FF  
With leading edge from X9, the following program inverts the output condition of Y10.



**Program Example 2**

FF  
With leading edge from X9, the following program inverts bit 10 (b10) of D10.



5.3.10 CHK

CPU

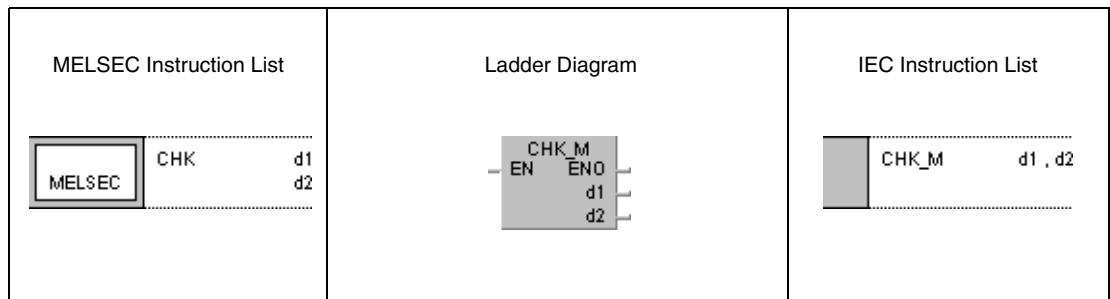
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●				

Devices  
MELSEC A

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag			
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level									
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N
d1	●	●	●	●	●	●																		
d2 ● <sup>1</sup>	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						K1 ↓ K4	5		

<sup>1</sup> Device d2 does not affect program execution (dummy device).

GX IEC  
Developer



Variables

Set Data	Meaning	Data Type
d1	Number of bit device of which the output signal is reversed.	bit
d2	Dummy device.	bit

**Functions Bit device output reverse (A series)**

**General notes**

The CHK instruction varies in function depending on the operation mode. In direct I/O control mode (except AnA and A2C CPUs) the CHK instruction performs a failure check. Using an AnS or AnN CPU in refresh I/O control mode the CHK instruction reverses the operation condition of an output device (flip-flop).

**CHK Bit device output reverse**

A complete CHK instruction consists of the CHK command, a device d1 of which the operation condition is to be reversed, and a dummy device d2.

If the input condition of the CHK instruction is set, the operation condition of the device designated by the CHK instruction is reversed. After resetting and setting the input condition once again the designated device is reset to its initial condition.

Although d2 is only a dummy device, it has to be specified (see table of usable devices). If a bit device is specified for d2, the digit has to be specified with K1 through K4. Any value can be specified because it is dummy data. The device d2 can be used freely for other purposes.

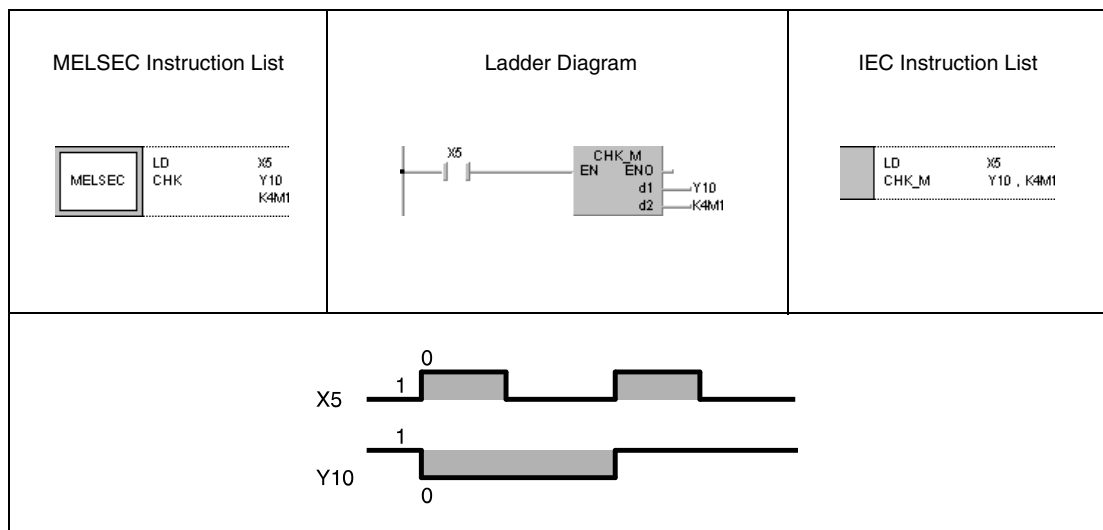
The CHK instruction described here, is only executed in refresh mode.

The reversal of the operation condition of an output device must maintain for at least one program scan time.

**Program Example**

CHK

With leading edge from X5, the following program reverses the output condition of Y10.



5.3.11 DELTA, DELTAP

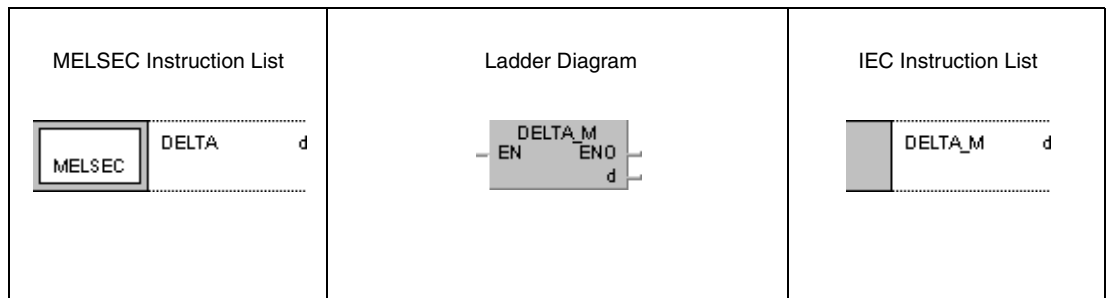
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

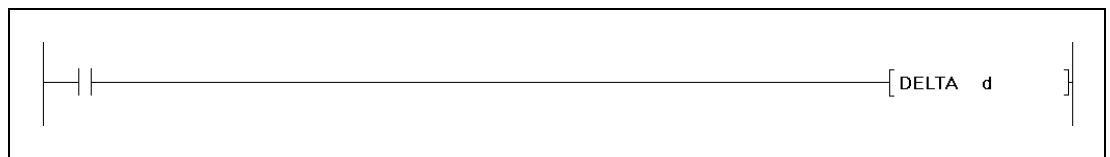
Devices  
MELSEC Q

d	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word				DY		
	—	—	—	—	—	—	—	—	●	SM0	2

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
d	Number of direct access output to generate pulse at.	bit ● <sup>1</sup>

<sup>1</sup> direct access outputs only

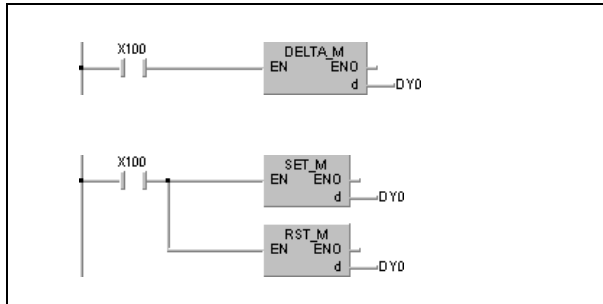
**Functions**     **Generating pulses at direct access outputs**

**DELTA**     **Pulse conversion of contacts**

The DELTA instruction generates a pulse at a direct access output (DY) designated by d, i.e. the output is set for a certain time only.

If the output designated by the DELTA instruction is DY0, the executed function is identical to that of the SET/RST instruction (see diagram).

The DELTA(P) instruction is used by commands for leading edge execution in special function units.



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The number of output designated by d exceeds the output range (error code: 4101).

**Program Example**

**DELTAP**

With leading edge from X20, the following program presets CH1 of the AD61 output unit mounted at slot 0 of the main base unit. The preset value 0 is stored at addresses 1 and 2 of the AD61 buffer memory. The DELTAP instruction outputs the preset instruction at DY11.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td></td> <td style="padding: 2px;">DMOV</td> <td style="padding: 2px;">K0</td> </tr> <tr> <td></td> <td style="padding: 2px;">U0V1</td> <td style="padding: 2px;">U0V1</td> </tr> <tr> <td></td> <td style="padding: 2px;">DELTAP</td> <td style="padding: 2px;">DY11</td> </tr> </table>	MELSEC	LD	X20		DMOV	K0		U0V1	U0V1		DELTAP	DY11	<p>Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td style="padding: 2px;">MOV_P_M</td> <td style="padding: 2px;">0, U0V1</td> </tr> <tr> <td style="padding: 2px;">MOV_P_M</td> <td style="padding: 2px;">0, U0V2</td> </tr> <tr> <td style="padding: 2px;">DELTAP_M</td> <td style="padding: 2px;">DY11</td> </tr> </table>	LD	X20	MOV_P_M	0, U0V1	MOV_P_M	0, U0V2	DELTAP_M	DY11
MELSEC	LD	X20																				
	DMOV	K0																				
	U0V1	U0V1																				
	DELTAP	DY11																				
LD	X20																					
MOV_P_M	0, U0V1																					
MOV_P_M	0, U0V2																					
DELTAP_M	DY11																					

## 5.4 Shift Instructions

### 5.4.1 SFT, SFTP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag					
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level											
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N	M9012	M9010 M9011
d	●	●	●	●	●	●																3 ● <sup>1</sup>	● <sup>2</sup>		

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

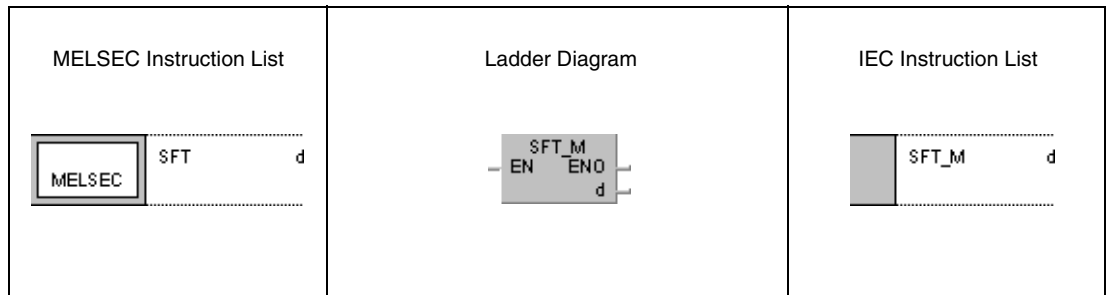
<sup>2</sup> Index qualification only supplied with AnA, AnAS, or AnU CPUs.

**Devices  
MELSEC Q**

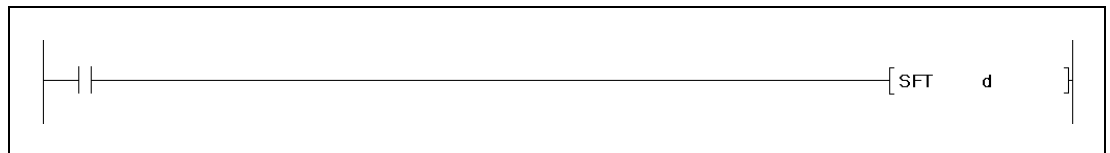
Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant DY	Other			
Bit	Word		Bit	Word				U			
d	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	—	—	●	—	2	

<sup>1</sup> Except T and C

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
d	Number of device to be shifted.	bit



**Functions Shift instruction**

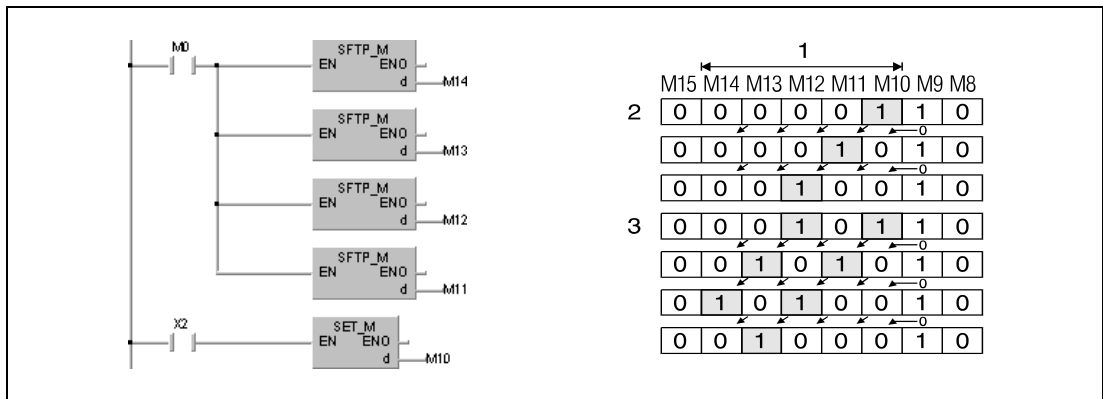
**SFT Shifting bit devices**

The SFT instruction shifts devices by one bit. Devices are only shifted via the SFT instruction, if the input condition is set (leading edge).

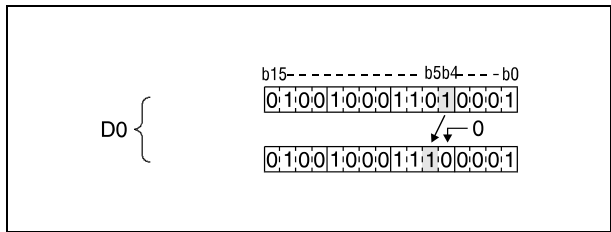
The instruction shifts the condition of a device (specified by d-1) to the destination address d. The condition of the device with the lower address d-1 is reset. The shifted number of device can be set via the SET instruction.

If several SFT instructions are applied consecutively, the program starts from the device with the higher number.

The program below sets the internal relay M10 if X2 is set (2,3). The condition of M10 (1) is shifted via the SFT P instruction within the shift range (1).



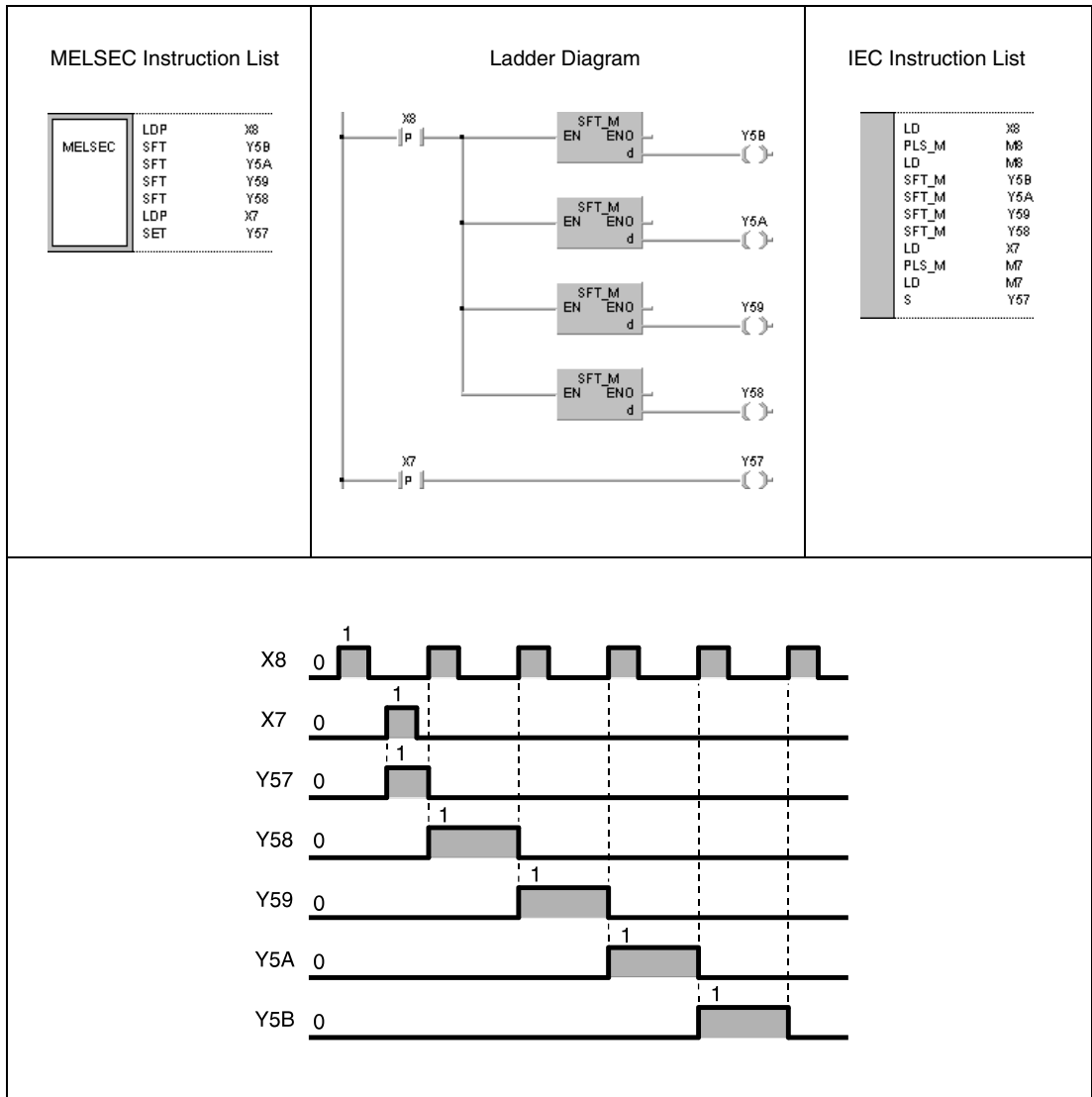
If bits in word devices are shifted, the condition (0/1) of the bit d-1 is shifted to d. The bit d-1 is reset after the SFT instruction. In the following illustration bit 5 (b5) in D0 is shifted. Bit 4 (b4) is reset after execution of the instruction.



**Program Example**

SFT

With leading edge from X8, the following program shifts the condition of Y57 to Y5B. With leading edge from X7, Y57 is set.



## 5.5 Master Control Instructions

### 5.5.1 MC, MCR

**NOTE** These instructions should not be used within the IEC editors.

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag						
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level												
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P	I	N	M9012
n																					●			3/5	● <sup>1</sup>		
d	●	●	●	●	●	●																		● <sup>2</sup>			

<sup>1</sup> Index qualification only supplied with AnA, AnAS, or AnU CPUs.

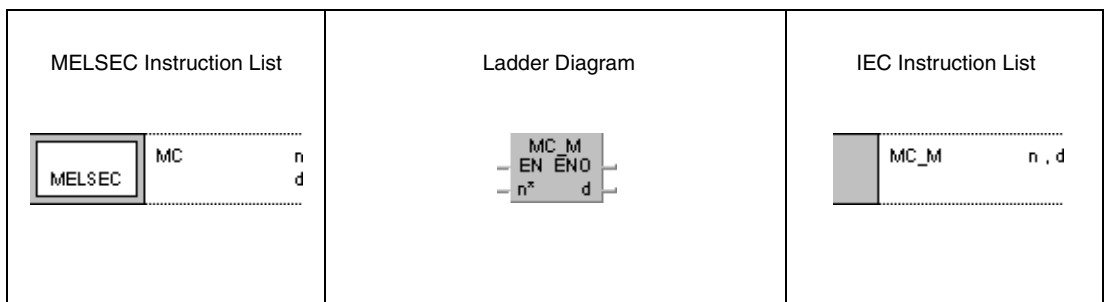
<sup>2</sup> The number of steps for the MC instruction is 5 and for the MCR instruction is 3. Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

**Devices  
MELSEC Q**

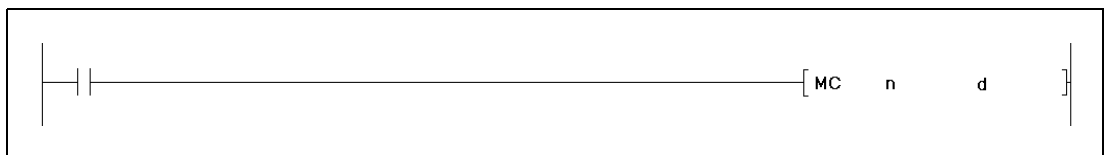
	Usable Devices										Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
	Bit	Word		Bit	Word				N	DY		
n	—	—	—	—	—	—	—	—	●		—	1/2
d	●	●	●	●	●	●	—	—		●	—	● <sup>1</sup>

<sup>1</sup> The number of steps is 2 for the MC instruction and 1 step for the MCR instruction.

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
n	Level of nesting (A series = N0 – N7, Q series and System Q = N0 – N14).	Nesting
d	Number of device to set nesting.	bit

**Functions**     **Setting and resetting master control**

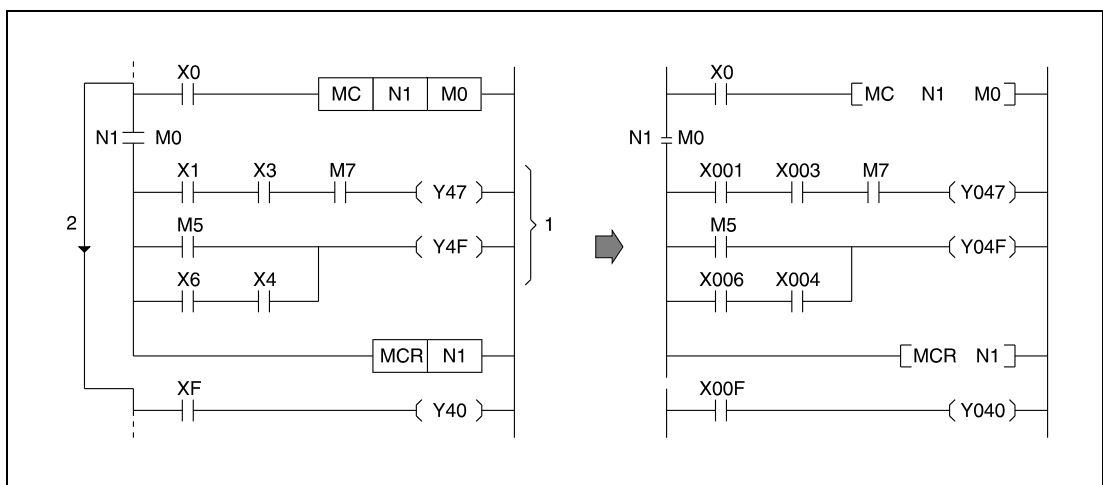
**General notes**

The MC instruction is applied to create highly efficient ladder switching sequence programs. After setting the input condition, the program part between the destination d and the MCR instruction is executed. The master control regions are distinguished by nesting (N0 through N7 for the A series, and N0 through N14 for the Q series and the System Q).

Since the GX IEC Developer Software does not allow a vivid programming of the MC/MCR instruction, here the ladder diagrams of the GX Developer Software are shown as an illustration.

The ladder diagram illustrates the function of the MC instruction. If the input X0 is reset, the program part in level 1 (designated by N1) is skipped (1). If X0 is set, the program part from N1 to the MCR instruction is executed (2).

When programming in the ladder mode, it is not necessary to input MC contacts on the vertical bus. These are displayed automatically.



**MC     Activating indicated program parts**

The MC instruction is the start instruction for master control to process a specified program part. If the input condition of the MC instruction is set, the devices between the MC and the MCR instruction are processed regularly.

The devices between the MC and the MCR instruction are even processed after the input condition of the MC instruction is reset. Therefore, the program scan time in this case is not decreased. When the input condition is reset, the devices between the MC and the MCR instruction are processed as follows:

Devices	Processing
10 ms timer 100 ms timer	Count value setting is reset to 0. Input and output contacts are reset (0).
Retentive 10 ms timer (Q series & System Qonly) Retentive 100 ms timer Counter	Count value setting and condition of input contacts remained. Output contact is reset (0).
Devices in the OUT instruction	All outputs are reset.
Devices in the SET, RST, and SFT instruction	Actual status remained.

**NOTE**

If an instruction that does not require any input condition (e.g. FOR/NEXT, EI, DI) is placed between the MC and MCR instructions, this instruction is executed by the PLC without regard to the input condition of the MC instruction.

For one MC instruction, identical nesting levels n are allowed, provided that different numbers (addresses) of devices are set.

After setting the MC instruction the device designated by d is set. If this device is designated as input condition elsewhere in the program, the contacts are processed as double contacts and set or reset in parallel. Therefore, the device designated by d should not be used within other instructions.

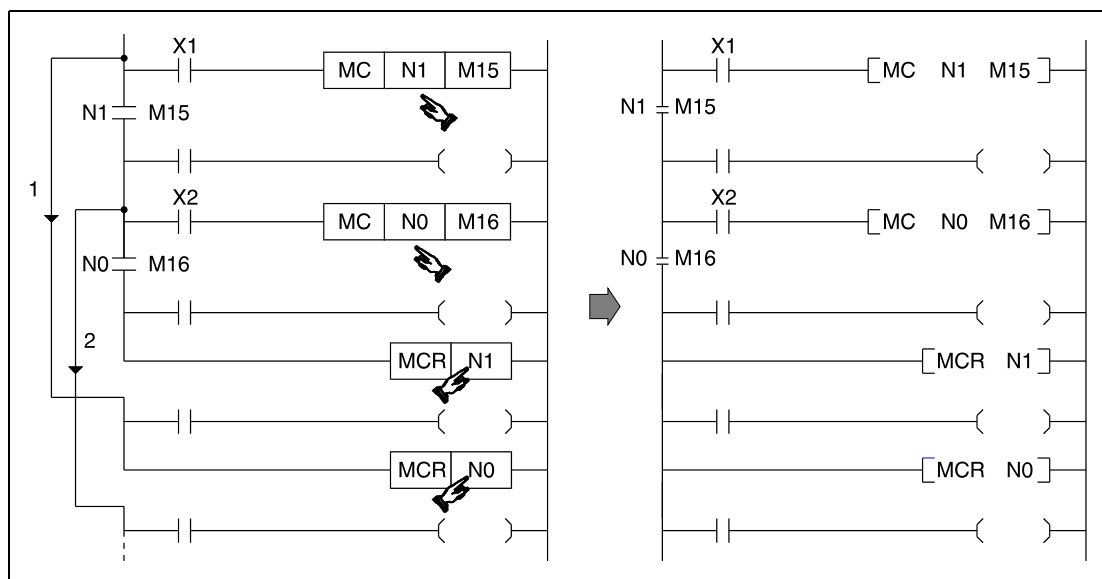
**MCR Deactivating indicated program parts**

The MCR instruction resets the MC instruction and indicates the end of the program part for master control.

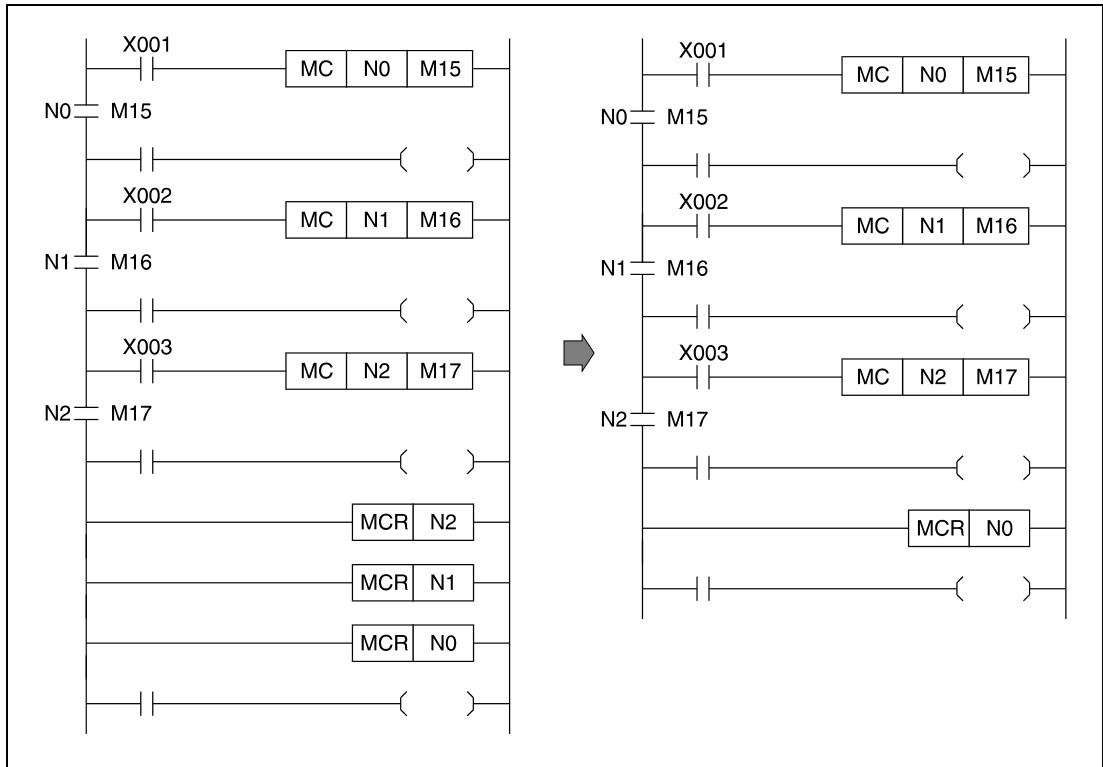
The MCR instruction must not be set via an input contact.

Notes on programming nesting numbers (addresses):

The Q series and the System Q provides 15 nesting levels from N0 to N14; the A series provides 8 nesting levels from N0 to N7. The first master control region designated by the MC instruction has to start with the lowest nesting address and the first MCR instruction has to start with the highest nesting address. If nesting addresses are designated in a different order, the nesting levels (1, 2) are not processed accurately by the PLC. The following diagram illustrates this case.



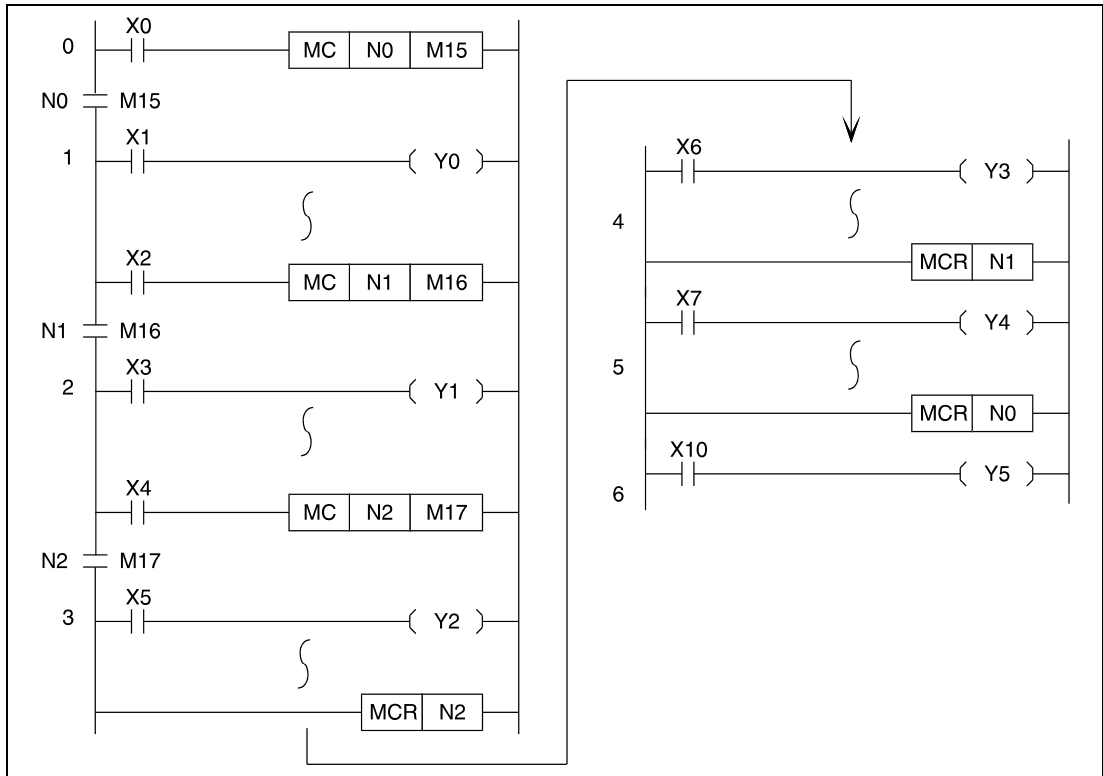
If several MCR instructions are programmed consecutively, the program can be shortened by placing one MCR instruction only with the lowest nesting address to finish all MC program parts.



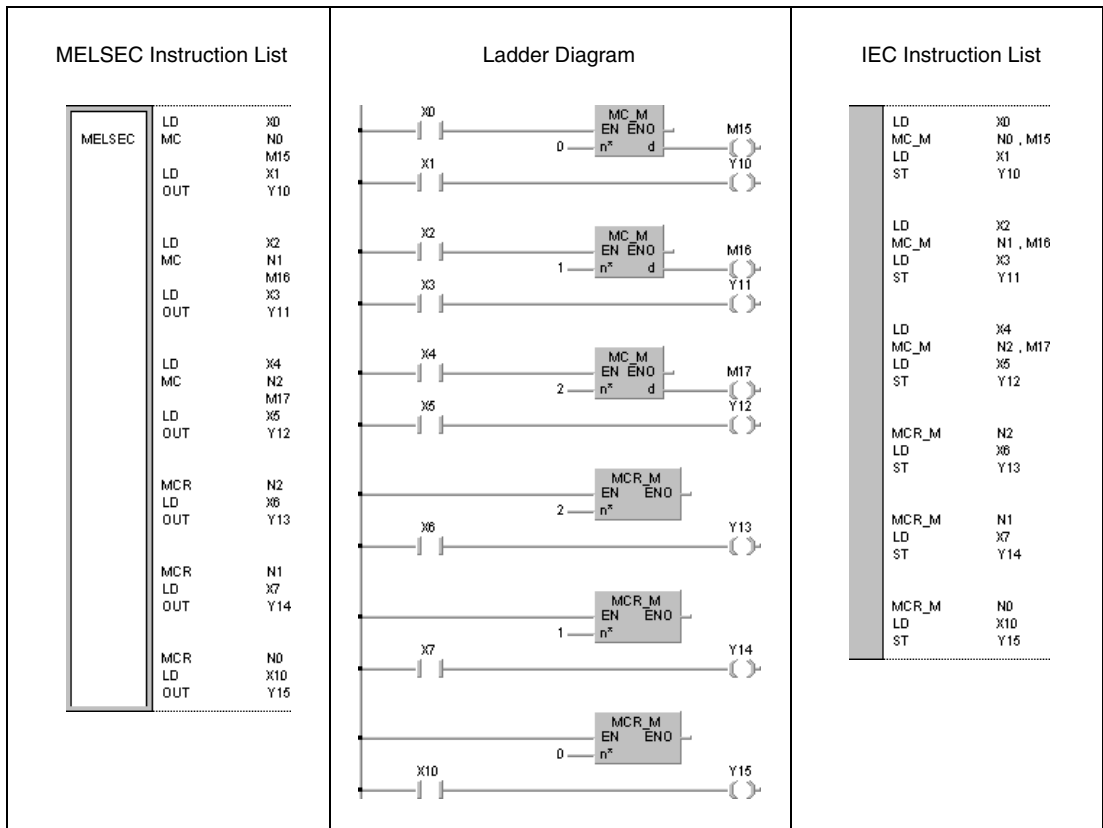
**Program Example** MC, MCR

The MC instruction designates a nesting address N to specify the nesting level. Nesting addresses can be designated within N0 to N14 for the Q series and the System Q, or within N0 to N7 for the A series respectively.

The nesting addresses determine the execution sequence of MC program parts. The following program illustrates designation of different execution levels by nesting addresses. For better comprehensibility the GX Developer ladder diagram is shown:



In addition the GX IEC Developer ladder diagram is shown:





## 5.6 Termination Instructions

### 5.6.1 FEND

**NOTE** This instruction should not be used within the IEC editors.

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

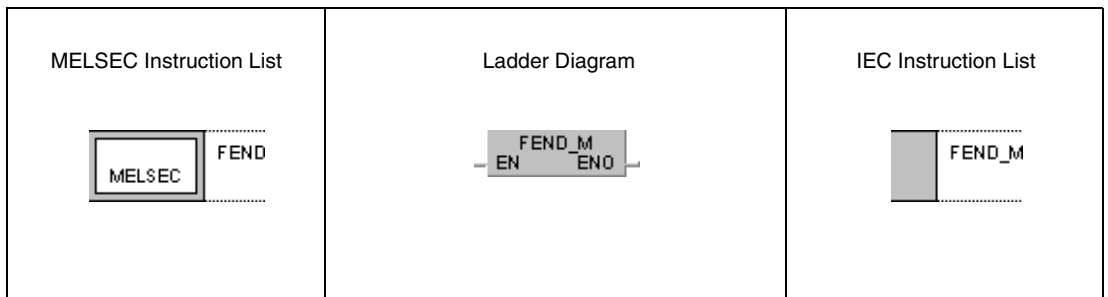
**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag			
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level									
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N
																						1	●

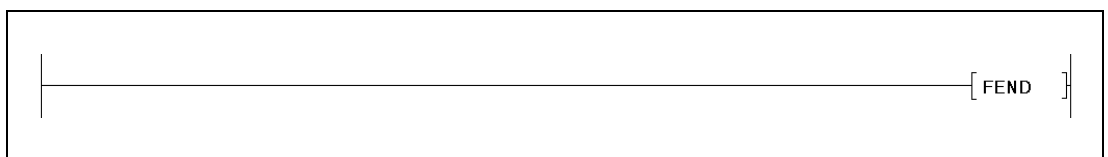
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	SM0	1

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions End of main routine program**

**FEND End of program branches**

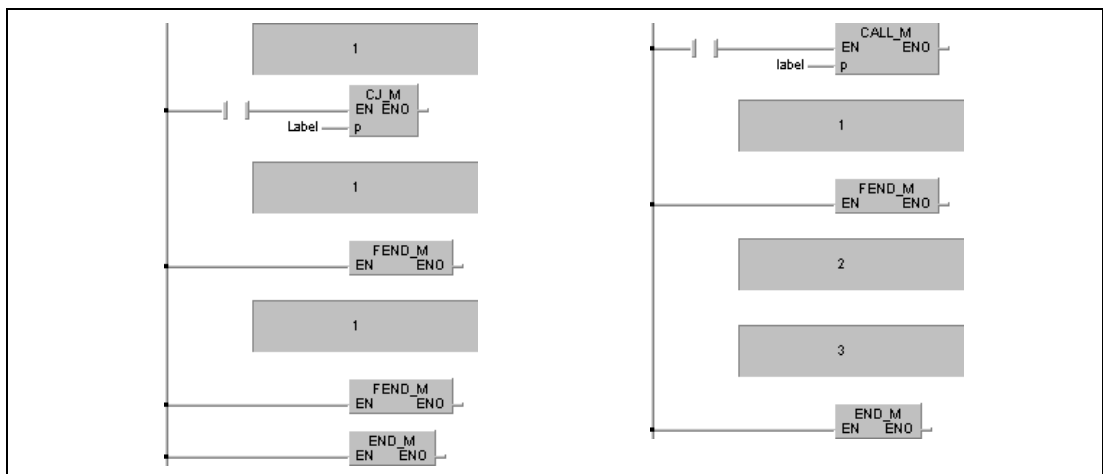
The FEND instruction specifies the end of a program branch. This branch can either be a main routine program or a subroutine program.

After execution of the FEND instruction the program jumps to the END instruction. The execution of internal processes like timer/counter processing or CPU self-diagnostics check begin at program step 1 again.

The program example on the left shows the termination of program branches invoked via the CJ (conditional jump) instruction.

After execution of the CJ instruction the invoked program part is executed up to the next FEND instruction. Without execution of the CJ instruction the program jumps back to program step 0 after the next FEND instruction.

The program example on the right shows the execution of the FEND instruction in order to split a main routine program from a sub-routine or interrupt program.



- 1 Main routine program
- 2 Subroutine program
- 3 Interrupt program

**NOTE**

*In the instruction list of the GX Developer the FEND instruction has to be programmed by the user. After this program organization unit has been processed no further one will be executed because it would follow the FEND instruction.*

*Alternatively to this programming the IEC editor can be used. In that case the FEND instruction would be set by the GX IEC Developer compiler automatically.*

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The FEND instruction is executed after a CALL, FCALL, ECALL, or EFCALL instruction and before a RET instruction (Q series and System Q = error code 4211).
- The FEND instruction is executed after a FOR instruction and before a NEXT instruction (Q series and System Q = error code 4200).
- The FEND instruction is executed during an interrupt program and before an IRET instruction (Q series and System Q = error code 4221).
- The FEND instruction is executed after a CHKCIR instruction and before a CHKEND instruction (Q series and System Q = error code 4230).
- The FEND instruction is executed after an IX instruction and before an IXEND instruction (Q series and System Q = error code 4231).

5.6.2 END

**NOTE** This instruction should not be used within the IEC editors.

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

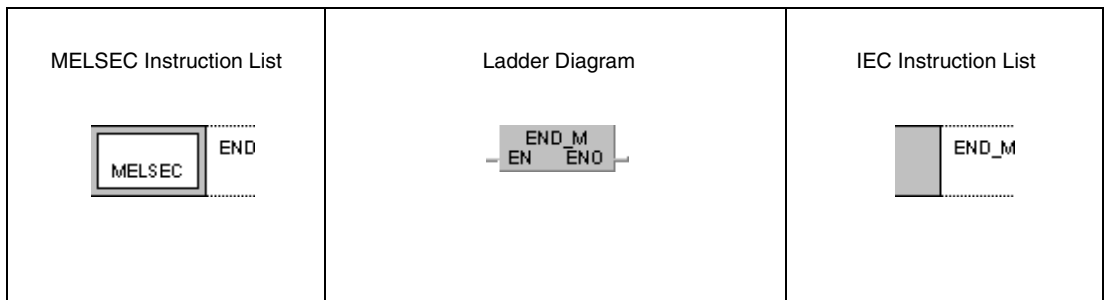
**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag			
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level									
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N
																						1	●

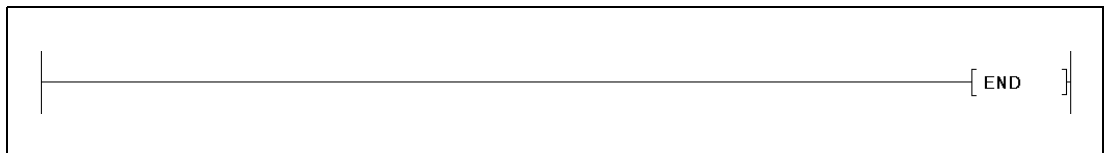
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word				U			
—	—	—	—	—	—	—	—	—	—	SM0	1

**GX IEC  
Developer**



**GX  
Developer**

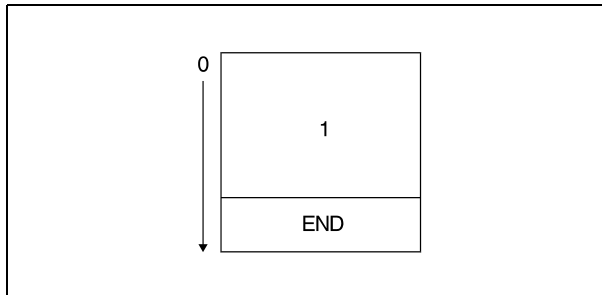


**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions**    **End of sequence program****END**    **End of sequence program**

The END instruction specifies the end of a program. Executing the END instruction the program jumps back to program step 0.

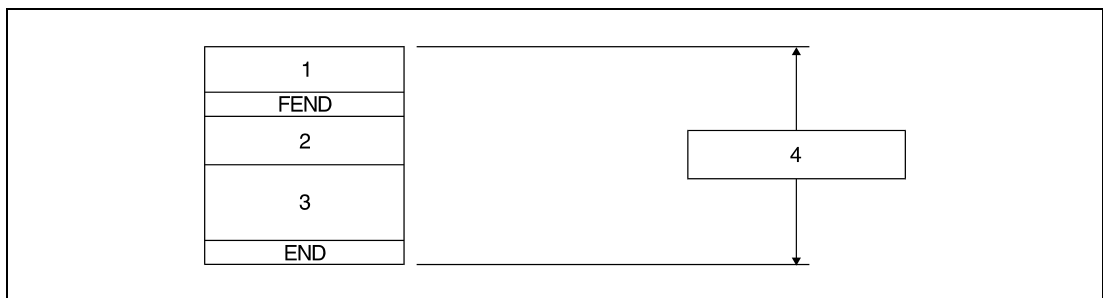


<sup>1</sup> Sequence program

The END instruction cannot be applied in a program routine. A program routine is terminated by the FEND instruction.

If the END instruction is missing in a program an error message is returned when starting the program, and the program execution is terminated by the PLC. Without the END instruction operation errors even occur, if the capacity of a subprogram is set by parameters.

The following diagram illustrates appropriate programming of the END and FEND instruction:



<sup>1</sup> Main routine program

<sup>2</sup> Subroutine program

<sup>3</sup> Interrupt program

<sup>4</sup> Sequence program

**NOTE**

*The FEND instruction will be set by both the GX IEC Developer and the GX Developer automatically.*

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The jump destination of a CJ, SCJ, or JMP instruction is allocated after the END instruction.
- A subprogram or interrupt routine allocated after the END instruction is called.
- The END instruction is executed after a CALL, FCALL, ECALL, or EFCALL instruction and before a RET instruction (Q series and System Q = error code 4211).
- The END instruction is executed after a FOR instruction and before a NEXT instruction (Q series and System Q = error code 4200).
- The END instruction is executed during an interrupt program and before an IRET instruction (Q series and System Q = error code 4221).
- The END instruction is executed after a CHKCIR instruction and before a CHKEND instruction (Q series and System Q = error code 4230).
- The END instruction is executed after an IX instruction and before an IXEND instruction (Q series and System Q = error code 4231).

## 5.7 Miscellaneous Instructions

### 5.7.1 STOP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

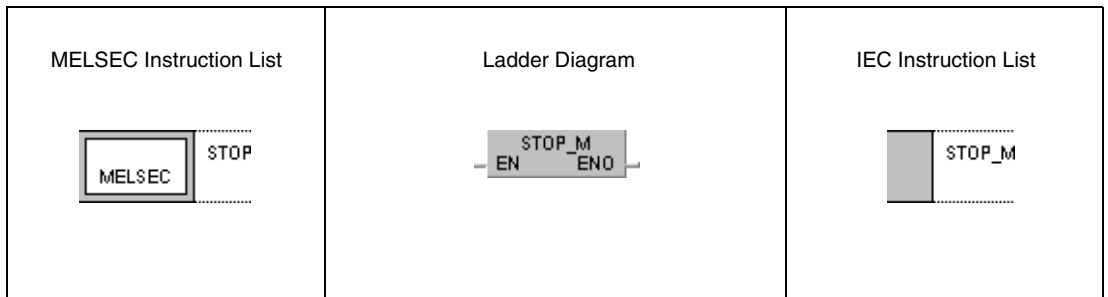
**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag				
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level										
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K H (16#)	P	I	N	M9012	M9010 M9011
																						1		

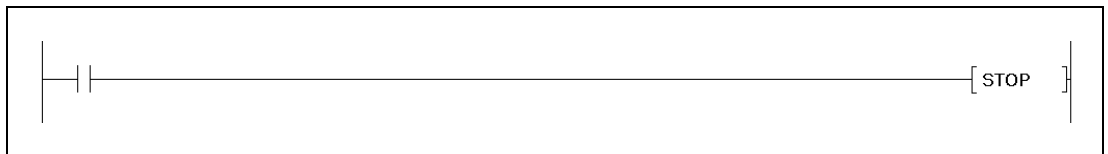
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word				U			
—	—	—	—	—	—	—	—	—	SM0	1	

**GX IEC Developer**



**GX Developer**



**Variables**

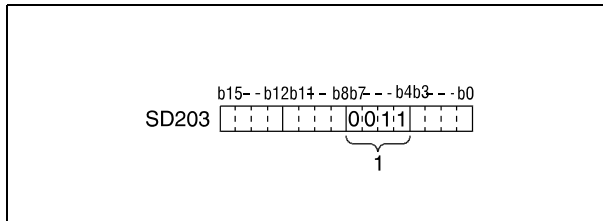
Set Data	Meaning	Data Type
—	—	—

**Functions**      **Sequence program stop**

**STOP**      **Stop instruction**

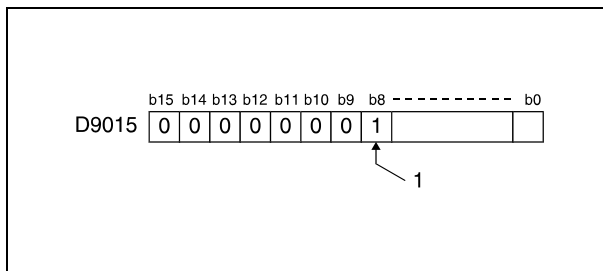
If the input condition of the STOP instruction is set, all outputs (Y) are reset and all operations of the PLC are terminated. The STOP instruction has the same function as the STOP position of the RUN/STOP key switch on the CPU.

On execution of the STOP instruction by a Q series or System Q CPU the 5th through the 8th bit (b4 through b7) in special register SD203 store the binary value 3.



<sup>1</sup> Binary value 3

On execution of the STOP instruction by an A series CPU the 9th bit (b8) in special register D9015 is set (1).



<sup>1</sup> Bit is set (1)

In order to restart the operation of the PLC the RUN/STOP switch has to be switched to STOP and then to RUN again.

Switching the RESET switch to LATCH CLEAR after execution of the STOP instruction does not affect the content of the buffer memory. In order to clear the buffer memory the RUN/STOP switch has to be switched to STOP first and then the RESET switch to L.CL. (LATCH CLEAR).



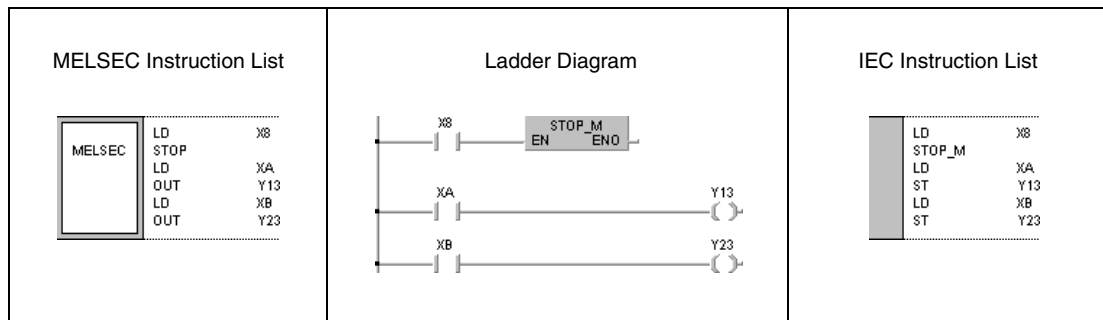
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The END instruction is executed after a CALL, FCALL, ECALL, or EFCALL instruction and before a RET instruction (Q series and System Q = error code 4211).
- The END instruction is executed after a FOR instruction and before a NEXT instruction (Q series and System Q = error code 4200).
- The END instruction is executed during an interrupt program and before an IRET instruction (Q series and System Q = error code 4221).
- The END instruction is executed after a CHKCIR instruction and before a CHKEND instruction (Q series and System Q = error code 4230).
- The END instruction is executed after an IX instruction and before an IXEND instruction (Q series and System Q = error code 4231).

**Program Example****STOP**

If X8 is set the following program terminates operation. All following program steps are executed after switching the RUN/STOP switch to STOP and to RUN again.



5.7.2 NOP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●


Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag			
Bit Devices							Word Devices (16-bit)						Constant	Pointer	Level									
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I	N	M9012
																						1		

Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

GX IEC  
Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p>	<p>IEC Instruction List</p>
--	-----------------------	-----------------------------

GX  
Developer

--

Variables

Set Data	Meaning	Data Type
—	—	—

**Functions**      **No operation program step**

**NOP**      **No operation program step**

The NOP instruction is a no-operation instruction that does not affect any other operations or program parts. The NOP instruction creates an empty logical program step that can be replaced by other program instructions during the development of a new program.

The NOP instruction is especially suitable for the following cases:

- To provide space for debugging sequence programs.
- To delete an instruction (overwrite it) without changing the number of steps.
- To delete an instruction temporarily for later editing.

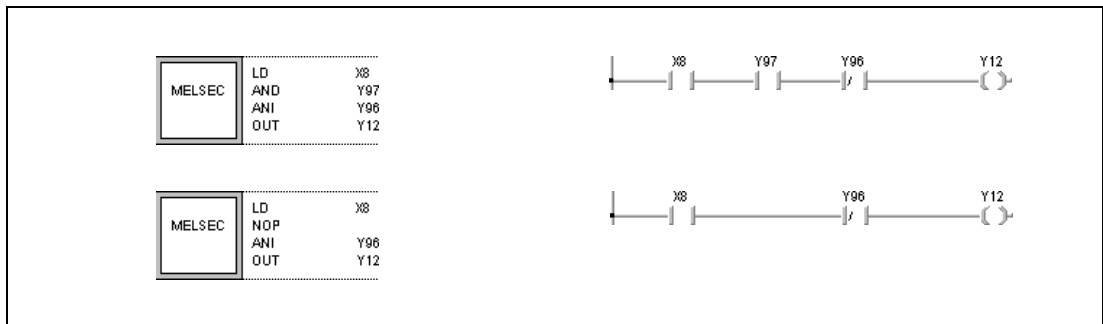
**NOTE**

*After finishing program editing the NOP instructions should be deleted where possible in order to shorten program scan time.*

**Program Example 1**

**NOP**

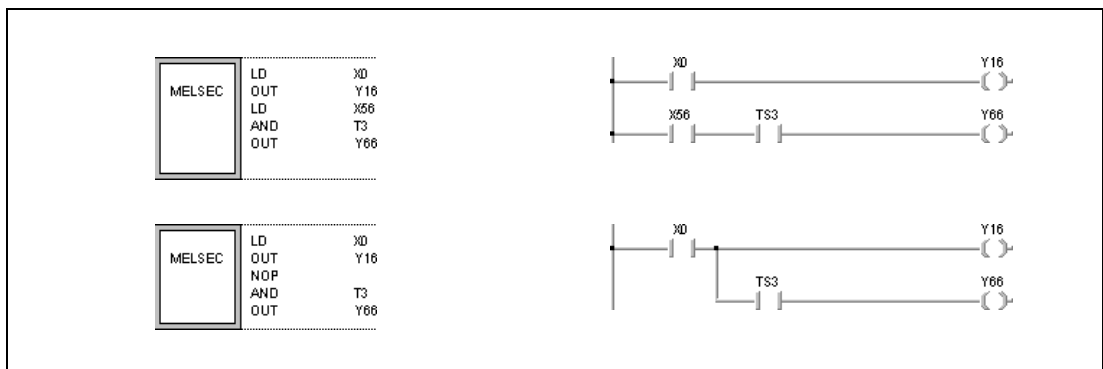
The following program contains a NOP instruction to replace the contact connection AND for debugging purposes.



**Program Example 2**

**NOP**

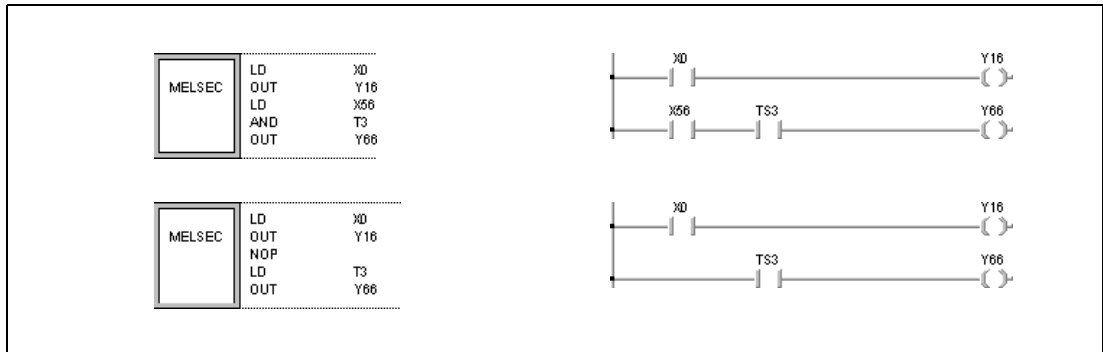
The following program example contains a NOP instruction to replace an LD instruction.



**Program Example 3**

**NOP**

The following program example contains a NOP instruction to replace an LD instruction.



**NOTE**

*Input contacts (LD, LDI) should be replaced by a NOP instruction carefully, because the logical structure of the program is changed considerably.*

---

## 6 Application Instructions, Part 1

The application instructions, part 1 comprise instructions that process numerical 16-bit and 32-bit data, floating point data, and character string data. Commonly, these basic instructions perform comparison and arithmetic operations.

Instruction	Meaning
Comparison operation instruction	Compares data to data (e.g. =, >, ≥)
Arithmetic operation instruction	Adds, subtracts, multiplies, divides, increments, and decrements BIN and BCD data, floating point data, and BIN block data Links character strings
Data conversion instruction	Converts data types (e.g. BCD -> BIN, BIN -> BCD)
Data transfer instruction	Transmits designated data
Program branch instruction	Program jump commands
Program execution control instruction	Enables and disables program interrupts
Refresh instruction	Refreshes bit devices, links, and I/O interfaces
Other convenient instructions	Count 1- or 2-phase input up or down, teaching timer, special function timer, rotary table near path rotation control, ramp signal, pulse density measurement, fixed cycle pulse output, pulse width modulation, matrix input

## 6.1 Comparison Operation Instructions

Comparison operation instructions compare data values (e.g. equal to =, greater than >, less than <). Programming the comparison operation instructions is similar to the corresponding basic instructions:

LD, LDI ⇒ LD=, LDD=

AND, ANI ⇒ AND=, ANDD=

OR, ORI ⇒ OR=, ORD=

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor	Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
= equal	LD=	LD_EQ_M	≤ less equal	LD<=	LD_LE_M
	AND=	AND_EQ_M		AND<=	AND_LE_M
	OR=	OR_EQ_M		OR<=	OR_LE_M
	LDD=	LDD_EQ_M		LDD<=	LDD_LE_M
	ANDD=	ANDD_EQ_M		ANDD<=	ANDD_LE_M
	ORD=	ORD_EQ_M		ORD<=	ORD_LE_M
	LDE=	LD_EEQ_M		LDE<=	LD_ELE_M
	ANDE=	AND_EEQ_M		ANDE<=	AND_ELE_M
	ORE=	OR_EEQ_M		ORE<=	OR_ELE_M
	LD\$=	LD_STRING_EQ_M		LD\$<=	LD_STRING_LE_M
	AND\$=	AND_STRING_EQ_M		AND\$<=	AND_STRING_LE_M
	OR\$=	OR_STRING_EQ_M		OR\$<=	OR_STRING_LE_M
	BKCOMP=	BKCOMP_EQ_M		BKCOMP<=	BKCOMP_LE_M
	BKCOMP=P	BKCOMP_EQP_M		BKCOMP<=P	BKCOMP_LEP_M
≠ not equal	LD<>	LD_NE_M	< less than	LD<	LD_LT_M
	AND<>	AND_NE_M		AND<	AND_LT_M
	OR<>	OR_NE_M		OR<	OR_LT_M
	LDD<>	LDD_NE_M		LDD<	LDD_LT_M
	ANDD<>	ANDD_NE_M		ANDD<	ANDD_LT_M
	ORD<>	ORD_NE_M		ORD<	ORD_LT_M
	LDE<>	LD_ENE_M		LDE<	LD_ELT_M
	ANDE<>	AND_ENE_M		ANDE<	AND_ELT_M
	ORE<>	OR_ENE_M		ORE<	OR_ELT_M
	LD\$<>	LD_STRING_NE_M		LD\$<	LD_STRING_LT_M
	AND\$<>	AND_STRING_NE_M		AND\$<	AND_STRING_LT_M
	OR\$<>	OR_STRING_NE_M		OR\$<	OR_STRING_LT_M
	BKCOMP<>	BKCOMP_NE_M		BKCOMP<	BKCOMP_LT_M
	BKCOMP<>P	BKCOMP_NEP_M		BKCOMP<P	BKCOMP_LTP_M

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor	Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
> greater	LD>	LD_GT_M	≥ greater equal	LD>=	LD_GE_M
	AND>	AND_GT_M		AND>=	AND_GE_M
	OR>	OR_GT_M		OR>=	OR_GE_M
	LDD>	LDD_GT_M		LDD>=	LDD_GE_M
	ANDD>	ANDD_GT_M		ANDD>=	ANDD_GE_M
	ORD>	ORD_GT_M		ORD>=	ORD_GE_M
	LDE>	LD_EGT_M		LDE>=	LD_EGE_M
	ANDE>	AND_EGT_M		ANDE>=	AND_EGE_M
	ORE>	OR_EGT_M		ORE>=	OR_EGE_M
	LD\$>	LD_STRING_GT_M		LD\$>=	LD_STRING_GE_M
	AND\$>	AND_STRING_GT_M		AND\$>=	AND_STRING_GE_M
	OR\$>	OR_STRING_GT_M		OR\$>=	OR_STRING_GE_M
	BKCOMP>	BKCOMP_GT_M		BKCOMP>=	BKCOMP_GE_M
	BKCOMP>P	BKCOMP_GTP_M		BKCOMP>=P	BKCOMP_GEP_M

**NOTE**

For the 16-bit comparison operation instructions, comparison commands with the same functional purpose are available in the IEC-standard library of the GX IEC Developer.

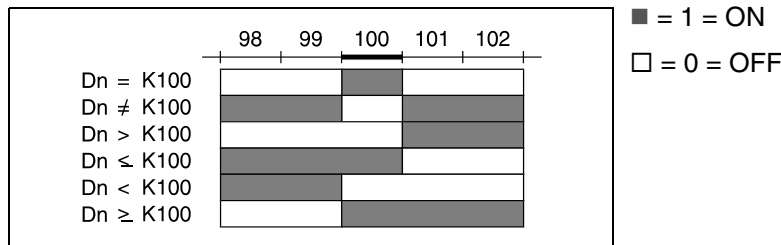
**IEC Commands**

Function	IEC Command	Meaning
=	EQ	Equal
<>	NE	Not Equal
<=	LE	Less Equal
<	LT	Less Than
>=	GE	Greater Equal
>	GT	Greater Than

Within the IEC editors please use the IEC commands.

**Execution Conditions**

The following illustration shows the execution conditions for the various comparison operation instructions.



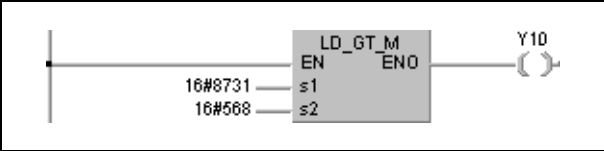
**NOTE** For comparison purposes, comparison operation instructions read all designated value types as negative BIN value numbers.

Comparison operation instructions process all designated data as binary data.

The result of the comparison operation  $16\#8000 > 16\#7999$  is FALSE (0), although TRUE (1) would be expected. The values are converted to BIN data and therefore bit 15 (b15) is set. If bit 15 is set, the value becomes negative.

**Program Example 1**

Comparison of two-digit BCD values:

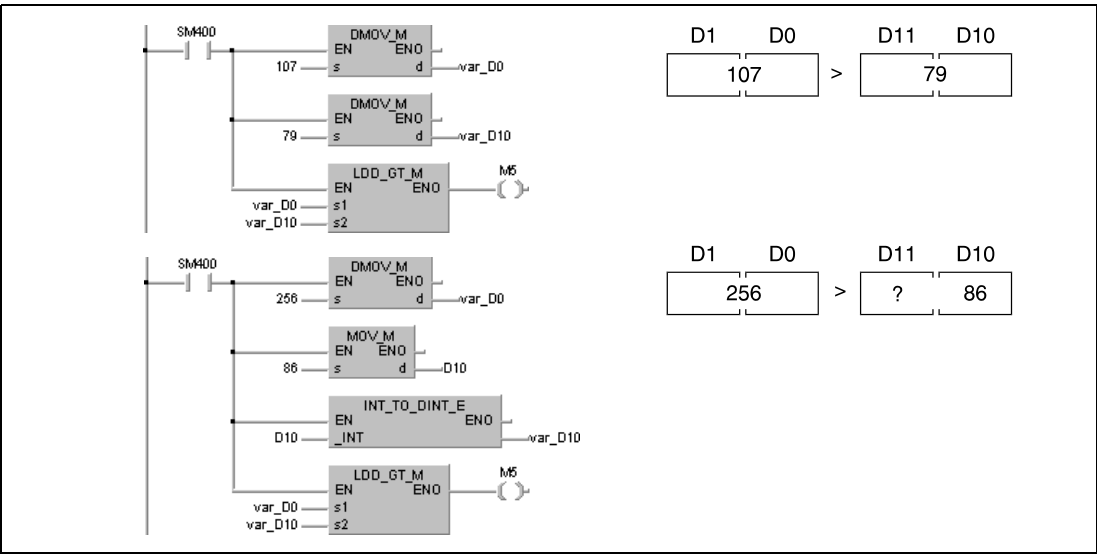


8731<sub>H</sub> is processed as -30927 and 568<sub>H</sub> as 1384. The comparison operation then is  $-30927 > 1384$  and Y10 is not set.

For comparison operation instructions with 32-bit data, the numerical input value has to be determined by a 32-bit instruction like DMOV. The instruction will not be carried out correctly, if the value was determined by a 16-bit instruction like MOV, because a 32-bit instruction always applies the n and (n+1) data value.

**Program Example 2**

Comparison instruction with 32-bit data:



The example shows two comparison operations with 32-bit data. The first program sets M5, because both values are determined by the 32-bit instruction DMOV.

The second program has no definite result, because the value in the upper bytes is not defined definitely.

**NOTE** This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.



6.1.1 =, <, >, >, <=, <, >=

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

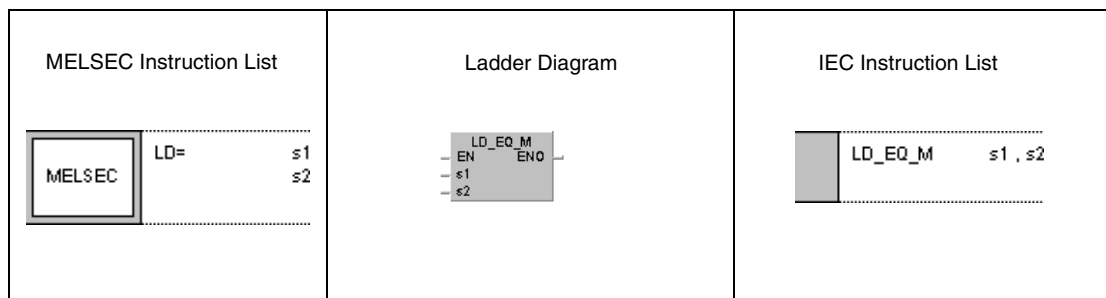
	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag						
	Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N	M9012	M9010 M9011	
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K4	5/7	● <sup>1</sup>	●		●	
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●										

<sup>1</sup> The number of steps is 7, provided the index function was started, the digit designation of a bit device is not K4, and the head address of a bit device is not a multiple of 8 (or 16 for the A3H, A3M, AnA, AnAS and AnU CPU). Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this Programming Manual for the according number of steps.

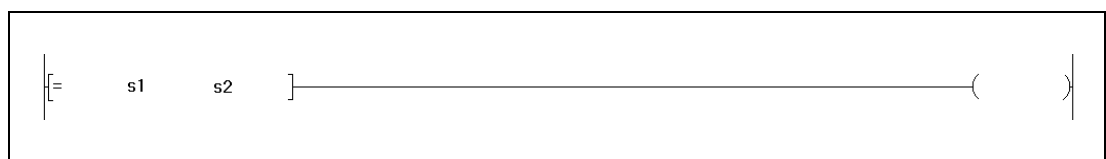
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	3	
s2	●	●	●	●	●	●	●	●	—		

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s1	Comparative data, or device storing comparative data	BIN 16-bit
s2		

**Functions BIN 16-bit data comparisons**

**=, <>, >, <=, <, >= Comparison operation instructions**

A 16-bit comparison operation instruction consists of the instruction itself and two designated devices s1 and s2 to be compared.

The comparison operation result is treated as NO contact.

The results of the comparison operations for the individual instructions are as follows:

Instruction Symbol	Comparison Operation Results	
	1	0
=	s1 = s2	s1 ≠ s2
<>	s1 ≠ s2	s1 = s2
>	s1 > s2	s1 ≤ s2
<=	s1 ≤ s2	s1 > s2
<	s1 < s2	s1 ≥ s2
>=	s1 ≥ s2	s1 < s2

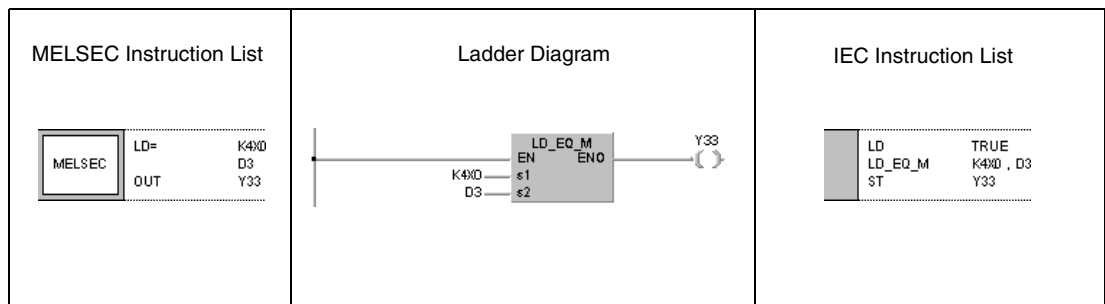
*For comparison purposes comparison operation instructions read all designated value types as negative BIN value numbers.*

*The result of the comparison operation 16#8000 > 16#7999 is FALSE (0), although TRUE (1) would be expected. The values are converted to BIN data and therefore bit 15 (b15) is set. If bit 15 is set, the value becomes negative.*

**Program Example 1**

Comparison operation instruction =

The following program compares the data at X0 to XF with the data in D3. It sets Y33, if the data are equal.



**Program Example 2** Comparison operation instruction <>

The following program compares BIN value 100 to the data in D3. It sets Y33, if the data in D3 is not equal to 100.

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LD</td> <td>M3</td> </tr> <tr> <td></td> <td>AND&lt;&gt;</td> <td>K100</td> </tr> <tr> <td></td> <td></td> <td>D3</td> </tr> <tr> <td></td> <td>OUT</td> <td>Y33</td> </tr> </table>	MELSEC	LD	M3		AND<>	K100			D3		OUT	Y33	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>LD</td> <td>M3</td> </tr> <tr> <td>AND_NE_M</td> <td>100, D3</td> </tr> <tr> <td>ST</td> <td>Y33</td> </tr> </table>	LD	M3	AND_NE_M	100, D3	ST	Y33
MELSEC	LD	M3																		
	AND<>	K100																		
		D3																		
	OUT	Y33																		
LD	M3																			
AND_NE_M	100, D3																			
ST	Y33																			

**Program Example 3** Comparison operation instruction >

The following program compares BIN value 100 to the data in D3. It sets Y33, if the data in D3 is less than 100 and M3 is set. Y33 is also set, if M8 and M3 are set.

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LD</td> <td>M3</td> </tr> <tr> <td></td> <td>LD&gt;</td> <td>K100</td> </tr> <tr> <td></td> <td></td> <td>D3</td> </tr> <tr> <td></td> <td>OR</td> <td>M8</td> </tr> <tr> <td></td> <td>ANB</td> <td></td> </tr> <tr> <td></td> <td>OUT</td> <td>Y33</td> </tr> </table>	MELSEC	LD	M3		LD>	K100			D3		OR	M8		ANB			OUT	Y33	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>LD</td> <td>M3</td> </tr> <tr> <td>AND&lt;</td> <td></td> </tr> <tr> <td></td> <td>M8</td> </tr> <tr> <td>OR_GT_M</td> <td>100, D3</td> </tr> <tr> <td>)</td> <td></td> </tr> <tr> <td>ST</td> <td>Y33</td> </tr> </table>	LD	M3	AND<			M8	OR_GT_M	100, D3	)		ST	Y33
MELSEC	LD	M3																														
	LD>	K100																														
		D3																														
	OR	M8																														
	ANB																															
	OUT	Y33																														
LD	M3																															
AND<																																
	M8																															
OR_GT_M	100, D3																															
)																																
ST	Y33																															

**Program Example 4** Comparison operation instruction <=

The following program compares the data in D0 to the data in D3. It sets Y33, if the data in D0 is less than or equal to D3. Y33 is also set, if M8 and M3 are set.

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LD</td> <td>M3</td> </tr> <tr> <td></td> <td>AND</td> <td>M8</td> </tr> <tr> <td></td> <td>OR&lt;=</td> <td>D0</td> </tr> <tr> <td></td> <td></td> <td>D3</td> </tr> <tr> <td></td> <td>OUT</td> <td>Y33</td> </tr> </table>	MELSEC	LD	M3		AND	M8		OR<=	D0			D3		OUT	Y33	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>LD</td> <td>M3</td> </tr> <tr> <td>AND</td> <td>M8</td> </tr> <tr> <td>OR_LE_M</td> <td>D0, D3</td> </tr> <tr> <td>ST</td> <td>Y33</td> </tr> </table>	LD	M3	AND	M8	OR_LE_M	D0, D3	ST	Y33
MELSEC	LD	M3																							
	AND	M8																							
	OR<=	D0																							
		D3																							
	OUT	Y33																							
LD	M3																								
AND	M8																								
OR_LE_M	D0, D3																								
ST	Y33																								

6.1.2 D=, D<>, D>, D<=, D<, D>=

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices MELSEC A

		Devices															Digit designation	Number of steps	Index	Carry Flag	Error Flag				
		Bit Devices								Word Devices (16-bit)						Constant						Pointer		Level	
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K						H (16#)	P	I	N
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						●		

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this Programming Manual for the according number of steps.

Devices MELSEC Q

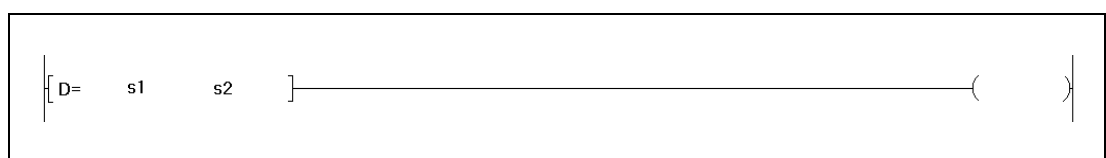
		Devices									Error Flag	Number of steps
		Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
s1	s2	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	●	—	—	3 <sup>1)</sup>
s2	●	●	●	●	●	●	●	●	●	—	—	

<sup>1</sup> The number of steps depends on the device and the type of CPU.  
 If a QnA-CPU or a single processor CPU of the System Q is used: 3  
 If a multi processor CPU of the System Q is used with internal word devices (except for file register ZR): 5  
 constants: 5  
 Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 5  
 If a System Q multi processor CPU is used with devices other than above mentioned: 3

GX IEC Developer

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;">                 MELSEC             </div> <p>LDD= s1 s2</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;">                 LDD_EQ_M s1, s2             </div>
---	-----------------------	---

GX Developer



Variables

Set Data	Meaning	Data Type
s1	Comparative data, or device storing comparative data.	BIN 32-bit
s2		

**Functions BIN 32-bit data comparison**

**D=, D<>, D>, D<=, D<, D>= Comparison operation instructions**

A 32-bit comparison operation instruction consists of the instruction itself and two designated devices s1 and s2 to be compared.

The comparison operation result is treated as NO contact. The comparison is performed with 32-bit data.

The results of the comparison operations for the individual instructions are as follows:

Instruction Symbol	Comparison Operation Results	
	1	0
D=	s1 = s2	s1 ≠ s2
D<>	s1 ≠ s2	s1 = s2
D>	s1 > s2	s1 ≤ s2
D<=	s1 ≤ s2	s1 > s2
D<	s1 < s2	s1 ≥ s2
D>=	s1 ≥ s2	s1 < s2

**NOTE**

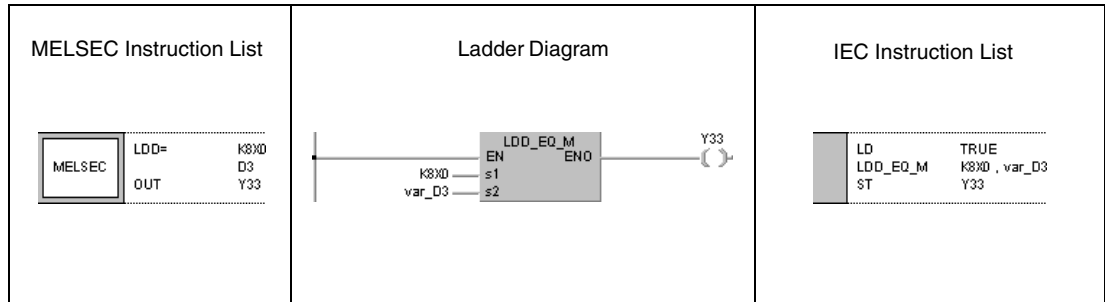
*For comparison purposes, comparison operation instructions read all designated value types as negative BIN value numbers.*

*The result of the comparison operation 16#8000 > 16#7999 is FALSE (0), although TRUE (1) would be expected. The values are converted to BIN data and therefore bit 15 (b15) is set. If bit 15 is set, the value becomes negative.*

**Program Example 1**

Comparison operation instruction D=

The following program compares the data at X0 to X1F with the data in D3 and D4. It sets Y33 if the data are equal.



**Program Example 2** Comparison operation instruction D<>

The following program compares BIN value 38000 to the data in D3 and D4. It sets Y33, if M3 is set and the data in D3 and D4 are not equal to 38000.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">M3</td> </tr> <tr> <td></td> <td style="padding: 2px;">ANDD&lt;&gt;</td> <td style="padding: 2px;">K38000</td> </tr> <tr> <td></td> <td style="padding: 2px;">OUT</td> <td style="padding: 2px;">D3</td> </tr> <tr> <td></td> <td></td> <td style="padding: 2px;">Y33</td> </tr> </table>	MELSEC	LD	M3		ANDD<>	K38000		OUT	D3			Y33	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">M3</td> </tr> <tr> <td style="padding: 2px;">ANDD_NE_M</td> <td style="padding: 2px;">38000 , var_D3</td> </tr> <tr> <td style="padding: 2px;">ST</td> <td style="padding: 2px;">Y33</td> </tr> </table>	LD	M3	ANDD_NE_M	38000 , var_D3	ST	Y33
MELSEC	LD	M3																		
	ANDD<>	K38000																		
	OUT	D3																		
		Y33																		
LD	M3																			
ANDD_NE_M	38000 , var_D3																			
ST	Y33																			

**Program Example 3** Comparison operation instruction D>

The following program compares BIN value -80000 to the data in D3 and D4. It sets Y33, if M3 is set and the data in D3 and D4 are less than -80000. Y33 is also set, if M3 and M8 are set.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">M3</td> </tr> <tr> <td></td> <td style="padding: 2px;">LDD&gt;</td> <td style="padding: 2px;">K-80000</td> </tr> <tr> <td></td> <td style="padding: 2px;">OR</td> <td style="padding: 2px;">M8</td> </tr> <tr> <td></td> <td style="padding: 2px;">ANB</td> <td></td> </tr> <tr> <td></td> <td style="padding: 2px;">OUT</td> <td style="padding: 2px;">Y33</td> </tr> </table>	MELSEC	LD	M3		LDD>	K-80000		OR	M8		ANB			OUT	Y33	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">M3</td> </tr> <tr> <td style="padding: 2px;">AND(</td> <td></td> </tr> <tr> <td style="padding: 2px;">ORD_GT_M</td> <td style="padding: 2px;">K-80000 , var_D3</td> </tr> <tr> <td style="padding: 2px;">)</td> <td></td> </tr> <tr> <td style="padding: 2px;">ST</td> <td style="padding: 2px;">Y33</td> </tr> </table>	LD	M3	AND(		ORD_GT_M	K-80000 , var_D3	)		ST	Y33
MELSEC	LD	M3																									
	LDD>	K-80000																									
	OR	M8																									
	ANB																										
	OUT	Y33																									
LD	M3																										
AND(																											
ORD_GT_M	K-80000 , var_D3																										
)																											
ST	Y33																										

**Program Example 4** Comparison operation instruction D<=

The following program compares the data in D0 and D1 to the data in D3 and D4. Y33 is set, if the data in D3 and D4 are greater than or equal to D0 and D1. Y33 is also set if M3 and M8 are set.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">M3</td> </tr> <tr> <td></td> <td style="padding: 2px;">AND</td> <td style="padding: 2px;">M8</td> </tr> <tr> <td></td> <td style="padding: 2px;">ORD&lt;=</td> <td style="padding: 2px;">D0</td> </tr> <tr> <td></td> <td></td> <td style="padding: 2px;">D3</td> </tr> <tr> <td></td> <td style="padding: 2px;">OUT</td> <td style="padding: 2px;">Y33</td> </tr> </table>	MELSEC	LD	M3		AND	M8		ORD<=	D0			D3		OUT	Y33	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">M3</td> </tr> <tr> <td style="padding: 2px;">AND</td> <td style="padding: 2px;">M8</td> </tr> <tr> <td style="padding: 2px;">ORD_LE_M</td> <td style="padding: 2px;">var_D0 , var_D3</td> </tr> <tr> <td style="padding: 2px;">ST</td> <td style="padding: 2px;">Y33</td> </tr> </table>	LD	M3	AND	M8	ORD_LE_M	var_D0 , var_D3	ST	Y33
MELSEC	LD	M3																							
	AND	M8																							
	ORD<=	D0																							
		D3																							
	OUT	Y33																							
LD	M3																								
AND	M8																								
ORD_LE_M	var_D0 , var_D3																								
ST	Y33																								

**NOTE** This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For more information see Chapter 3.5.2 of this manual..

**6.1.3 E=, E<>, E>, E<=, E<, E>=**

**CPU**

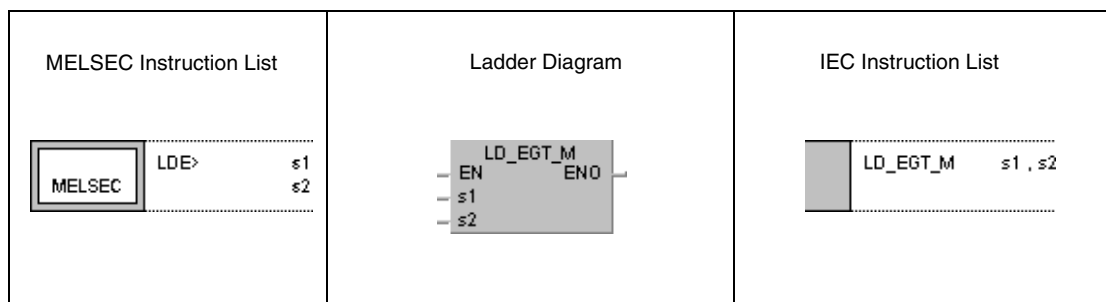
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

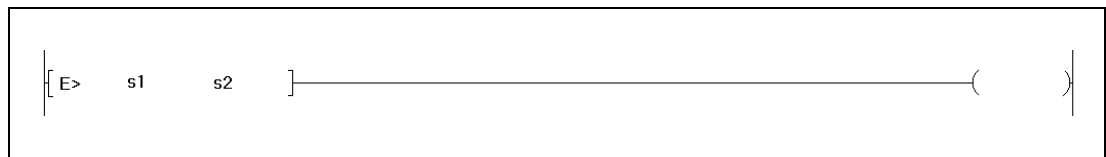
**Devices  
MELSEC Q**

	Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant E	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	●	●	—	●	—	—	3
s2	—	●	●	—	●	●	—	●	—	—	

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
s1	Comparative data, or device storing comparative data.	Real number
s2		

**Functions Floating point data comparisons**

**E=, E<>, E>, E<=, E<, E>= Comparison operation instructions**

A comparison operation instruction for floating point data consists of the instruction itself and two designated devices s1 and s2 to be compared.

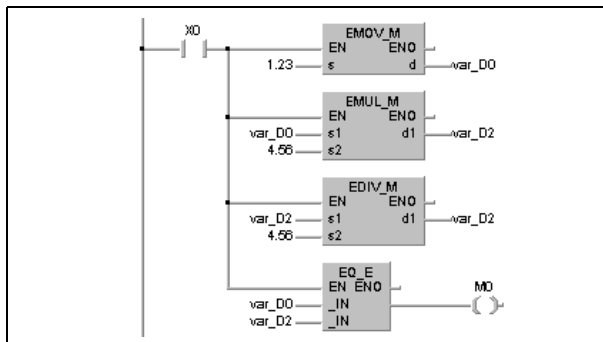
The comparison operation result is treated as NO contact. The comparison is performed with floating point data.

The results of the comparison operations for the individual instructions are as follows:

Instruction Symbol	Comparison Operation Results	
	1	0
E=	s1 = s2	s1 ≠ s2
E<>	s1 ≠ s2	s1 = s2
E>	s1 > s2	s1 ≤ s2
E<=	s1 ≤ s2	s1 > s2
E<	s1 < s2	s1 ≥ s2
E>=	s1 ≥ s2	s1 < s2

**NOTE**

*In some cases, rounding errors appear and floating point values that were equal before the comparison operation are not equal afterwards. In the following example M0 is not set:*



**NOTE**

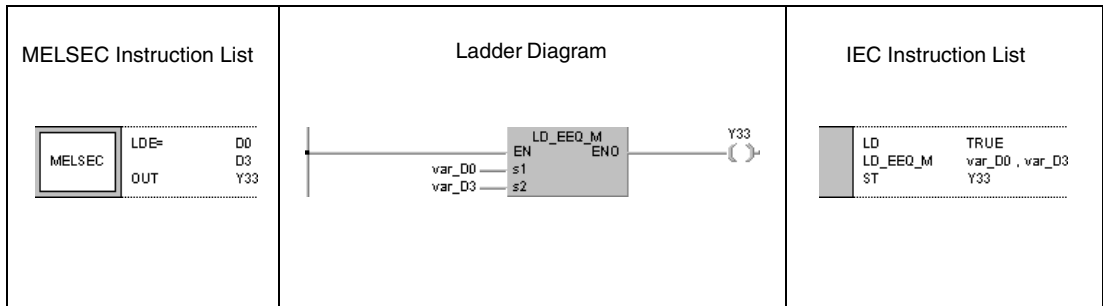
*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For more information see Chapter 3.5.2 of this manual.*



**Program Example 1**

Comparison operation instruction E=

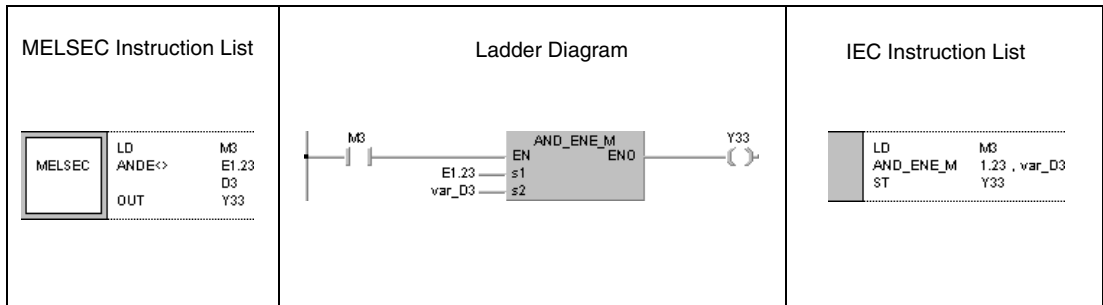
The following program compares floating point data in D0 and D1 to floating point data in D3 and D4. It sets Y33, if the data are equal.



**Program Example 2**

Comparison operation instruction E<>

The following program compares the floating point real number 1.23 to a floating point real number in D3 and D4. It sets Y33, if M3 is set and the data in D3 and D4 are not equal to 1.23.

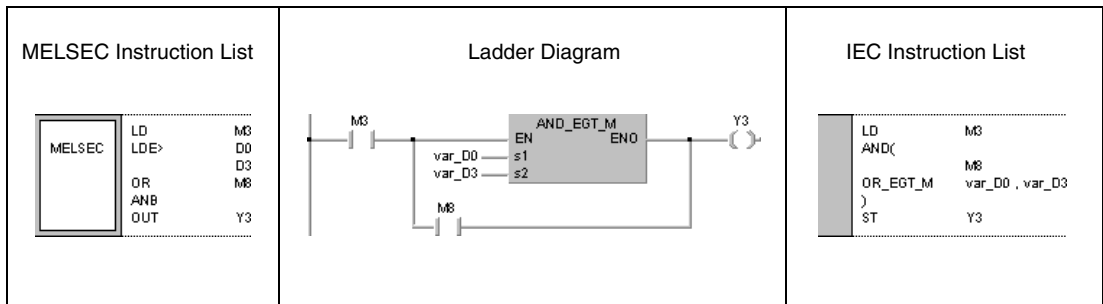


**Program Example 3**

Comparison operation instruction E>

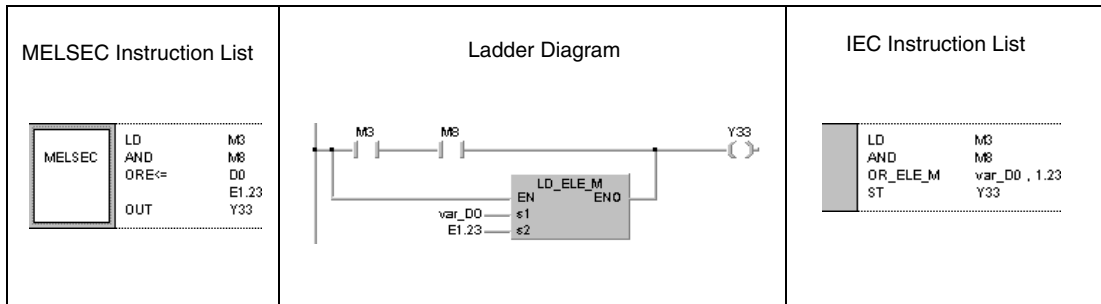
The following program compares floating point data in D0 and D1 to floating point data in D3 and D4. It sets Y3, if M3 is set and the data in D3 and D4 are less than the data in D0 and D1.

Y3 is also set, if M3 and M8 are set.



**Program Example 4** Comparison operation instruction E<=

The following example compares a floating point number in D0 and D1 to the floating point number 1.23. It sets Y33, if the data in D0 and D1 are less than or equal to 1.23. Y33 is also set, if M3 and M8 are set.



**NOTE** *This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 of this manual.*

6.1.4 \$ =, \$ < >, \$ >, \$ < =, \$ <, \$ > =

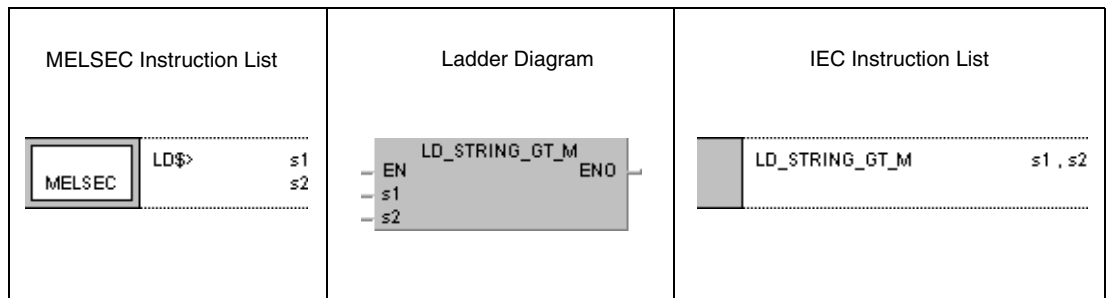
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

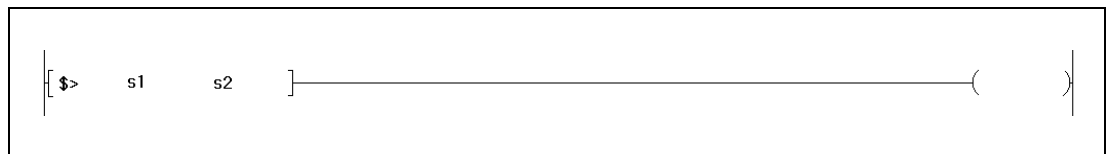
Devices  
MELSEC Q

	Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—	SM0	3
s2	—	●	●	—	—	—	—	●	—		

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s1	Comparative data, or device storing comparative data.	Character string
s2		

**Functions Character string data comparison**

**\$=, \$<>, \$>, \$<=, \$<, \$>= Comparison operation instructions**

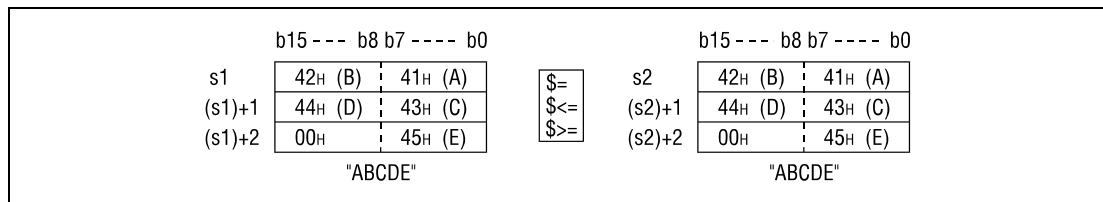
A comparison operation instruction for character string data consists of the instruction itself and two designated devices s1 and s2 to be compared.

The comparison operation result is treated as NO contact.

The comparison is performed with character string data in ASCII code character by character, beginning with the first character in the string.

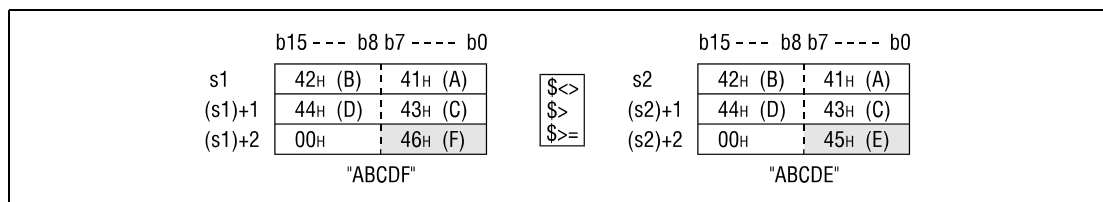
The s1 and s2 character strings include all characters from the designated device number up to the next device storing the code "00H".

If all character strings match, the comparison result for the operations \$=, \$<=, \$>= is 1.



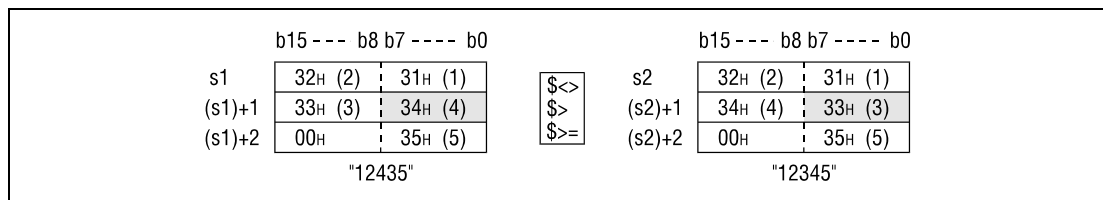
If the character strings are different, the character string with the larger character code will be the larger one.

Below, the comparison result for the operations \$<>, \$>, \$>= is 1.



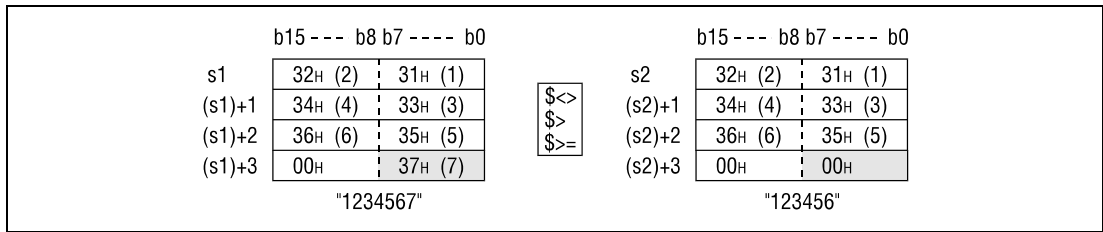
If the character strings are different, the first different sized character code determines whether the character string is larger or smaller.

Below, the comparison result for the operations \$<>, \$>, \$>= is 1.



If the character strings are of different lengths, the data with the longer character string will be larger.

Below, the comparison result for the operations \$<>, \$>, \$>=, is 1.



**Operation Errors**

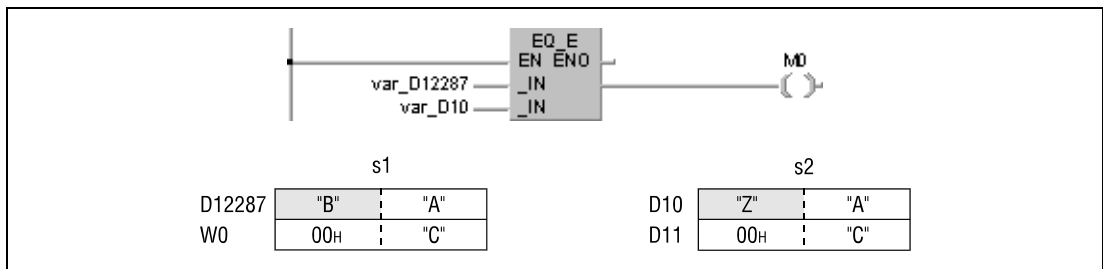
In the following cases an operation error occurs and the error flag is set:

- The code "00H" does not exist within the relevant device range of s1 and s2 (error code: 4101).

**NOTE**

The character string data comparison instruction also checks the device range.

Even though, in cases where one character string exceeds the device range, character string data is being compared and non-matching characters within the device range are detected. The comparison operation results are output without returning an error code.

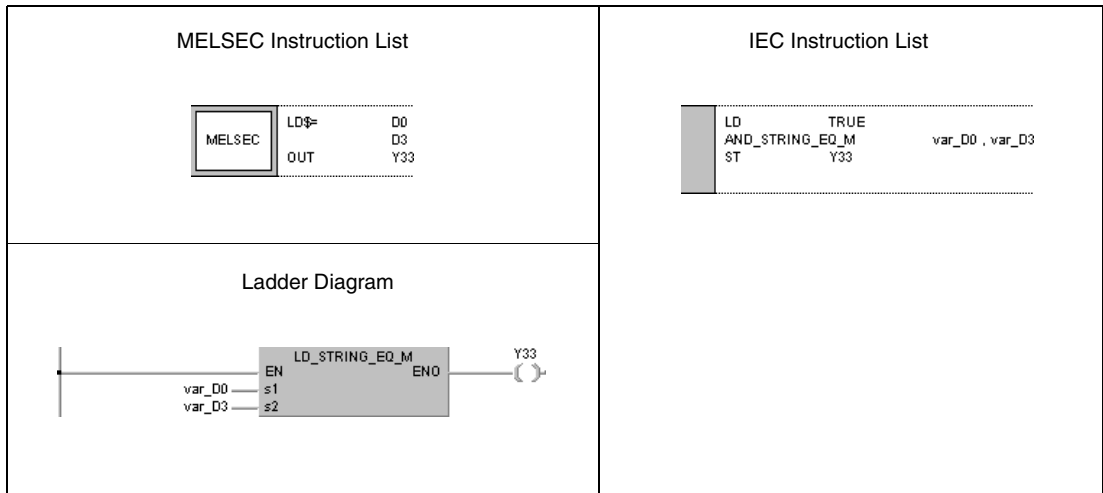


In the example shown above, the s1 character string exceeds the device range, and the most significant 16 bits (D12288) were renamed W0. Nevertheless, the comparison result is 0, because the second character in s1 is detected as different from that in s2. In this case no error code regarding the device range is returned.

This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

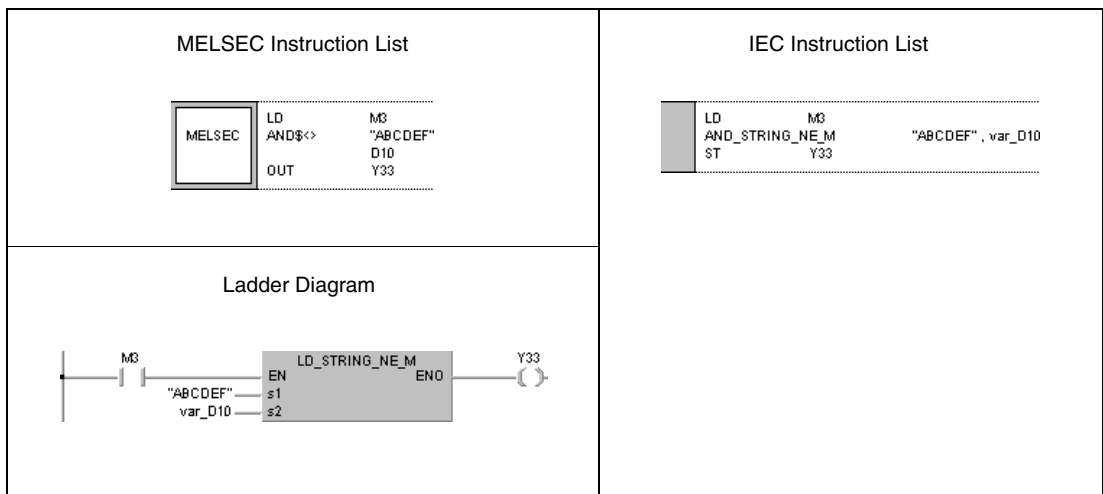
**Program Example 1** Comparison operation instruction \$=

The following program compares character string data in D0 to character string data in D3. It sets Y33, if the data are equal.



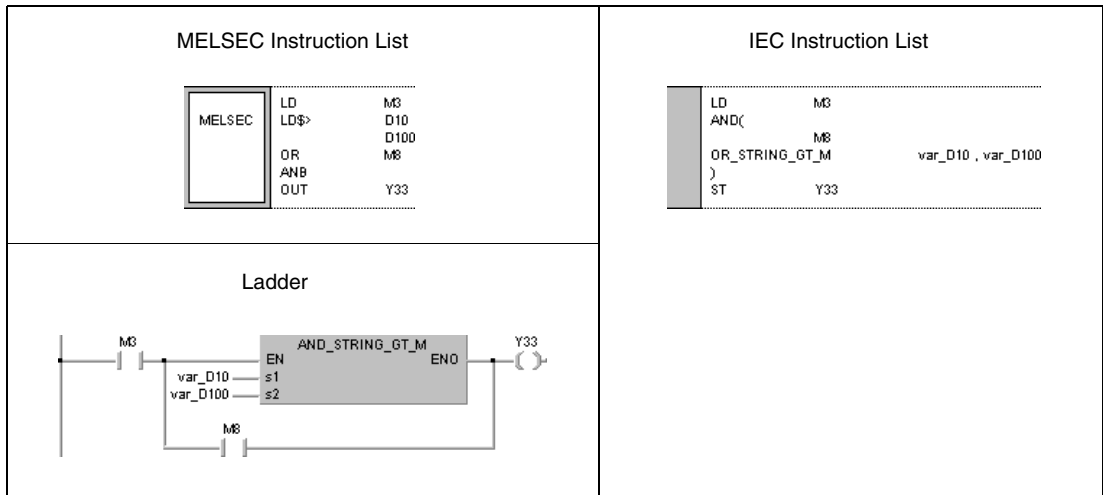
**Program Example 2** Comparison operation instruction \$<>

The following program compares the character string "ABCDEF" to character string data in D10. It sets Y33, if the data are not equal.



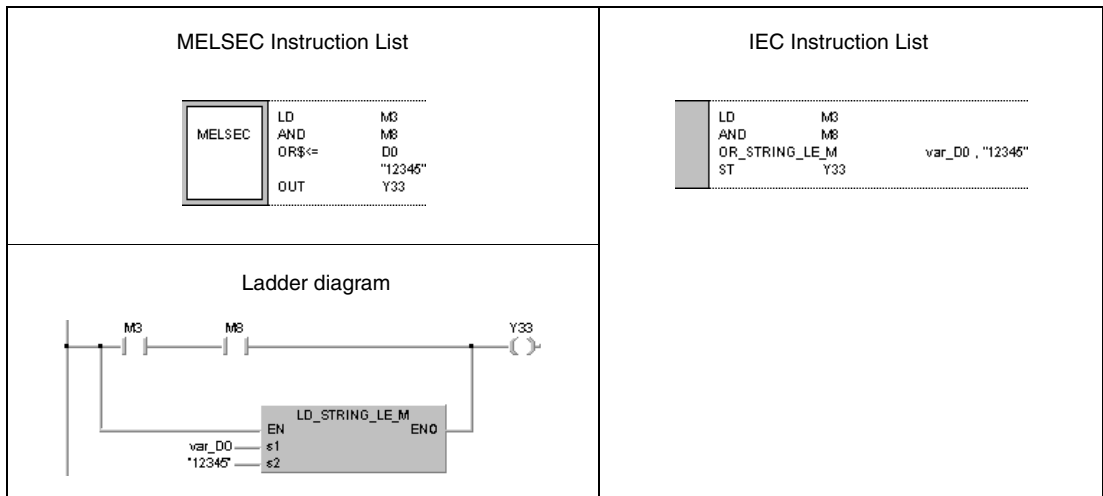
**Program Example 3** Comparison operation instruction \$>

The following program compares character string data in D10 to character string data in D100. It sets Y33, if character string data in D10 is greater.



**Program Example 4** Comparison operation instruction \$<=

The following program compares character string data in D0 to the character string "12345". Y33 is set, if character string data in D0 is less than or equal to "12345".



**NOTE** This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

6.1.5 BKCMP, BKCMPP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

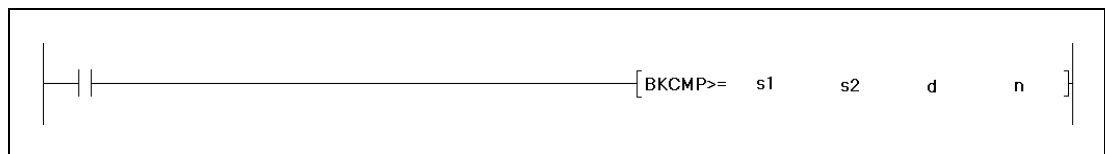
Devices  
MELSEC Q

	Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—	SM0	5
s2	—	●	●	—	—	—	—	—	—		
d	●	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <p>MELSEC</p> </div> <p>BKCMP&gt;=    s1                   s2                   d                   n</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <p>BKCMP_GE_M</p> </div> <p>BKCMP_GE_M s1 , s2 , n , d</p>
---	-----------------------	---

GX Developer



Variables

Set Data	Meaning	Data Type
s1	Comparative data, or device storing comparative data	BIN 16-bit
s2	Comparative data, or device storing comparative data	BIN 16-bit
d	First number of device storing results of comparison operation	Bit
n	Number of data blocks compared	BIN 16-bit



**Functions BIN block data comparisons**

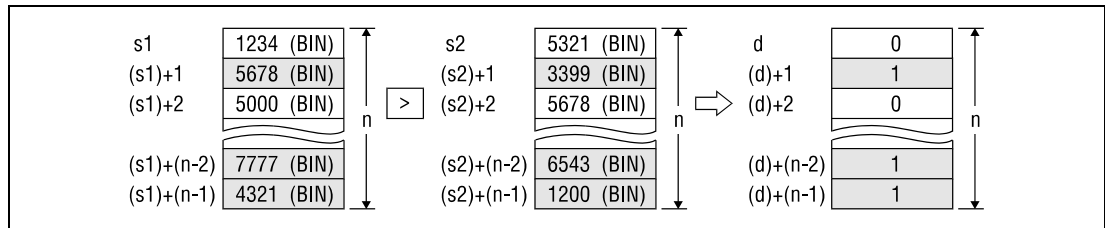
**BKCMP Comparison operation instructions**

A comparison operation instruction for BIN block data consists of the instruction itself, two designated devices s1 and s2 to be compared, a device d to store the result, and the number of datablocks to be compared.

It compares the nth BIN 16-bit block in s1 to the nth BIN 16-bit block in s2, beginning with the first number of device. The result of each block comparison is stored in d.

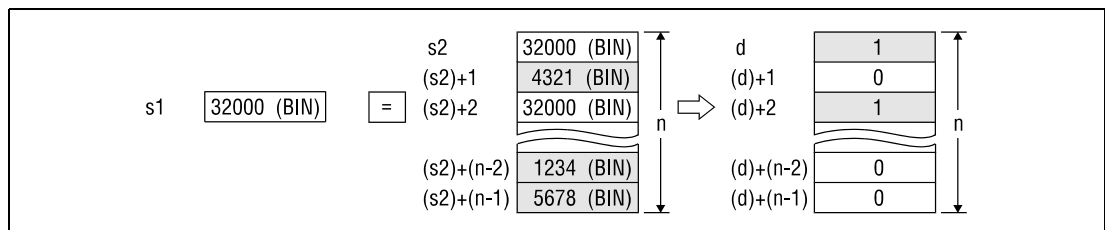
If the block comparison result is 1, then 1 is stored in d.

If the block comparison result is 0, then 0 is stored in d.



The comparison operation is conducted in 16-bit units.

The constant designated by s1 must be BIN 16-bit data ranging from -32768 to 32767.



The results of the comparison operations for the individual instructions are as follows:

Instruction Symbol	Comparison Operation Results for nth 16-bit Block	
	1	0
BKCMP=	s1 = s2	s1 ≠ s2
BKCMP<>	s1 ≠ s2	s1 = s2
BKCMP>	s1 > s2	s1 ≤ s2
BKCMP<=	s1 ≤ s2	s1 > s2
BKCMP<	s1 < s2	s1 ≥ s2
BKCMP>=	s1 ≥ s2	s1 < s2

If all comparison results stored in d are 1, the block comparison signal SM704 is set.

If the device designated by d is already set (1), that device will not change. If the conditions designated by s1 and s2 are changed and the BKCMP\_P instruction is executed, the device designated by d should be reset (0) before.

### Operation Errors

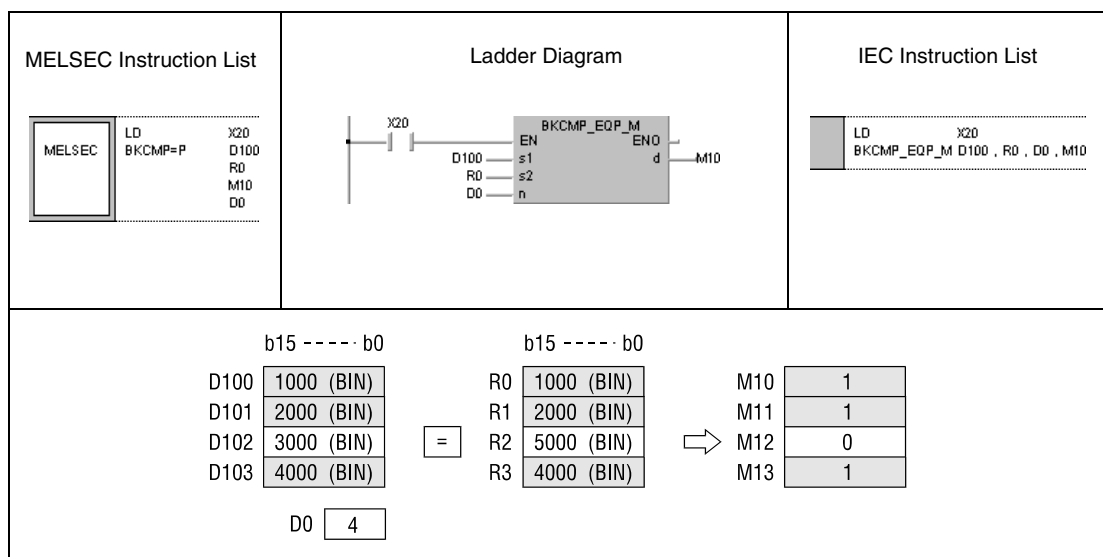
In following case an operation error occurs and the error flag is set:

- The BIN block data at s1, s2, or d exceeds the relevant device range (error code: 4101).
- The device range from [s1 to (s1) + (n-1)] overlaps with the device range [d to (d) + (n-1)] (error code: 4101).
- The device range from [s2 to (s2) + (n-1)] overlaps with the device range [d to (d) + (n-1)] (error code: 4101).
- The device range from [s1 to (s1) + (n-1)] overlaps with the device range [s2 to (s2) + (n-1)] (error code: 4101).

### Program Example 1

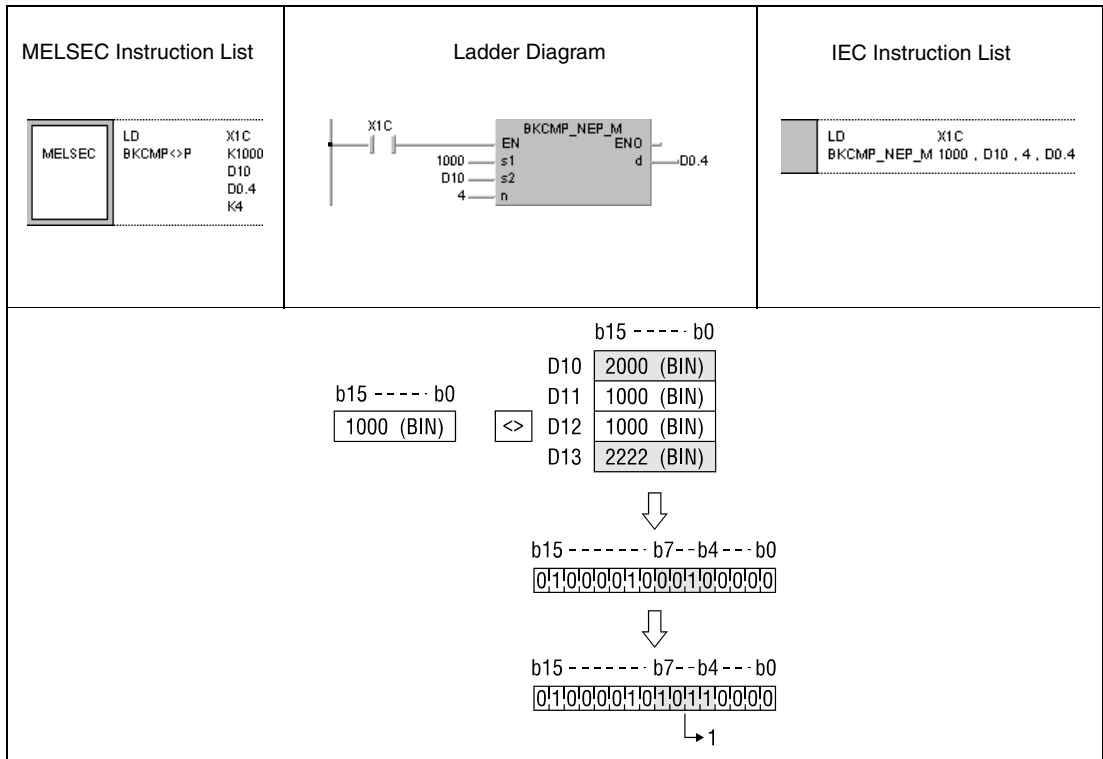
Comparison operation instruction BKCMP=P

With leading edge from X20, the following program compares BIN block data in D100 to BIN block data in R0. The results of the comparison are stored from M10 onward. The number of blocks (4) to be compared is stored in D0



**Program Example 2** Comparison operation instruction BKCOMP<>P

With leading edge from X1C, the following program compares the constant K1000 to the block data beginning from D10. The number of blocks (4) to be compared is determined by the constant K4. The results of the comparison are stored in b4 through b7 of D0.

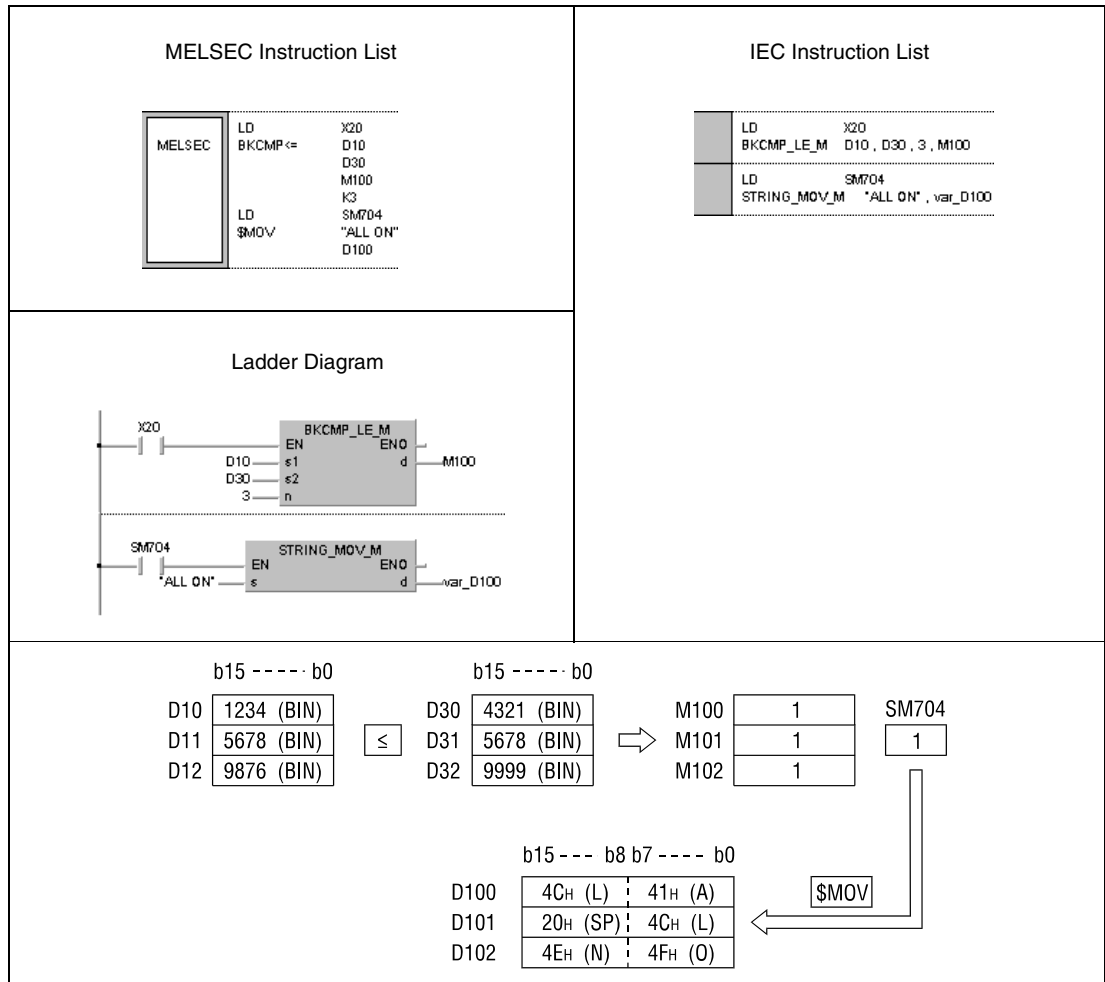


<sup>1</sup> Bits already in this state do not change (see function).

**Program Example 3**

Comparison operation instruction BKCOMP<=

As long as X20 is set, the following program compares block data beginning from D10 to block data beginning from D30. The number of blocks (3) to be compared is determined by the constant K3. The results of the comparison are stored from M100 onward. When all comparison results stored in M100 are 1, the block comparison signal SM704 is set and the character string "ALL ON" is transferred to D100.



**NOTE**

This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

## 6.2 Arithmetic Operation Instructions

Arithmetic operation instructions perform simple calculations like addition, subtraction, multiplication, and division.

The total number of arithmetic operation instructions is 54 (Q series and System Q) and 40 (A series) respectively.

Function	BIN		BCD	
	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
+	+	PLUS_M, PLUS_3_M	B+	BPLUS_M, BPLUS_3_M
	+P	PLUSP_M, PLUSP_3_M	B+P	BPLUSP_M, BPLUSP_3_M
	D+	DPLUS_M, DPLUS_3_M	DB+	DBPLUS_M, DBPLUS_3_M
	D+P	DPLUSP_M, DPLUSP_3_M	DB+P	DBPLUSP_M, DBPLUSP_3_M
—	-	MINUS_M, MINUS_3_M	B-	BMINUS_M, BMINUS_3_M
	-P	MINUSP_M, MINUSP_3_M	B-P	BMINUSP_M, BMINUSP_3_M
	D-	DMINUS_M, DMINUS_3_M	DB-	DBMINUS_M, DBMINUS_3_M
	D-P	DMINUSP_M, DMINUSP_3_M	DB-P	DBMINUSP_M, DBMINUSP_3_M
×	×	MULTI_3_M	B×	BMULTI_M
	×P	MULTIP_3_M	B×P	BMULTIP_M
	D×	DMULTI_3_M	DB×	DBMULTI_M
	D×P	DMULTIP_3_M	DB×P	DBMULTIP_M
/	/	DIVID_3_M	B/	BDIVID_M
	/P	DIVIDP_3_M	B/P	BDIVIDP_M
	D/	DDIVID_3_M	DB/	DBDIVID_M
	D/P	DDIVIDP_3_M	DB/P	DBDIVIDP_M
+1	INC	INC_M		
	INCP	INCP_M		
	DINC	DINC_M		
	DINCP	DINCP_M		
-1	DEC	DEC_M		
	DECP	DECP_M		
	DDEC	DDEC_M		
	DDECP	DDECP_M		

**NOTE** Within the IEC editors please use the IEC commands.

Function	Floating Point Data		BIN Block Data	
	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
+	E+	EPLUS_M, EPLUS_3_M	BK+	BKPLUS_M
	E+P	EPLUSP_M, EPLUSP_3_M	BK+P	BKPLUSP_M
-	E-	EMINUS_M, EMINUS_3_M	BK-	BKMINUS_M
	E-P	EMINUSP_M, EMINUSP_3_M	BK-P	BKMINUSP_M
×	E×	EMUL_M		
	E×P	EMULP_M		
/	E/	EDIV_M		
	E/P	EDIVP_M		

Function	Character String Data	
	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
+	\$+	STRING_PLUS_M, STRING_PLUS_3_M
	\$+P	STRING_PLUSP_M, STRING_PLUSP_3_M

**NOTE** *Within the IEC editors please use the IEC commands.*

The arithmetic operation instructions for floating point data, BIN block data, and character string data are only available with the Q series and the System Q.

**BIN data arithmetic operation instructions**

If the result of the addition exceeds a BIN value 32767 (2147483647 for a 32-bit instruction), a negative value is generated (overflow).

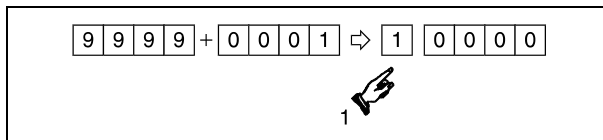
If the result of the subtraction falls below a BIN value -32768 (-2147483647 for a 32-bit instruction), a positive value is generated (underflow).

The calculation of positive and negative values appears as follows:

- 5 + 8 = 13
- 5 - 8 = -3
- 5 × 3 = 15
- 5 × 3 = -15
- 5 × (-3) = 15
- 5 / 3 = 1 remainder 2
- 5 / 3 = -1 remainder -2
- 5 / (-3) = -1 remainder 2
- 5 / (-3) = 1 remainder -2

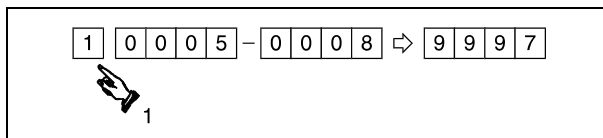
**BCD data arithmetic operation instructions**

If the result of the addition exceeds 9999 (99999999 for a 32-bit instruction), the higher bits are ignored (overflow). The carry flag in this case is not set.



<sup>1</sup> Carry ignored

If the result of the subtraction falls below 0000 (underflow), the carry is processed as shown:



<sup>2</sup> Carry

6.2.1 +, +P, -, -P

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices								Word Devices (16-bit)								Constant		Pointer					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K4	5	●	●
d		●	●	●	●	●	●	●	●	●	●	●	●	●						1				
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				7				
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				1				
d1		●	●	●	●	●	●	●	●	●	●	●	●	●										

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

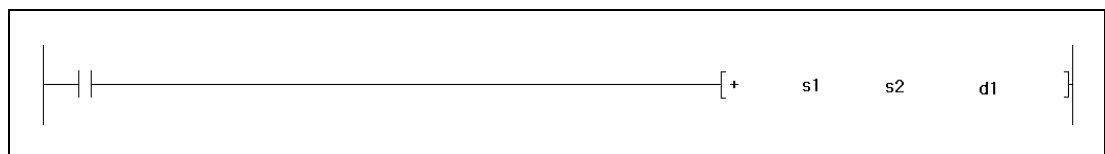
Devices  
MELSEC Q

	Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	3	
d	●	●	●	●	●	●	●	—	—		
s1	●	●	●	●	●	●	●	●	—	4	
s2	●	●	●	●	●	●	●	●	—		
d1	●	●	●	●	●	●	●	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%; text-align: center;">MELSEC</td> <td style="width: 20%;">\$+</td> <td style="width: 20%; text-align: right;">s</td> <td style="width: 20%; text-align: right;">d</td> </tr> <tr> <td style="text-align: center;">MELSEC</td> <td>\$+</td> <td style="text-align: right;">s1</td> <td style="text-align: right;">s2</td> </tr> <tr> <td></td> <td></td> <td style="text-align: right;">d1</td> <td></td> </tr> </table>	MELSEC	\$+	s	d	MELSEC	\$+	s1	s2			d1		<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 20%; text-align: center;">PLUS_3_M</td> <td style="width: 20%; text-align: right;">s1, s2, d1</td> </tr> </table>	PLUS_3_M	s1, s2, d1
MELSEC	\$+	s	d													
MELSEC	\$+	s1	s2													
		d1														
PLUS_3_M	s1, s2, d1															

GX Developer





**Variables**

Set Data	Meaning	Data Type
s	Addition or subtraction data, or first number of device storing addition or subtraction data	BIN 16-bit
d	Data to be added to or subtracted from, or first number of device storing such data	
s1	Data to be added to or subtracted from, or first number of device storing such data	
s2	Addition or subtraction data, or first number of device storing addition or subtraction data	
d1	First number of device storing addition or subtraction data	

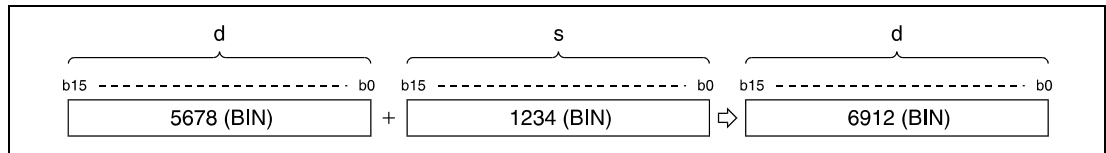
**Functions**

**BIN 16-bit addition and subtraction operations**

**+ BIN addition (16-bit)**

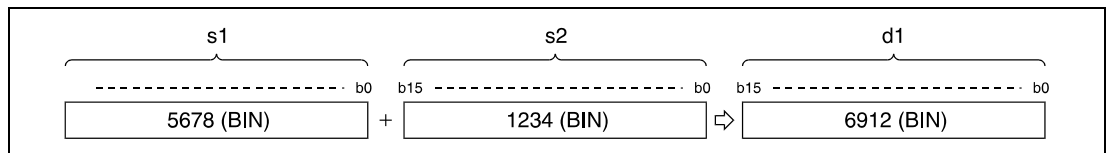
● Variation 1:

BIN 16-bit data in d is added to BIN 16-bit data in s. The result of the addition is stored in d.



● Variation 2:

BIN 16-bit data in s1 is added to BIN 16-bit data in s2. The result of the addition is stored in d1.



BIN 16-bit data designated by s, d, s1, s2, and d1 have to range within -32768 and 32767.

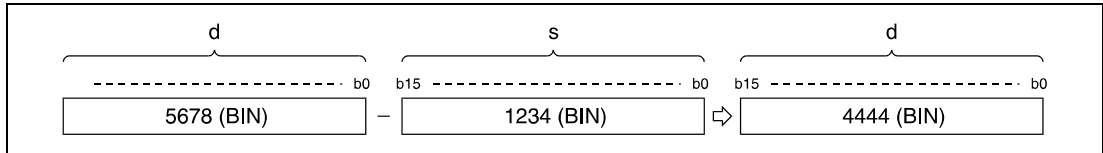
The most significant bit (b15) determines, whether data in s, d, s1, or d1 are positive (bit = 0) or negative (bit = 1).

If the least significant bit (b0) is fallen below or the most significant bit (b15) is exceeded, the carry flag is not set.

**- BIN subtraction (16-bit)**

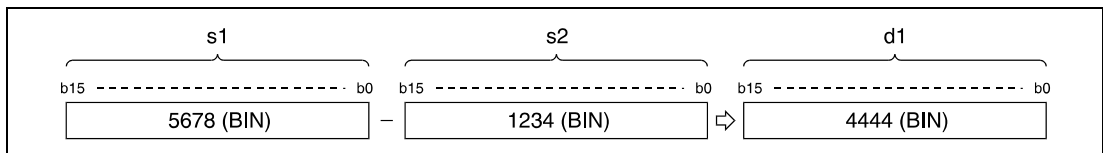
● Variation 1:

BIN 16-bit data in s is subtracted from BIN 16-bit data in d. The result of the subtraction is stored in d.



● Variation 2:

BIN 16-bit data in s2 is subtracted from BIN 16-bit data in s1. The result is stored in d1.



BIN 16-bit data designated by s, d, s1, s2, and d1 have to range within -32768 and 32767.

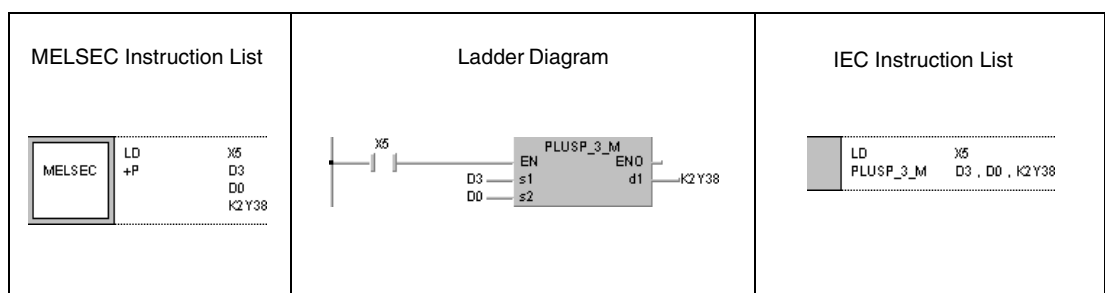
The most significant bit (b15) determines, whether data in s, d, s1, or d1 are positive (bit = 0) or negative (bit = 1).

If the least significant bit (b0) is fallen below or the most significant bit (b15) is exceeded, the carry flag is not set.

**Program Example 1**

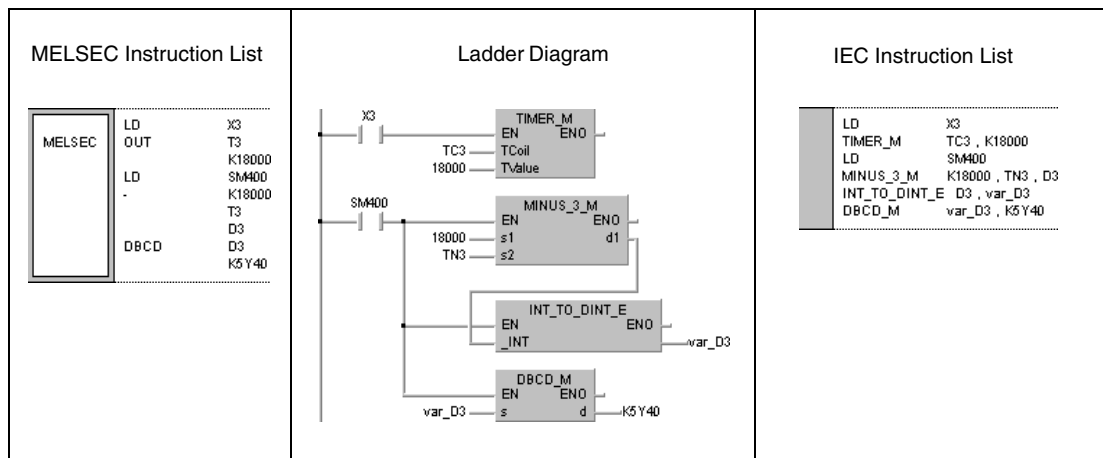
+P

With leading edge from X5, the following program adds data in D3 to data in D0. The result is stored from Y38 to Y3F.



**Program Example 2** -

The following program outputs the difference between the nominal and the actual value of timer T3 to Y40 through Y53 in BCD.



**NOTE** *This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.2.2 D+, D+P, D-, D-P

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices								Word Devices (16-bit)								Constant		Pointer					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K8	9	●	●
d		●	●	●	●	●	●	●	●	●	●	●	●	●						11				
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				11				
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				11				
d1		●	●	●	●	●	●	●	●	●	●	●	●	●						11				

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps

Devices  
MELSEC Q

	Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	—	3 <sup>1)</sup>
d	●	●	●	●	●	●	●	—	—		
s1	●	●	●	●	●	●	●	●	—	—	4 <sup>2)</sup>
s2	●	●	●	●	●	●	●	●	—	—	
d1	●	●	●	●	●	●	●	—	—	—	

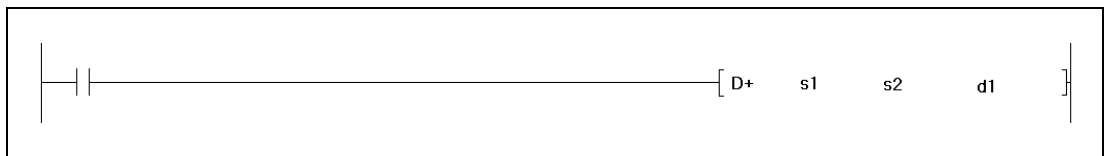
<sup>1</sup> The number of steps depends on the device and the type of CPU.  
 If a QnA-CPU or a System Q single processor CPU is used: 3  
 If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 5  
 constants: 5  
 Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 5  
 If a System Q multi processor CPU is used with devices other than above mentioned: 3

<sup>2</sup> The number of steps depends on the device and the type of CPU.  
 If a QnA-CPU or a System Q single processor CPU is used: 4  
 If a System Q multi processor CPU is used with internal word devices (except for file register ZR): 6  
 constants: 6  
 Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification: 6  
 If a System Q multi processor CPU is used with devices other than above mentioned: 4

**GX IEC Developer**

MELSEC Instruction List	Ladder Diagram	IEC Instruction List														
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 100px;">MELSEC</td> <td style="width: 50px;">D+</td> <td style="width: 50px;">s</td> <td style="width: 50px;">d</td> </tr> <tr> <td style="text-align: center;">MELSEC</td> <td>D+</td> <td>s1</td> <td>s2</td> </tr> <tr> <td></td> <td></td> <td></td> <td>d1</td> </tr> </table>	MELSEC	D+	s	d	MELSEC	D+	s1	s2				d1		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 100px;">DPLUS_3_M</td> <td>s1, s2, d1</td> </tr> </table>	DPLUS_3_M	s1, s2, d1
MELSEC	D+	s	d													
MELSEC	D+	s1	s2													
			d1													
DPLUS_3_M	s1, s2, d1															

**GX Developer**



**Variables**

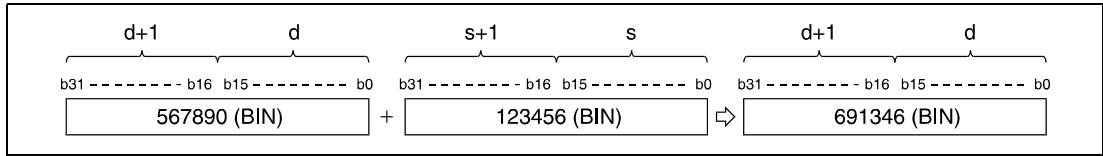
Set Data	Meaning	Data Type
s	Addition or subtraction data, or first number of device storing addition or subtraction data	BIN 32-bit
d	Data to be added to or subtracted from, or first number of device storing such data	
s1	Data to be added to or subtracted from, or first number of device storing such data	
s2	Addition or subtraction data, or first number of device storing addition or subtraction data	
d1	First number of device storing addition or subtraction data	

**Functions BIN 32-bit addition and subtraction operations**

**D+ BIN addition (32-bit)**

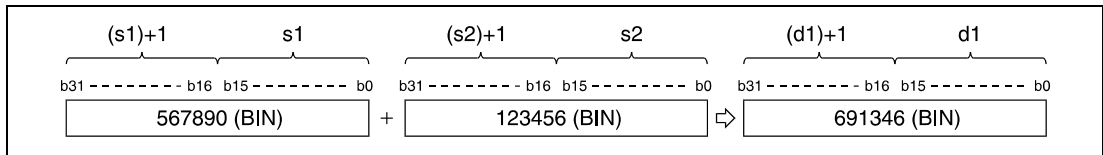
- Variation 1:

BIN 32-bit data in d is added to BIN 32-bit data in s. The result of the addition is stored in d.



- Variation 2:

BIN 32-bit data in s1 is added to BIN 32-bit data in s2. The result of the addition is stored in d1.



BIN 32-bit data designated by s, d, s1, s2, and d1 have to range within -2147483648 and 2147483647.

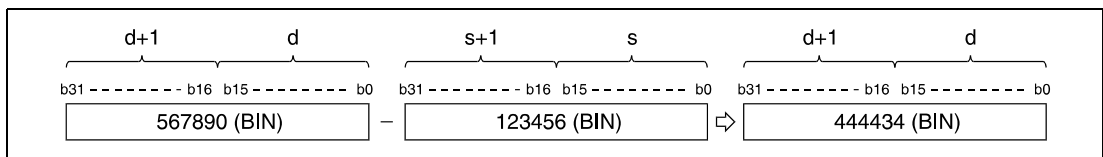
The most significant bit (b31) determines, whether data in s, d, s1, or d1 are positive (bit = 0) or negative (bit = 1).

If the least significant bit (b0) is fallen below or the most significant bit (b31) is exceeded, the carry flag is not set.

**D- BIN subtraction (32-bit)**

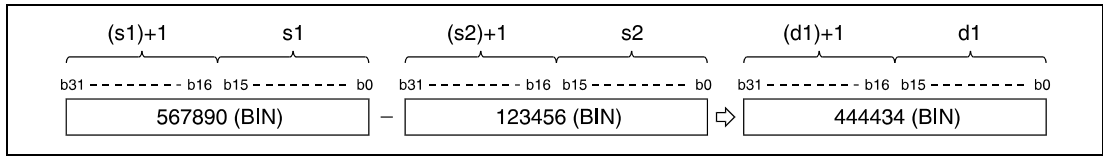
- Variation 1:

BIN 32-bit data in s is subtracted from BIN 32-bit data in d. The result of the subtraction is stored in d.



● Variation 2:

BIN 32-bit data in s2 is subtracted from BIN 32-bit data in s1. The result is stored in d1.



BIN 32-bit data designated by s, d, s1, s2, and d1 have to range within -2147483648 and 2147483647.

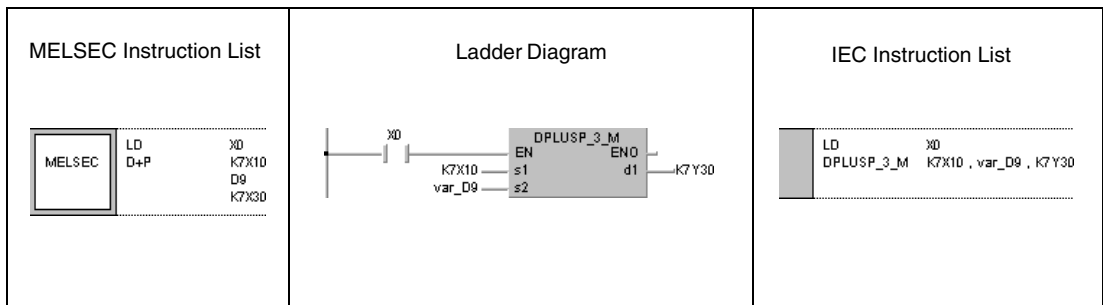
The most significant bit (b31) determines, whether data in s, d, s1, or d1 are positive (bit = 0) or negative (bit = 1).

If the least significant bit (b0) is fallen below or the most significant bit (b31) is exceeded, the carry flag is not set.

**Program Example 1**

D+P

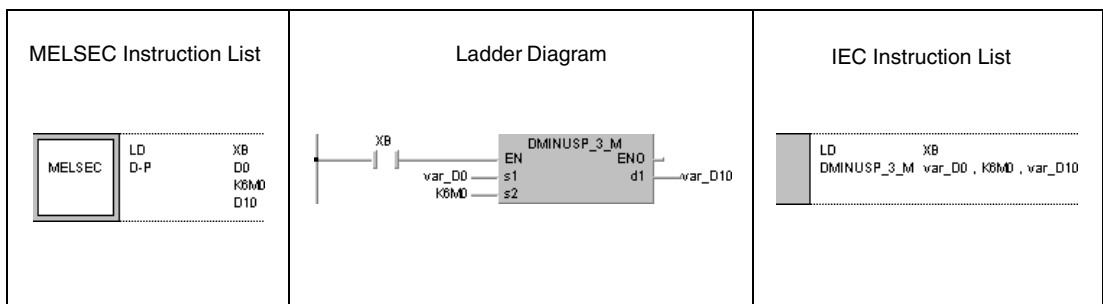
With leading edge from X0, the following program adds data in X10 through X2B to D9 and D10. The result is stored in Y30 through Y4B.



**Program Example 2**

D-P

With leading edge from XB, the following program subtracts data in M0 through M23 from data in D0 and D1. The result is stored in D10 and D11.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.2.3 x, xP, /, /P

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Devices																			Digit designation K1 ↓ K4	Number of steps 7 <sup>1</sup>	Index ●	Carry Flag M9012	Error Flag M9010 M9011
	Bit Devices						Word Devices (16-bit)						Constant		Pointer		Level							
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							
d1		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

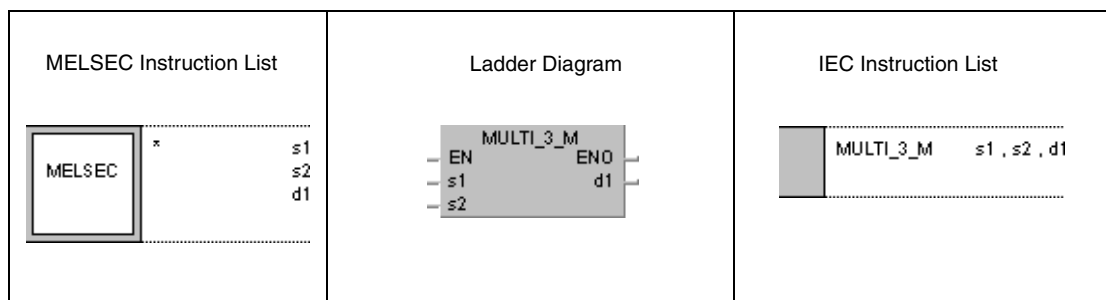
Devices  
MELSEC Q

	Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module Index Register U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	SM0	4 <sup>1)</sup>
s2	●	●	●	●	●	●	●	—			
d1	●	●	●	●	●	●	—	—			

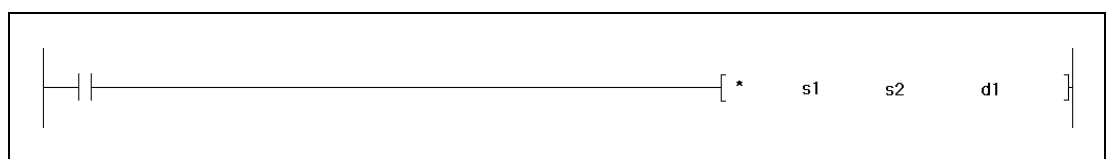
<sup>1</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU is used: 4
- If a CPU of the System Q is used with internal word devices (except for file register ZR): 3
- constants: 3
- Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K4, and which use no index qualification: 3
- If a CPU of the System Q is used with devices other than above mentioned: 4

GX IEC  
Developer



GX  
Developer





**Variables**

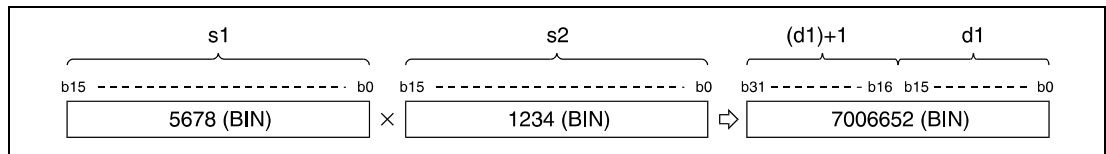
Set Data	Meaning	Data Type
s1	Data that will be multiplied or divided, or first number of device storing data that will be multiplied or divided	BIN 16-bit
s2	Data to multiply or divide by, or first number of device storing such data	BIN 16-bit
d1	First number of device storing the operation results of multiplication or division operation	BIN 32-bit

**Functions**

**BIN 16-bit multiplication and division**

**x BIN multiplikation (16-bit)**

BIN 16-bit data in s1 is multiplied with BIN 16-bit data in s2. The result is stored in d1.



If the result in d1 is a bit device, designation is made from the lower bits.

Example:

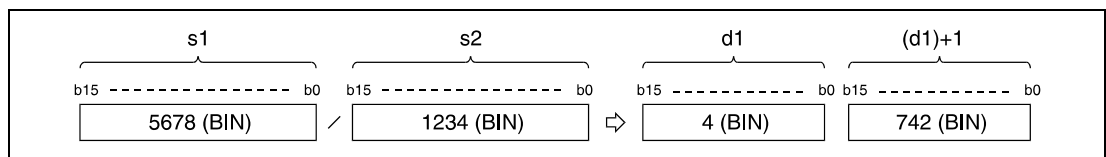
- K1: lower 4 bits (b0 to b3)
- K4: lower 16 bits (b0 to b15)
- K8: 32 bits (b0 to b31)

BIN 16-bit data designated by s1 and s2 have to range within -32768 and 32767.

The most significant bit (b15 or b31) in d1 determines, whether data in s1, s2 or d1 are positive (bit = 0) or negative (bit = 1).

**/ BIN division (16-bit)**

BIN 16-bit data in s1 is divided by BIN 16-bit data in s2. The result is stored in d1.



If a word device is used, the result of the operation is stored as 32-bits, and both, the quotient and remainder are stored. The quotient is stored in the least significant 16-bits. The remainder is stored in the most significant 16-bits.

If a bit device is used, 16-bits are used and only the quotient is stored.

BIN 16-bit data designated by s1 and s2 have to range within -32768 and 32767.

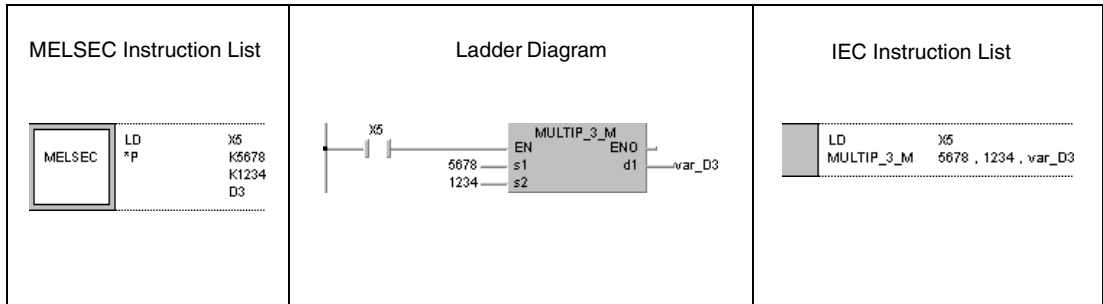
The most significant bit (b15) in d1 determines, whether data in s1, s2, d1 or (d1)+1 is positive (bit = 0) or negative (bit = 1).

**Operation Errors** In the following cases an operation error occurs and the error flag is set:

- A1 or V were designated by d1 (A Series).
- Division by 0 (Q Series and System Q = error code 4100).

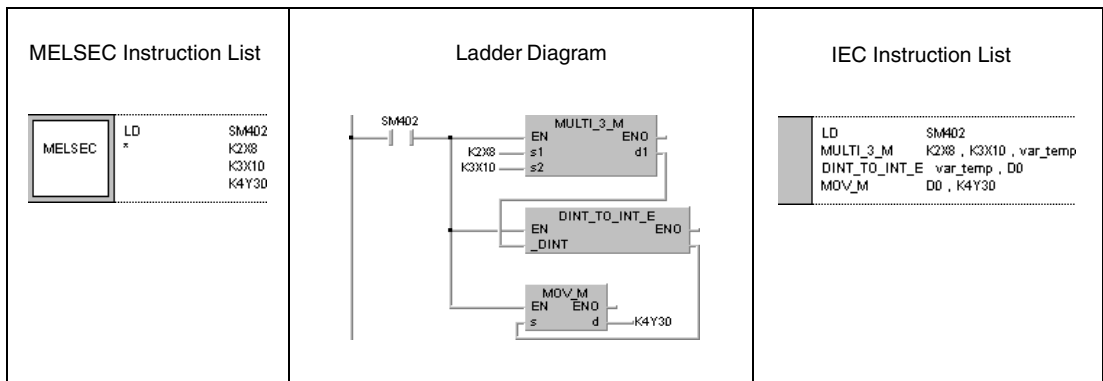
**Program Example 1**

xP  
 With leading edge from X5, the following program multiplies 5678 and 1234. The result is stored in D3 and D4.



**Program Example 2**

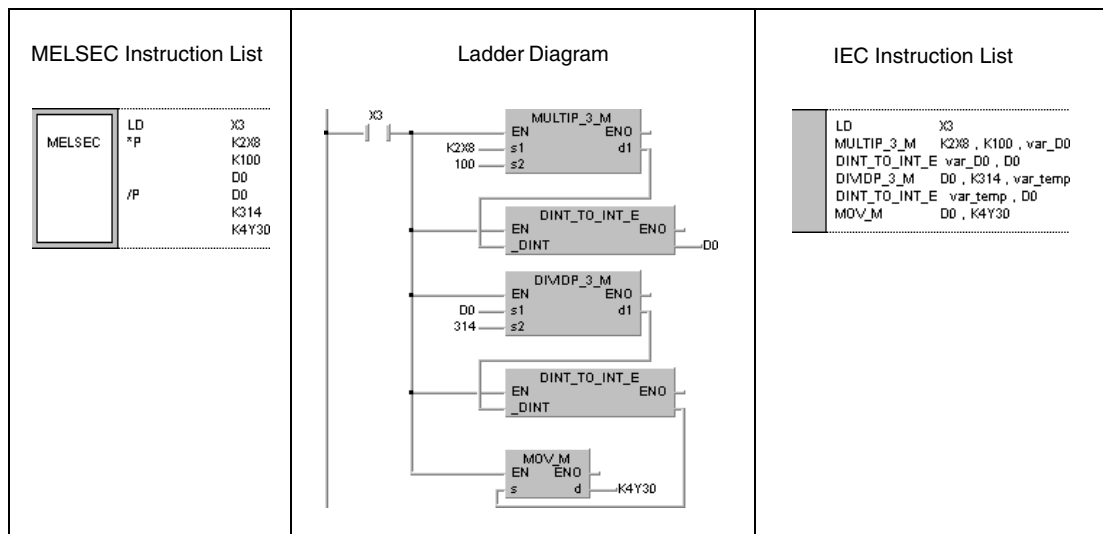
x  
 The following program multiplies BIN data at X8 through XF and BIN data at X10 through X1B. The result is output at Y30 through Y3F.



**Program** /P

**Example 3**

With leading edge from X3, the following program divides data at X8 through XF by 3.14. The result is output at Y30 through Y3F.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.2.4 Dx, DxP, D/, D/P

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

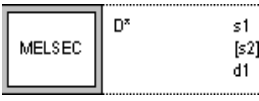
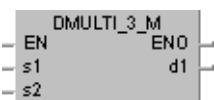
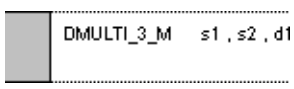
	Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices								Word Devices (16-bit)								Constant		Pointer					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K8	11 ● <sup>1</sup>	●	●
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
d1		●	●	●	●	●	●	●	●	●	●													

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

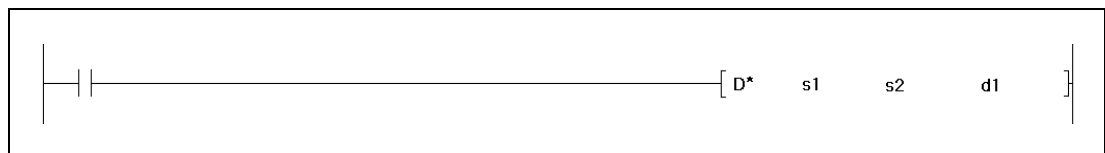
Devices  
MELSEC Q

	Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module Special Function Module U□NG□	Index Register Index Register Zn	Constant Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	SM0	4
s2	●	●	●	●	●	●	●	—			
d1	●	●	●	—	—	—	—	—			

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

GX Developer



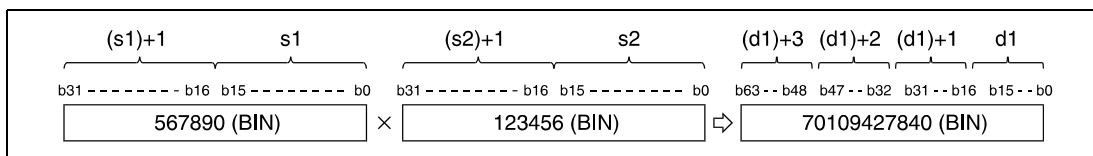
Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s1	Data that will be multiplied or divided, or first number of device storing data that will be multiplied or divided	BIN 32-bit	ANY32
s2	Data to multiply or divide by, or first number of device storing such data	BIN 32-bit	ANY32
d1	First number of device storing the operation results of multiplication or division operation	BIN 64-bit	Array [1..2] of ANY32

**Functions BIN 32-bit multiplication and division**

**Dx BIN multiplication (32-bit)**

BIN 32-bit data in s1 is multiplied with BIN 32-bit data in s2. The result is stored in d1.



If the result in d1 is a bit device, designation is made from the lower bits.

Example:

- K1: lower 4 bits (b0 to b3)
- K4: lower 16 bits (b0 to b15)
- K8: 32 bits (b0 to b31)

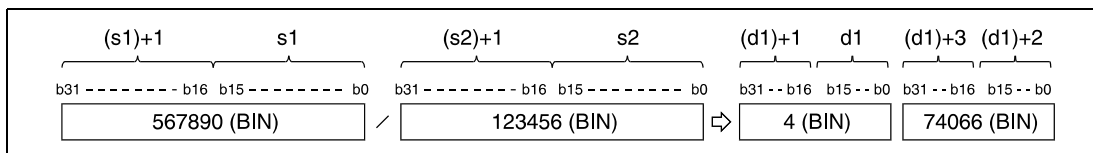
If the upper 32 bits of the bit device are required for the result of the multiplication operation, first temporarily store the data in a word device, then transfer the word device data to the bit device designated by (d1)+2 and (d1)+3.

BIN 32-bit data designated by s1 and s2 has to range within -2147483648 and 2147483647.

The most significant bit (b31 or b63) in d1 determines, whether data in s1, s2 or d1 is positive (bit = 0) or negative (bit = 1).

**D/ BIN division (32-bit)**

BIN 32-bit data in s1 is divided by BIN 32-bit data in s2. The result is stored in d1.



If a word device is used, the result of the division operation is stored as array of DINT (64-bit), and both the quotient and remainder are stored. The quotient is stored in the lower array elements (32-bit). The remainder is stored in the upper array elements (32-bit).

If a bit device is used, 32 bits are used and only the quotient is stored.

BIN 32-bit data designated by s1 and s2 has to range within -2147483648 and 2147483647.

The most significant bit (b31) in d1 determines, whether data in s1, s2, d1 or (d1)+2 is positive (bit = 0) or negative (bit = 1).

**Operation Errors**

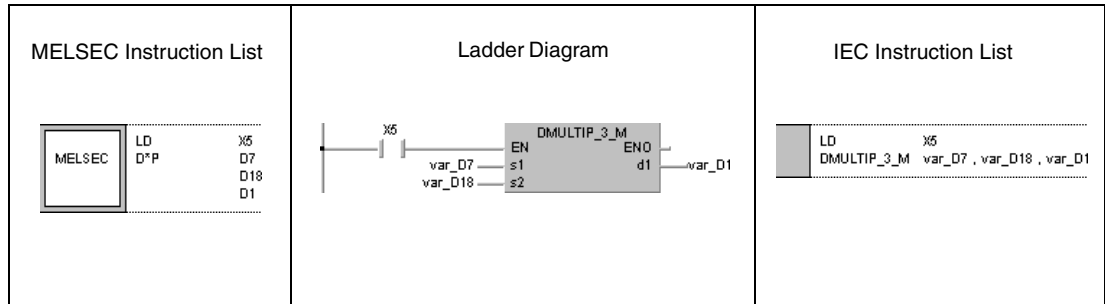
In the following cases an operation error occurs and the error flag is set:

- A1 or V were designated by s1 or s2 and A0, A1, Z, and V were designated by d1 (A Series).
- Division by 0 (Q Series and System Q = error code 4100).

**Program Example 1**

DxP

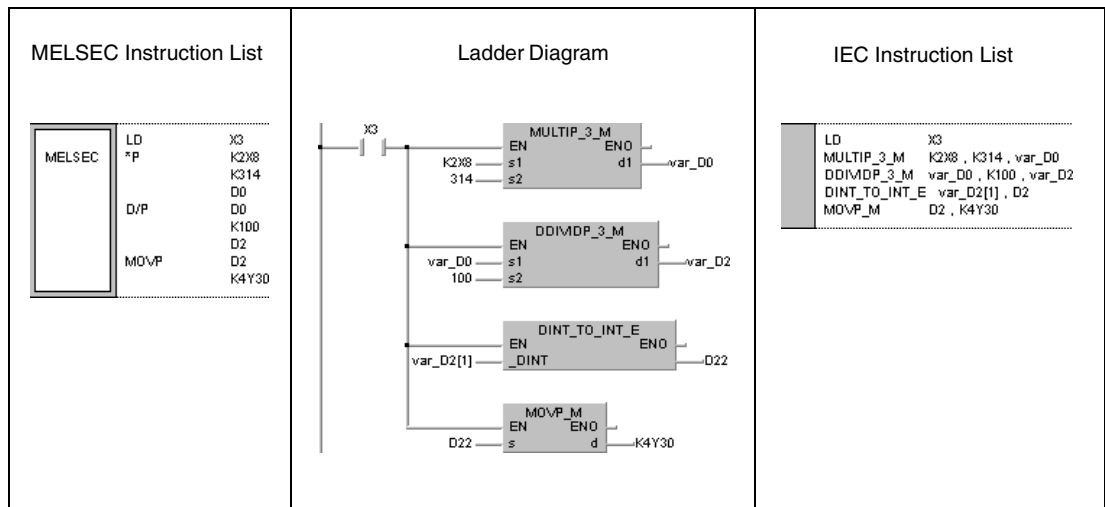
With leading edge from X5, the following program multiplies BIN data in D7 and D8 with BIN data in D18 and D19. The result is stored in D1 through D4.



**Program Example 2**

xP

With leading edge from X3, the following program multiplies data at X8 through XF and 3.14. The result is output at Y30 through Y3F



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.2.5 B+, B+P, B-, B-P

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

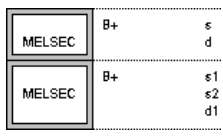
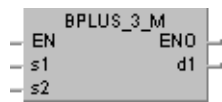
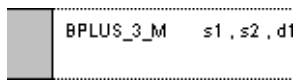
	Usable Devices															Digit Designation	Number of steps	Index	Carry Flag		Error Flag					
	Bit Devices					Word Devices (16-bit)					Constant		Pointer		Level				M9012	M9010 M9011						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z						V	K	H (16#)	P	I	N
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K4	7 ● <sup>1</sup> 9 ● <sup>1</sup>	●		
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●											
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									
d1		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●										

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

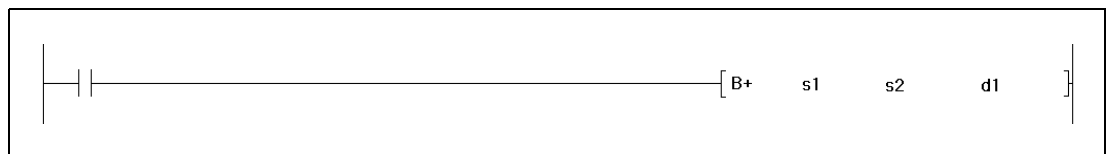
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	●	●	●	●	●	●	●	—			
s1	●	●	●	●	●	●	●	●	—	SM0	4
s2	●	●	●	●	●	●	●	●			
d1	●	●	●	●	●	●	●	—			

GX IEC  
Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

GX  
Developer



Variables

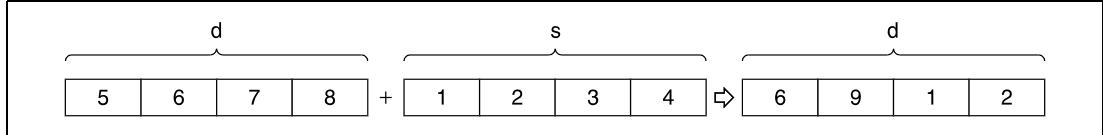
Set Data	Meaning	Data Type
s	Addition or subtraction data, or first number of device storing addition or subtraction data	BCD 4-digit
d	Data to be added to or subtracted from, or first number of device storing such data	
s1	Data to be added to or subtracted from, or first number of device storing such data	
s2	Addition or subtraction data, or first number of device storing addition or subtraction data	
d1	First number of device storing addition or subtraction data	

**Functions BCD 4-digit addition and subtraction operations**

**B+ BCD addition (4-digit)**

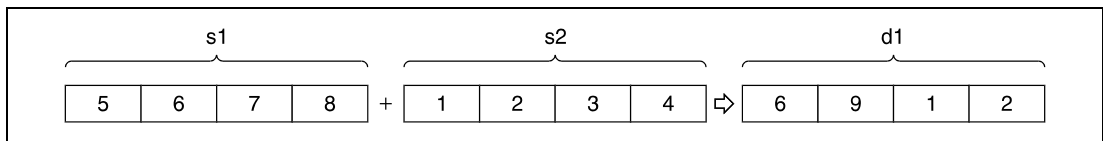
● Variation 1:

BCD 4-digit data in d is added to BCD 4-digit data in s. The result of the addition is stored in d.



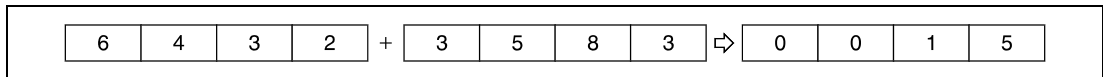
● Variation 2:

BCD 4-digit data in s1 is added to BCD 4-digit data in s2. The result is stored in d1.



BCD 4-digit data designated by s, d, s1, s2, and d1 have to range within 0 and 9999. Undesignated digits are read as 0 (e.g. 12 = 0012).

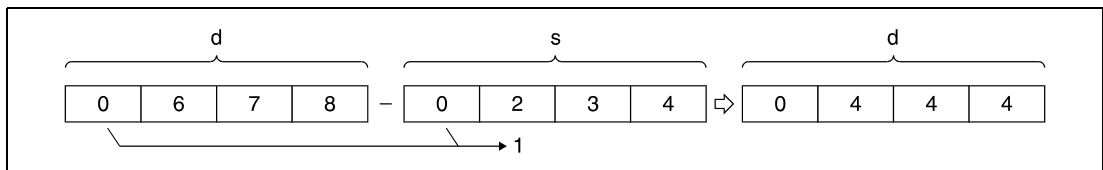
If the result of the addition exceeds 9999, the higher bits are ignored (overflow). The carry flag in this case is not set.



**B- BCD subtraction (4-digit)**

● Variation 1:

BCD 4-digit data in s is subtracted from BCD 4-digit data in d. The result is stored in d.

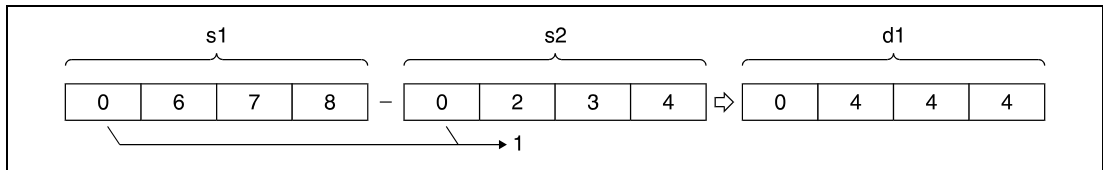


<sup>1</sup> Undesignated digits are read as 0.



● Variation 2:

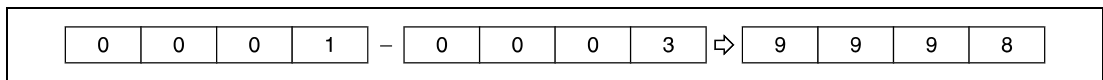
BCD 4-digit data in s2 is subtracted from BCD 4-digit data in s1. The result is stored in d1.



<sup>1</sup> Undesignated digits are read as 0.

BCD 4-digit data designated by s, d, s1, s2, and d1 have to range within 0 and 9999.

If the result of the subtraction operation is negative, the minuend is reduced by the number of steps determined by the subtrahend. The carry flag in this case is not set.



In the further course of a program, make sure that either positive or negative results are treated adequately.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The BCD 4-digit data designated by s, d, s1, s2, or d1 exceed the relevant device range of 0 to 9999 (Q series and System Q = error code 4100).

# B+, B+P, B-, B-P

## Program Example 1

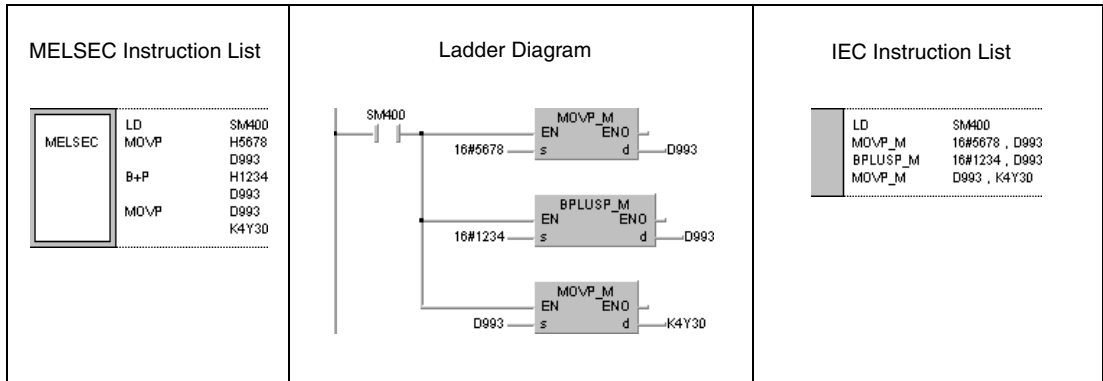
B+P (s, d)

The following program adds BCD data 5678 to BCD data 1234. The result is stored in D993 and output at Y30 through Y3F.

The first line of the program, with leading edge from SM400 stores the value 5678 in D993.

The following program step adds BCD data 1234 to BCD data in D993.

The MOV instruction in the last program step outputs the result in D993 at Y30 through Y3F.



## Program Example 2

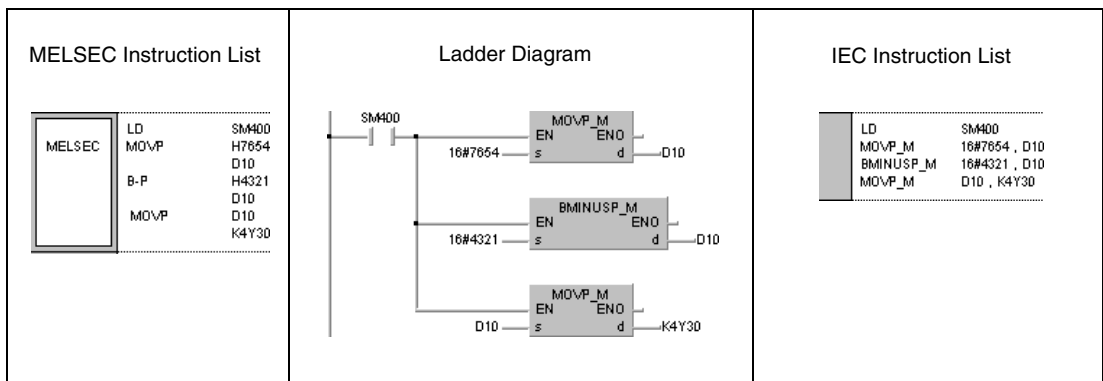
B-P (s, d)

The following program subtracts BCD data 4321 from BCD data 7654. The result is stored in D10 and output at Y30 through Y3F.

The first line of the program, with leading edge from SM400 stores the value 7654 in D10.

The following program step subtracts BCD data 4321 from BCD data in D10.

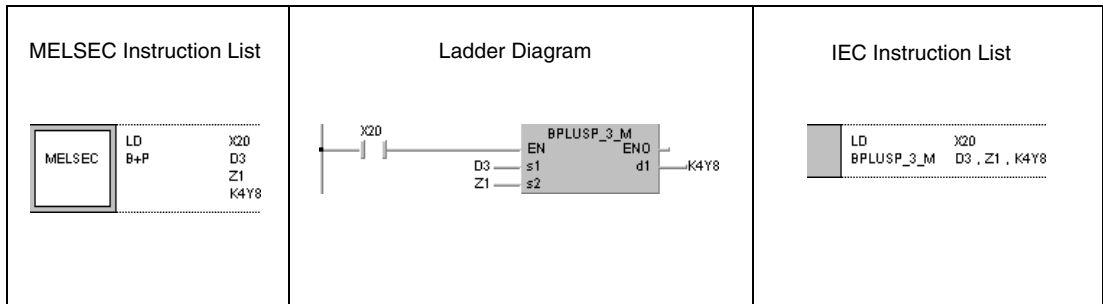
The MOV instruction in the last program step outputs the result in D10 at Y30 through Y3F.



**Program Example 3**

B+P (s1, s2, d1)

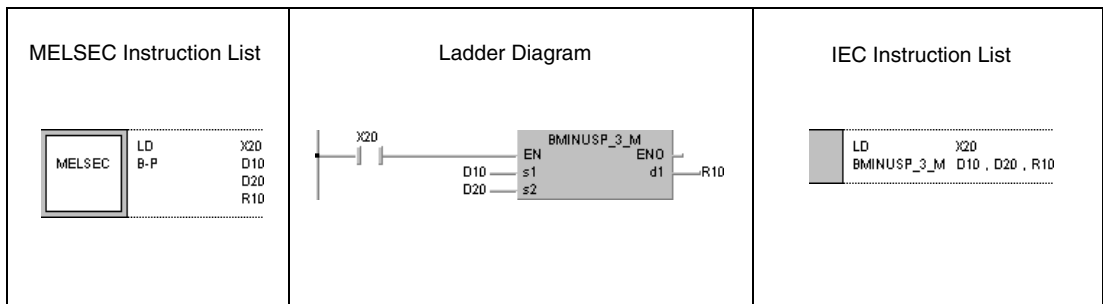
With leading edge from X20, the following program adds BCD data in D3 to BCD data in Z1. The result is output at Y8 through Y17.



**Program Example 4**

B-P (s1, s2, d1)

With leading edge from X20, the following program subtracts BCD data in D20 from BCD data in D10. The result is stored in R10.



# DB+, DB+P, DB-, DB-P

## 6.2.6 DB+, DB+P, DB-, DB-P

### CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

### Devices MELSEC A

	Usable Devices															Digit Designation	Number of steps	Index	Carry Flag		Error Flag					
	Bit Devices					Word Devices (16-bit)					Constant		Pointer		Level				M9012	M9010 M9011						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z						V	K	H (16#)	P	I	N
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K8	9 ● <sup>1</sup>	●	●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●											
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									
d1		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●										

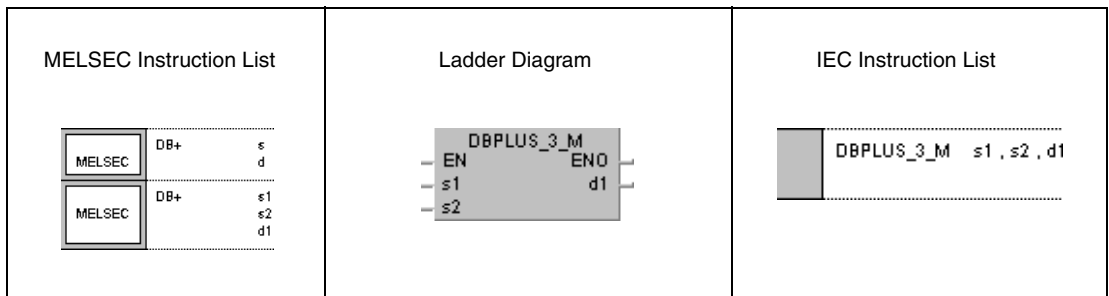
<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

<sup>2</sup> AnA, AnAS, and AnU CPUs only.

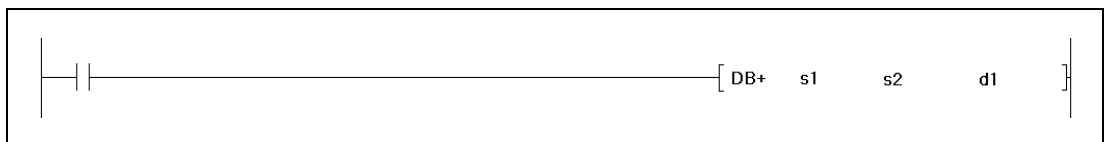
### Devices MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□□		Special Function Module U□G□□	Index Register Zn	Constants K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	●	●	●	●	●	●	●	—			
s1	●	●	●	●	●	●	●	●	—	SM0	4
s2	●	●	●	●	●	●	●	●			
d1	●	●	●	●	●	●	●	—			

### GX IEC Developer



### GX Developer



### Variables

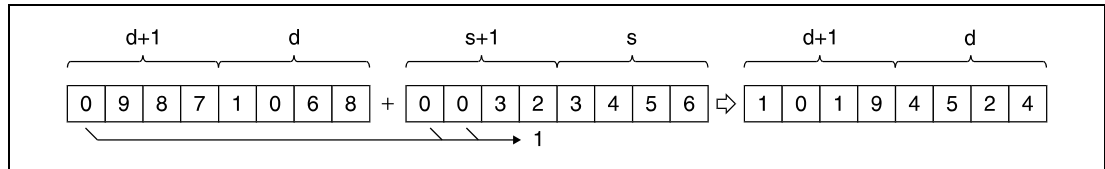
Set Data	Meaning	Data Type
s	Addition or subtraction data, or first number of device storing addition or subtraction data	BCD 8-digit
d	Data to be added to or subtracted from, or first number of device storing such data	
s1	Data to be added to or subtracted from, or first number of device storing such data	
s2	Addition or subtraction data, or first number of device storing addition or subtraction data	
d1	First number of device storing addition or subtraction data	

**Functions BCD 8-digit addition and subtraction operations**

**DB+ BCD addition (8-digit)**

● Variation 1:

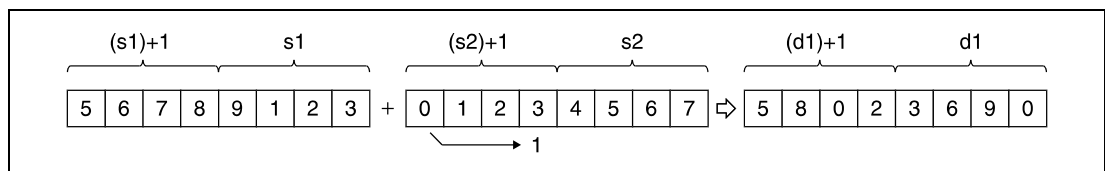
BCD 8-digit data in d is added to BCD 8-digit data in s. The result is stored in d.



<sup>1</sup> Undesignated digits are read as 0.

● Variation 2:

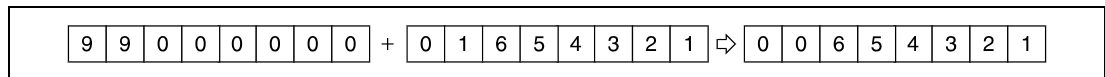
BCD 8-digit data in s1 is added to BCD 8-digit data in s2. The result is stored in d1.



<sup>1</sup> Undesignated digits are read as 0.

BCD 8-digit data designated by s, d, s1, and d1 have to range within 0 and 99999999. Undesignated digits are read as 0 (e.g. 12345 = 00012345).

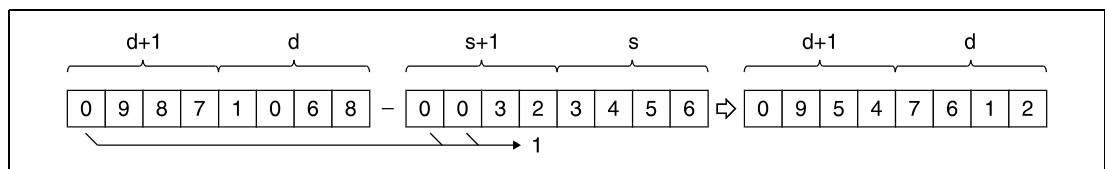
If the result of the addition exceeds 99999999, the higher bits are ignored (overflow). The carry flag in this case is not set.



**DB- BCD subtraction (8-digit)**

● Variation 1:

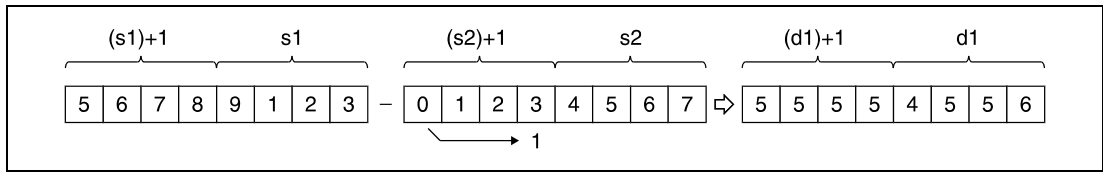
BCD 8-digit data in s is subtracted from BCD 8-digit data in d. The result is stored in d.



<sup>1</sup> Undesignated digits are read as 0

● Variation 2:

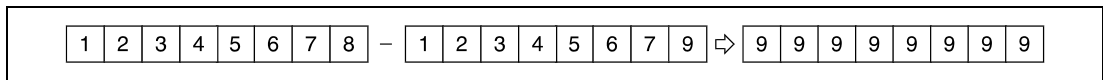
BCD 8-digit data in s2 is subtracted from BCD 8-digit data in s1. The result is stored in d1.



<sup>1</sup> Undesignated digits are read as 0

BCD 8-digit data designated by s, d, s1, and d1 have to range within 0 and 99999999. Undesignated digits are read as 0 (e.g. 12345 = 00012345).

If the result of the subtraction operation is negative, the minuend is reduced by the number of steps determined by the subtrahend. The carry flag in this case is not set.



In the further course of a program, make sure that either positive or negative results are treated adequately.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The BCD 8-digit data designated by s, d, s1, s2, or d1 exceed the relevant device range of 0 to 99999999 (Q series and System Q = error code 4100).

**Program Example 1**

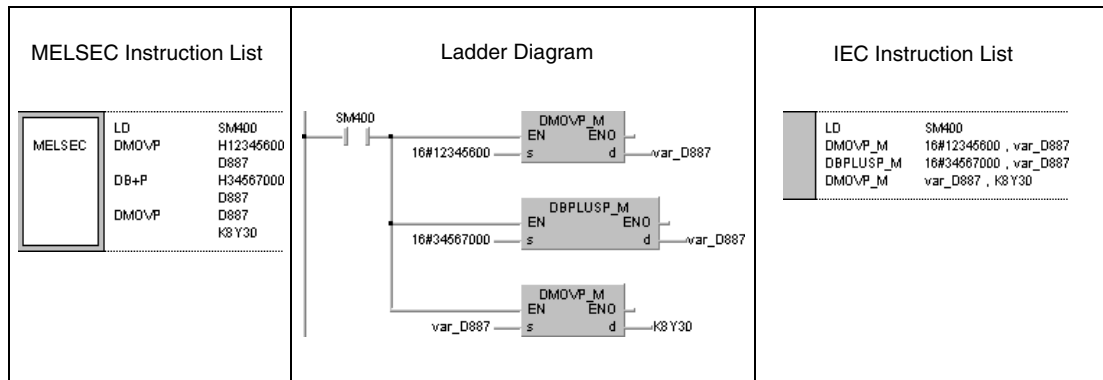
DB+P (s, d)

The following program adds BCD data 12345600 to BCD data 34567000. The result is stored in D887 and D888 and output at Y30 through Y4F.

The first line of the program, with leading edge from SM400 stores the value 12345600 in D887 and D888.

The following program step adds BCD data 34567000 to BCD data in D887 and D888.

The DMOV instruction in the last program step outputs the result in D887 and D888 at Y30 through Y4F.



**Program Example 2**

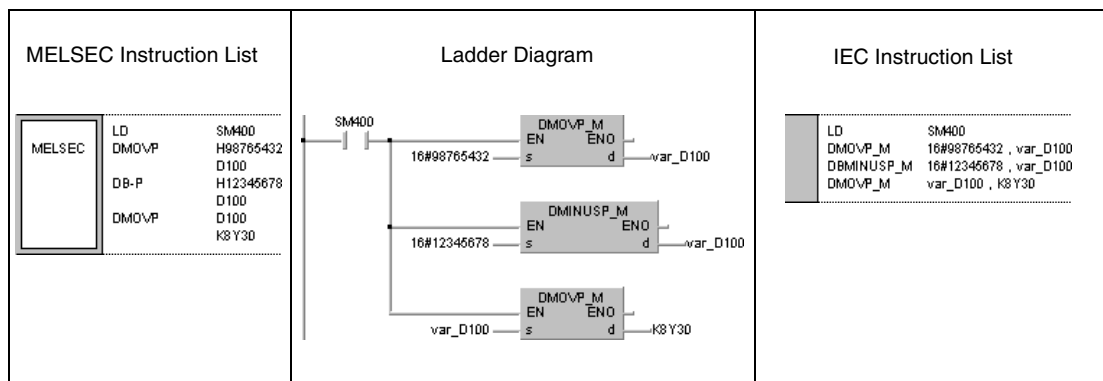
DB-P (s, d)

The following program subtracts BCD data 12345678 from BCD data 98765432. The result is stored in D100 and D101 and output at Y30 through Y4F.

The first line of the program, with leading edge from SM400 stores the value 98765432 in D100 and D101.

The following program step subtracts BCD data 12345678 from BCD data in D100 and D101.

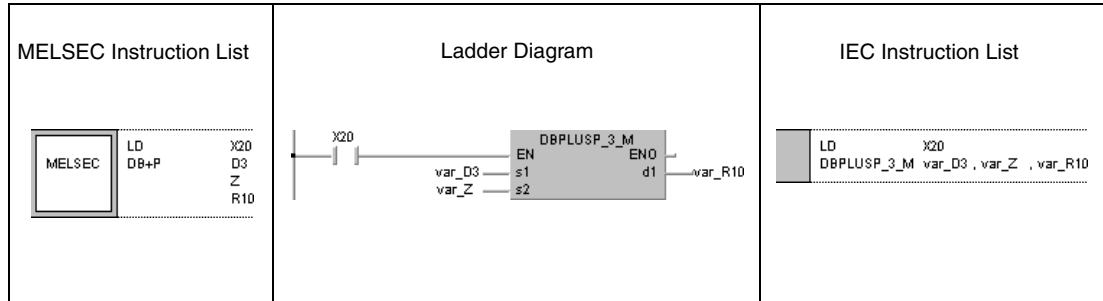
The DMOV instruction in the last program step outputs the result in D100 and D101 at Y30 through Y4F.



## DB+, DB+P, DB-, DB-P

**Program Example 3** DB+P (s1, s2, d1)

With leading edge from X20, the following program adds BCD data in D3 and D4 to BCD data in Z and V. The result is stored in R10 and R11.



### NOTE

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details refer to Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



6.2.7 Bx, BxP, B/, B/P

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

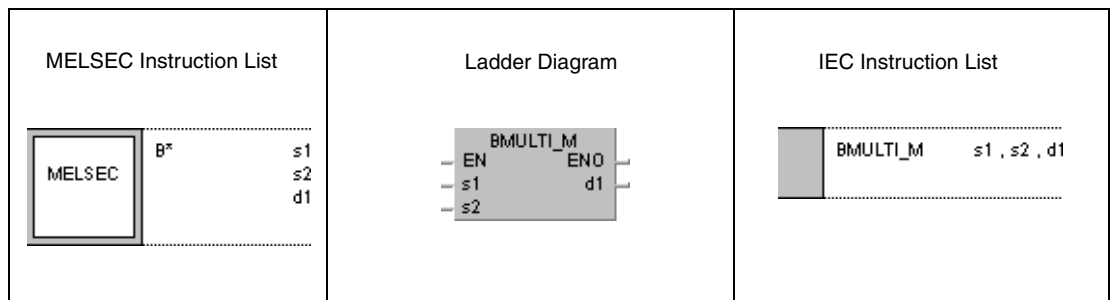
	Usable Devices															Digit Designation	Number of steps	Index	Carry Flag	Error Flag						
	Bit Devices					Word Devices (16-bit)					Constant	Pointer		Level												
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z						V	K	H (16#)	P	I	N
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K4	9 1	●		●
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K8				
d1		●	●	●	●	●	●	●	●	●	●			●												

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

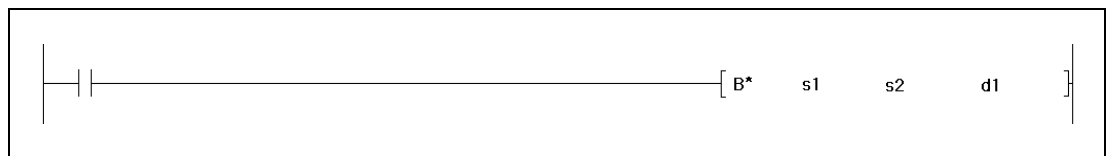
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	SM0	4
s2	●	●	●	●	●	●	●	●	—		
d1	●	●	●	●	●	●	—	—	—		

GX IEC  
Developer



GX  
Developer



Variables

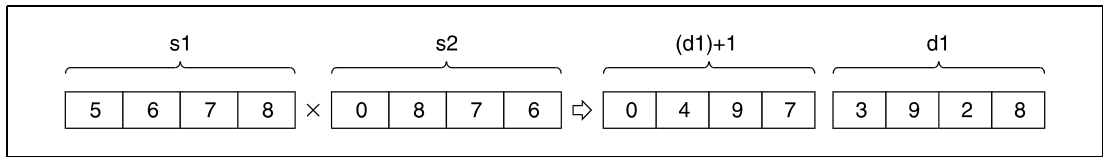
Set Data	Meaning	Data Type	
		MELSEC	IEC
s1	Data that will be multiplied or divided, or first number of device storing data that will be multiplied or divided	BCD 4-digit	WORD
s2	Data to multiply or divide by, or first number of device storing such data	BCD 4-digit	WORD
d1	First number of device storing the operation results of multiplication or division operation	BCD 8-digit	2 Arrays of WORD

## Bx, BxP, B/, B/P

**Functions**    **BCD 4-digit multiplication and division operations**

**Bx**        **BCD multiplication (4-digit)**

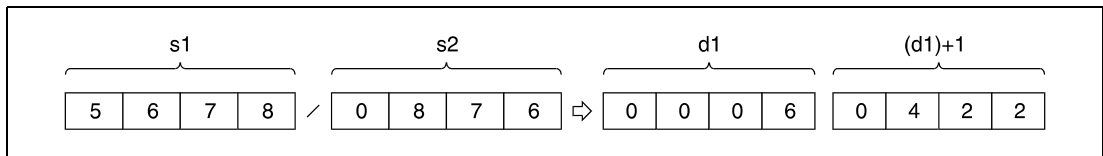
BCD 4-digit data in s1 is multiplied with BCD 4-digit data in s2. The result is stored in d1.



BCD 4-digit data designated by s1 and s2 have to range within 0 and 9999.

**B/**        **BCD division (4-digit)**

BCD 4-digit data in s1 is divided by BCD 4-digit data in s2. The result is stored in d1.



The result of the division is stored in two 16-bit WORD arrays. The lower array stores the quotient (BCD 4-digit) and the upper array stores the remainder (BCD 4-digit).

If d is a bit device, the remainder of the division is not stored.

**Operation Errors**

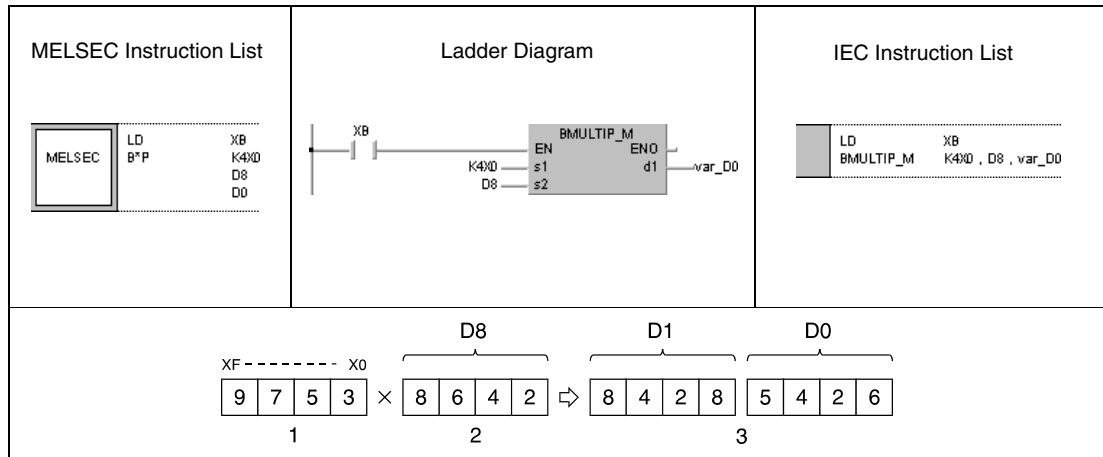
In the following cases an operation error occurs and the error flag is set:

- The BCD 4-digit data at s1, s2, or d exceed the relevant device range (error code: 4101).
- Division by 0 (Q series and System Q = error code 4100).

**Program Example 1**

BxP

With leading edge from XB, the following program multiplies BCD data at X0 through XF with BCD data in D8. The result is stored in D0 and D1.

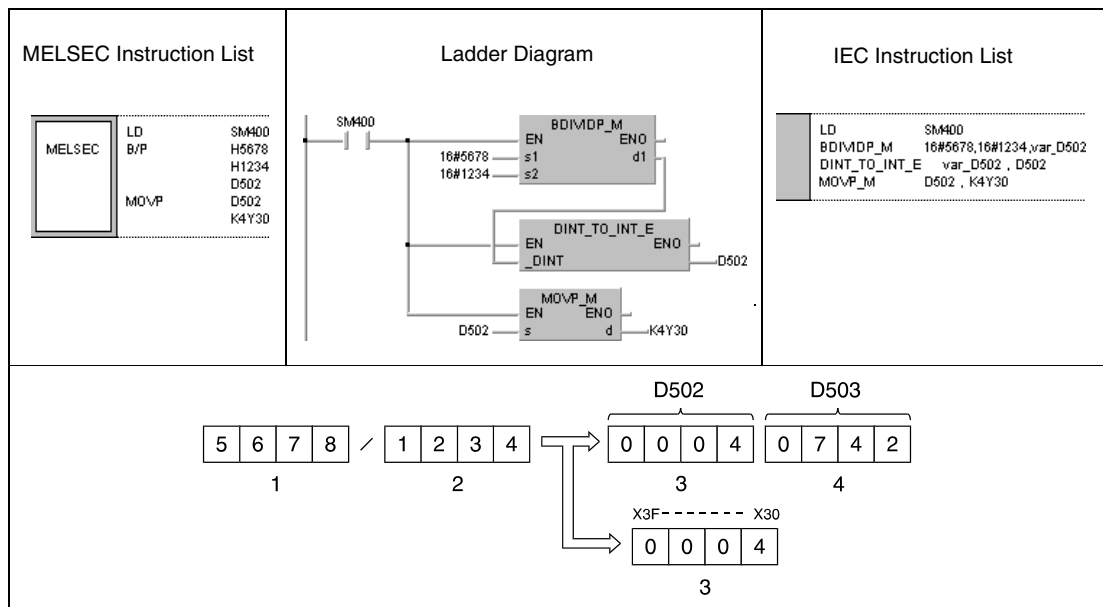


- <sup>1</sup> Multiplicand
- <sup>2</sup> Multiplier
- <sup>3</sup> Result of multiplication

**Program Example 2**

B/P

With leading edge from SM400, the following program divides BCD data 5678 by BCD data 1234. The result is stored in D502 and the remainder is stored in D503. The last program step outputs the quotient in D502 at Y30 through Y3F.



- <sup>1</sup> Dividend
- <sup>2</sup> Divisor
- <sup>3</sup> Quotient
- <sup>4</sup> Remainder

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.2.8 DBx, DBxP, DB/, DB/P

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

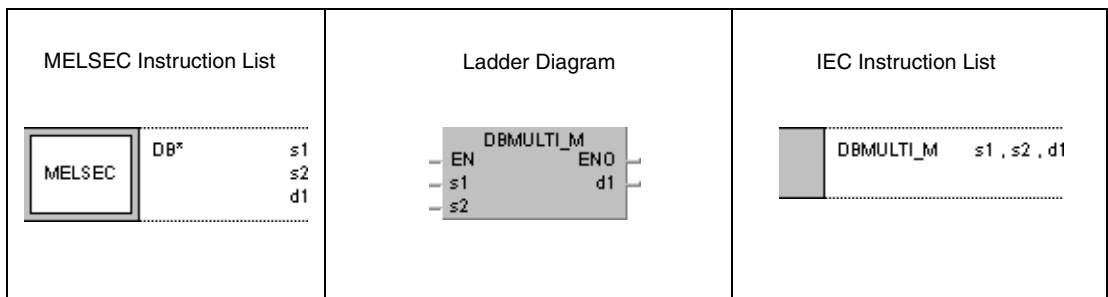
	Usable Devices															Digit Designation	Number of steps	Index	Carry Flag	Error Flag							
	Bit Devices					Word Devices (16-bit)					Constant		Pointer		Level												
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1							Z	V	K	H (16#)	P	I	N
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1	11	●		●
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K8	1			
d1		●	●	●	●	●	●	●	●	●	●	●	●														

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

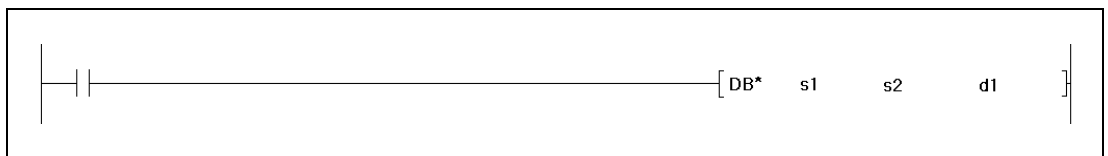
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	SM0	4
s2	●	●	●	●	●	●	●	●	—		
d1	●	●	●	—	—	—	—	—	—		

GX IEC Developer



GX Developer



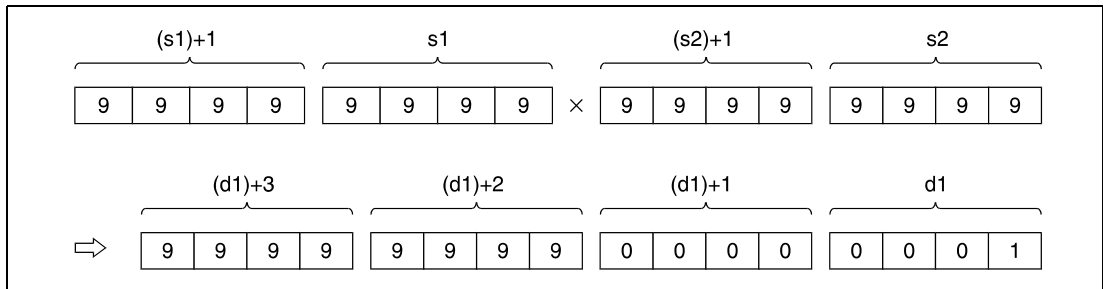
Variables

Set DataSet Data	Meaning	Data Type
s1	Data that will be multiplied or divided, or first number of device storing data that will be multiplied or divided	BCD 8-digit
s2	Data to multiply or divide by, or first number of device storing such data	BCD 8-digit
d1	First number of device storing the operation results of multiplication or division operation	BCD 16-digit

**Functions**    **BCD 8-digit multiplication and division operations**

**DBx**    **BCD multiplication (8-digit)**

BCD 8-digit data in s1 is multiplied with BCD 8-digit data in s2. The result is stored in d1.



If the result in d1 is a bit device, designation is made from the lower bits.

Example:

K1: lower 4 bits (b0 to b3)

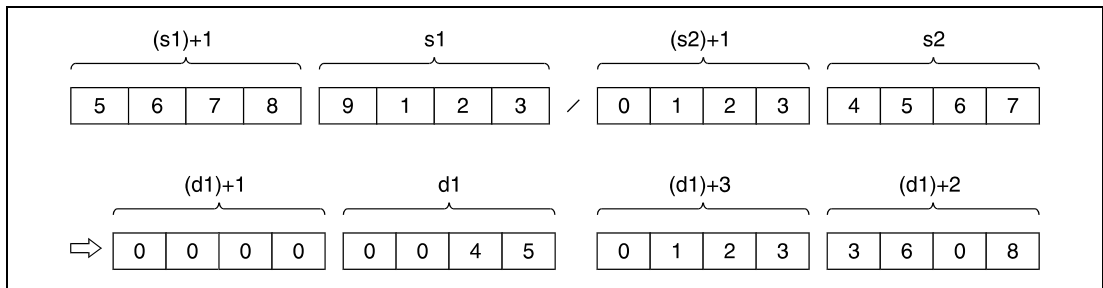
K4: lower 16 bits (b0 to b15)

K8: 32 bits (b0 to b31)

BCD 8-digit data designated by s1 and s2 have to range within 0 and 99999999. Undesignated digits are read as 0 (e.g. 12345 = 00012345).

**DB/**    **BCD division (8-digit)**

BCD 8-digit data in s1 is divided by BCD 8-digit data in s2. The result is stored in d1.



The result of the division is stored in two 32-bit WORD arrays. The lower array stores the quotient (BCD 8-digit) and the upper array stores the remainder (BCD 8-digit).

If d is a bit device, the remainder of the division is not stored.

**Operation Errors**

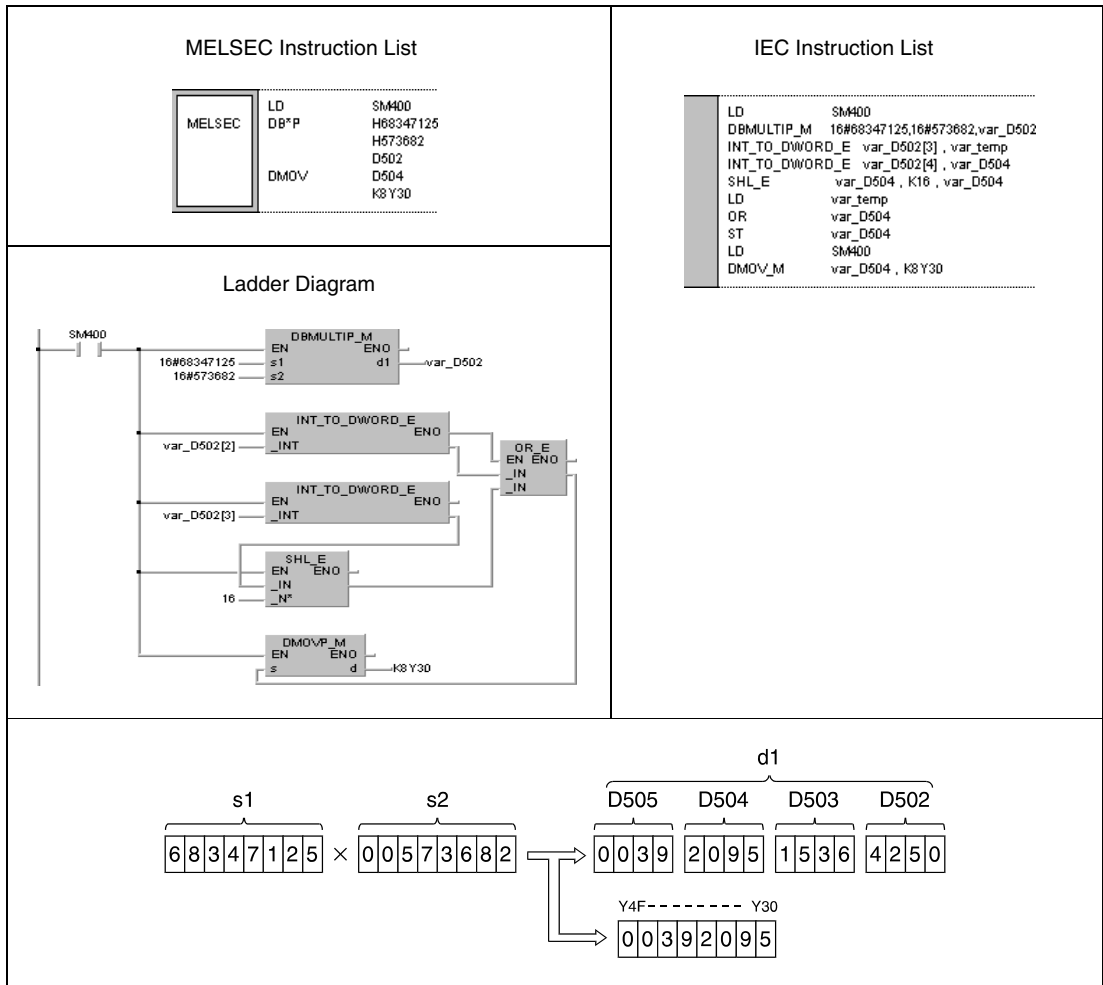
In the following cases an operation error occurs and the error flag is set:

- The BCD 8-digit data at s1, s2, or d exceed the relevant device range (error code: 4101).
- Division by 0 (Q series and System Q = error code 4100).

# DBx, DBxP, DB/, DB/P

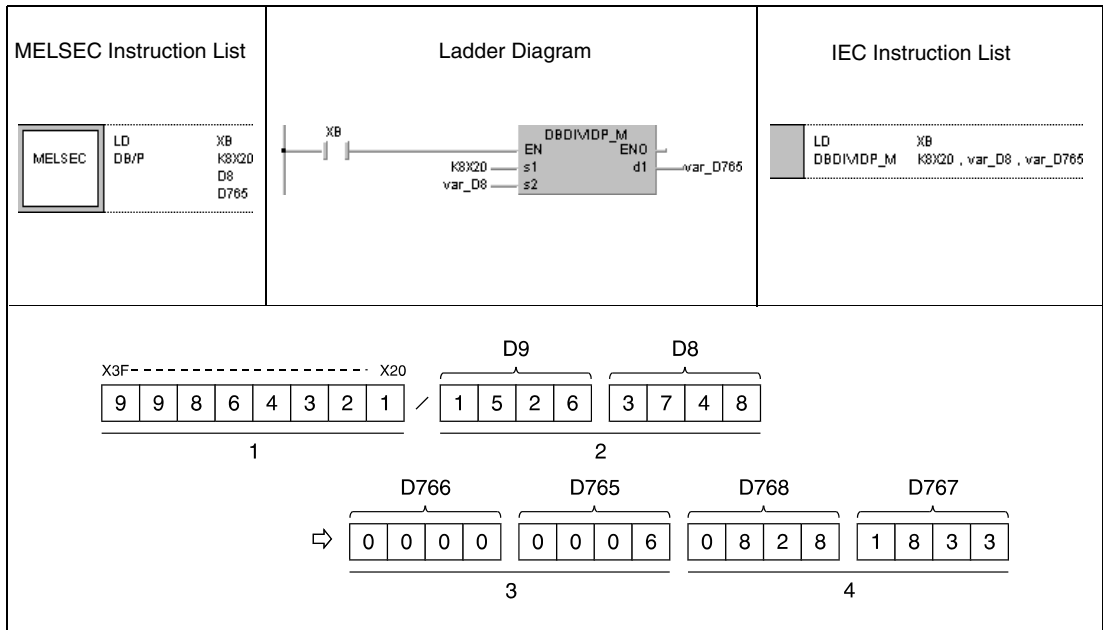
**Program Example 1** DBxP

With leading edge from SM400, the following program multiplies BCD data 68347125 with BCD data 576682. The result is stored in D502 through D505. The following program step outputs the upper eight digits (D504, D505) at Y30 through Y4F.



**Program Example 2** DB/P

With leading edge from XB, the following program divides BCD data at X20 through X3F by BCD data in D8 and D9. The result is stored in D765 through D768.



- 1 Dividend
- 2 Divisor
- 3 Quotient
- 4 Remainder

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

# E+, E+P, E-, E-P

## 6.2.9 E+, E+P, E-, E-P

### CPU

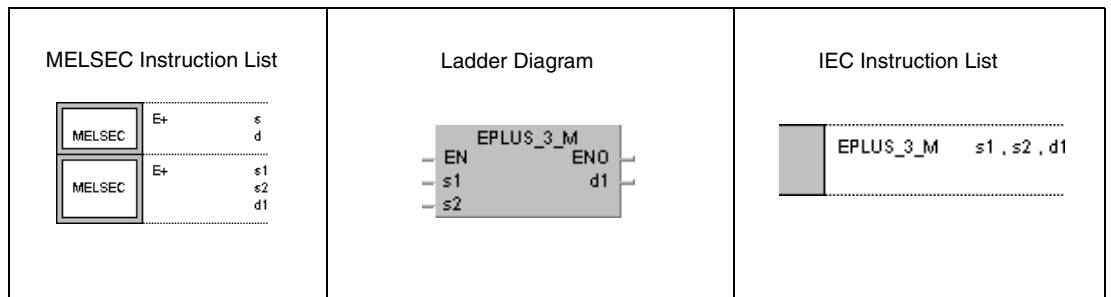
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

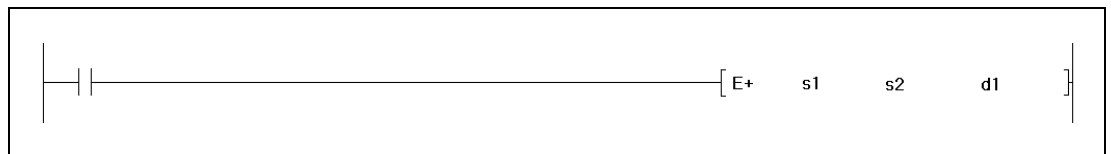
### Devices MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants E	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		
s1	—	●	●	—	●	●	—	●	—	SM0	4
s2	—	●	●	—	●	●	—	●	—		
d1	—	●	●	—	●	●	—	—	—		

### GX IEC Developer



### GX Developer



### Variables

Set Data	Meaning	Data Type
s	Addition or subtraction data, or first number of device storing addition or subtraction data	Real number
d	Data to be added to or subtracted from, or first number of device storing such data	
s1	Data to be added to or subtracted from, or first number of device storing such data	
s2	Addition or subtraction data, or first number of device storing addition or subtraction data	
d1	First number of device storing addition or subtraction data	

### NOTE

Within the IEC editors please use the IEC commands.

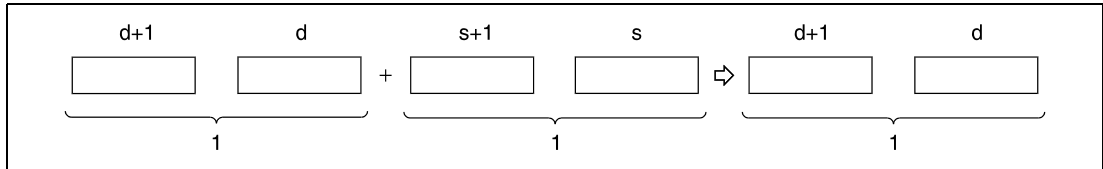


**Functions Floating point data addition and subtraction operations**

**E+ Floating point data addition**

● Variation 1:

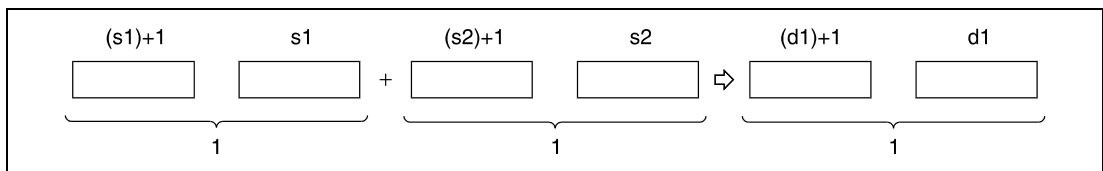
Floating point data in d is added to floating point data in s. The result is stored in d.



<sup>1</sup> Floating point data, data type real number

● Variation 2:

Floating point data in s1 is added to floating point data in s2. The result is stored in d1.



<sup>1</sup> Floating point data, data type real number

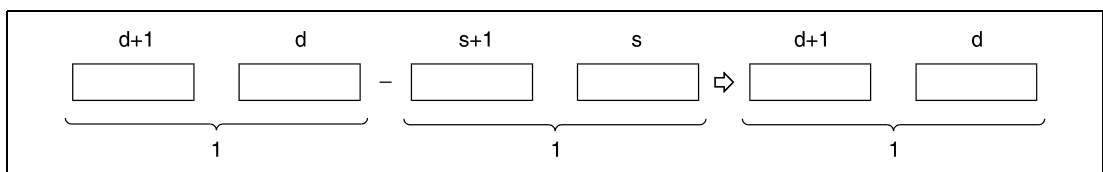
Floating point data designated by s, d, s1, s2, and d1 have to range within:

$$0, \pm 2^{-127} \leq (s, d, s1, s2, d1) < \pm 2^{129}$$

**E- Floating point data subtraction**

● Variation 1:

Floating point data in s is subtracted from floating point data in d. The result is stored in d.

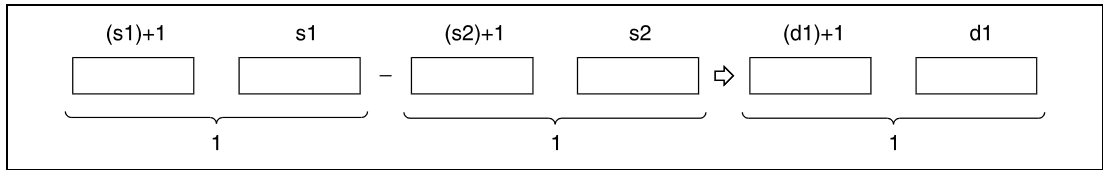


<sup>1</sup> Floating point data, data type real number

## E+, E+P, E-, E-P

- Variation 2:

Floating point data in s2 is subtracted from floating point data in s1. The result is stored in d1.



<sup>1</sup> Floating point data, data type real number

Floating point data designated by s, d, s1, s2, and d1 have to range within:

$$0, \pm 2^{-127} \leq (s, d, s1, s2, d1) < \pm 2^{129}$$

### Operation Errors

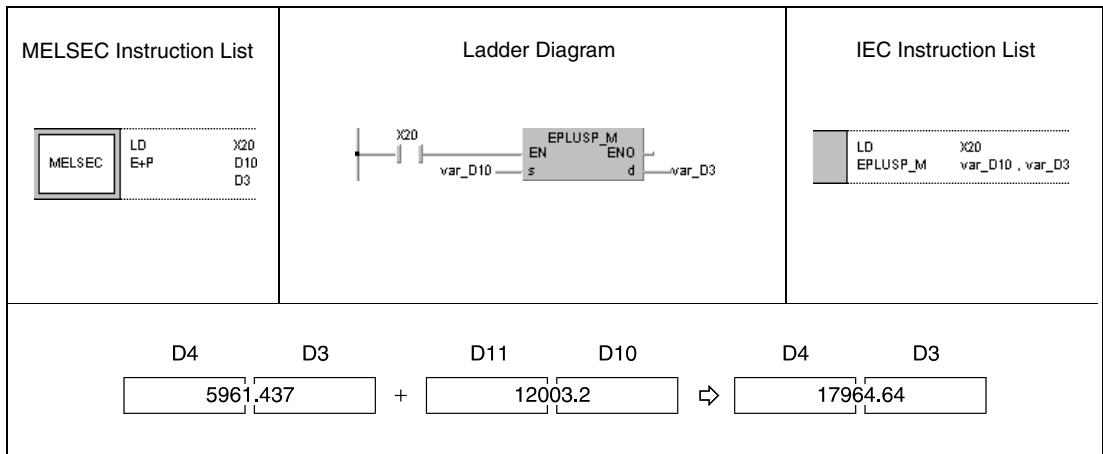
In the following cases an operation error occurs and the error flag is set:

- The floating point data in s, d, s1, s2, or d1 exceed the relevant device range (error code 4100).

### Program Example 1

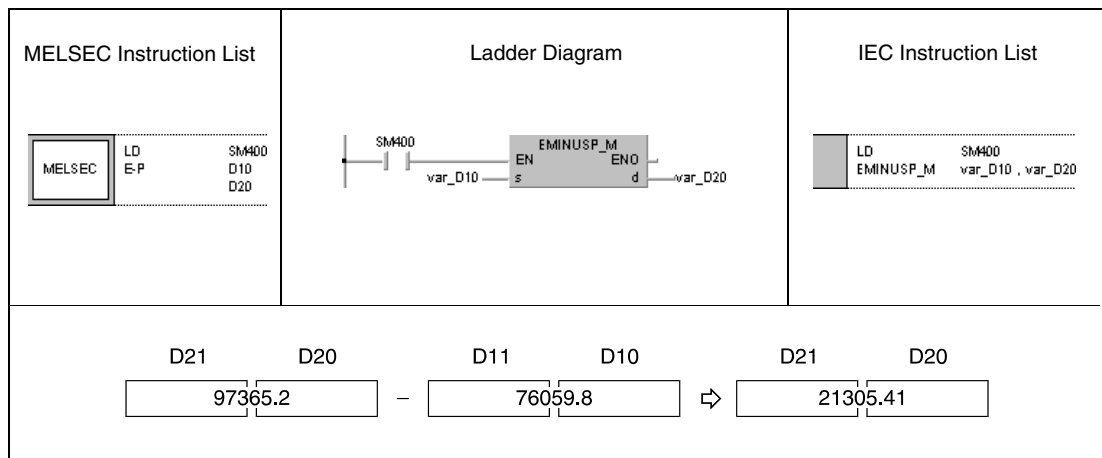
E+P (s, d)

With leading edge from X20, the following program adds floating point data in D3 and D4 to floating point data in D10 and D11. The result is stored in D3 and D4.



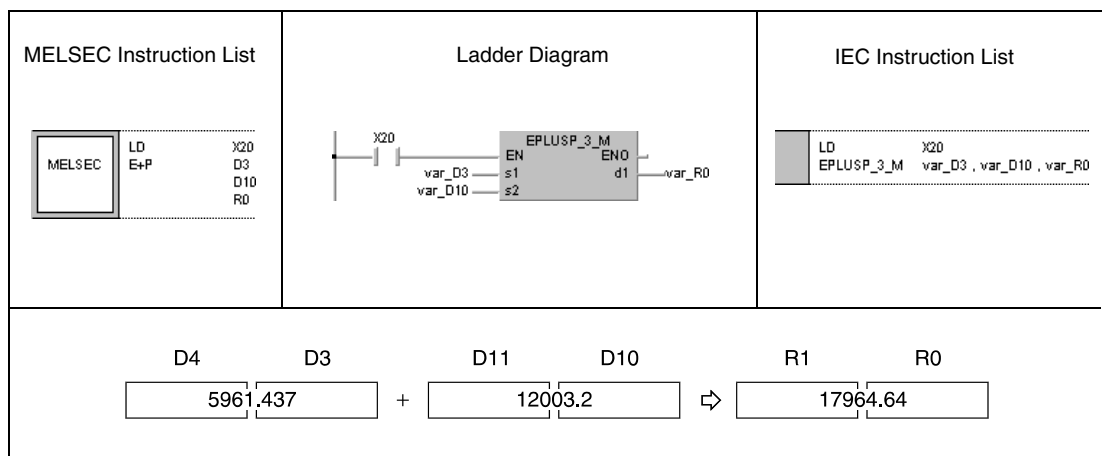
**Program Example 2** E-P (s, d)

With leading edge from SM400, the following program subtracts floating point data in D10 and D11 from floating point data in D20 and D21. The result is stored in D20 and D21.



**Program Example 3** E+P (s1, s2, d)

With leading edge from X20, the following program adds floating point data in D3 and D4 to floating point data in D10 and D11. The result is stored in R0 and R1.

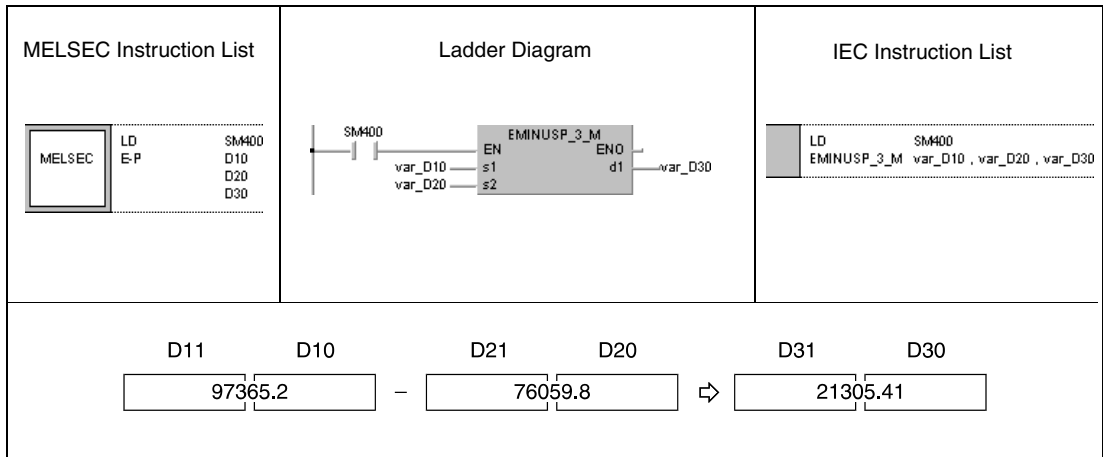


# E+, E+P, E-, E-P

**Program** E-P (s1, s2, d)

**Example 4**

With leading edge from SM400, the following program subtracts floating point data in D20 and D21 from floating point data in D10 and D11. The result is stored in D30 and D31.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.2.10 Ex, ExP, E/, E/P

CPU

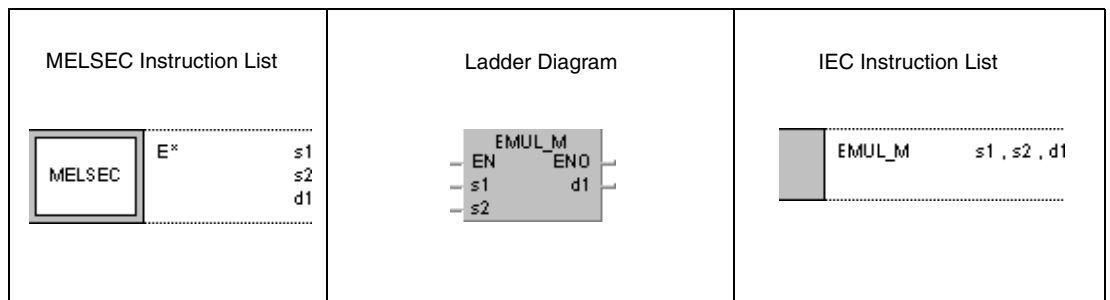
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

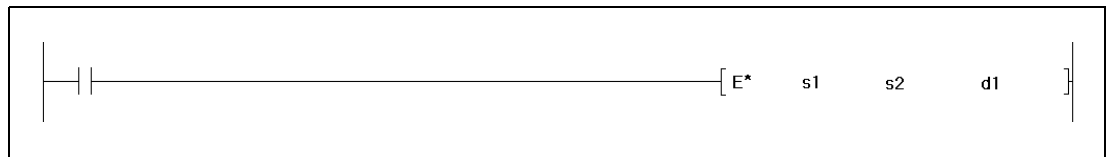
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constants E	Other U		
	Bit	Word		Bit	Word						
s1	—	●	●	—	●	●	—	●	—	SM0	4
s2	—	●	●	—	●	●	—	●	—		
d1	—	●	●	—	●	●	—	—	—		

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s1	Data that will be multiplied or divided, or first number of device storing data that will be multiplied or divided	Real number
s2	Data to multiply or divide by, or first number of device storing such data	
d1	First number of device storing the operation results of multiplication or division operation	

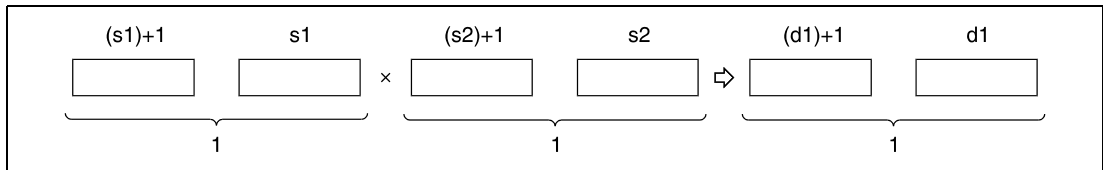
NOTE

Within the IEC editors please use the IEC commands.

**Functions Floating point data multiplication and division operations**

**Ex Floating point data multiplication**

Floating point data in s1 is multiplied with floating point data in s2. The result is stored in d1.



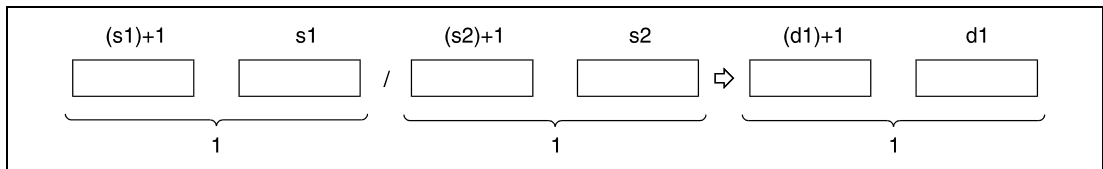
<sup>1</sup> Floating point data, data type real number

Floating point data designated by s1, s2, and d1 have to range within:

$$0, \pm 2^{-127} \leq (s1, s2, d1) < \pm 2^{129}$$

**E/ Floating point data division**

Floating point data in s1 is divided by floating point data in s2. The result is stored in d1.



<sup>1</sup> Floating point data, data type real number

Floating point data designated by s1, s2, and d1 have to range within:

$$0, \pm 2^{-127} \leq (s1, s2, d1) < \pm 2^{129}$$

**Operation Errors**

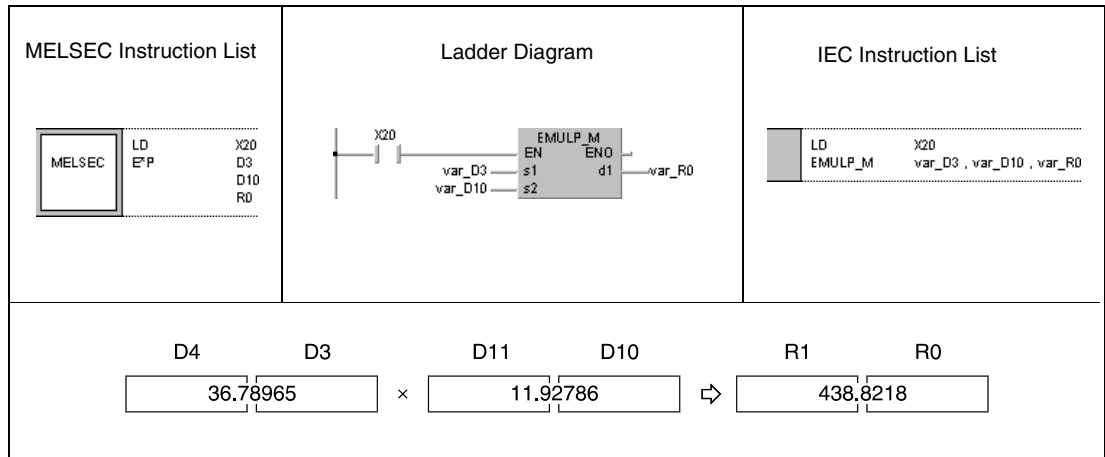
In the following cases an operation error occurs and the error flag is set:

- The floating point data at s1, s2, or d1 exceed the relevant device range (error code 4100).
- Division by 0 (error code 4100).

**Program Example 1**

ExP

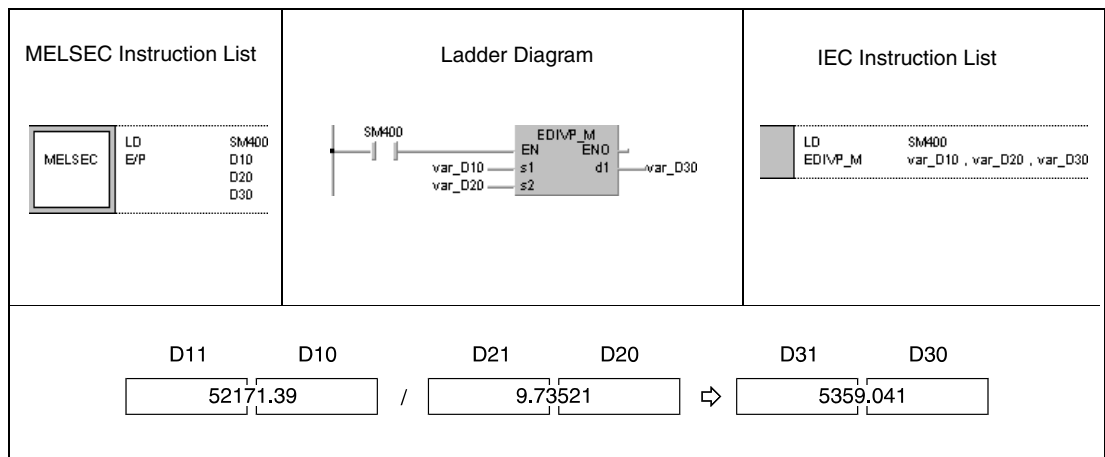
With leading edge from X20, the following program multiplies floating point data in D3 and D4 with floating point data in D10 and D11. The result is stored in R0 and R1.



**Program Example 2**

E/P

With leading edge from SM400, the following program divides floating point data in D10 and D11 by floating point data in D20 and D21. The result is stored in D30 and D31.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

# BK+, BK+P, BK-, BK-P

## 6.2.11 BK+, BK+P, BK-, BK-P

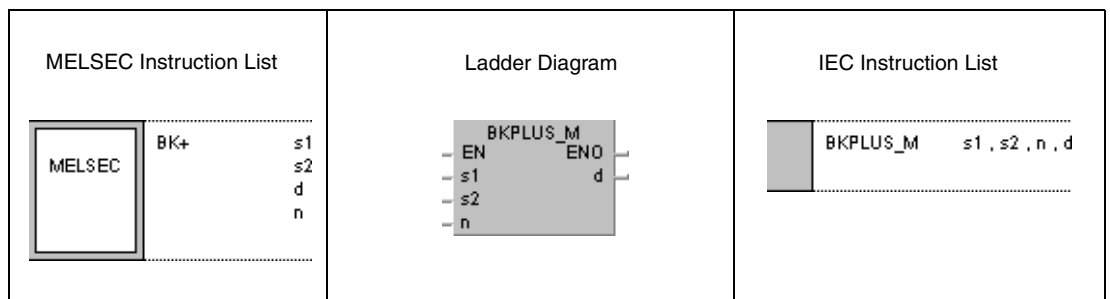
### CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

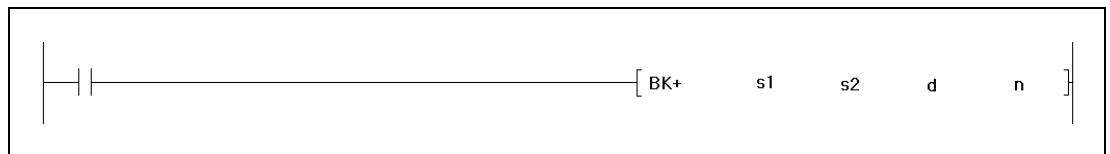
### Devices MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	5
s2	—	●	●	—	—	—	—	●	—		
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

### GX IEC Developer



### GX Developer



### Variables

Set Data	Meaning	Data Type
s1	Data to be added to or subtracted from, or first number of device storing such data	BIN 16-bit
s2	Addition or subtraction data, or first number of device storing addition or subtraction data	
d	First number of device storing results of operation	
n	Number of data blocks	

### NOTE

Within the IEC editors please use the IEC commands.

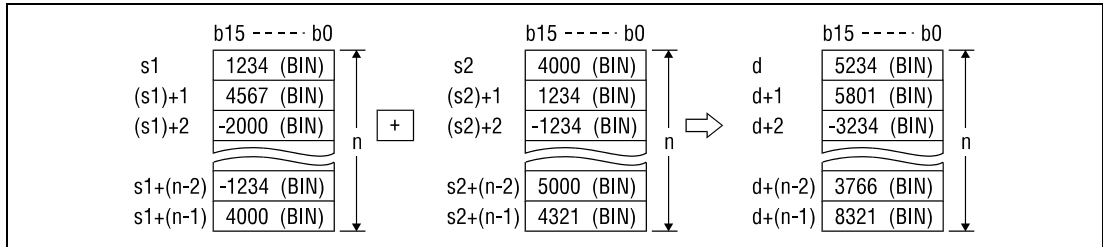


**Functions BIN block addition and subtraction operations**

**BK+ BIN block addition**

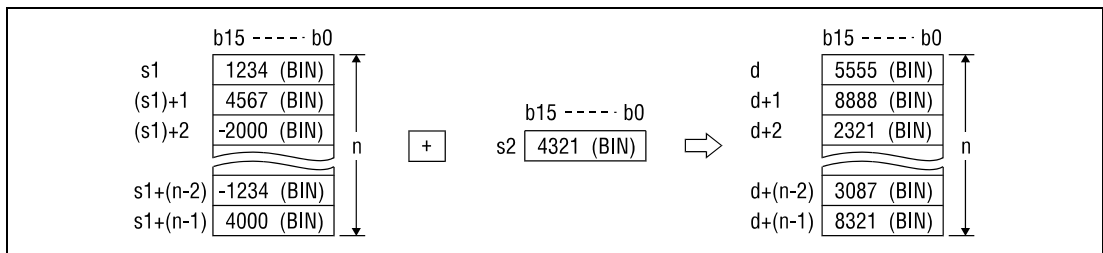
An addition operation instruction for BIN block data consists of the instruction itself, two designated devices s1 and s2 to be added, a device d to store the result, and the number of data blocks to be added.

It adds the nth 16-bit block in s1 to the nth 16-bit block in s2, beginning with the first number of device. The result of each block addition is stored in d.



The addition operation is conducted in 16-bit units.

The constant designated by s1 must be BIN 16-bit data ranging from -32768 to 32767.



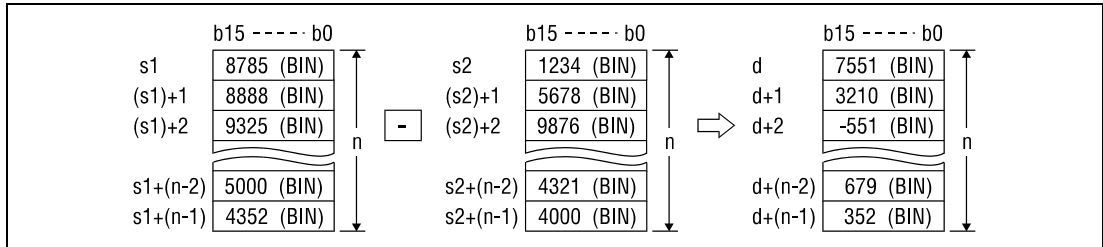
The most significant bit of each block determines, whether data in s1, s2 or d are positive (bit = 0) or negative (bit = 1).

If the least significant bit of a block is fallen below or the most significant bit of a block is exceeded, the carry flag is not set.

## BK- BIN block subtraction

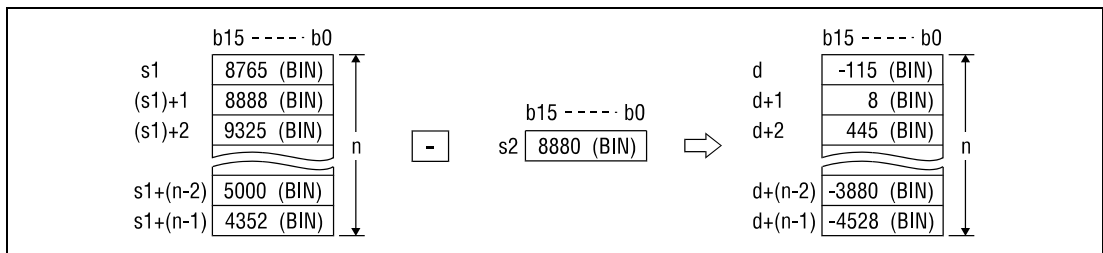
A subtraction operation instruction for BIN block data consists of the instruction itself, two designated devices s1 and s2 to be added, a device d to store the result, and the number of data blocks to be subtracted.

It subtracts the nth 16-bit block in s2 from the nth 16-bit block in s1, beginning with the first number of device. The result of each block addition is stored in d.



The subtraction operation is conducted in 16-bit units.

The constant designated by s2 must be BIN 16-bit data ranging from -32768 to 32767.



The most significant bit of each block determines, whether data in s1, s2 or d are positive (bit = 0) or negative (bit = 1).

If the least significant bit of a block is fallen below or the most significant bit of a block is exceeded, the carry flag is not set.

### Operation Errors

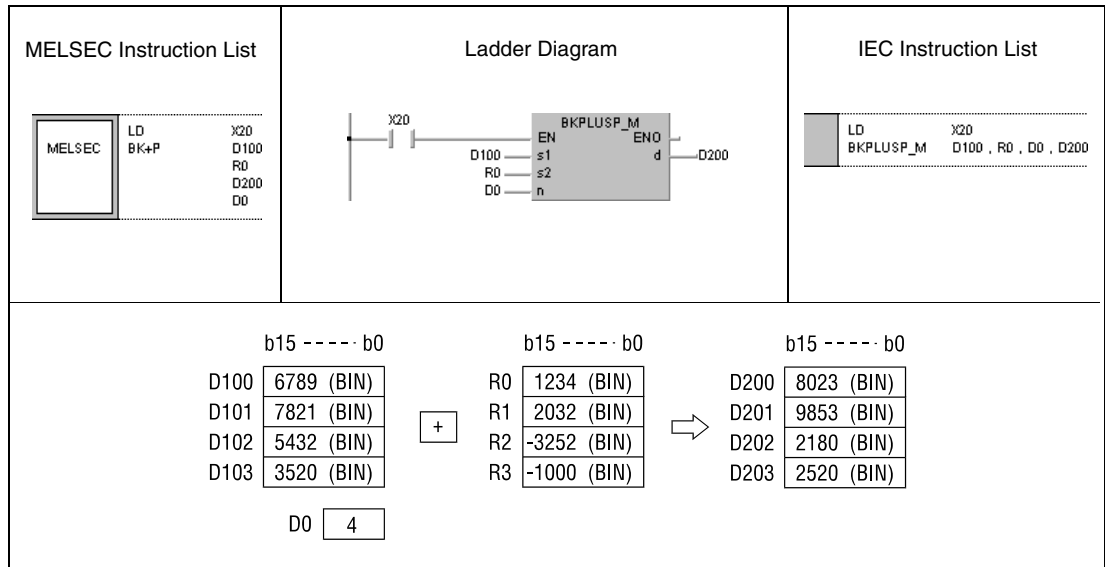
In the following cases an operation error occurs and the error flag is set:

- The number of data blocks in s1, s2 or d exceeds the relevant device range.
- The device s1 overlaps with the devices s2 or d.

**Program Example 1**

**BK+P**

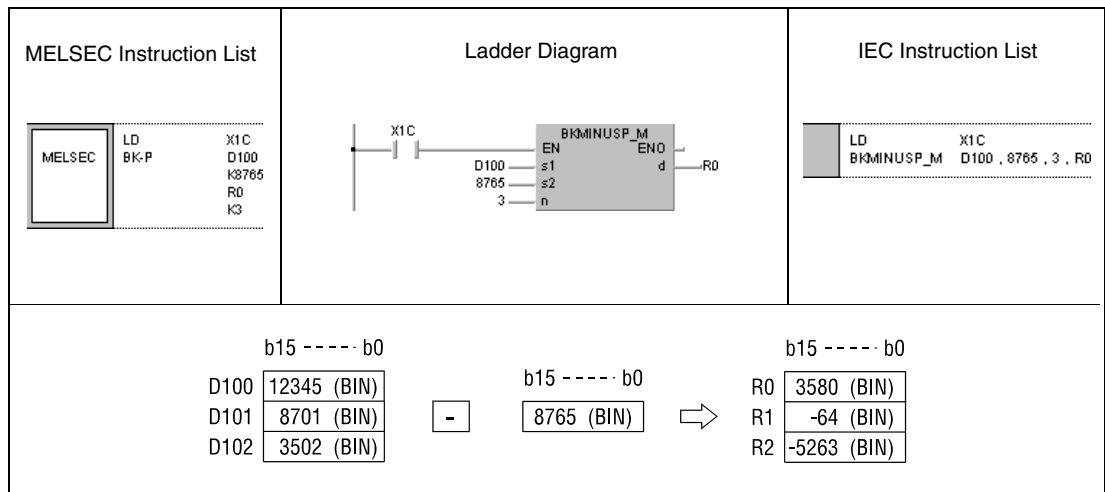
With leading edge from X20, the following program adds BIN block data beginning from D100 to BIN block data beginning from R0. The result of the operation is stored beginning from D200. The number of blocks (4) added is stored in D0.



**Program Example 2**

**BK-P**

With leading edge from X1C, the following program subtracts a constant 8765 from BIN block data beginning from D100. The result of the operation is stored beginning from R0. The number of data blocks (3) subtracted is designated by a constant K3.



# \$+, \$+P

## 6.2.12 \$+, \$+P

### CPU

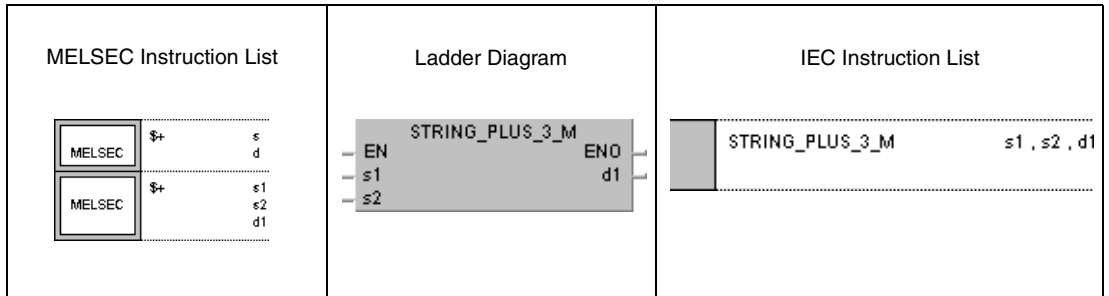
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

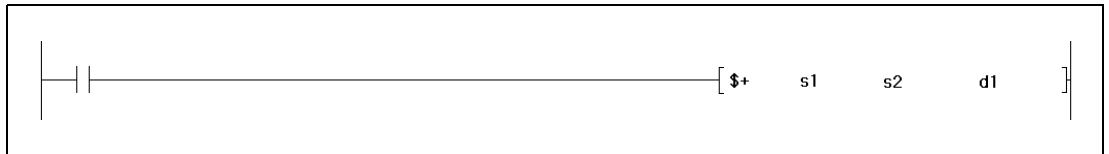
### Devices MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants \$	Other U		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	3
d	—	●	●	—	—	—	—	—	—		
s1	—	●	●	—	—	—	—	●	—	SM0	4
s2	—	●	●	—	—	—	—	●	—		
d1	—	●	●	—	—	—	—	—	—		

### GX IEC Developer



### GX Developer



### Variables

Set Data	Meaning	Data Type
s	Data to be linked, or first number of device storing such data	Character string
d	First number of device storing results of operation	
s1	Data to be linked, or first number of device storing such data	
s2	Data to be linked, or first number of device storing such data	
d1	First number of device storing results of operation	

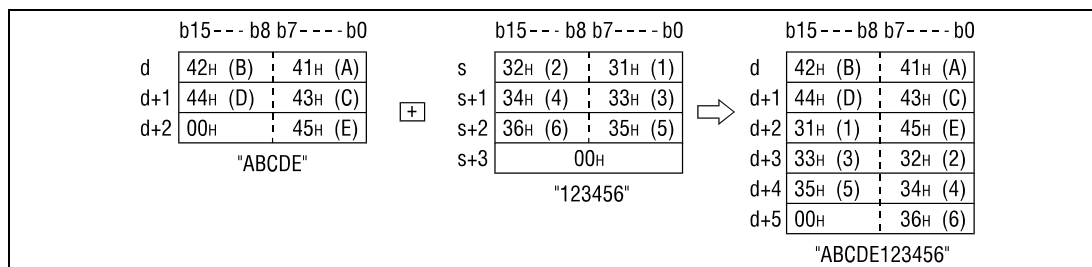
**Functions Character string linking operations**

**\$+ Character string linking**

● Variation 1:

Character string data in s is appended to character data in d. The linked character string is stored in d.

The linked character string begins with the character at the least significant byte in d and ends with the code "00H" in s.

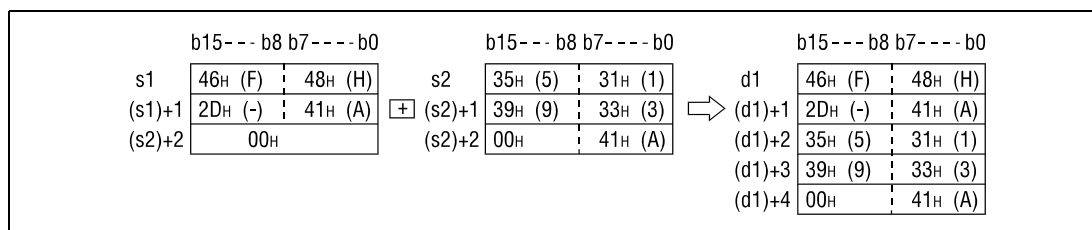


The code "00H" indicates the end of a character string. When two strings are linked, in the first string this code is ignored and the "00H" of the second string marks the end of the linked string.

● Variation 2:

Character string data in s2 is appended to character string data in s1. The linked character string is stored in d1.

The linked character string begins with the character at the least significant byte in s1 and ends with the code "00H" in s2.



The code "00H" indicates the end of a character string. When two strings are linked, in the first string this code is ignored and the "00H" of the second string marks the end of the linked string.

**Operation Errors**

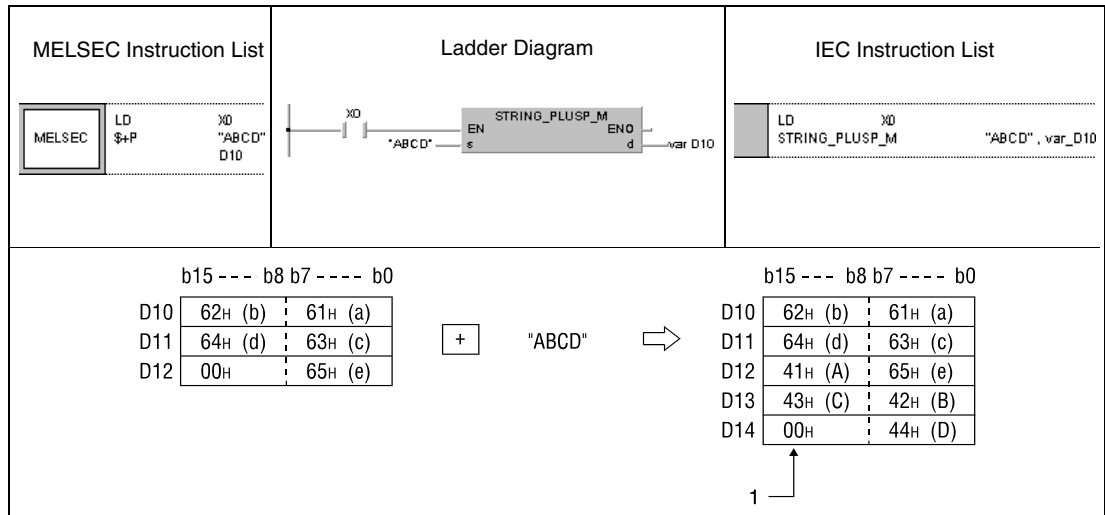
In the following cases an operation error occurs and the error flag is set:

- The linked character string cannot be stored (error code 4100).
- The storage device numbers designated by s, d, s1, s2, and d1 overlap (error code 4101).

**Program Example 1**

S+P

With leading edge from X0, the following program links character string data in D10 through D12 to the character string "ABCD". The linked character string is stored in D10 through D14.

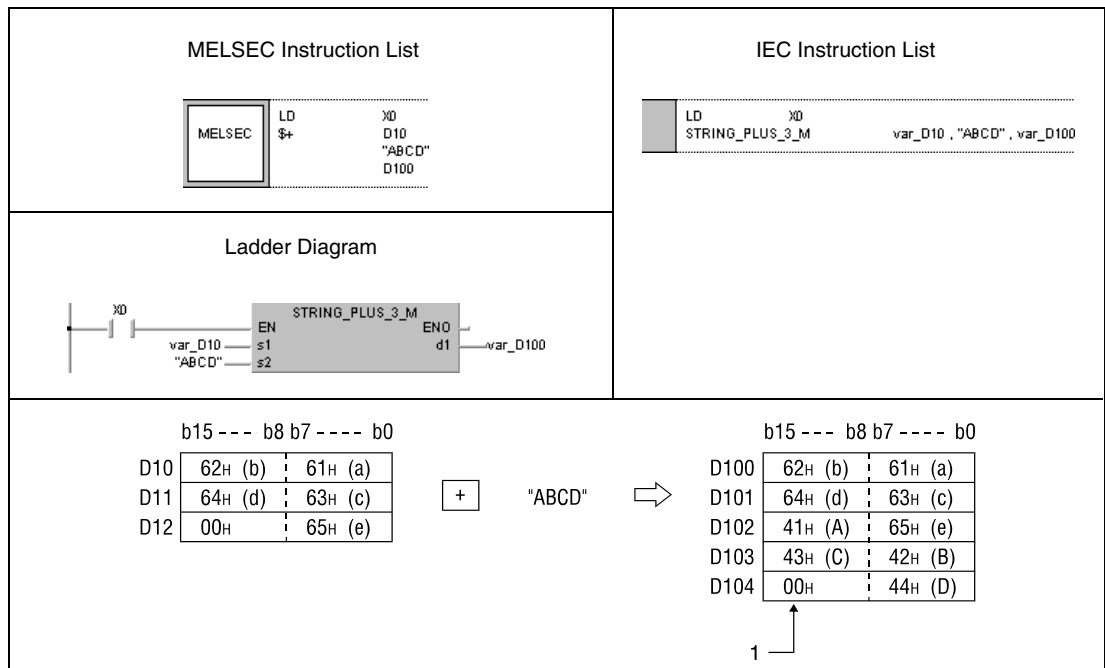


<sup>1</sup> "00H" indicates the end of character strings and is stored automatically.

**Program Example 2**

S+

While X0 is set (1), the following program links character string data in D10 through D12 to a character string "ABCD". The linked character string is stored from D101 through D104.



<sup>1</sup> "00H" indicates the end of character strings and is stored automatically.

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.2.13 INC, INCP, DEC, DECP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

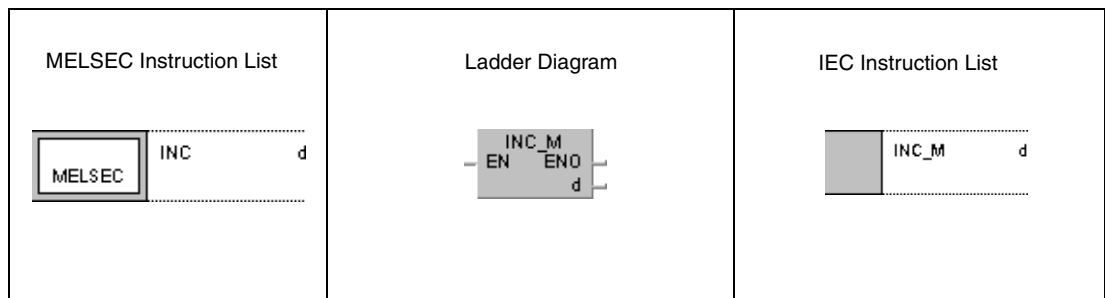
	Usable Devices															Digit Designation	Number of steps	Index	Carry Flag	Error Flag					
	Bit Devices					Word Devices (16-bit)					Constant		Pointer		Level										
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z				V	K	H (16#)	P	I	N	M9012
d	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						K1 ↓ K4	3 1	●		●

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

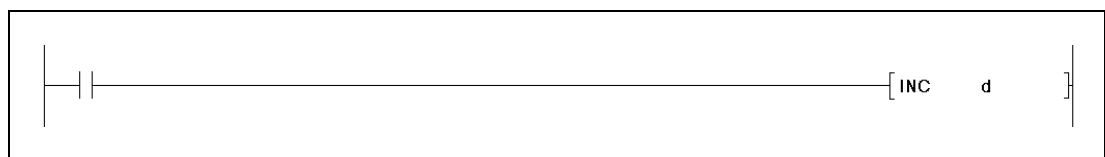
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)			Other
	Bit	Word		Bit	Word						
d	●	●	●	●	●	●	●	—	—	—	2

GX IEC  
Developer



GX  
Developer



Variables

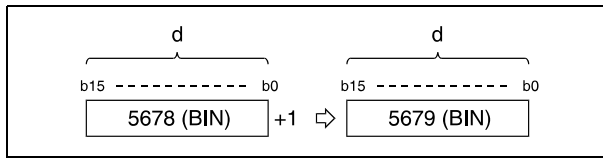
Set Data	Meaning	Data Type
d	First number of device conducted by INC (add 1) or DEC (subtract 1) operation.	BIN 16-bit

# INC, INCP, DEC, DECP

## Functions BIN 16-bit increment and decrement operations

### INC BIN 16-bit increment

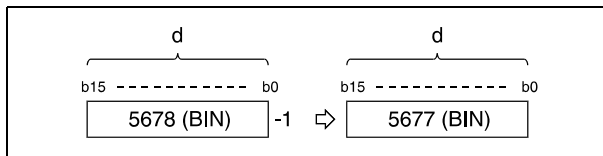
Adds 1 to device designated by d (16-bit).



If the content of d is 32767, the result after incrementing is -32768.

### DEC BIN 16-bit decrement

Subtracts 1 from device designated by d (16-bit).



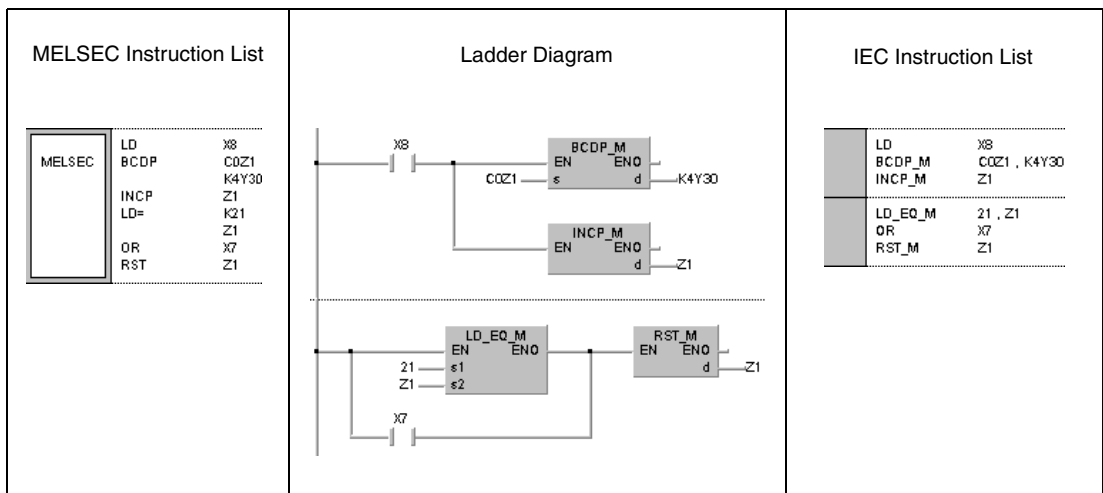
If the content of d is 0, the result after decrementing is -1.

If the content of d is -32768, the result after decrementing is 32767.

## Program Example 1

### INCP

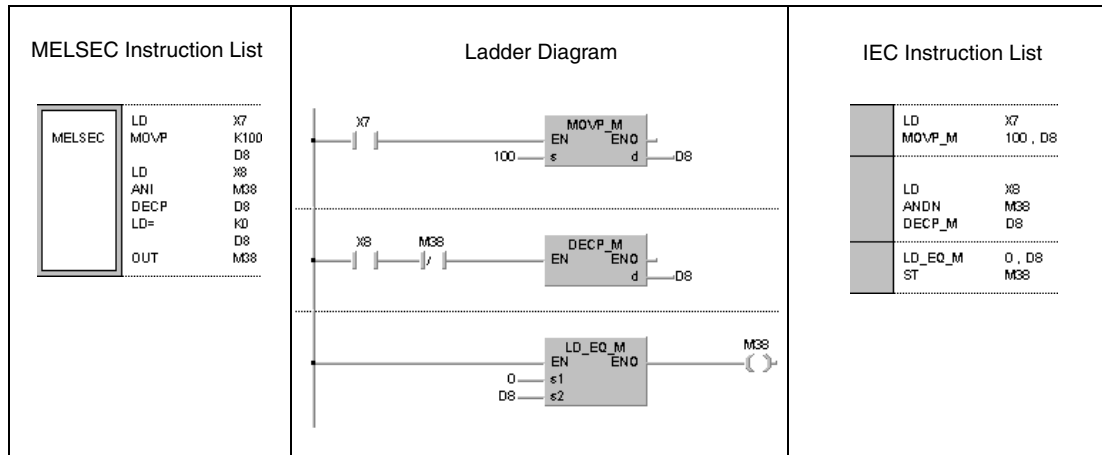
With leading edge from X8, the following program outputs the actual value of the counter (nominal value = 9999) C0 through C20 (C0 plus Z1) at Y30 through Y3F as BCD data. Z1 is reset (RST Z1), if Z1 is equal to 21 (LD = K21 Z1) or if the reset input X7 is set.





**Program Example 2**      **DECP**

The following example shows a down counter program. With leading edge from X7, this program stores a value 100 in D8. While M38 is not set, data in D8 is decremented by 1 with leading edge from X8. At D8 = 0, M38 is set.



# DINC, DINCP, DDEC, DDECP

## 6.2.14 DINC, DINCP, DDEC, DDECP

### CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

### Devices MELSEC A

	Usable Devices															Digit Designation	Number of steps	Index	Carry Flag	Error Flag						
	Bit Devices					Word Devices (16-bit)					Constant	Pointer		Level	M9012				M9010 M9011							
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1						Z	V	K	H (16#)	P	I	N
d	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	● <sup>1</sup>						K1 ↓ K8	3 ● <sup>2</sup>	●		●

<sup>1</sup> Except for AnN CPU

<sup>2</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

### Devices MELSEC Q

	Usable Devices									Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)	Other U			
	Bit	Word		Bit	Word							
d	●	●	●	●	●	●	●	●	—	—	—	2 <sup>1)</sup>

<sup>1</sup> The number of steps depends on the device and the type of CPU.

If a QnA-CPU or a System Q single processor CPU is used:

2

If a System Q multi processor CPU is used with internal word devices (except for file register ZR):

3

constants:

3

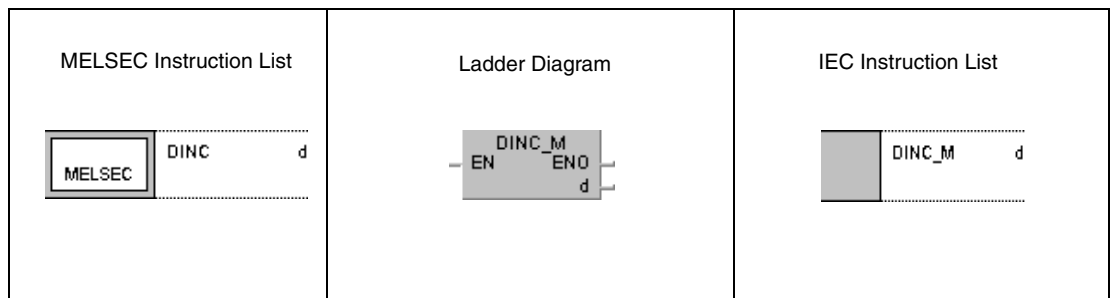
Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K8, and which use no index qualification:

3

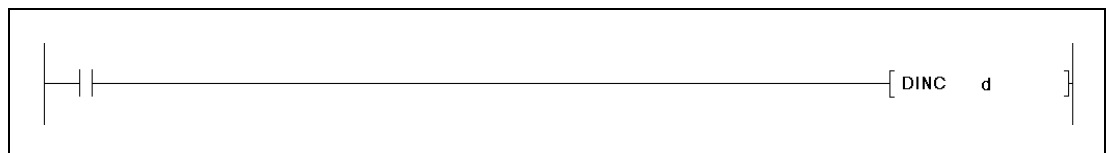
If a System Q multi processor CPU is used with devices other than above mentioned:

2

### GX IEC Developer



### GX Developer



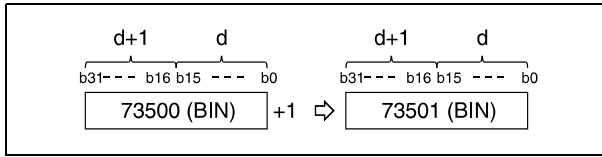
### Variables

Set Data	Meaning	Data Type
d	First number of device conducted by INC (add 1) or DEC (subtract 1) operation.	BIN 32-bit

**Functions BIN 32-bit increment and decrement operations**

**DINC BIN 32-bit increment**

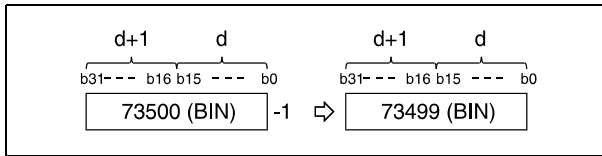
Adds 1 to device designated by d (32-bit).



If the content of d is 2147483647, the result after incrementing is -2147483648.

**DDEC BIN 32-bit decrement**

Subtracts 1 from device designated by d (16-bit).



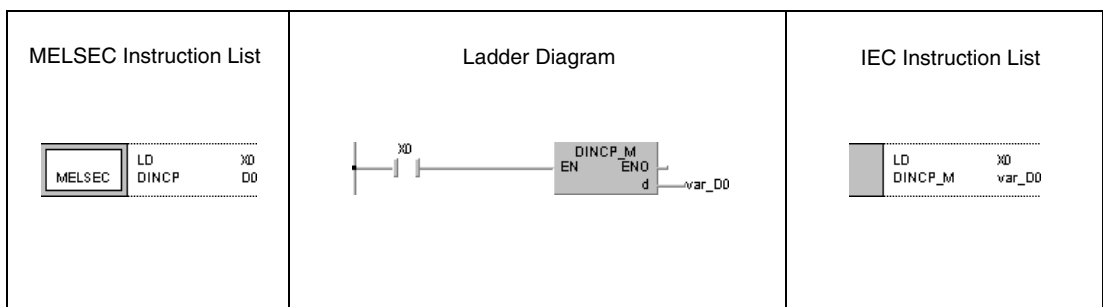
If the content of d is 0, the result after decrementing is -1.

If the content of d is -2147483647, the result after decrementing is 2147483647.

**Program Example 1**

**DINCP**

With leading edge from X0, the following program adds 1 to data in D0.

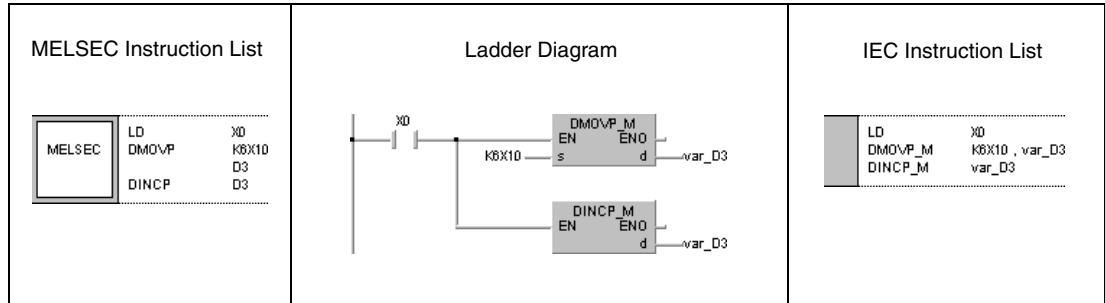


# DINC, DINCP, DDEC, DDECP

## Program Example 2 DINC

### Example 2

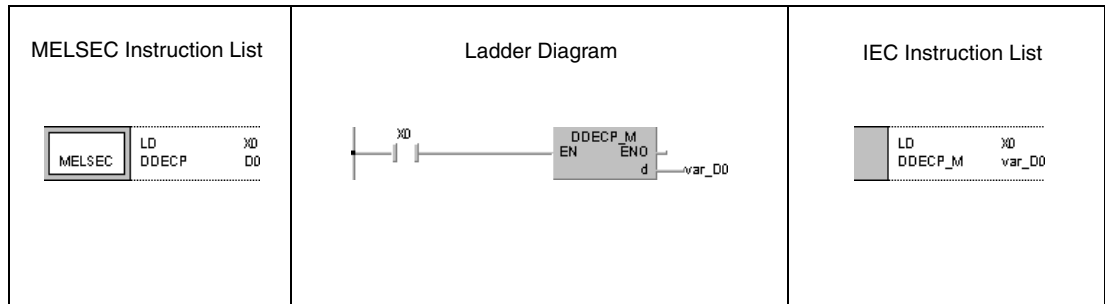
With leading edge from X0, the following program adds 1 to data at X10 through X27. The result is stored in D3 and D4.



## Program Example 3 DDEC

### Example 3

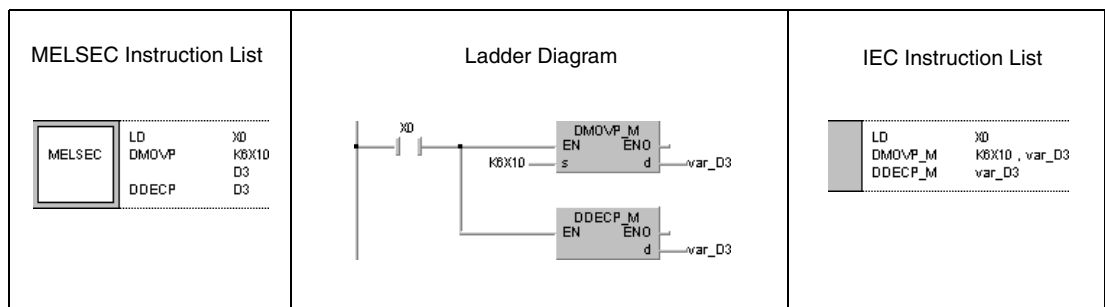
With leading edge from X0, the following program subtracts 1 from data in D0.



## Program Example 4 DDECP

### Example 4

With leading edge from X0, the following program subtracts 1 from data in X10 through X27. The result is stored in D3 and D4.



## NOTE

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 6.3 Data Conversion Instructions

The following instructions convert different data types:

Conversion	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
BIN (16-/32-bit) ↓ BCD (4-/8-digit)	BCD	BCD_M
	BCDP	BCDP_M
	DBCDCD	DBCDCD_M
	DBCDCP	DBCDCP_M
BCD (4-/8-digit) ↓ BIN (16-/32-bit)	BIN	BIN_M
	BINP	BINP_M
	DBIN	DBIN_M
	DBINP	DBINP_M
BIN (16-/32-bit) ↓ Floating Point Data	FLT	FLT_M
	FLTP	FLTP_M
	DFLT	DFLT_M
	DFLTP	DFLTP_M
Floating Point Data ↓ BIN (16-/32-bit)	INT	INT_MD
		INT_E_MD
	INTP	INT_P_MD
		INT_P_E_MD
	DINT	DINT_MD
		DINT_E_MD
	DINTP	DINT_P_MD
		DINT_P_E_MD
BIN 16-bit ↓ BIN 32-bit	DBL	DBL_M
	DBLP	DBLP_M
BIN 32-bit ↓ BIN 16-bit	WORD	WORD_M
	WORDP	WORDP_M
BIN (16-/32-bit) ↓ GRAY CODE Data	GRY	GRY_M
	GRYP	GRYP_M
	DGRY	DGRY_M
	DGRYP	DGRYP_M
GRAY CODE Data ↓ BIN (16-/32-bit)	GBIN	GBIN_M
	GBINP	GBINP_M
	DGBIN	DGBIN_M
	DGBINP	DGBINP_M
Sign Reversal BIN (16-/32-bit) (Complement of 2)	NEG	NEG_M
	NEGP	NEGP_M
	DNEG	DNEG_M
	DNEGP	DNEGP_M
Sign Reversal Floating Point Data	ENEG	ENEG_M
	ENEGP	ENEGP_M
BIN Block (16-bit) ↓ BCD Block (4-digit)	BKBCD	BKBCD_M
	BKBCDP	BKBCDP_M

Conversion	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
BCD Block (4-digit) ↓ BIN Block (16-bit)	BKBIN	BKBIN_M
	BKBINP	BKBINP_M

6.3.1 BCD, BCDP, DBCD, DBCDP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

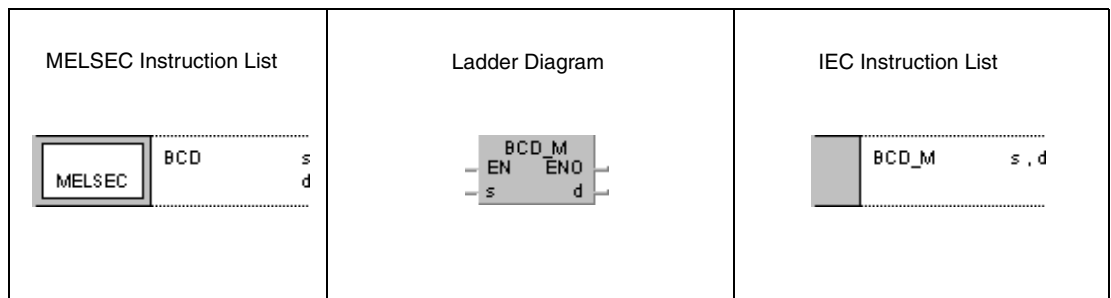
Usable Devices																				Digit designation	Number of steps	Index	Carry Flag	Error Flag																	
Bit Devices								Word Devices (16-bit)								Constant		Pointer							Level																
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I				N	M9012	M9010 M9011																
BCD																																									
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●																										
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●																										
DBCD																																									
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●																										
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●																										

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this Programming Manual for the according number of steps.

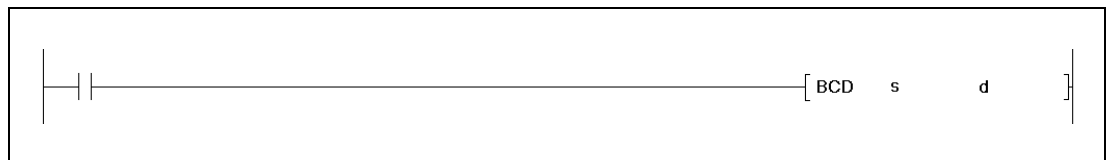
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
s	●	●	●	●	●	●	●	—		SM0	3
d	●	●	●	●	●	●	—	—			

GX IEC Developer



GX Developer



Variables

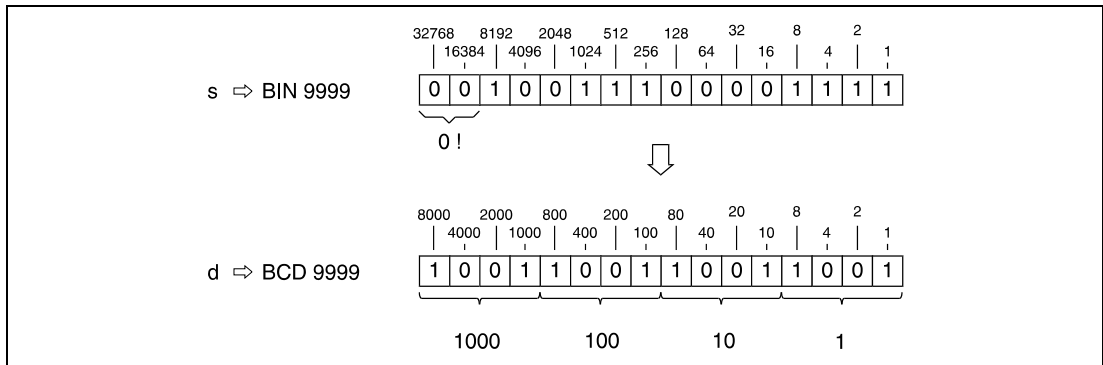
Set Data	Meaning	Data Type
s	BIN data, or first number of device storing BIN data.	BIN 16-/32-bit
d	First number of device storing BCD data.	BCD 4-/8-digit

**Functions Conversion from BIN data into BCD data**

**BCD Conversion from BIN 16-bit data into BCD 4-digit data**

BIN data in s (0 to 9999) is converted into BCD data. The result is stored in d.

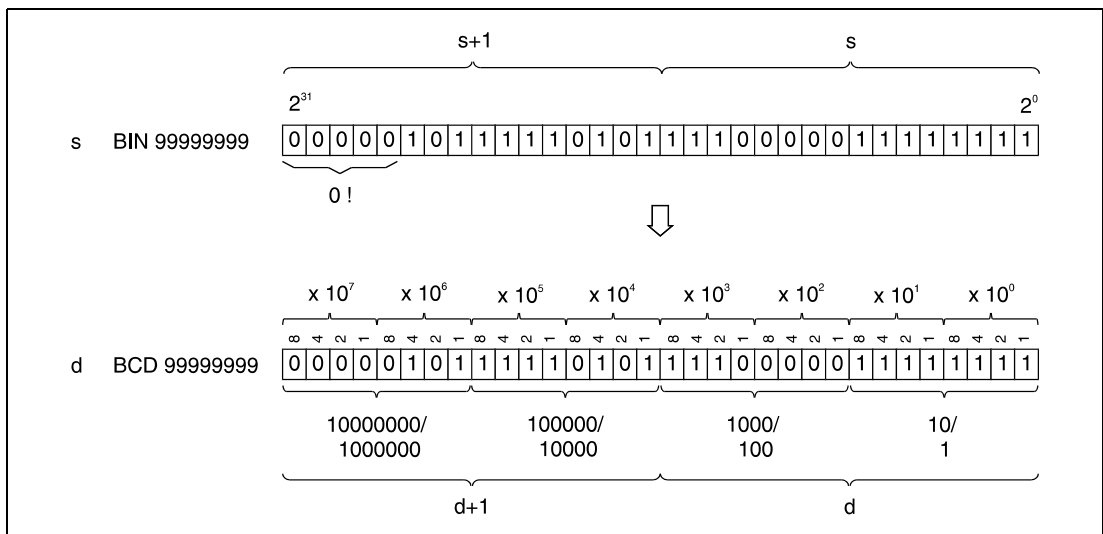
The most significant two bits of BIN data in s must be reset (0) when converted into BCD 4-digit data.



**DBCD Conversion from BIN 32-bit data into BCD 8-digit data**

BIN data in s (0 to 99999999) is converted into BCD data. The result is stored in d.

The most significant five bits of BIN data in s must be reset (0) when converted to BCD 8-digit data.





**Operation Errors**

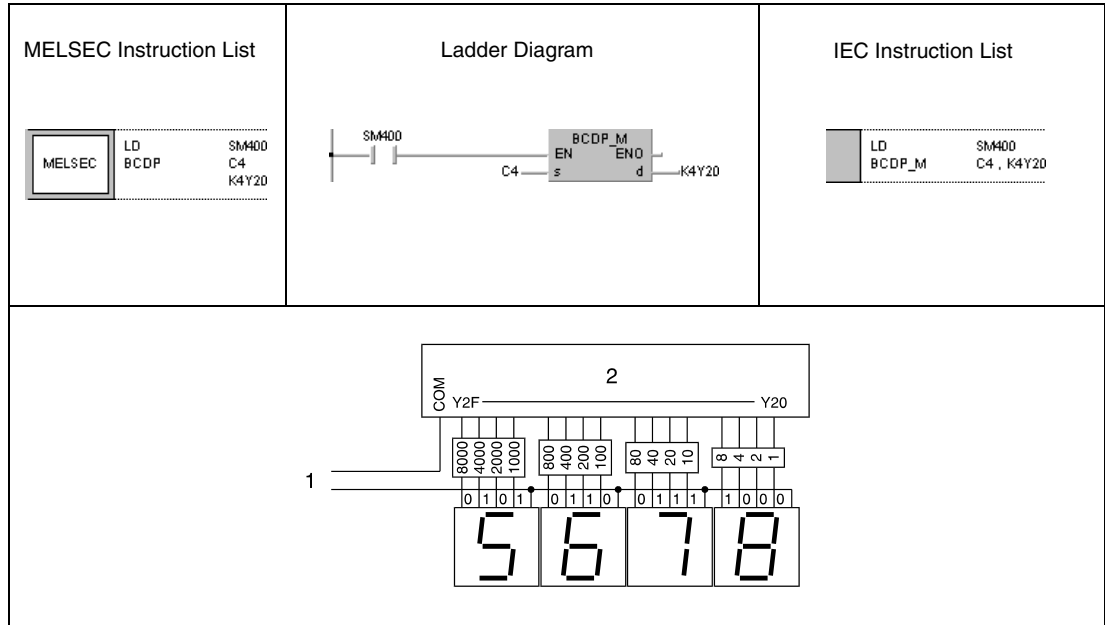
In the following cases an operation error occurs and the error flag is set:

- BIN 16-bit data in s exceeds the relevant device range of 0 to 9999 (Q series and System Q = error code 4100).
- BIN 32-bit data in s+1 or s exceed the relevant device range of 0 to 99999999 (Q series and System Q = error code 4100).

**Program Example**

**BCDP**

With leading edge from SM400, the following program outputs the current value in C4 (5678) to Y20 through Y2F. The output module displays the value on the display unit.



- <sup>1</sup> Output power supply
- <sup>2</sup> Output module

**6.3.2 BIN, BINP, DBIN, DBINP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

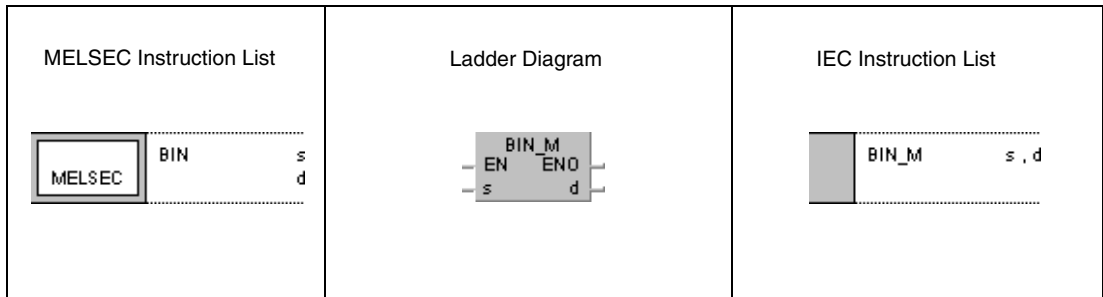
Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag						
Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level											
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I	N				M9012	M9010 M9011		
<b>BIN</b>																											
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								K1 ↓ K4	5	●		●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●									1			
<b>DBIN</b>																											
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								K1 ↓ K8	9	●		●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●									1			●

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this Programming Manual for the according number of steps.

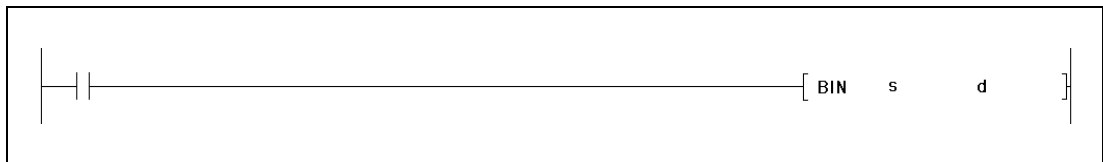
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word				U			
s	●	●	●	●	●	●	●	—	—	SM0	3
d	●	●	●	●	●	●	—	—	—		

**GX IEC Developer**



**GX Developer**



**Variables**

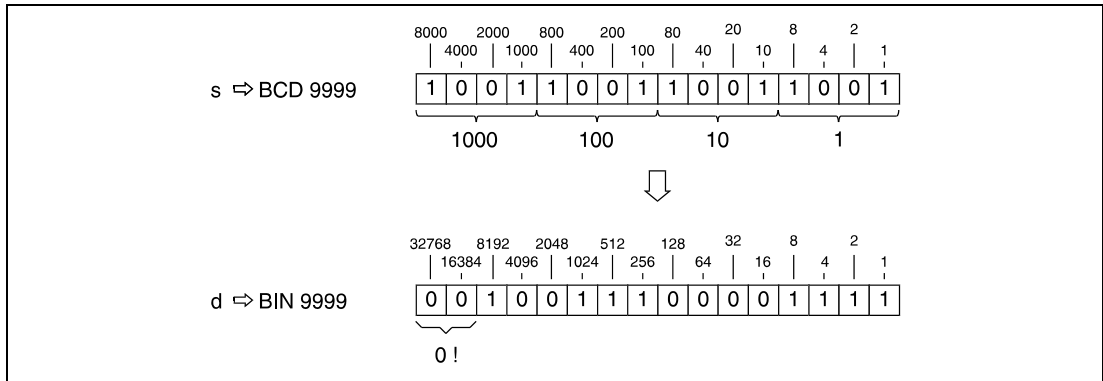
Set Data	Meaning	Data Type
s	BCD data, or first number of device storing BCD data.	BCD 4-/8-digit
d	First number of device storing BIN data.	BIN 16-/32-bit

**Functions Conversion from BCD data into BIN data**

**BIN Conversion from BCD 4-digit data into BIN 16-bit data**

BCD data in s (0 to 9999) is converted into BIN data. The result is stored in d.

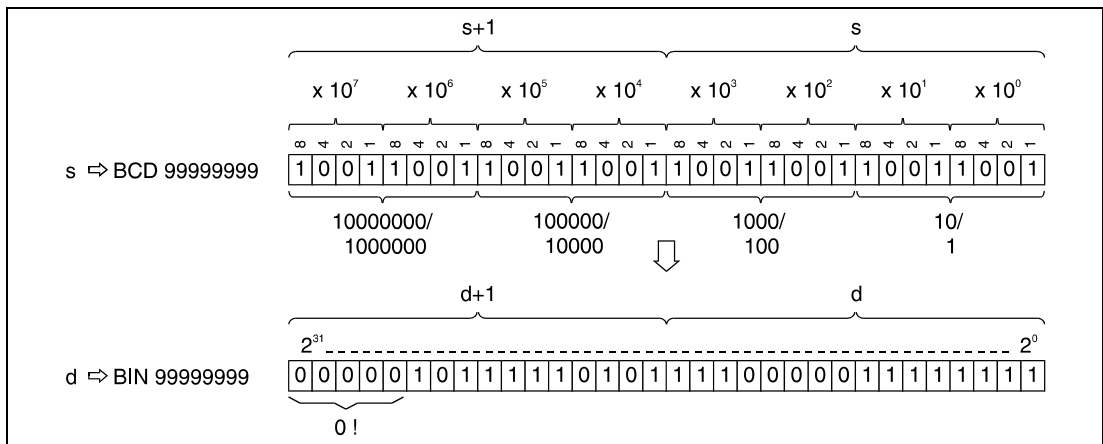
The most significant two bits of BIN data in d must be reset (0) when converted from BCD 4-digit it data.



**DBIN Conversion from BCD 8-digit data into BIN 32-bit data**

BCD data in s (0 to 99999999) is converted to BIN data. The result is stored in d.

The most significant five bits of BIN data in d must be reset (0) when converting from BCD 8-digit data.



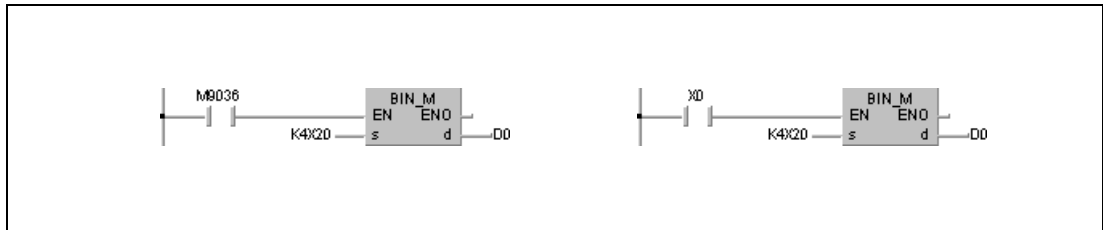
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The individual digits in s do not range within 0 to 9.
- When a Q series CPU or a CPU of the System Q is used, this error can be suppressed by turning SM722 ON. However, the instruction is not executed regardless of the status of SM722 if the specified value in s is out of range.

**NOTE**

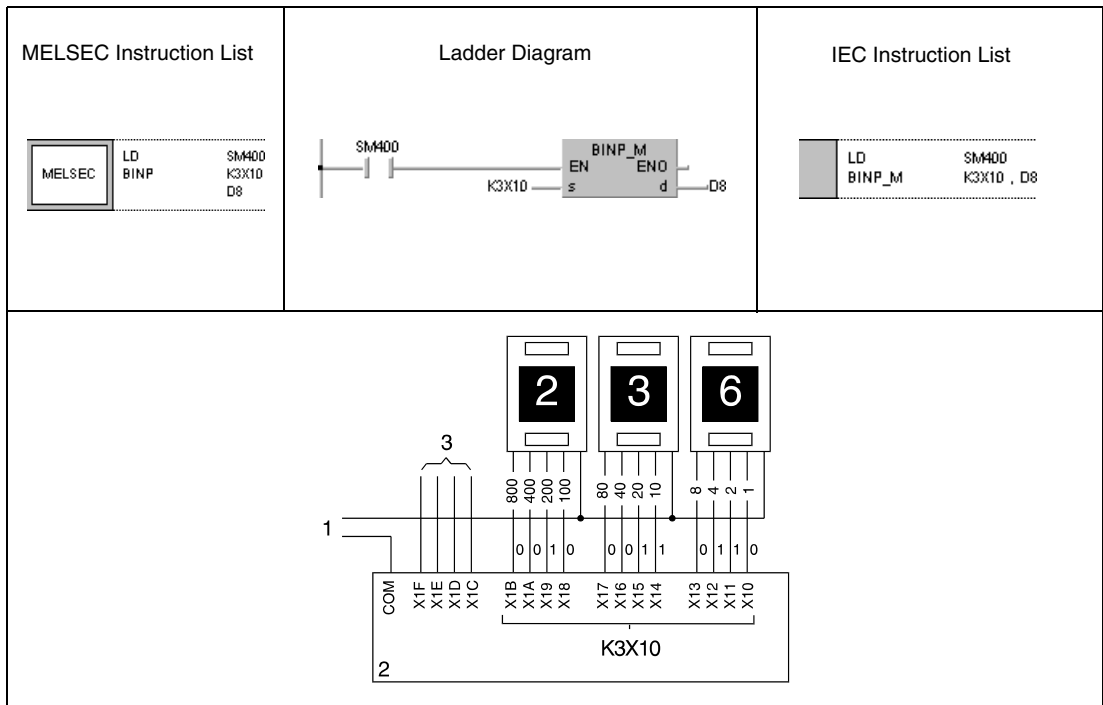
Due to the switching delay of BCD display units, errors in the program execution might occur with special relays M9036 or M9037 as input condition. In this case BCD data should first be set by a regular input device and then converted (A series only).



**Program Example 1**

**BINP**

With leading edge from SM400, the following program converts BCD data in X10 through X1B into BIN data. The result is stored in D8.

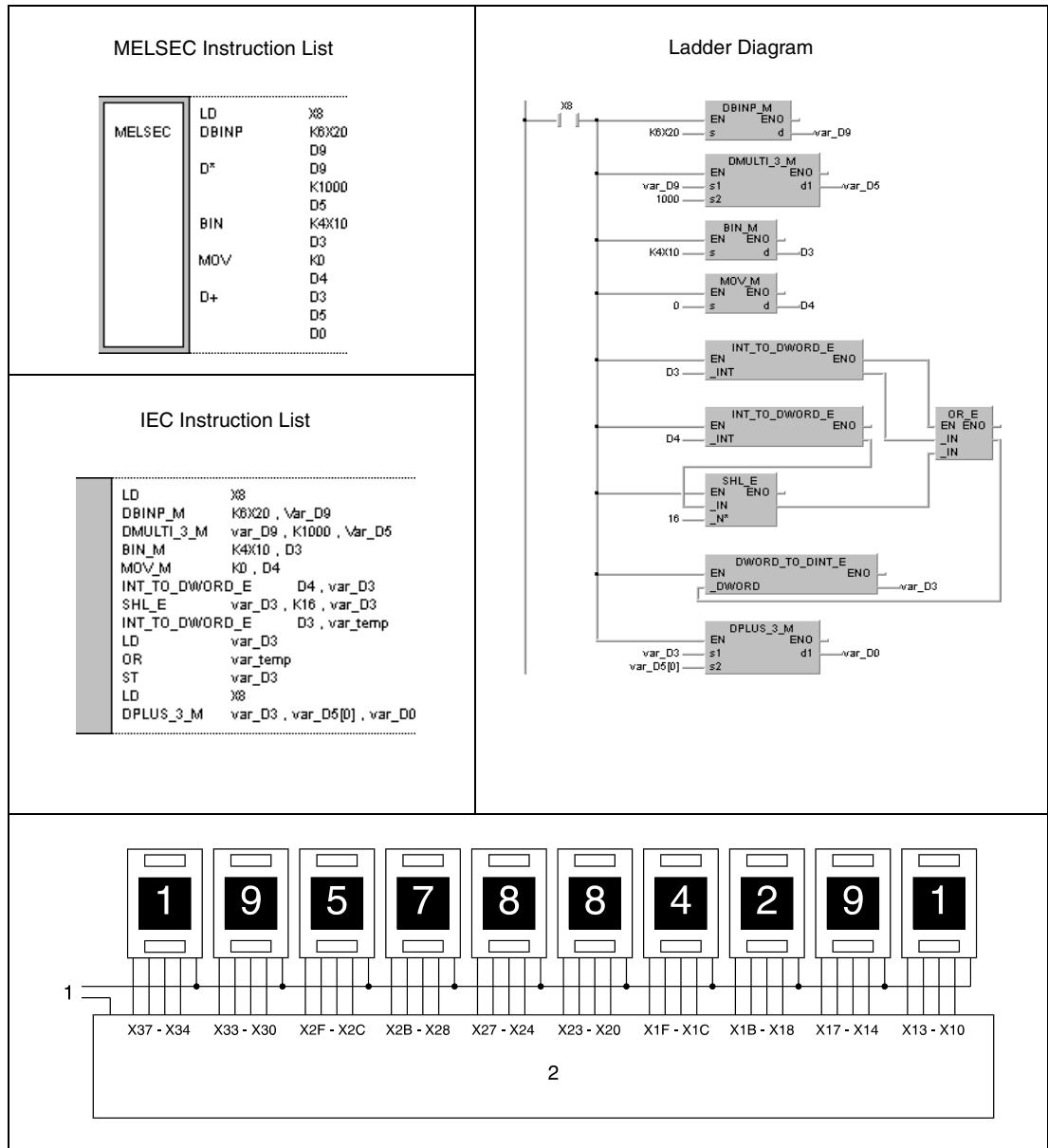


- 1 Input power supply
- 2 Input module
- 3 Available inputs

**Program** DBINP

**Example 2**

With leading edge from X8, the following program converts BCD data at X10 through X37 into BIN data. The result is stored in D0 through D1.



- <sup>1</sup> Input power supply
- <sup>2</sup> Input module

**NOTE**

*BCD data at X10 through X37 exceeding the relevant device range of 2147483647 cannot be processed by 32-bit devices! In this case the values in D0 and D1 become negative. For further details see chapter "Processing numerical data" in the Programming Manual.*

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual..*

6.3.3 FLT, FLTP, DFLT, DFLTP

CPU

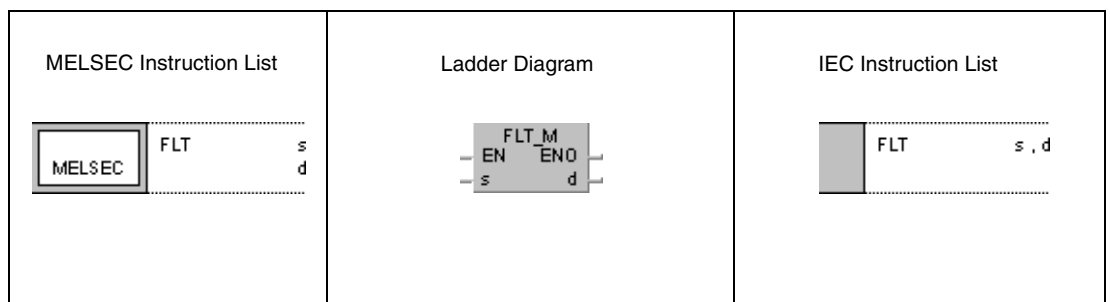
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

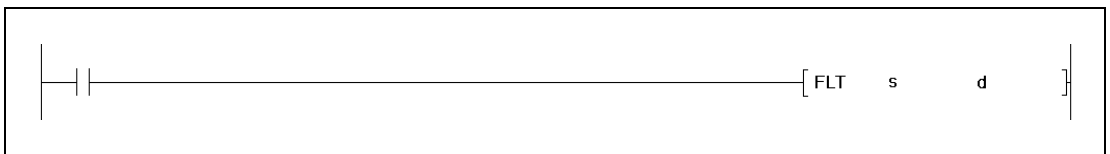
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	3	
d	—	●	●	—	●	—	—	—	—		

GX IEC Developer



GX Developer



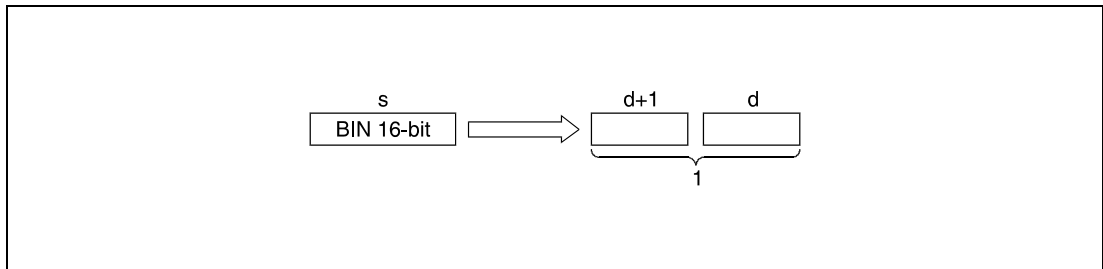
Variables

Set Data	Meaning	Data Type
s	BIN data, or first number of device storing BIN data.	BIN 16-/32-bit
d	First number of device storing floating point data.	Real number

**Functions Conversion from BIN data into floating point data**

**FLT Conversion from BIN 16-bit data into floating point data**

BIN 16-bit data in s is converted into floating point data. The result is stored in d.

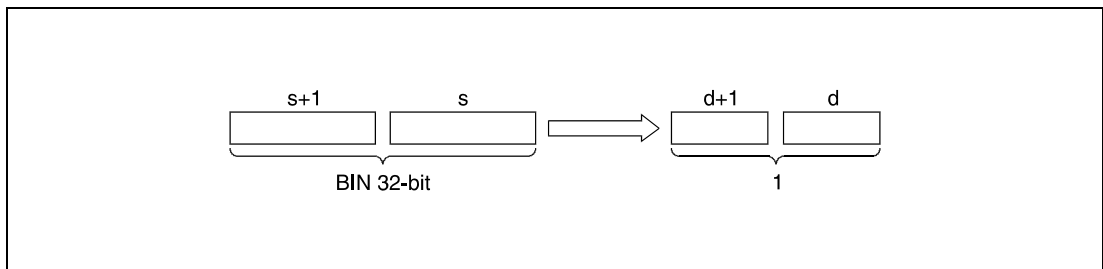


<sup>1</sup> Floating point data, data type real number

BIN 16-bit data designated by s has to range within -32768 and 32767.

**DFLT Conversion from BIN-32 bit data into floating point data**

BIN 32-bit data in s is converted into floating point data. The result is stored in d.

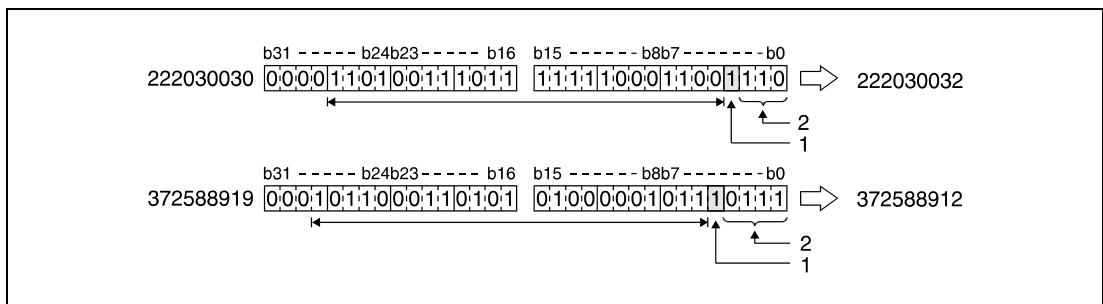


<sup>1</sup> Floating point data, data type real number

BIN 32-bit data designated by s and s+1 have to range within -2147483648 and 2147483647.

Due to the fact that floating point data (data type real number) is processed by simple 32-bit procedures, the number of significant bits is 24 for a binary display, or approx. 7 digits for a decimal display.

The result of the conversion is rounded off at the 25th bit. All higher bits are eliminated. For this reason, if the resulting integer exceeds a range of -16777216 to 16777215 (BIN 24-bit value), errors may occur in the conversion.



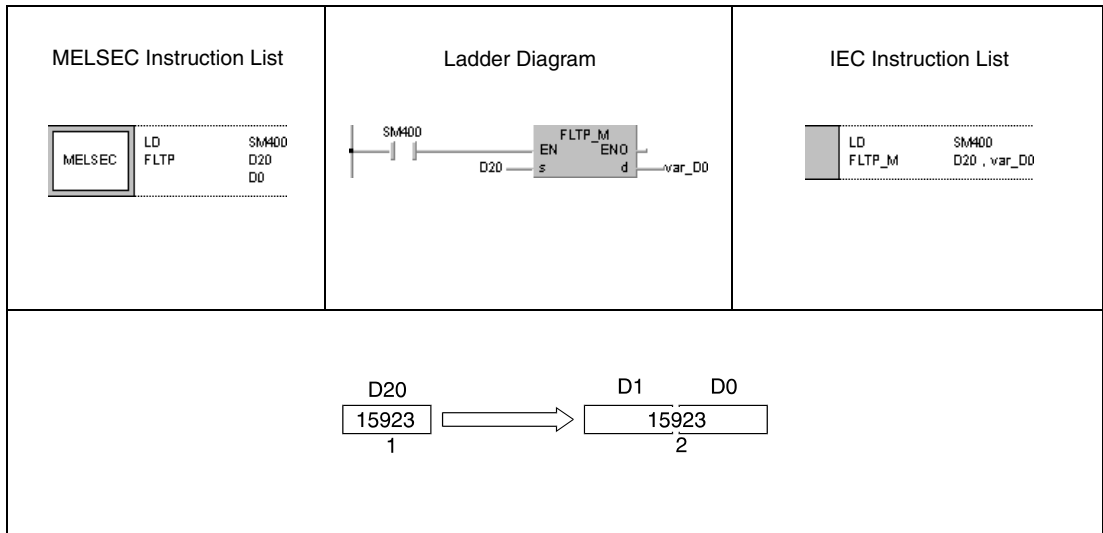
<sup>1</sup> Rounded off

<sup>2</sup> Eliminated

**Program Example 1**

FLTP

With leading edge from SM400, the following program converts BIN 16-bit data in D20 into floating point data. The result is stored in D0 and D1.

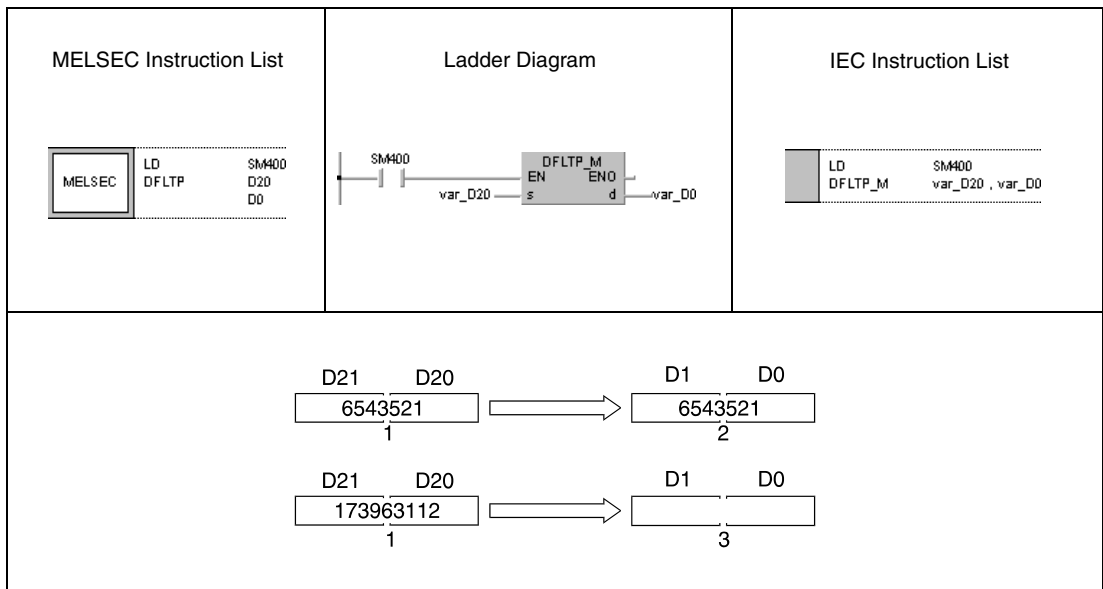


- <sup>1</sup> BIN 16-bit data
- <sup>2</sup> Floating point data, data type real number

**Program Example 2**

DFLTP

With leading edge from SM400, the following program converts BIN 32-bit data in D20 and D21 into floating point data. The result is stored in D0 and D1.



- <sup>1</sup> BIN 32-bit data
- <sup>2</sup> Floating point data, data type real number
- <sup>3</sup> Conversion error, because there are 7 significant digits

**NOTE**

*These programs will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



6.3.4 INT, INTP, DINT, DINTP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

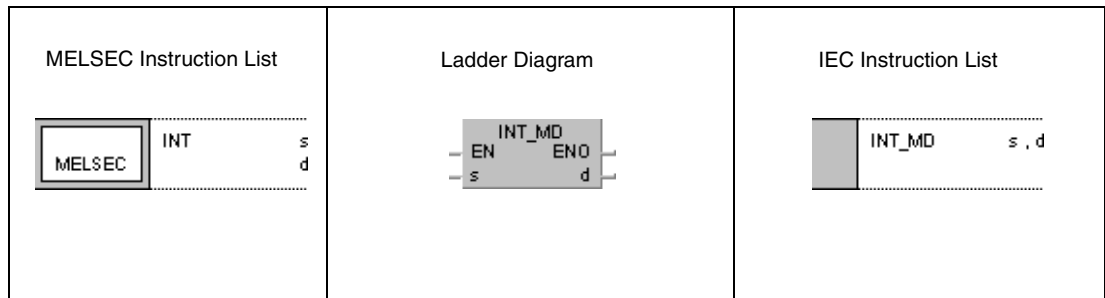
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

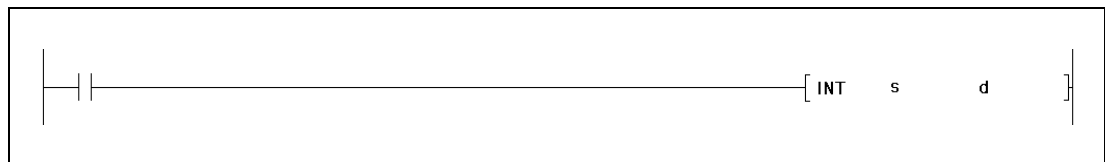
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



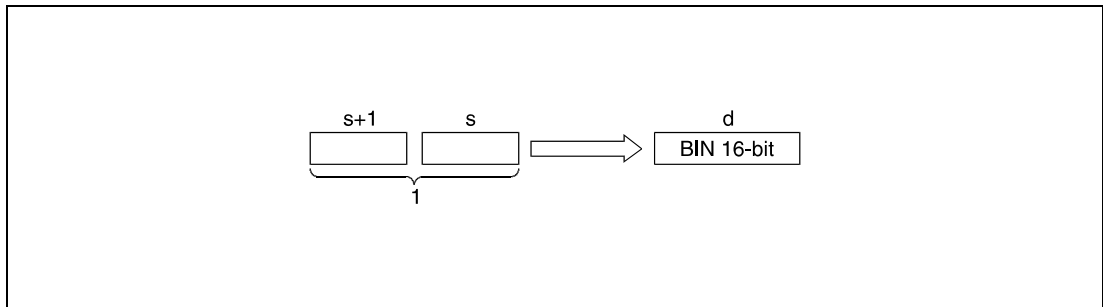
Variables

Set Data	Meaning	Data Type
s	Floating point data, or first number of device storing floating point data.	Real number
d	First number of device storing BIN data.	BIN 16-/32-bit

**Functions Conversion from floating point data into BIN data**

**INT Conversion from floating point data into BIN 16-bit data**

Floating point data in s is converted into BIN 16-bit data. The result is stored in d.



<sup>1</sup> Floating point data, data type real number

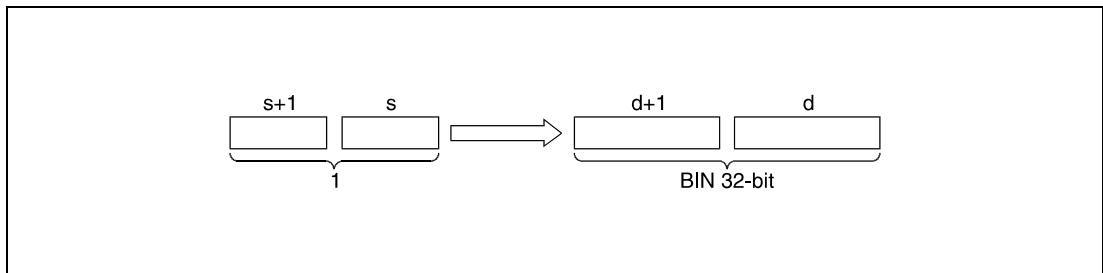
Floating point data in s and s+1 have to range within -32768 and 32767.

The converted integer value is stored as BIN 16-bit data.

The converted integer value is rounded off at the first digit after the decimal point.

**DINT Conversion from floating point data into BIN 32-bit data**

Floating point data in s is converted to BIN 32-bit data. The result is stored in d.



<sup>1</sup> Floating point data, data type real number

Floating point data in s and s+1 have to range within -2147483648 and 2147483647.

The converted integer value is stored as BIN 32-bit data.

The converted integer value is rounded off at the first digit after the decimal point.

**Operation Errors**

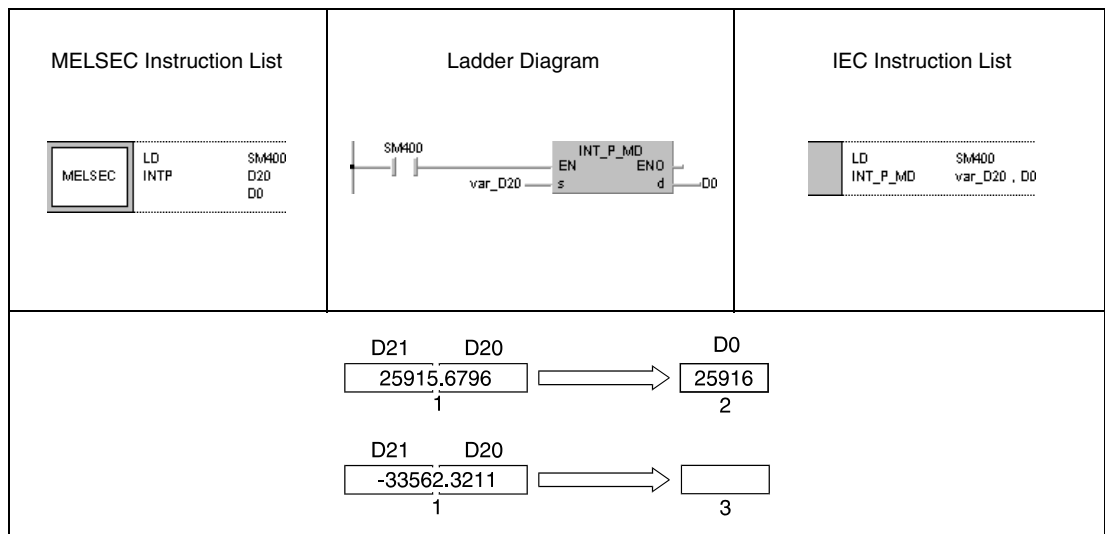
In the following cases an operation error occurs and the error flag is set:

- Performing an INT instruction, floating point data designated by s exceeds the relevant device range of -32768 to 32767.
- Performing a DINT instruction, floating point data designated by s exceeds the relevant device range of -2147483648 to 2147483647.

**Program Example 1**

INTP

With leading edge from SM400, the following program converts floating point data in S20 and D21 into BIN 16-bit data. The result is stored in D0.

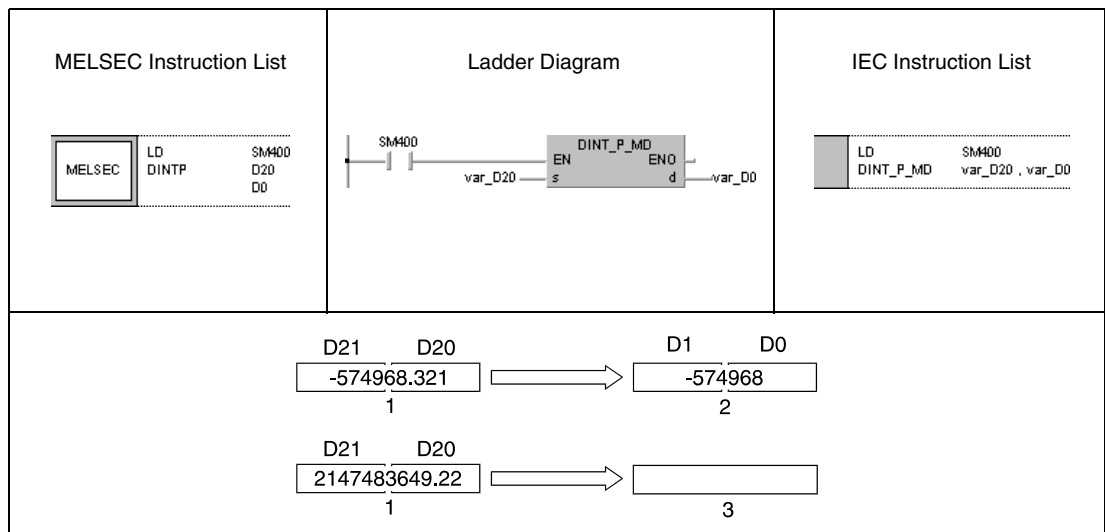


- <sup>1</sup> Floating point data, data type real number
- <sup>2</sup> BIN 16-bit data
- <sup>3</sup> No result. Value exceeds relevant device range of INT instruction. Error code is returned.

**Program Example 2**

DINTP

With leading edge from SM400, the following program converts floating point data in D20 and D21 into BIN 32-bit data. The result is stored in D0.



- <sup>1</sup> Floating point data, data type real number
- <sup>2</sup> BIN 32-bit data
- <sup>3</sup> No result. Value exceeds relevant device range of DINT instruction. Error code is returned.

**NOTE**

*These programs will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.3.5 DBL, DBLP

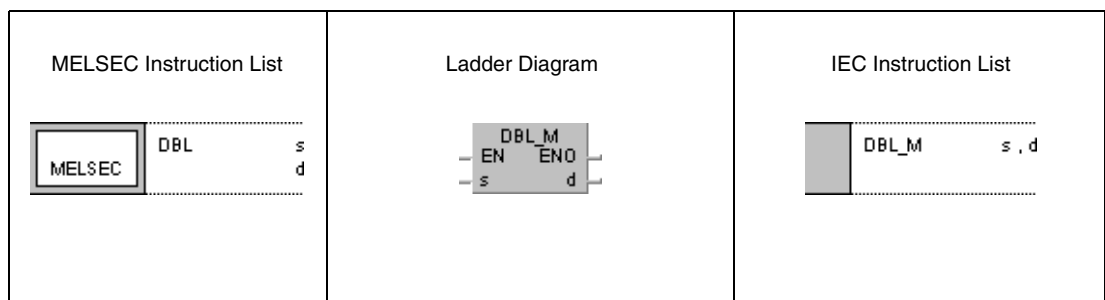
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

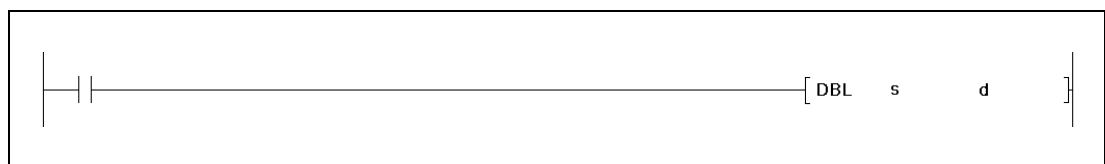
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	—	3
d	●	●	●	●	●	●	—	—	—		

GX IEC Developer



GX Developer



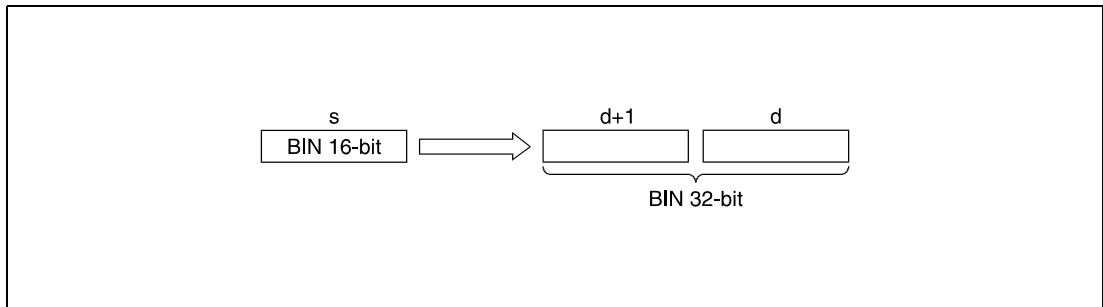
Variables

Set Data	Meaning	Data Type
s	First number of device storing data to be converted.	BIN 16-bit
d	First number of device storing converted data.	BIN 32-bit

**Functions**      **Conversion from BIN 16-bit data into BIN 32-bit data**

**DBL**      **Conversion from BIN 16-bit data into BIN 32-bit data**

BIN 16-bit data in s is converted into BIN 32-bit data with sign. The result is stored in d.



**Program Example**

DBLP

With leading edge from X20, the following program converts BIN 16-bit data in D100 into BIN 32-bit data. The result is stored in R0 and R1.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td style="padding: 2px;"></td> <td style="padding: 2px;">DBLP</td> <td style="padding: 2px;">D100 R0</td> </tr> </table>	MELSEC	LD	X20		DBLP	D100 R0	<p>Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td style="padding: 2px;">DBLP_M</td> <td style="padding: 2px;">D100 , var_R0</td> </tr> </table>	LD	X20	DBLP_M	D100 , var_R0
MELSEC	LD	X20										
	DBLP	D100 R0										
LD	X20											
DBLP_M	D100 , var_R0											

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.3.6 WORD, WORDP

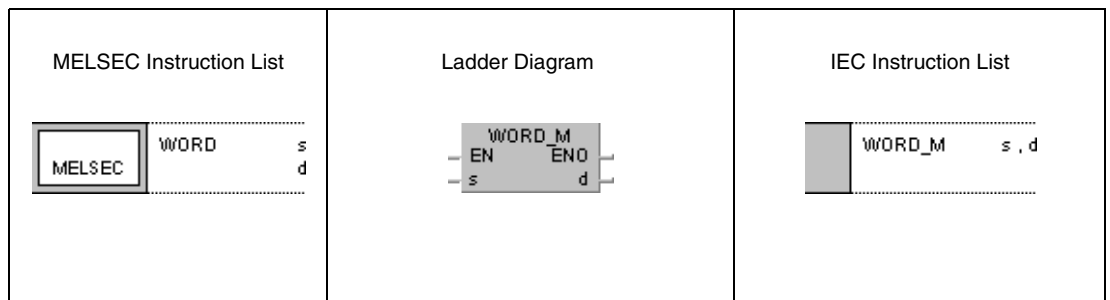
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

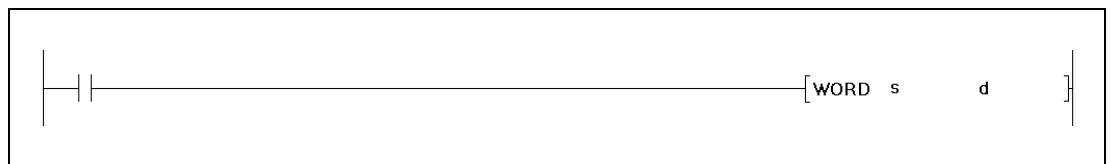
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



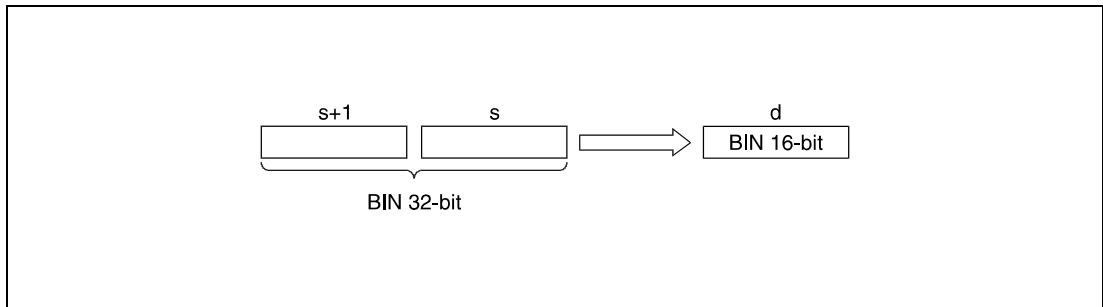
Variables

Set Data	Meaning	Data Type
s	First number of device storing data to be converted.	BIN 32-bit
d	First number of device storing converted data.	BIN 16-bit

**Functions** Conversion from BIN 32-bit data into BIN 16-bit data

**WORD Conversion from BIN 32-bit data into BIN 16-bit data**

BIN 32-bit data in s is converted into BIN 16-bit data. The result is stored in d.



**Operation Errors** In the following cases an operation error occurs and the error flag is set:

- The BIN data designated by s and s+1 exceed the relevant device range of -32768 to 32767 (error code: 4100).

**Program Example** WORDP

With leading edge from X20, the following program converts BIN 32-bit data in D100 and D101 into BIN 16-bit data. The result is stored in R0.

<p style="text-align: center;">MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 5px auto; width: fit-content;"> <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;"></td> <td style="padding: 2px;">WORDP</td> <td style="padding: 2px;">D100 R0</td> </tr> </table> </div>	MELSEC	LD	X20		WORDP	D100 R0	<p>Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 5px auto; width: fit-content;"> <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td style="padding: 2px;">WORDP_M</td> <td style="padding: 2px;">var_D100, R0</td> </tr> </table> </div>	LD	X20	WORDP_M	var_D100, R0		
MELSEC	LD	X20												
	WORDP	D100 R0												
LD	X20													
WORDP_M	var_D100, R0													
<table style="margin: auto;"> <tr> <td style="text-align: center;">R1</td> <td style="text-align: center;">R0</td> <td style="width: 20px;"></td> <td style="text-align: center;">D100</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">HFFFF8253</td> <td style="border: 1px solid black; padding: 2px;">(-32173)</td> <td style="font-size: 2em;">→</td> <td style="border: 1px solid black; padding: 2px;">H8253</td> </tr> <tr> <td></td> <td></td> <td></td> <td style="border: 1px solid black; padding: 2px;">(-32173)</td> </tr> </table>			R1	R0		D100	HFFFF8253	(-32173)	→	H8253				(-32173)
R1	R0		D100											
HFFFF8253	(-32173)	→	H8253											
			(-32173)											

**NOTE** *This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.3.7 GRY, GRYP, DGRY, DGRYP

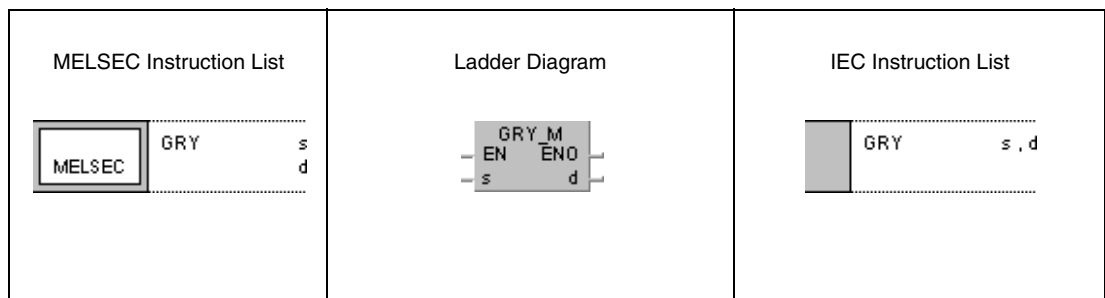
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

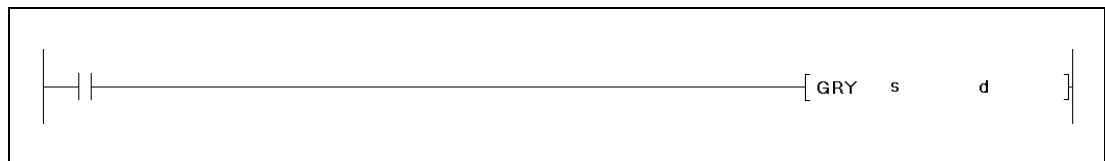
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



Variables

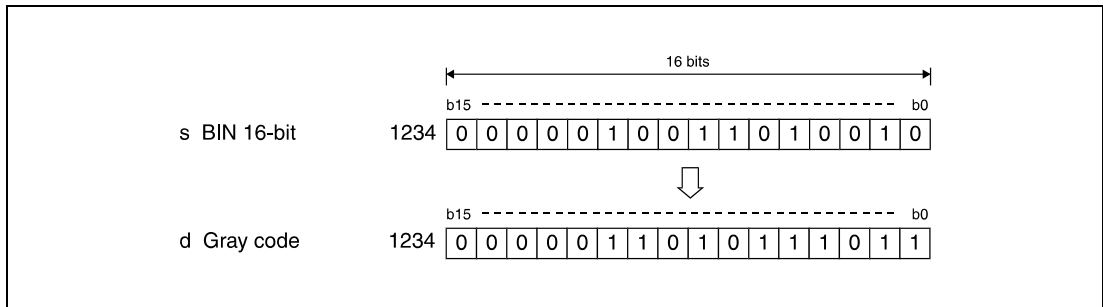
Set Data	Meaning	Data Type
s	BIN data, or first number of device storing BIN data.	BIN 16-/32-bit
d	First number of device storing converted Gray code data.	Gray code data 16-/32-bit



**Functions**      **Conversion from BIN data into Gray code data**

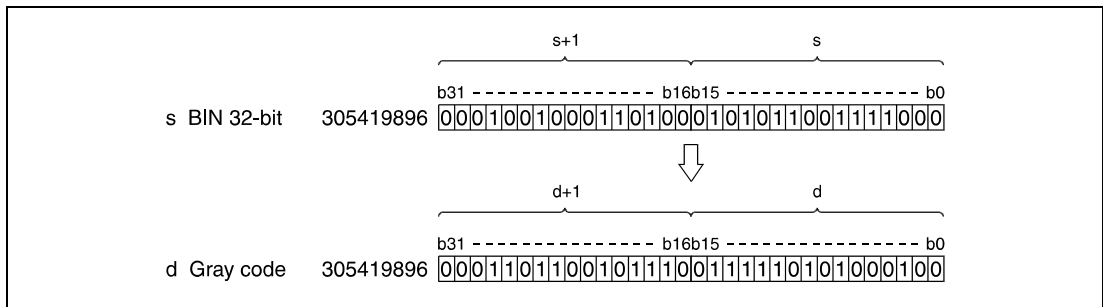
**GRY**      **Conversion from BIN 16-bit data into Gray code data**

BIN 16-bit data in s is converted into Gray code data. The result is stored in d.



**DGRY**      **Conversion from BIN 32-bit data into Gray code data**

BIN 32-bit data in s is converted into Gray code data. The result is stored in d.



**Operation Errors**

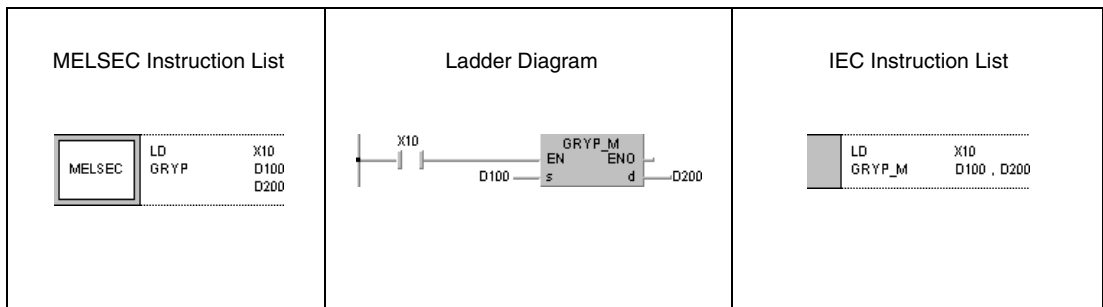
In the following cases an operation error occurs and the error flag is set:

- Data in s is negative.

**Program Example 1**

GRYP

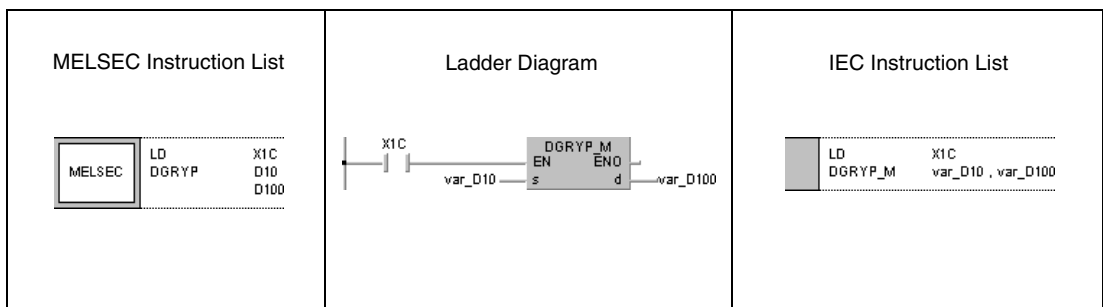
With leading edge from X10, the following program converts BIN 16-bit data in D100 into Gray code data. The result is stored in D200.



**Program Example 2**

DGRYP

With leading edge from X1C, the following program converts BIN 32-bit data in D10 and D11 into Gray code data. The result is stored in D100 and D101.



**NOTE**

*The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

**6.3.8 GBIN, GBINP, DGBIN, DGBINP**

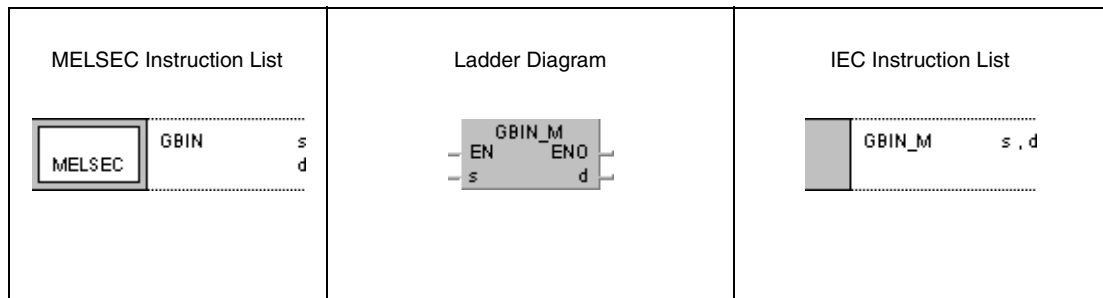
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

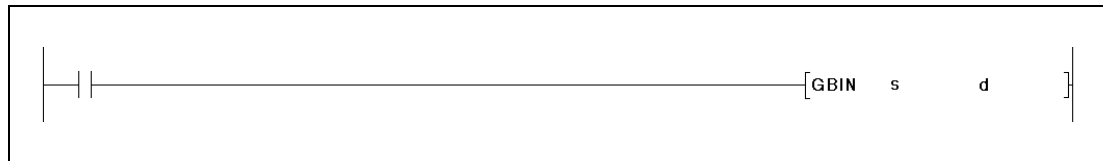
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

**GX IEC Developer**



**GX Developer**



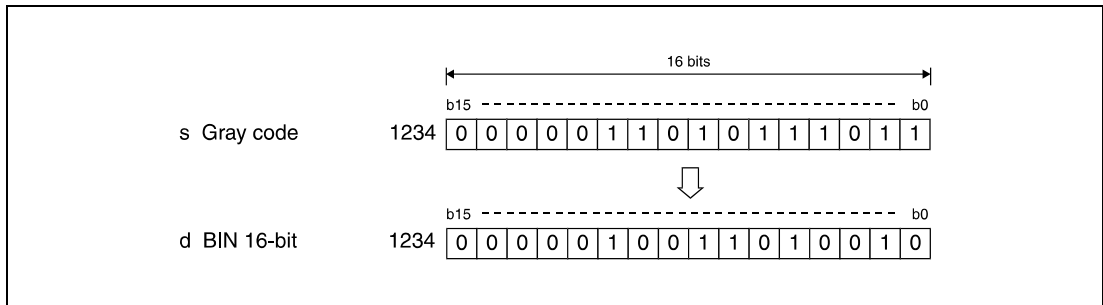
**Variables**

Set Data	Meaning	Data Type
s	Gray code data, or first number of device storing Gray code data.	Gray code data 16-/32-bit
d	First number of device storing converted BIN data.	BIN 16-/32-bit

**Functions**      **Conversion from Gray code data into BIN data**

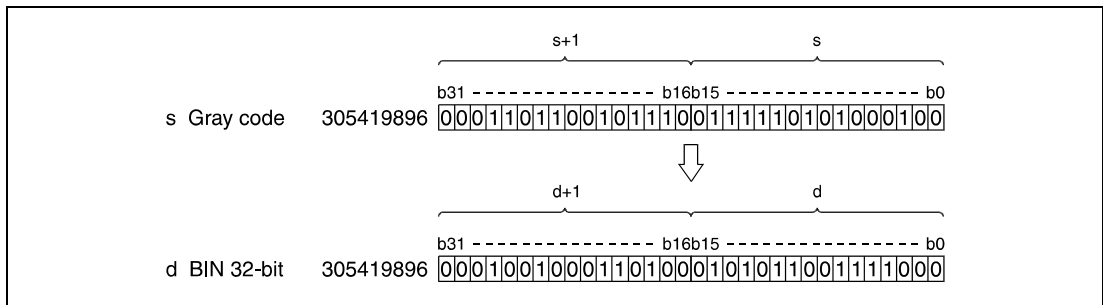
**GBIN**      **Conversion from Gray code data into BIN 16-bit data**

Gray code data in s is converted into BIN 16-bit data. The result is stored in d.



**DGBIN**      **Conversion from Gray code data into BIN 32-bit data**

Gray code data in s is converted into BIN 32-bit data. The result is stored in d.



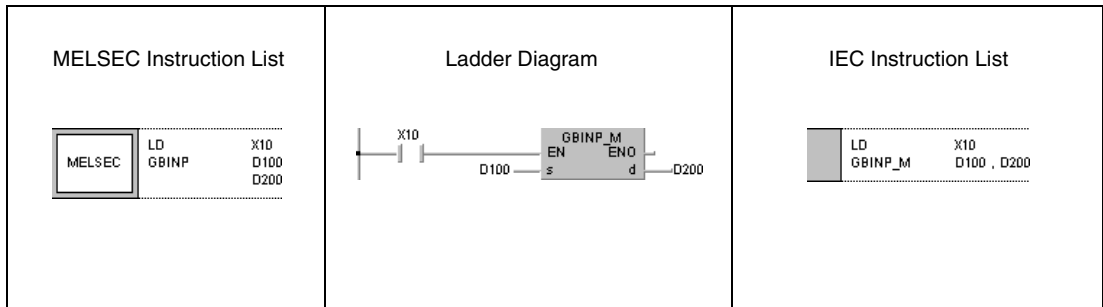
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- Performing a GBIN instruction, data in s exceeds the relevant device range of 0 to 32767.
- Performing a DGBIN instruction, data in s exceeds the relevant device range of 0 to 2147483647.

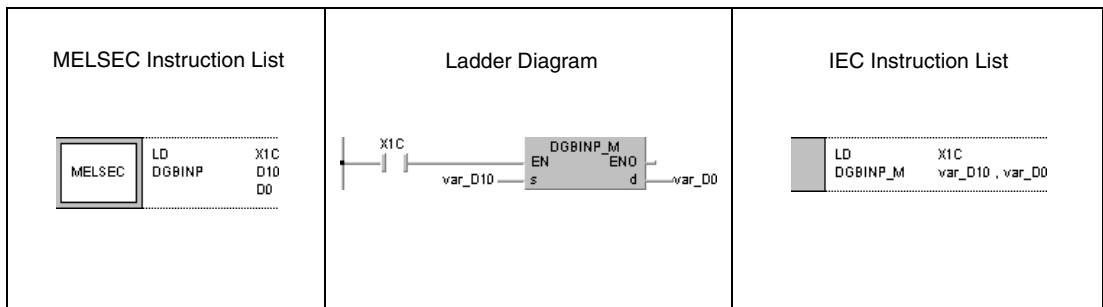
**Program Example 1** GBINP

With leading edge from X10, the following program converts Gray code data in D100 into BIN 16-bit data. The result is stored in D200.



**Program Example 2** DGBINP

With leading edge from X1C, the following program converts Gray code data in D10 and D11 into BIN 32-bit data. The result is stored in D0 and D1.



**NOTE**

*The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.3.9 NEG, NEGP, DNEG, DNEGP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																	Digit designation K1 ↓ K4	Number of steps 3	Index ●	Carry Flag M9012	Error Flag M9011			
Bit Devices							Word Devices (16-bit)							Constant		Pointer						Level		
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K						H (16#)	P	I
d	●	●	●	●	●	●	●	●	●	●	●	●	●	●	● <sup>2</sup>						K1 ↓ K4	3 ● <sup>1</sup>	●	●

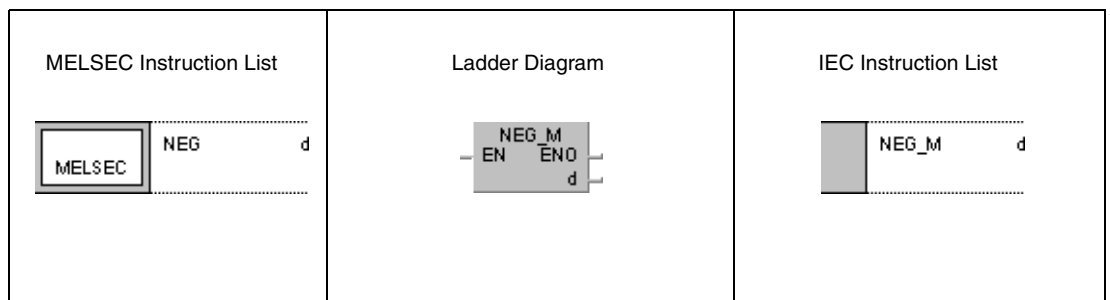
<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

<sup>2</sup> With DNEG and DNEGP not valid for AnN and AnS

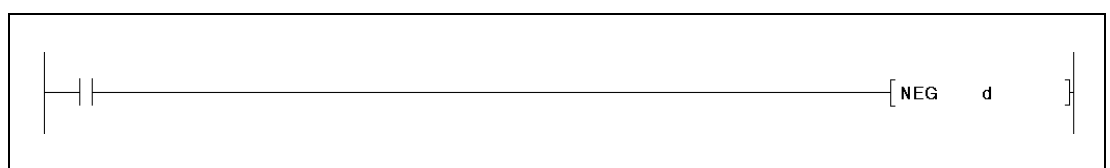
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
d	●	●	●	●	●	●	—	—	—	2	

GX IEC Developer



GX Developer



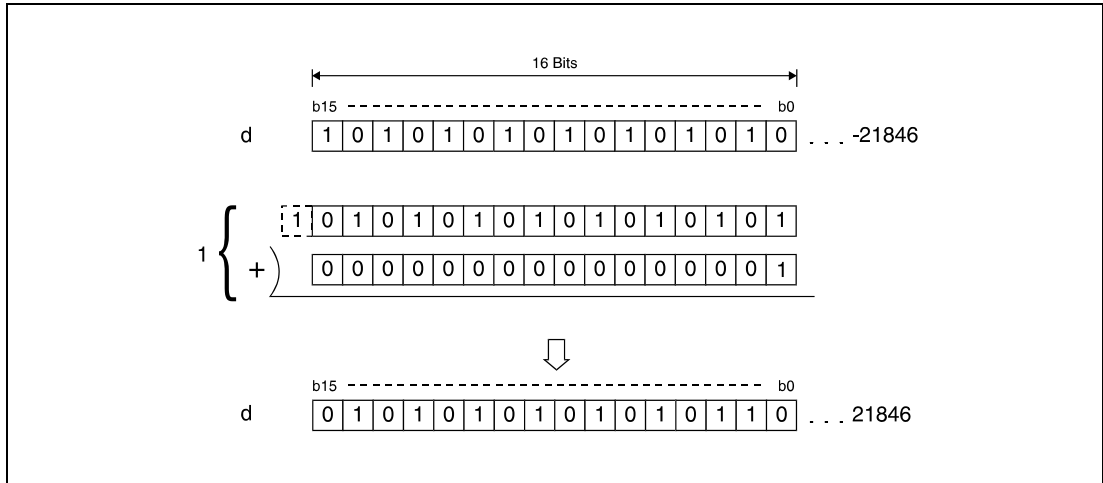
Variables

Set Data	Meaning	Data Type
d	First number of device storing data for the sign reversal.	BIN 16-/32-bit

**Functions Sign reversal for BIN data (complement of 2)**

**NEG Negation of BIN 16-bit data**

The NEG instruction (complement of 2 / NOT operation) reverses the sign of BIN 16-bit data. BIN 16-bit data in d is inverted first and then the value "1" is added. The result is stored in d.

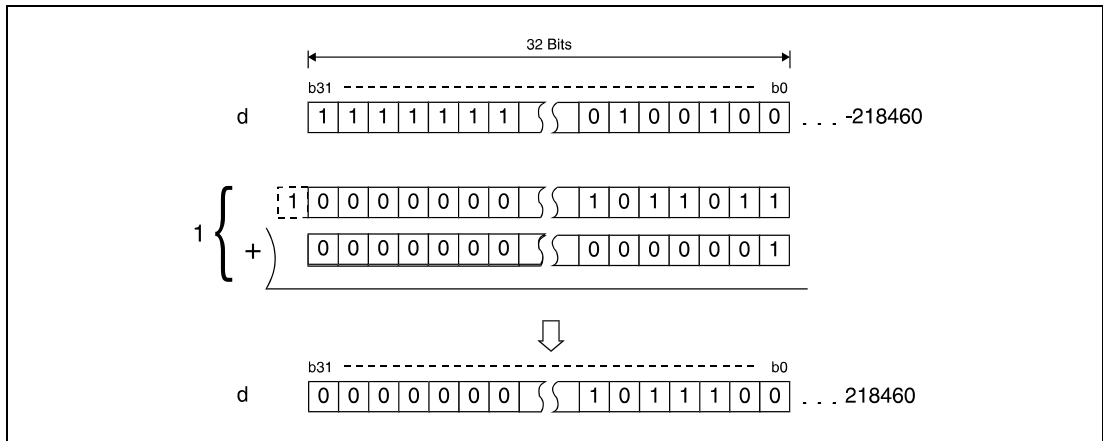


<sup>1</sup> Inversion with following addition

The function of this instruction is to change a negative sign into a positive one, or to change a positive sign into a negative one.

**DNEG Negation of BIN 32-bit data (Q-series and System Q only)**

The DNEG instruction (complement of 2 / NOT operation) reverses the sign of BIN 32-bit data. BIN 32-bit data in d is inverted first and then the value "1" is added. The result is stored in d.

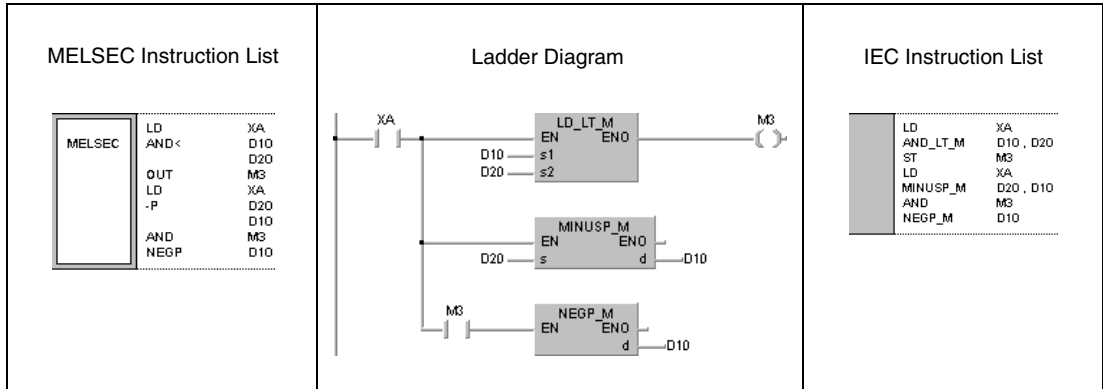


<sup>1</sup> Inversion with following addition

**Program Example**

**NEGP**

With leading edge from XA, the following program subtracts data in D10 from data in D20. M3 is set, if D10 is less than D20. If M3 is set, the result in D10 is the absolute value (complement of 2) and becomes positive.





**6.3.10 ENEG, ENEGP**

**CPU**

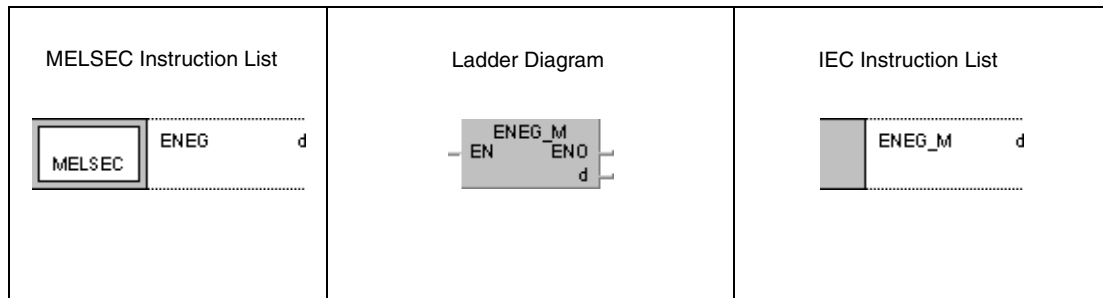
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

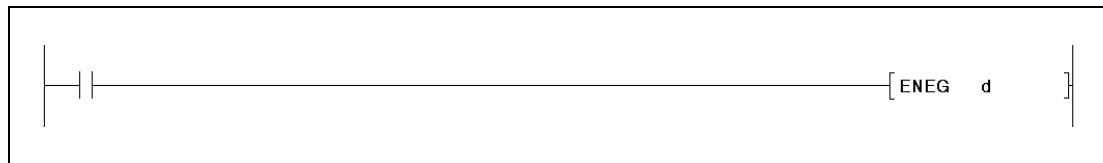
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
d	—	●	●	—	●	●	—	—	—	—	2

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
d	First number of device storing floating point data for the sign reversal.	Real number

**Functions** Sign reversal for floating point data

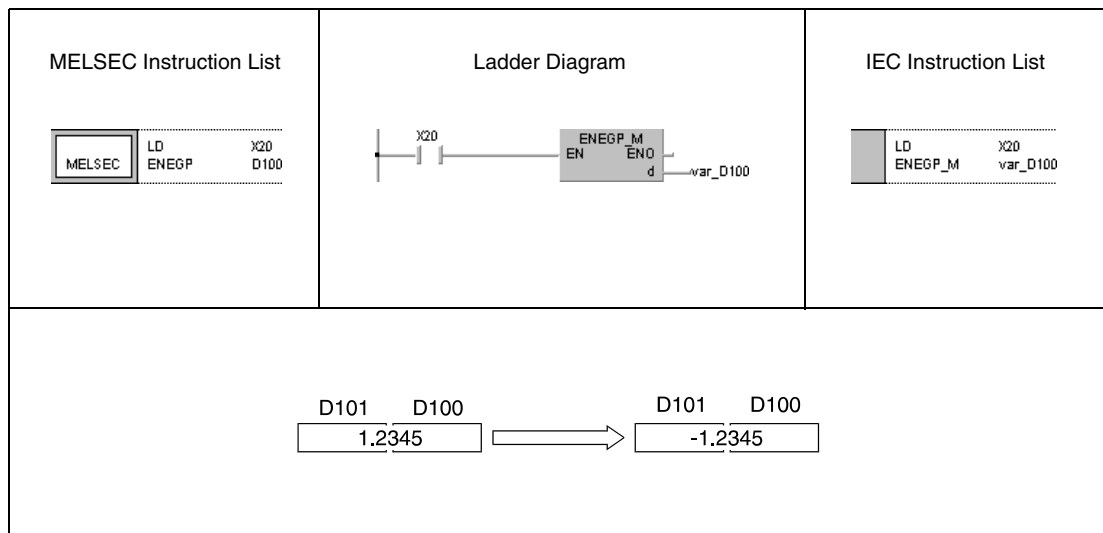
**ENEG** Negation of floating point data

These instructions negate floating point data in d. The result is stored in d.

The function of these instructions is to change a negative sign into a positive one, or a positive sign into a negative one.

**Program Example** ENEGP

With leading edge from X20, the following program negates floating point data in D100 and D101. The result is stored in D100 and D101.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

**6.3.11 BKBCD, BKBCDP**

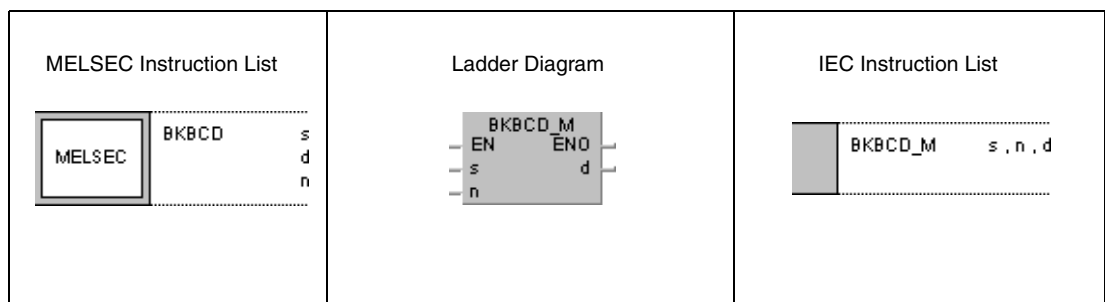
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

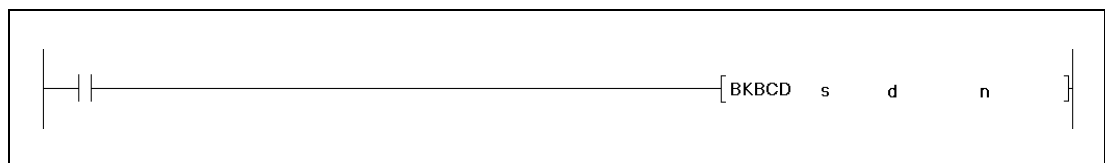
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	First number of device storing BIN data to be converted.	BIN 16-bit
d	First number of device storing converted BCD data.	BCD 4-digit
n	Number of data blocks to be converted.	BIN 16-bit

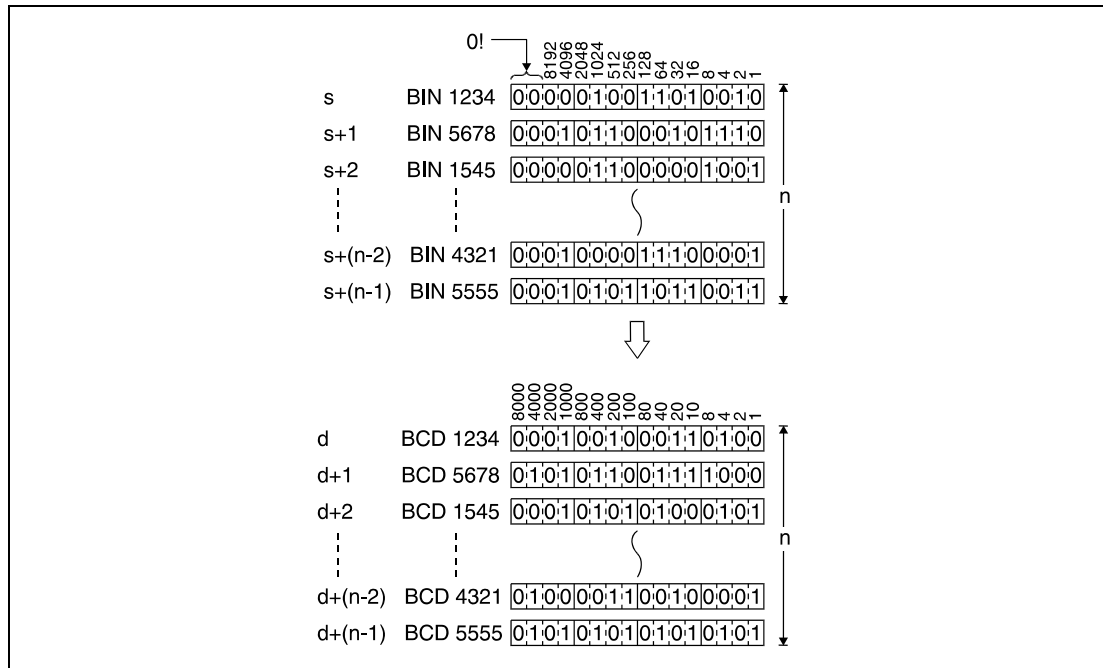
**Functions**      **Conversion from BIN block data into BCD block data**

**BKBCD Conversion from BIN 16-bit block data into BCD 4-digit block data**

This instruction converts each nth BIN 16-bit block in s into the nth BCD 4-digit block. Converted data is stored in d.

BIN 16-bit block data in s has to range within 0 and 9999.

The most significant two bits of the BIN 16-bit data blocks in s must be reset (0).



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The number of data blocks determined by n exceeds the storage device numbers designated by s and d (error code: 4101).
- BIN block data in s exceeds the relevant device range of 0 to 9999 (error code: 4100).
- The storage device numbers designated by s and d overlap (error code: 4101).

For details on index qualification refer to chapter 3.6.

**Program Example** BKBCDP

With leading edge from X20, the following program converts BIN 16-bit block data in D100 into BCD 4-digit block data. Converted data is stored in D200. The number of data blocks (3) converted is stored in D0.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																																																																																																																																													
<table border="1" style="border-collapse: collapse; width: 80%; margin: auto;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">BKBCDP</td> <td style="padding: 2px;">D100 D200 D0</td> </tr> </table>	MELSEC	LD	X20	MELSEC	BKBCDP	D100 D200 D0		<table border="1" style="border-collapse: collapse; width: 80%; margin: auto;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">X20</td> </tr> <tr> <td style="padding: 2px;">BKBCDP_M</td> <td style="padding: 2px;">D100 , D0 , D200</td> </tr> </table>	LD	X20	BKBCDP_M	D100 , D0 , D200																																																																																																																																			
MELSEC	LD	X20																																																																																																																																													
MELSEC	BKBCDP	D100 D200 D0																																																																																																																																													
LD	X20																																																																																																																																														
BKBCDP_M	D100 , D0 , D200																																																																																																																																														
<table style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">D100</td> <td style="padding: 2px;">BIN 5432</td> <td style="padding: 2px;"> <table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table> </td> </tr> <tr> <td style="padding: 2px;">D101</td> <td style="padding: 2px;">BIN 4444</td> <td style="padding: 2px;"> <table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table> </td> </tr> <tr> <td style="padding: 2px;">D102</td> <td style="padding: 2px;">BIN 3210</td> <td style="padding: 2px;"> <table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table> </td> </tr> <tr> <td colspan="2"></td> <td style="text-align: center; padding: 5px;"> <div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 5px;">↓</div> <div style="margin-right: 10px;">D0</div> <div style="border: 1px solid black; padding: 2px 5px;">3</div> </div> </td> </tr> <tr> <td style="padding: 2px;">D200</td> <td style="padding: 2px;">BCD 5432</td> <td style="padding: 2px;"> <table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table> </td> </tr> <tr> <td style="padding: 2px;">D201</td> <td style="padding: 2px;">BCD 4444</td> <td style="padding: 2px;"> <table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table> </td> </tr> <tr> <td style="padding: 2px;">D202</td> <td style="padding: 2px;">BCD 3210</td> <td style="padding: 2px;"> <table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table> </td> </tr> </table>			D100	BIN 5432	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0	0	0	0	0	D101	BIN 4444	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	0	1	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0	D102	BIN 3210	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0			<div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 5px;">↓</div> <div style="margin-right: 10px;">D0</div> <div style="border: 1px solid black; padding: 2px 5px;">3</div> </div>	D200	BCD 5432	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	1	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	D201	BCD 4444	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	D202	BCD 3210	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
D100	BIN 5432	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0	0	0	0	0																																																																																																																									
0	0	0	1																																																																																																																																												
0	1	0	1																																																																																																																																												
1	0	0	1																																																																																																																																												
1	1	1	0																																																																																																																																												
0	0	0	0																																																																																																																																												
D101	BIN 4444	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	0	1	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0																																																																																																																									
0	0	0	1																																																																																																																																												
0	0	0	1																																																																																																																																												
1	0	1	1																																																																																																																																												
1	1	1	0																																																																																																																																												
0	0	0	0																																																																																																																																												
D102	BIN 3210	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0																																																																																																																									
0	0	0	0																																																																																																																																												
1	1	0	0																																																																																																																																												
1	0	0	0																																																																																																																																												
1	0	0	1																																																																																																																																												
0	0	0	0																																																																																																																																												
		<div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 5px;">↓</div> <div style="margin-right: 10px;">D0</div> <div style="border: 1px solid black; padding: 2px 5px;">3</div> </div>																																																																																																																																													
D200	BCD 5432	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	1	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0																																																																																																																									
0	1	0	1																																																																																																																																												
0	1	0	0																																																																																																																																												
0	0	1	1																																																																																																																																												
0	0	0	1																																																																																																																																												
0	0	0	0																																																																																																																																												
D201	BCD 4444	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0																																																																																																																									
0	1	0	0																																																																																																																																												
0	1	0	0																																																																																																																																												
0	1	0	0																																																																																																																																												
0	0	0	1																																																																																																																																												
0	0	0	0																																																																																																																																												
D202	BCD 3210	<table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">1</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">1</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> <tr><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td><td style="border: 1px solid black; padding: 1px;">0</td></tr> </table>	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0																																																																																																																									
0	0	1	1																																																																																																																																												
0	0	1	0																																																																																																																																												
0	0	0	0																																																																																																																																												
1	0	0	0																																																																																																																																												
0	0	0	0																																																																																																																																												

6.3.12 BKBIN, BKBINP

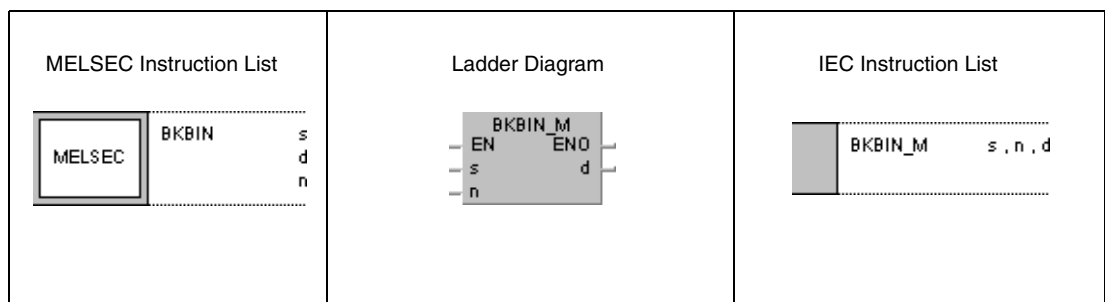
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

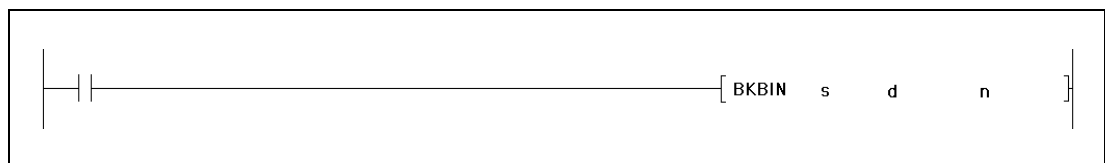
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer



GX Developer



Variables

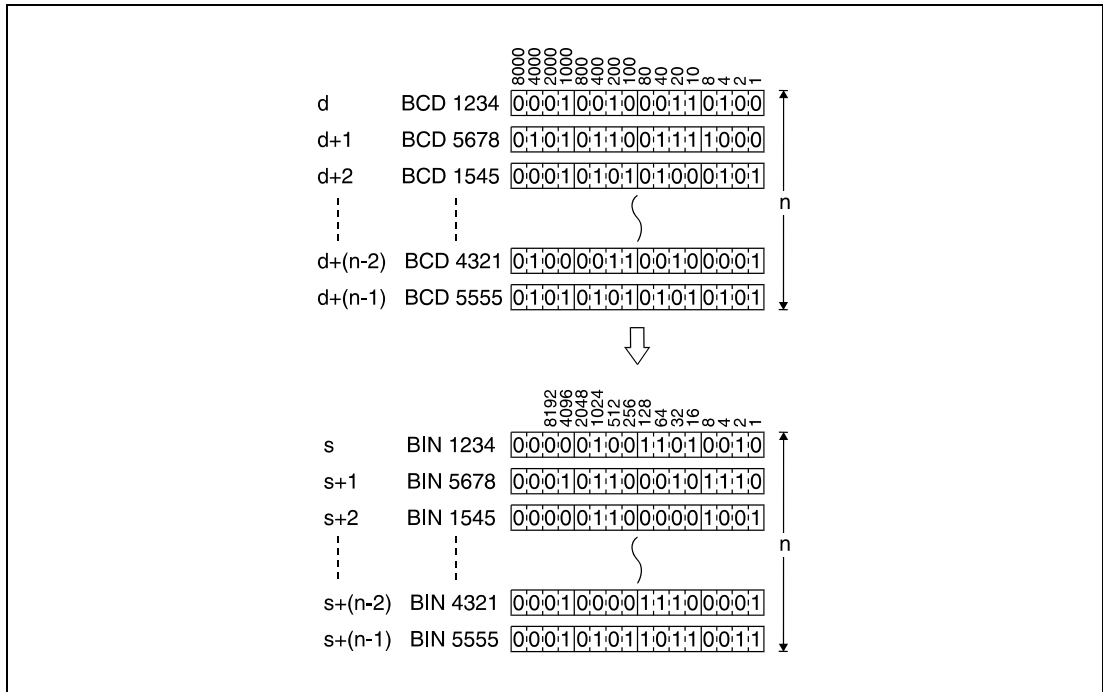
Set Data	Meaning	Data Type
s	First number of device storing BCD data to be converted.	BCD 4-digit
d	First number of device storing converted BIN data.	BIN 16-bit
n	Number of data blocks to be converted.	BIN 16-bit

**Functions Conversion from BCD block data into BIN block data**

**BKBIN Conversion from BCD 4-digit block data into BIN 16-bit block data**

This instruction converts each nth BCD 4-digit block in s into the nth BIN 16-bit block. Converted data is stored in d.

BIN 16-bit block data in s has to range within 0 to 9999.



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

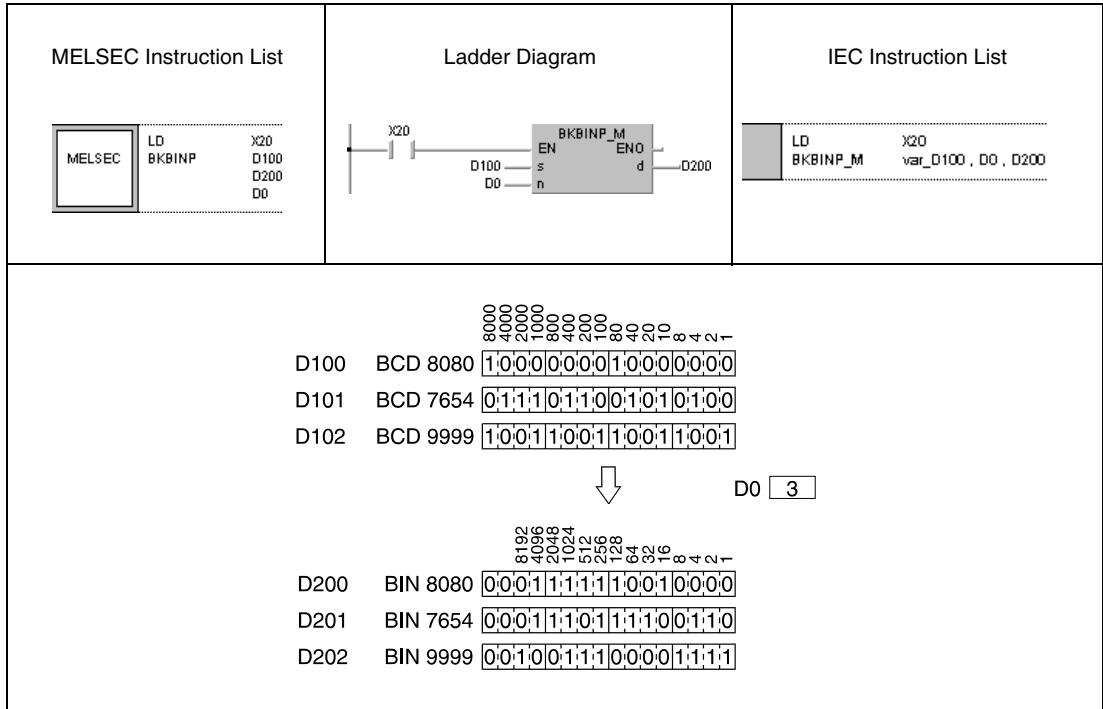
- The number of data blocks determined by n exceeds the storage device numbers designated by s and d.
- BCD block data in s exceeds the relevant device range of 0 to 9999.
- The storage device numbers designated by s and d overlap.

For details on index qualification refer to chapter 3.6.

**Program Example**

BKBINP

With leading edge from X20, the following program converts BCD 4-digit block data in D100 into BIN 16-bit block data. Converted data is stored in D200. The number of data blocks (3) converted is stored in D0.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



## 6.4 Data transfer instructions

These instructions transfer, invert, or exchange data. In total, 24 different instructions are supplied:

**NOTE** *Transferred data remain stored until they are replaced. Therefore, data even remain stored if the input condition of the transfer instruction is reset.*

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
BIN Data Transfer (16-/32-bit)	MOV	MOV_M
	MOVP	MOVP_M
	DMOV	DMOV_M
	DMOVP	DMOVP_M
Transfer of Floating Point Data	EMOV	EMOV_M
	EMOVP	EMOVP_M
Transfer of Character String Data	\$MOV	STRING_MOV_M
	\$MOVP	STRING_MOVP_M
Inverted BIN Data Transfer (16-/32-bit)	CML	CML_M
	CMLP	CMLP_M
	DCML	DCML_M
	DCMLP	DCMLP_M
Block Data Transfer	BMOV	BMOV_M
	BMOVP	BMOVP_M
Block Transfer of identical Data	FMOV	FMOV_M
	FMOVP	FMOVP_M
BIN Data Exchange (16-/32-bit)	XCH	XCH_M
	XCHP	XCHP_M
	DXCH	DXCH_M
	DXCHP	DXCHP_M
BIN Data Exchange (16-bit blocks)	BXCH	BXCH_M
	BXCHP	BXCHP_M
Byte Exchange (upper and lower byte)	SWAP	SWAP_MD
	SWAPP	SWAP_P_MD

**NOTE** *Within the IEC editors please use the IEC commands.*

6.4.1 MOV, MOVP, DMOV, DMOVP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag					
Bit Devices				Word Devices (16-bit)						Constant		Pointer		Level												
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I	N			
<b>MOV</b>																										
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K4	5	●		●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●								1			
<b>DMOV</b>																										
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K8	7	●		●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●								1			

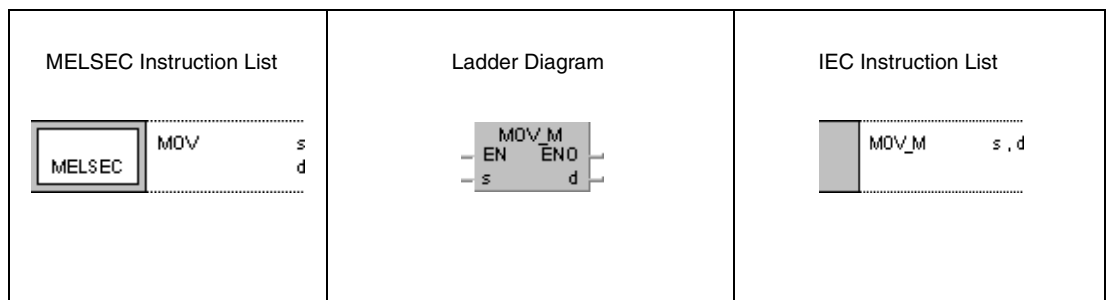
<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

Devices  
MELSEC Q

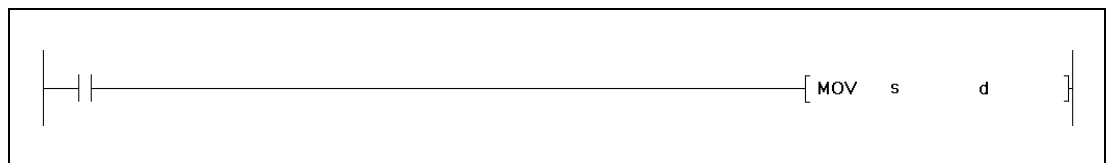
Usable Devices								Error Flag	Number of steps	
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	3 <sup>1)</sup>
d	●	●	●	●	●	●	—	—	—	

<sup>1</sup> The number of steps depends on the device and the type of CPU.  
 If a QnA-CPU or a single processor CPU of the System Q is used: 3  
 If a System Q multi processor CPU is used with internal word devices (except for file register ZR) or constants: 2  
 Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K4, and which use no index qualification: 2  
 If a System Q multi processor CPU is used with devices other than above mentioned: 3

GX IEC Developer



GX Developer



**Variables**

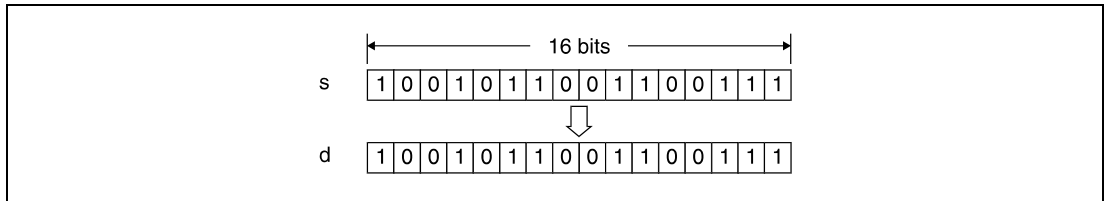
Set Data	Meaning	Data Type
s	Source data, or first number of device storing data to be transferred.	BIN 16-/32-bit
d	First number of destination device to store transferred data.	

**Functions**

**BIN data transfer**

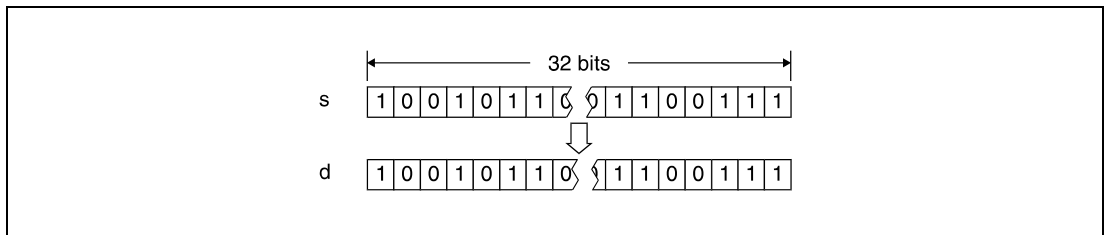
**MOV BIN 16-bit data transfer**

The MOV instruction transfers BIN 16-bit data in s to the device designated by d.



**DMOV BIN 32-bit data transfer**

The DMOV instruction transfers BIN 32-bit data in s to the device designated by d.



**Program Example 1**

**MOVP**

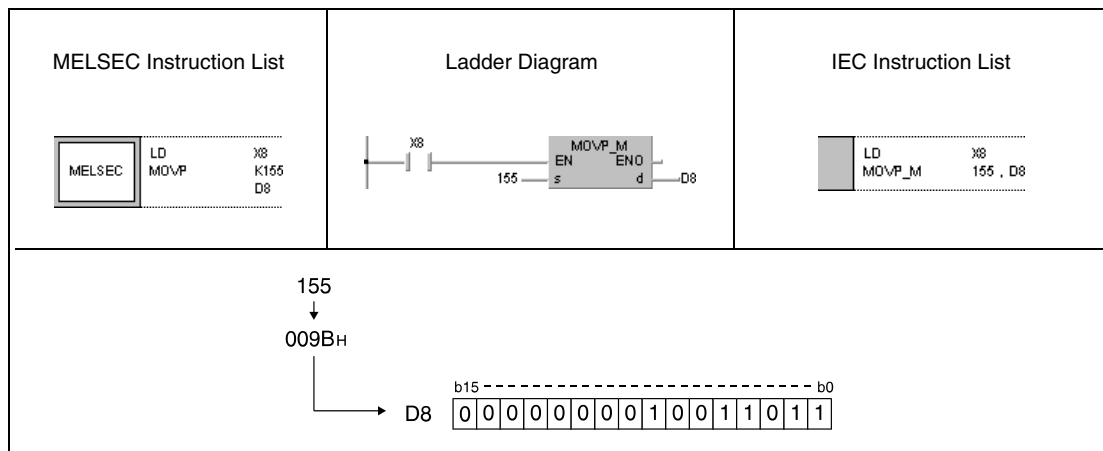
With leading edge from SM400, this program transfers data at X0 through XB to D8.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List
<pre> MELSEC ----- LD      SM400 MOVP   K3#0       D8         </pre>		<pre> IEC ----- LD      SM400 MOV_P_M K3#0, D8         </pre>

**Program Example 2**

MOVP

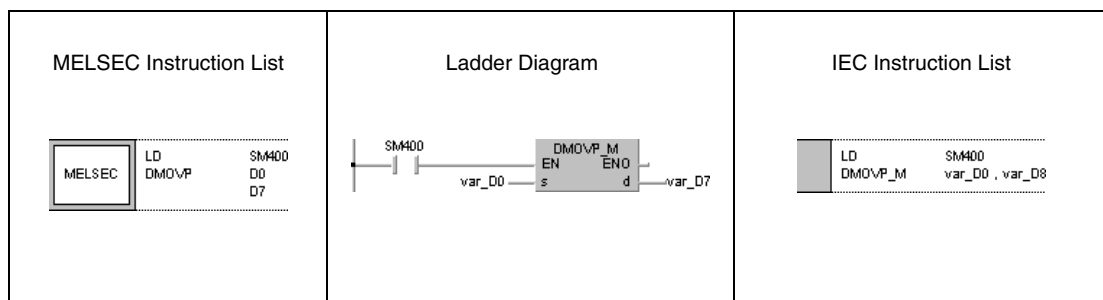
With leading edge from X8, the following program transfers the constant 155 as BIN value to D8.



**Program Example 3**

DMOVP

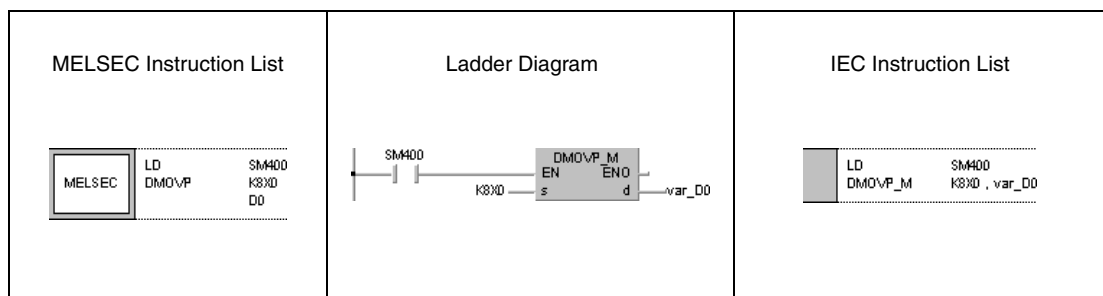
With leading edge from SM400, the following program transfers data in D0 and D1 to D7 and D8.



**Program Example 4**

DMOVP

With leading edge from SM400, the following program transfers data at X0 through X1F to D0 and D1.



**NOTE**

The program examples 3 and 4 will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

**6.4.2 EMOV, EMOVP**

**CPU**

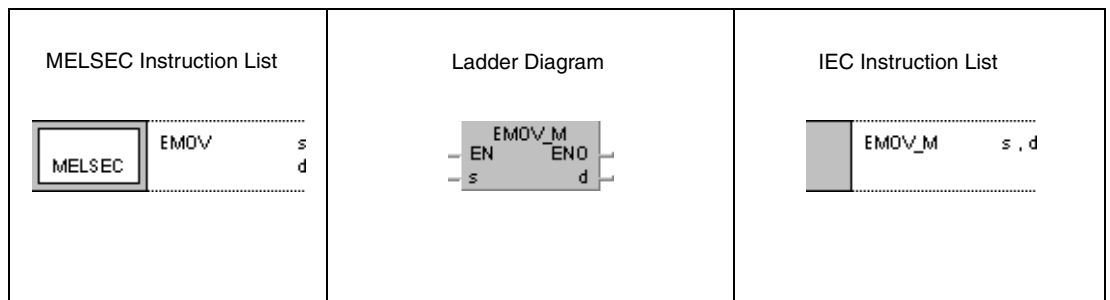
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

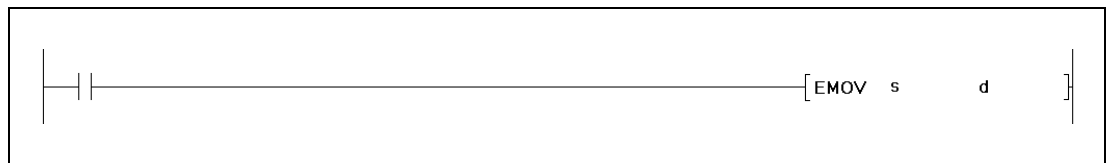
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	3	
d	—	●	●	—	●	●	—	—	—		

**GX IEC  
Developer**



**GX  
Developer**



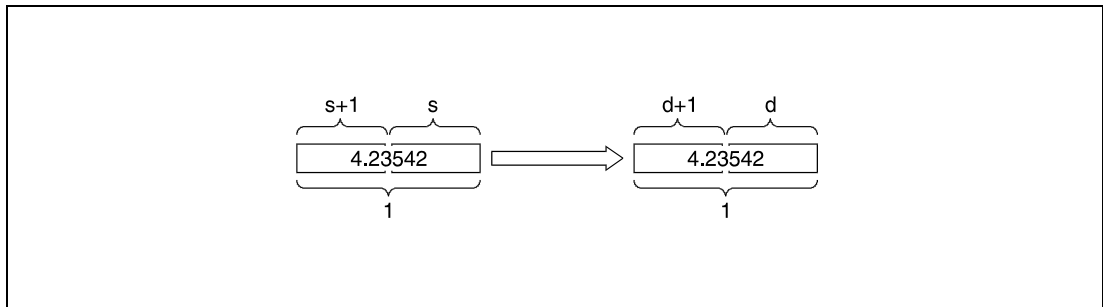
**Variables**

Set Data	Meaning	Data Type
s	Floating point data, or first number of device storing data to be transferred.	Real number
d	First number of device storing transferred floating point data.	

**Functions** Floating point data transfer

**EMOV Floating point data transfer**

The EMOV instruction transfers floating point data in s to the device designated by d.



<sup>1</sup> Floating point number, data type real number

**Program Example 1**

EMOVP

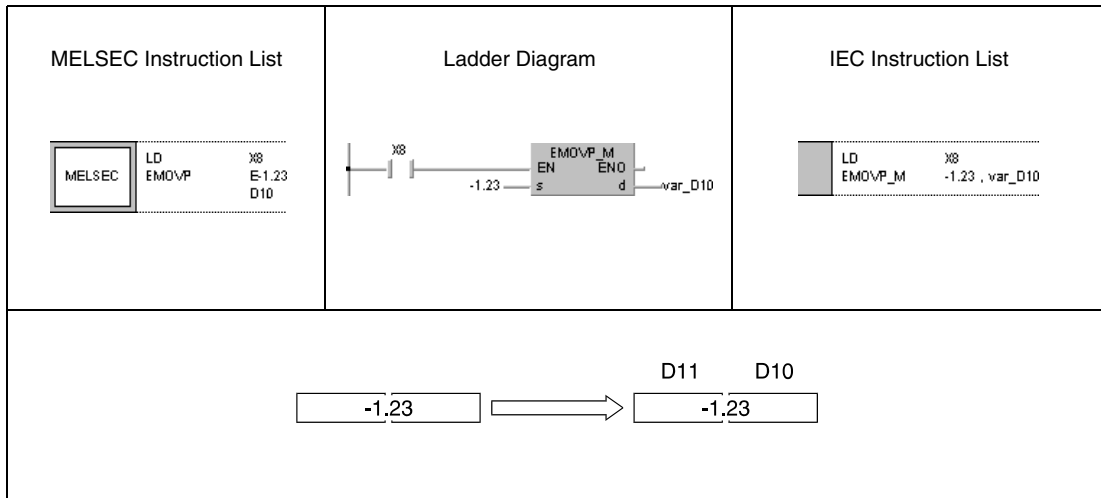
With leading edge from SM400, the following program transfers floating point data in D10 and D11 to D0 and D1.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List										
<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;"> <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">SM400</td> </tr> <tr> <td style="border: none;"></td> <td style="padding: 2px;">EMOVP</td> <td style="padding: 2px;">D10 D0</td> </tr> </table> </div>	MELSEC	LD	SM400		EMOVP	D10 D0		<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;"> <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 2px;">LD</td> <td style="padding: 2px;">SM400</td> </tr> <tr> <td style="padding: 2px;">EMOVP_M</td> <td style="padding: 2px;">var_D10 , var_D0</td> </tr> </table> </div>	LD	SM400	EMOVP_M	var_D10 , var_D0
MELSEC	LD	SM400										
	EMOVP	D10 D0										
LD	SM400											
EMOVP_M	var_D10 , var_D0											

**Program** EMOVP

**Example 2**

With leading edge from X8, the following program transfers the real number 1,23 to D10 and D11.



**NOTE**

*These programs will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 6.4.3 \$MOV, \$MOVP

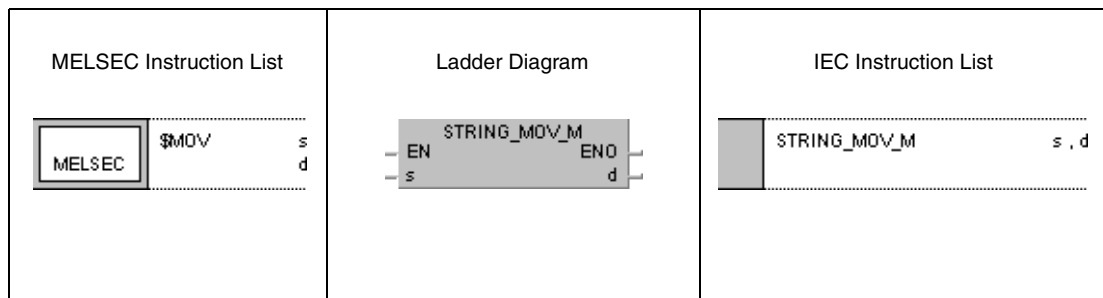
### CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

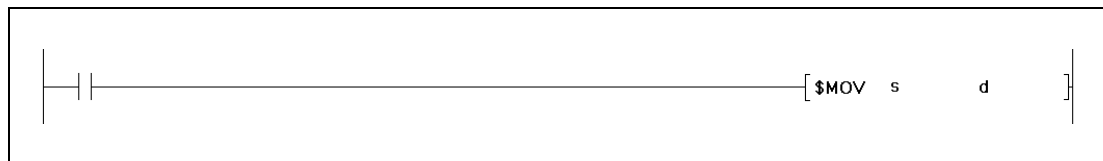
### Devices MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	3
d	—	●	●	—	—	—	—	—	—		

### GX IEC Developer



### GX Developer



### Variables

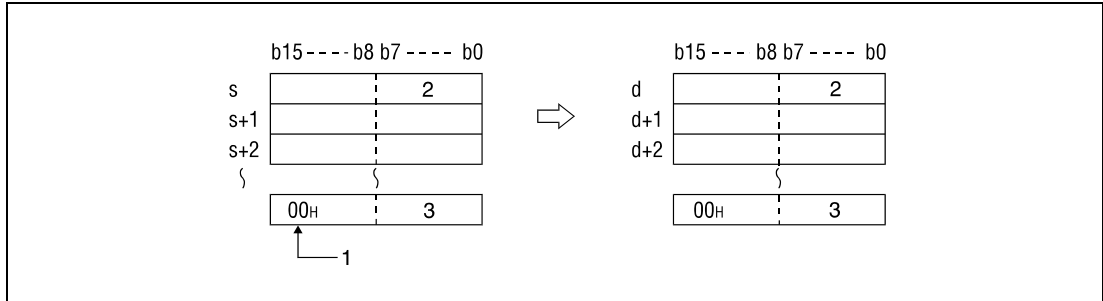
Set Data	Meaning	Data Type
s	Character string data, or first number of device storing data to be transferred.	Character string
d	First number of device storing transferred character string data.	



**Functions Character string data transfer**

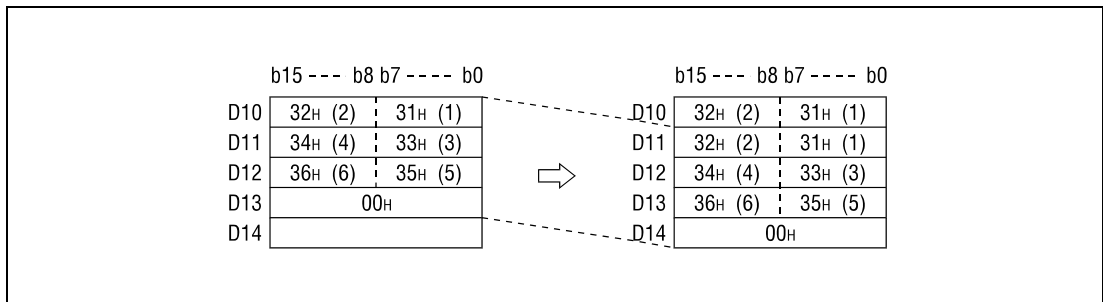
**\$MOV Character string data transfer**

The \$MOV instruction transfers character string data in s to d. The instruction transfers character string data from the first number of device designated by s up to the number of device storing the code "00H" (end of string) in one operation.

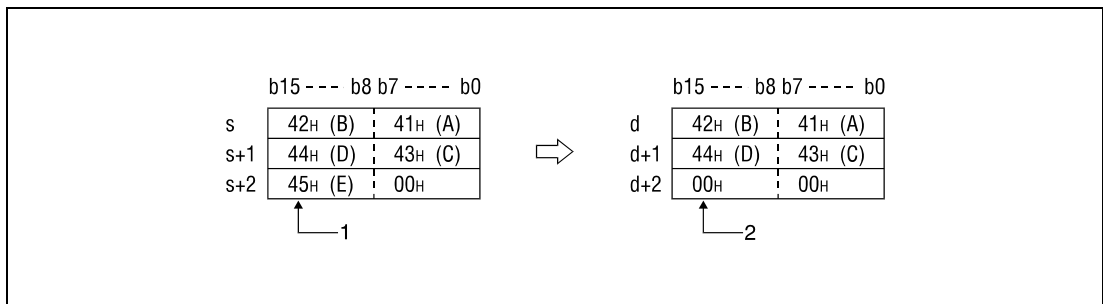


- <sup>1</sup> Indicates end of character string
- <sup>2</sup> 1st character
- <sup>3</sup> nth character

The \$MOV instruction is even performed without error messages, if the range of devices storing character string data to be transferred (s through s+n) overlaps with the range of devices storing transferred data (d through d+n). The \$MOV instruction performs as follows, if character string data in D10 through D13 is transferred to D11 through D14:



If the code "00H" is stored at lower bytes of s+n, the characters following at the higher bytes are omitted. In d+n, the transferred code "00H" will be stored at both, the higher bytes and the lower bytes:



- <sup>1</sup> Character is not transferred.
- <sup>2</sup> "00H" is stored automatically.

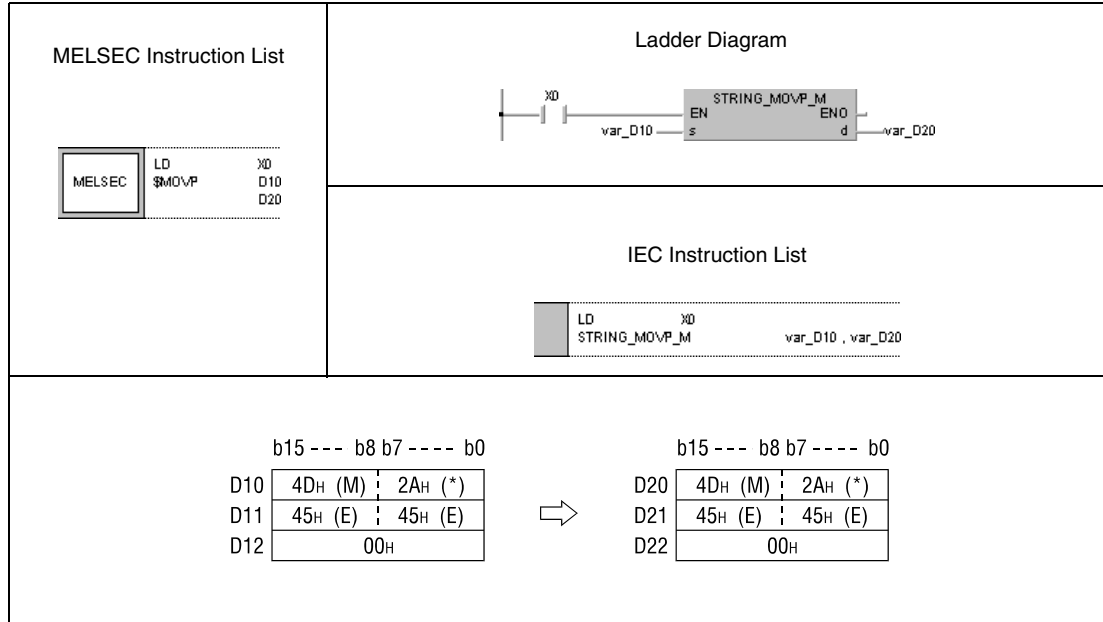
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The code "00H" does not exist in character string data designated by s (error code: 4101).
- Character string data in s cannot be transferred completely to d.

**Program Example**

With leading edge from X0, the following program transfers character string data at D10 through D12 to D20 through D22.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.4.4 CML, CMLP, DCML, DCMLP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011	
Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level						
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K						H (16#)
CML																	K1 ↓ K4	5 ↓ 1	●		●	
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●							
DCML																	K1 ↓ K8	7 ↓ 1	●		●	
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●							

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

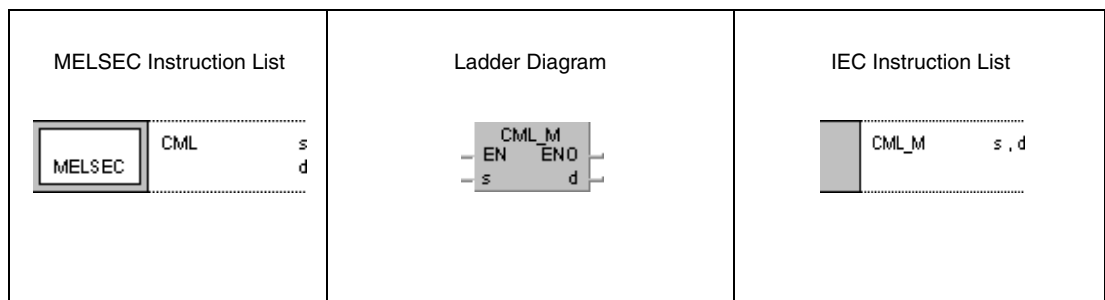
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
s	●	●	●	●	●	●	●	—	—	—	3 <sup>1)</sup>
d	●	●	●	●	●	●	—	—	—	—	

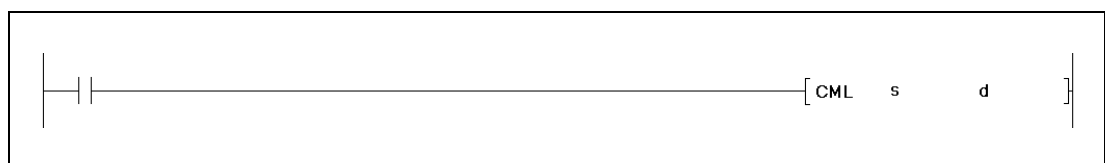
<sup>1</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU or a System Q single processor CPU is used: 3
- If a System Q multi processor CPU is used with internal word devices (except for file register ZR) or constants: 2
- If a System Q multi processor CPU is used with Bit Devices, whose device numbers are multiplies of 16, whose digit designation is K4, and which use no index qualification: 2
- If a System Q multi processor CPU is used with devices other than above mentioned: 3

GX IEC Developer



GX Developer



**Variables**

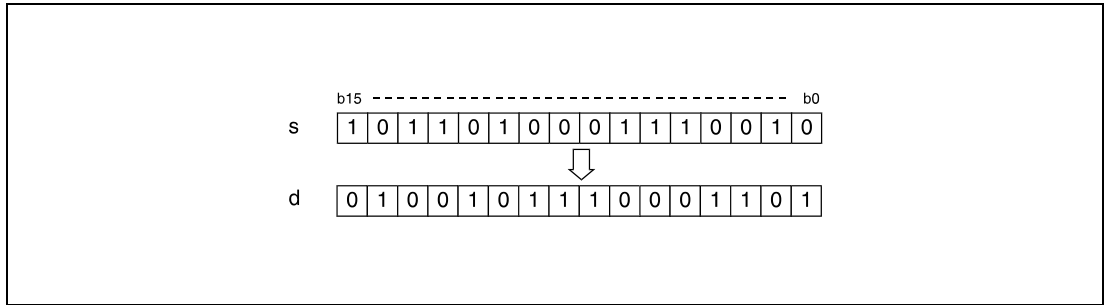
Set Data	Meaning	Data Type
s	BIN data, or first number of device storing data to be inverted.	BIN 16-/32-bit
d	First number of device storing inverted data.	

**Functions**

**BIN data inversion**

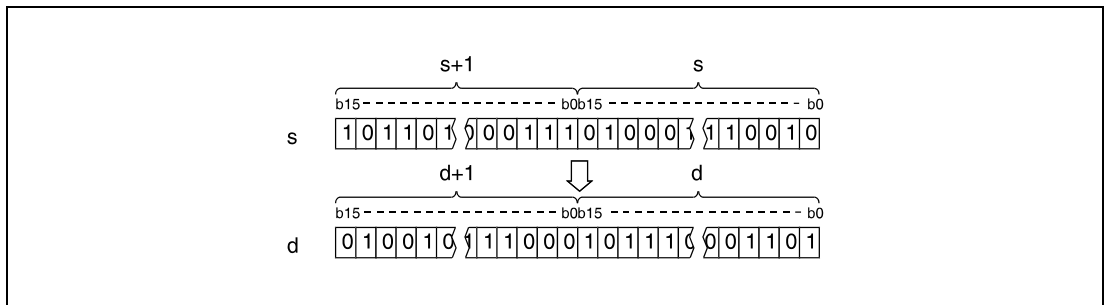
**CML BIN 16-bit data inversion**

BIN 16-bit data in s is inverted bit by bit. The result is stored in d.



**DCML BIN 32-bit data inversion**

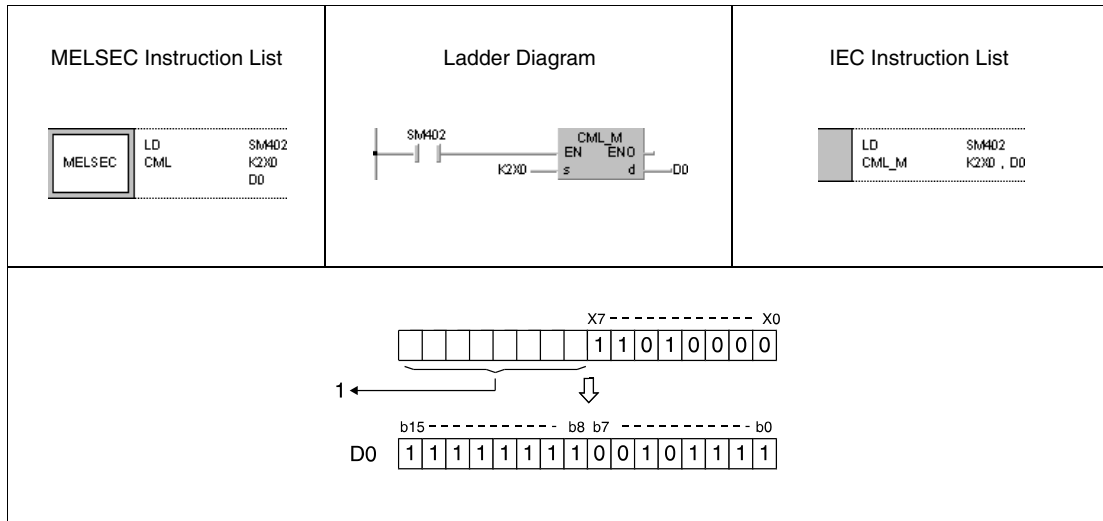
BIN 32-bit data in s is inverted bit by bit. The result is stored in d.



**Program Example 1**

CML

While SM402 is set, the following program transfers data at X0 through X7 inverted to D0.



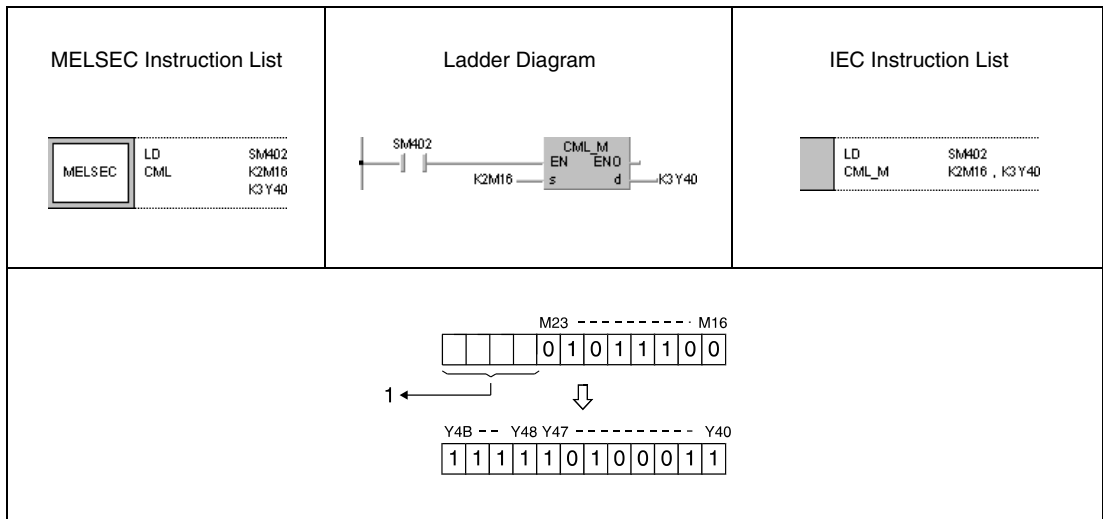
<sup>1</sup> Undesignated bits are read as 0.

The number of bits in s must be smaller than the number of bits in d.

**Program Example 2**

CML

While SM402 is set, the following program transfers data in M16 through M23 inverted to K3 Y40 (Y40 through Y4F). Y48 through Y4B are all set (1), because they were read as 0.

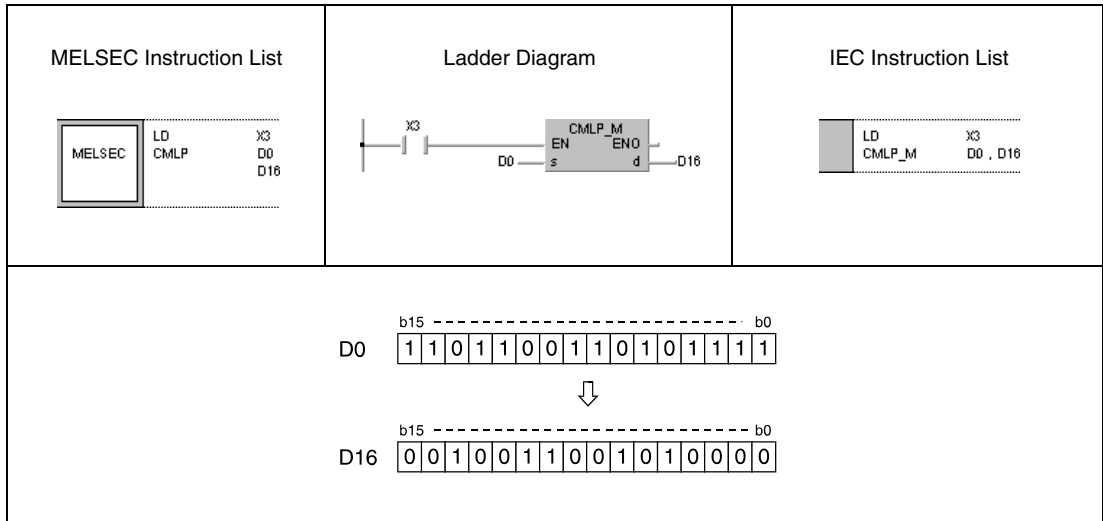


<sup>1</sup> Undesignated bits are read as 0.

The number of bits in s must be smaller than the number of bits in d.

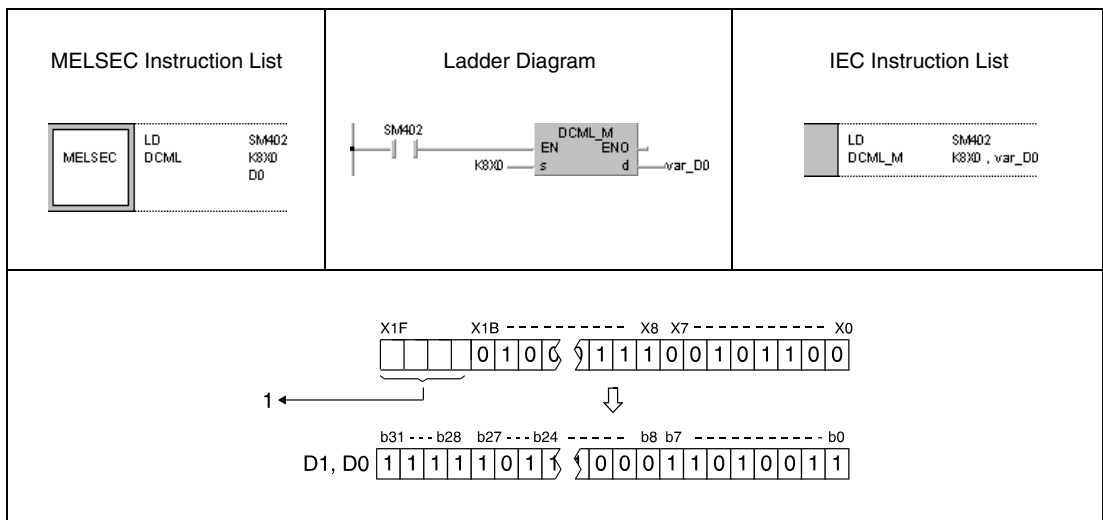
**Program Example 3**

CMLP  
 With leading edge from X3, the following program transfers data in D0 inverted to D16.



**Program Example 4**

DCML  
 While SM402 is set, the following program transfers data at X0 through X1F inverted to D0 and D1.



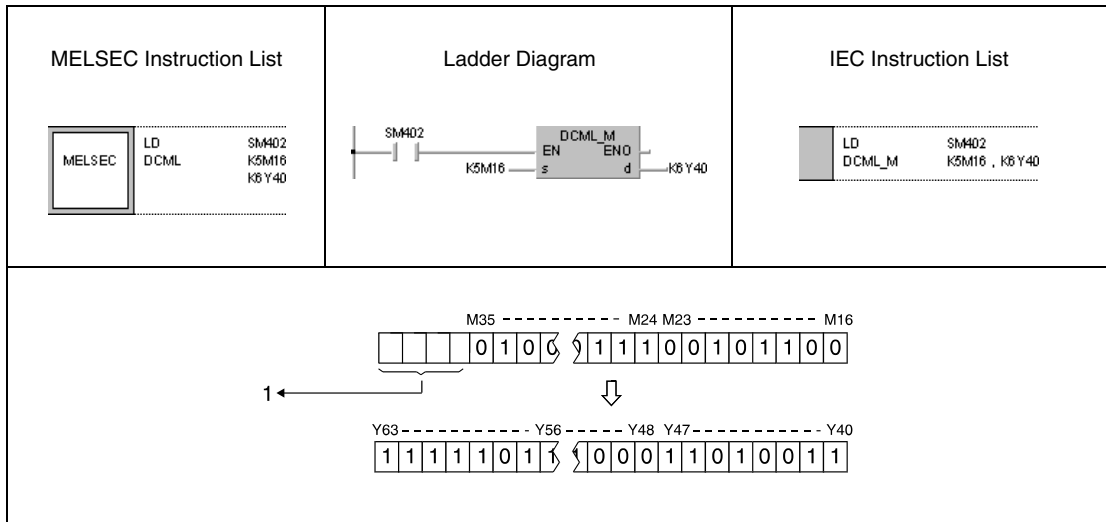
<sup>1</sup> Undesignated bits are read as 0.

The number of bits in s must be smaller than the number of bits in d.

**Program Example 5**

**DCML**

While SM402 is set, the following program transfers data in M16 through M35 inverted to Y40 and Y57.

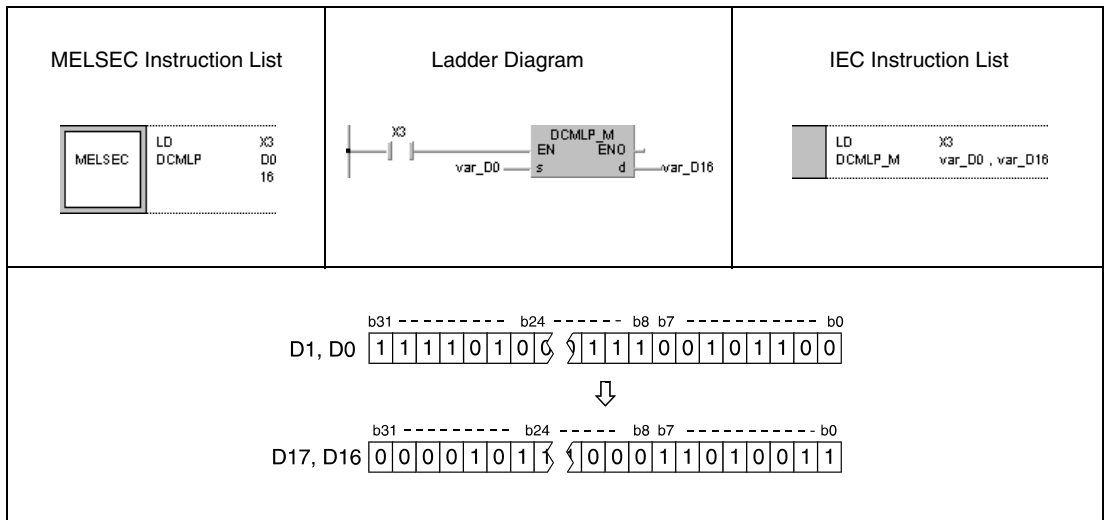


<sup>1</sup> Undesignated bits are read as 0.

**Program Example 6**

**DCMLP**

With leading edge from X3, the following program transfers data in D0 and D1 inverted to D16 and D17.



The number of bits in s must be smaller than the number of bits in d.

**NOTE**

The program examples 4 and 6 will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

6.4.5 BMOV, BMOVP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

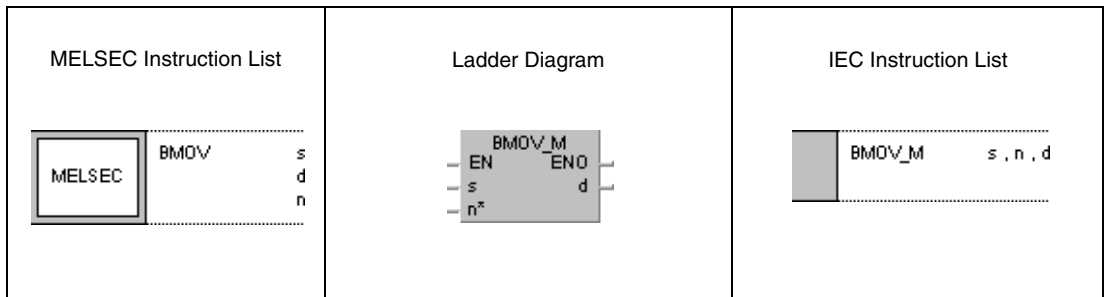
	Usable Devices																			Digit designation K1 L K4	Number of steps g 1	Index ●	Carry Flag M9012	Error Flag M9010 M9011
	Bit Devices						Word Devices (16-bit)						Constant		Pointer		Level							
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P					
s	●	●	●	●	●	●	●	●	●	●	●													
d		●	●	●	●	●	●	●	●	●	●													
n																●	●							

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this Programming Manual for the according number of steps.

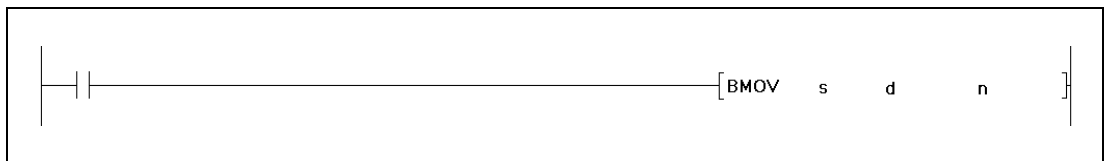
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	—	—	—	SM0	4
d	●	●	●	●	●	●	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer



GX Developer



Variables

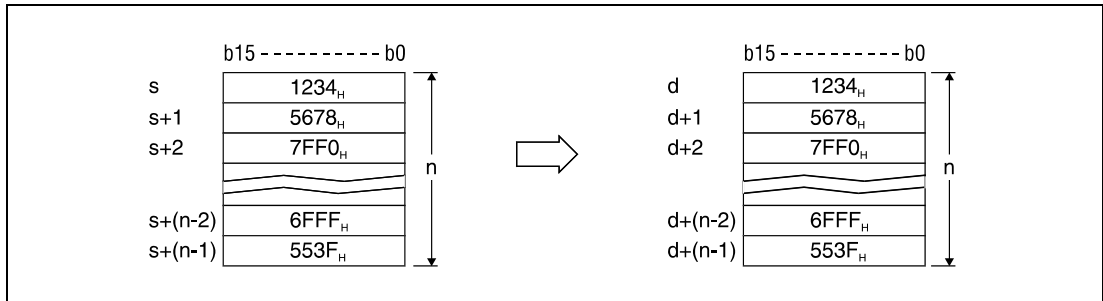
Set Data	Meaning	Data Type
s	First number of device storing data to be transferred.	BIN 16-bit
d	First number of device storing transferred data.	
n	Number of data blocks to be transferred.	



**Functions BIN block data transfer**

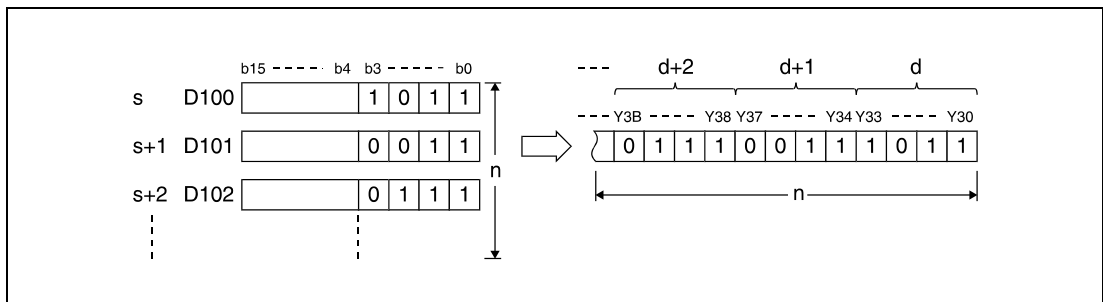
**BMOV BIN 16-bit block data transfer**

The BMOV instruction transfers successive data blocks in a batch. The first number of device storing block data is designated by s. The number of successive data blocks to be transferred is determined by n. The data are transferred to the device designated by d onwards.



A transfer can even be performed without operation errors, if the source and the destination devices overlap. Transfer to the smaller device number begins from s. Transfer to the larger device number begins from s+(n-1).

If s is a word device and d is a bit device, the number of bits designated by digit designation for the bit device will be the object bits for the word device. If K1Y30 is designated by d, the object bits for the word device s are the lower 4 bits.



If s and d are bit devices, the number of bits in s and d must equal.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

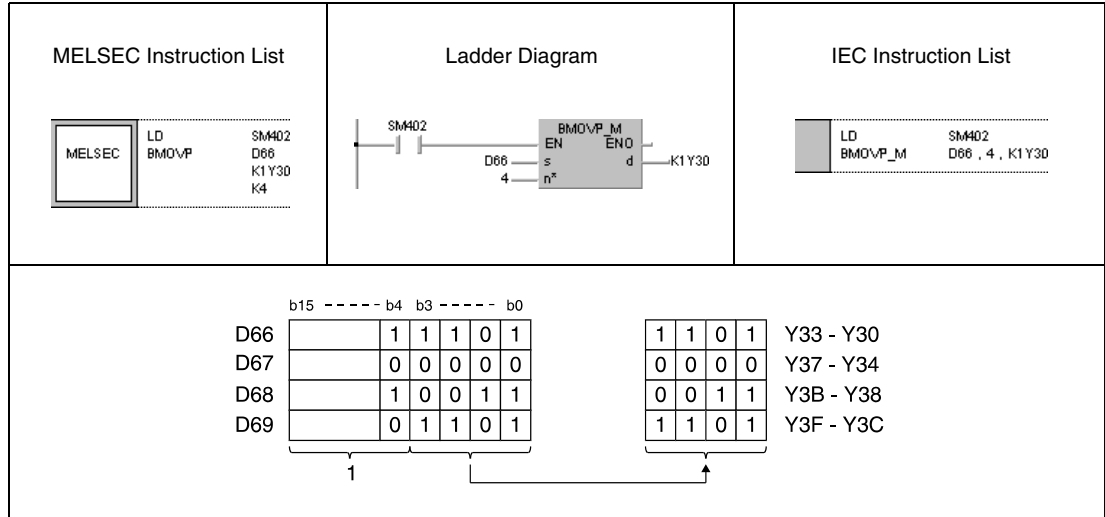
- The number of data blocks determined by n exceeds the storage device numbers designated by s and d (Q series and System Q = error code 4101).

**Program Example 1**

**BMOV P**

With leading edge from SM402, the following program transfers the lower 4 bits of data (b0 through b3) in D66 through D69 to the outputs Y30 through Y3F. The number of blocks (4) to be transferred is determined by the constant K4.

The bit patterns show the structure of bits before and after the transfer.



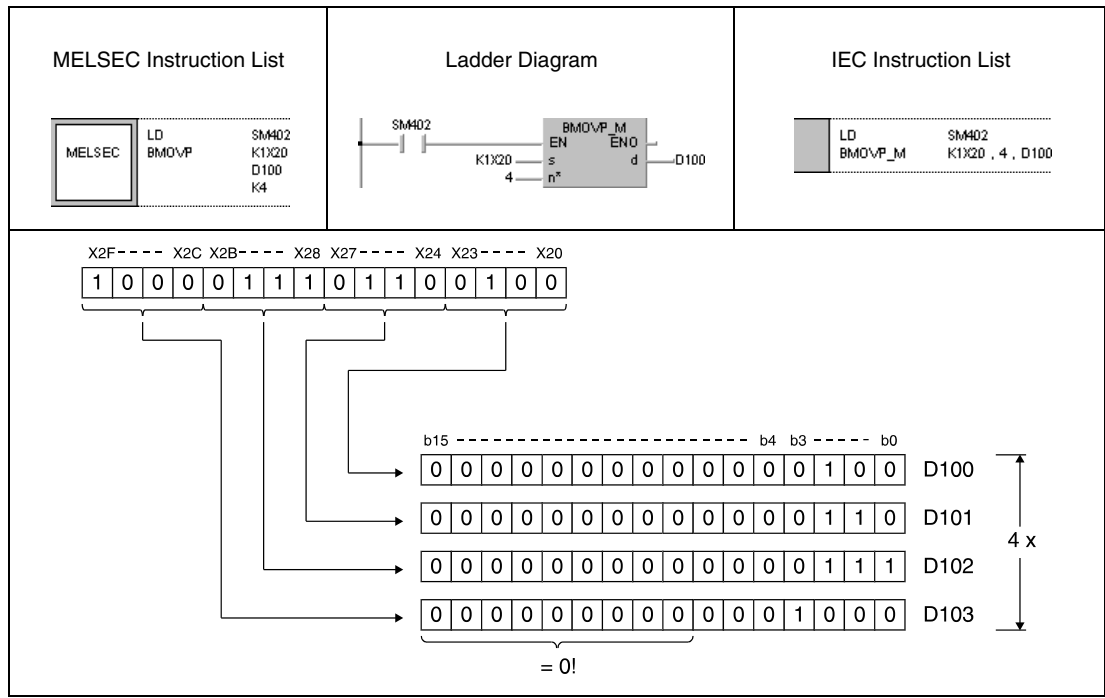
<sup>1</sup> These bits are ignored.

**Program Example 2**

**BMOV P**

With leading edge from SM402, the following program transfers data at X20 through X2F to D100 through 103. The number of blocks (4) to be transferred is determined by the constant K4.

The bit patterns show the structure of bits before and after the transfer.



### 6.4.6 FMOV, FMOVP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

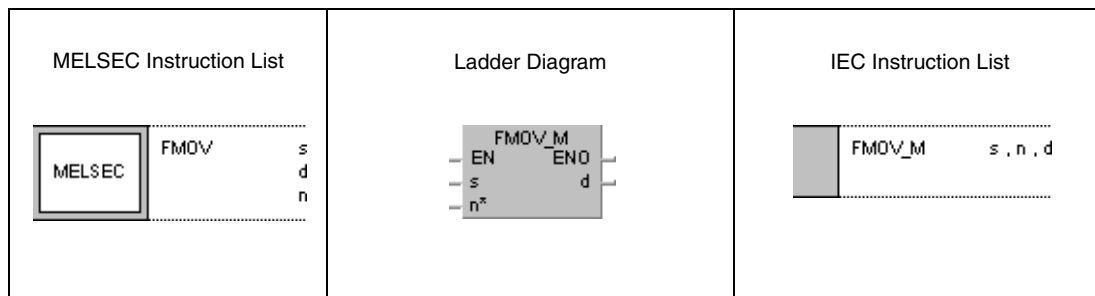
	Usable Devices																		Digit designation K1 ↓ K4	Number of steps 9 ↓ 1	Index	Carry Flag M9012	Error Flag M9010 M9011
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)					
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						
d		●	●	●	●	●	●	●	●	●	●												
n																●	●						

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this Programming Manual for the according number of steps.

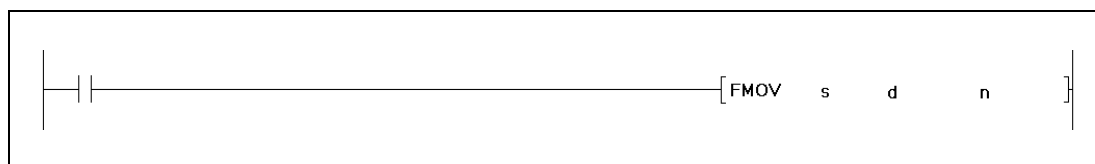
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	4
d	●	●	●	●	●	●	—	—	—		
n	●	●	●	●	●	●	●	●	—		

**GX IEC  
Developer**



**GX  
Developer**



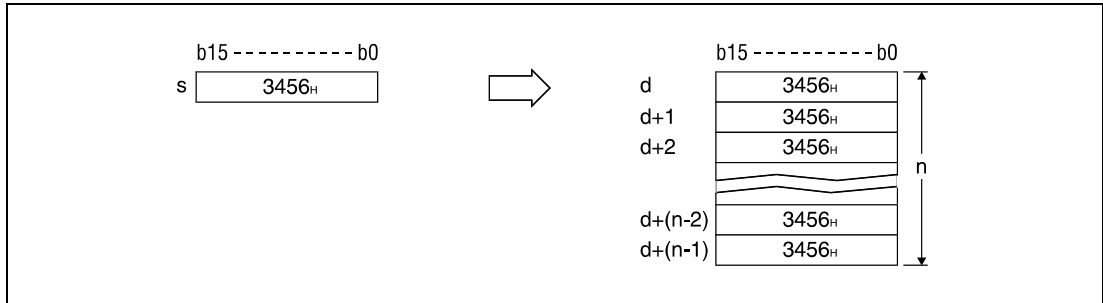
**Variables**

Set Data	Meaning	Data Type
s	First number of device storing data to be transferred.	BIN 16-bit
d	First number of device storing transferred data.	
n	Number of data blocks to be transferred.	

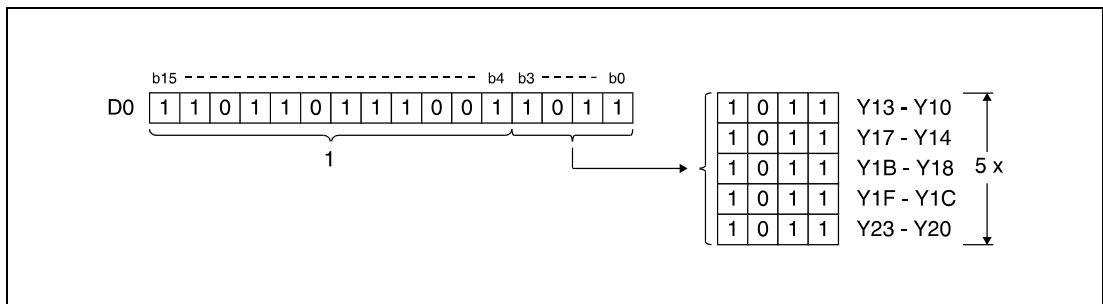
**Functions Identical BIN block data transfer**

**FMOV Identical BIN 16-bit block data transfer**

The FMOV instruction transfers data in s to d through d+(n-1). Each device of the data block from d through d+(n-1) stores the value from s.



If s is a word device and d is a bit device, the number of bits designated by digit designation for the bit device will be the object bits for the word device.



<sup>1</sup> These bits are ignored.

If s and d are bit devices, the number of bits in s and d must equal.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

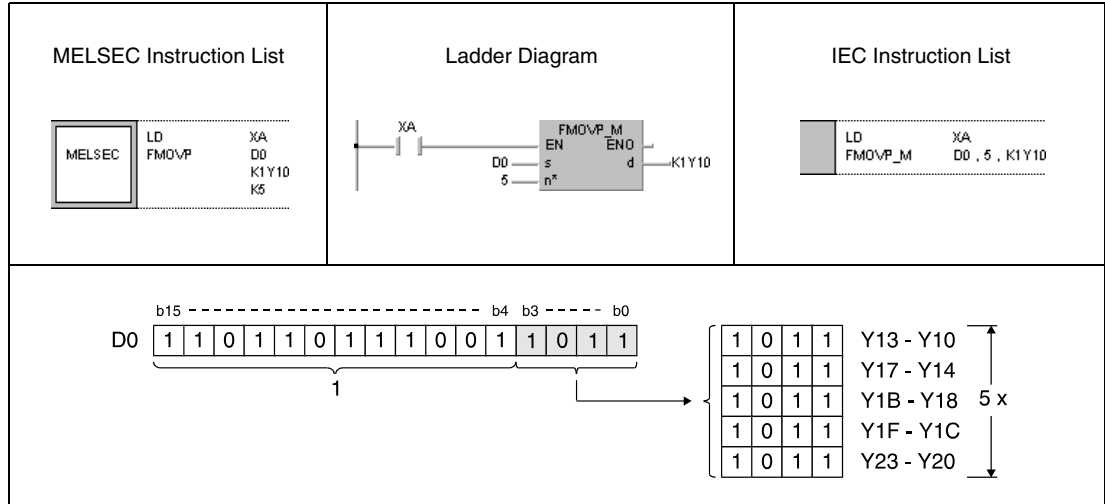
- The number of data blocks determined by n exceeds the storage device numbers designated by s and d (Q series and System Q = error code 4101).

**Program Example 1**

FMOVP

With leading edge from XA, the following program transfers the lower 4 bits of data (b0 through b3) in D0 to the outputs Y10 through Y23. The number of blocks (5) is determined by the constant K5.

The bit patterns show the structure of bits before and after the transfer.



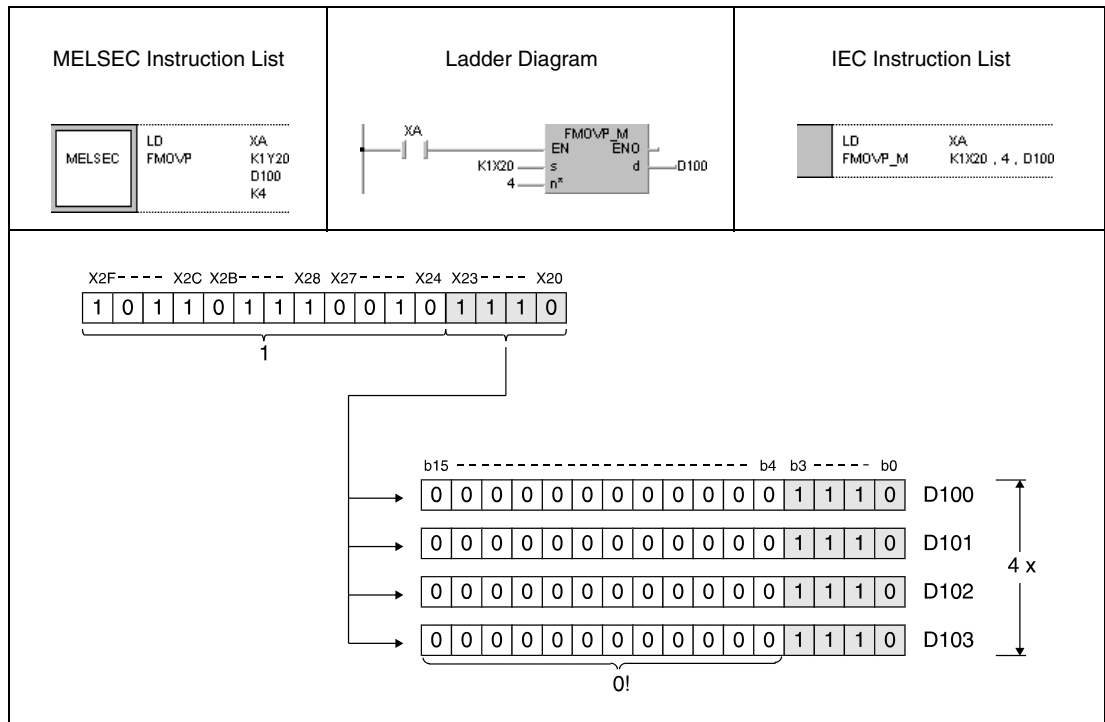
<sup>1</sup> These bits are ignored.

**Program Example 2**

FMOVP

With leading edge from XA, the following program transfers data at X20 through X23 to D100 through D103. The number of blocks (4) to be transferred is determined by the constant K4.

The bit patterns show the structure of bits before and after the transfer.



<sup>1</sup> These bits are ignored.

6.4.7 XCH, XCHP, DXCH, DXCHP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

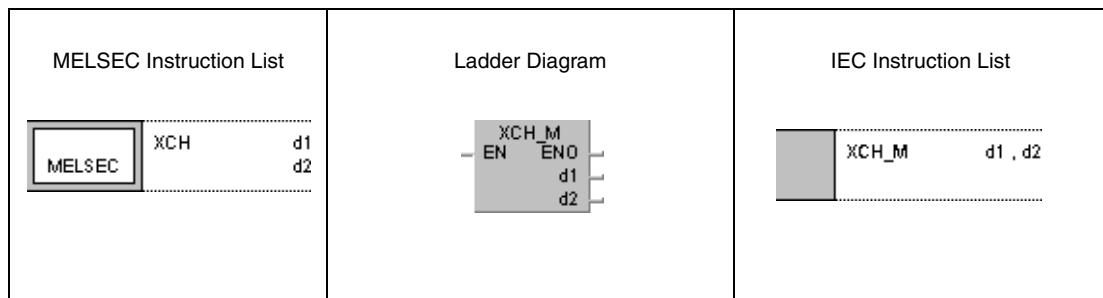
Usable Devices																	Digit designation ↓ Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011
Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level				
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				
XCH																	K1 ↓ K4	5 ↓ 1	●	●
d1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					
d2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					
DXCH																	K1 ↓ K8	7 ↓ 1	●	●
d1	●	●	●	●	●	●	●	●	●	●	●	●	●	●						
d2	●	●	●	●	●	●	●	●	●	●	●	●	●	●						

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

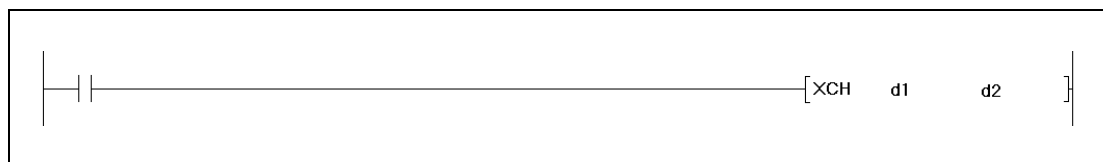
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
d1	●	●	●	●	●	●	—	—	—	3	
d2	●	●	●	●	●	●	—	—	—		

GX IEC Developer



GX Developer



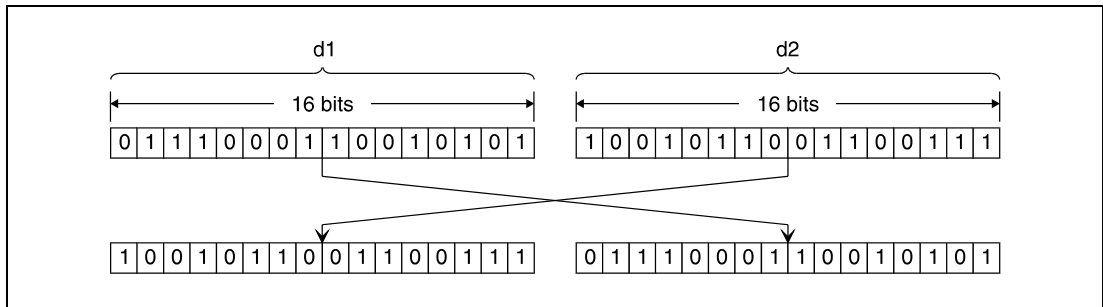
Variables

Set Data	Meaning	Data Type
d1	First number of device storing data to be exchanged.	BIN 16-/32-bit
d2		

**Functions BIN data exchange**

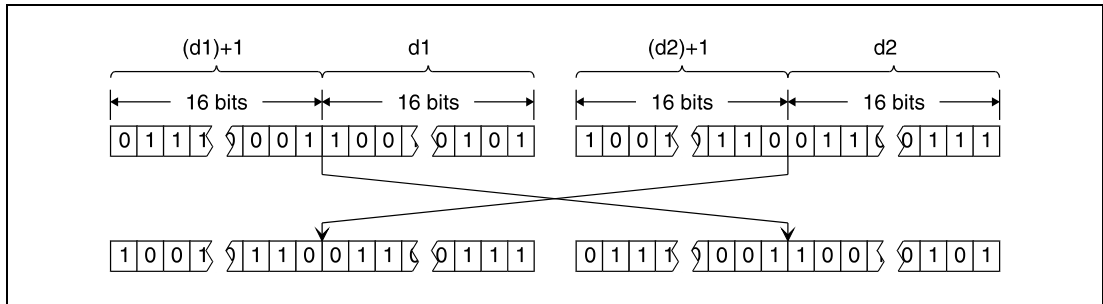
**XCH BIN 16-bit data exchange**

The XCH instruction exchanges BIN 16-bit data in d1 and BIN 16-bit data in d2.



**DXCH BIN 32-bit data exchange**

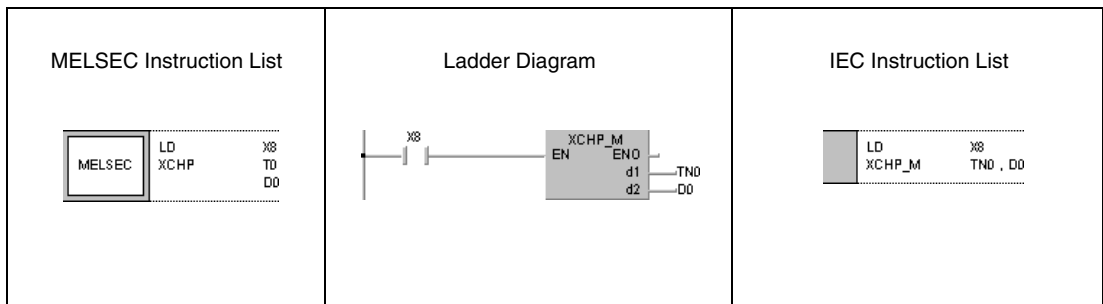
The DXCH instruction exchanges BIN 32-bit data in (d1)+1, d1 and BIN 32-bit data in (d2)+1, d2.



**Program Example 1**

**XCHP**

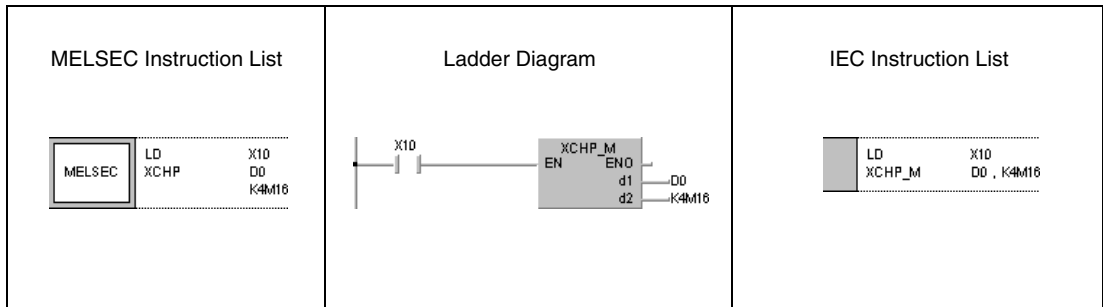
With leading edge from X8, the following program exchanges data in D0 and the actual value in T0.



**Program Example 2**

XCHP

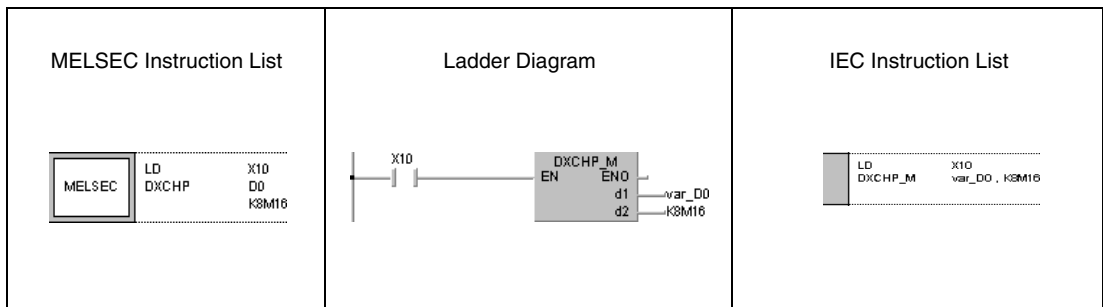
With leading edge from X10, the following program exchanges data in D0 and data in M16 through M31.



**Program Example 3**

DXCHP

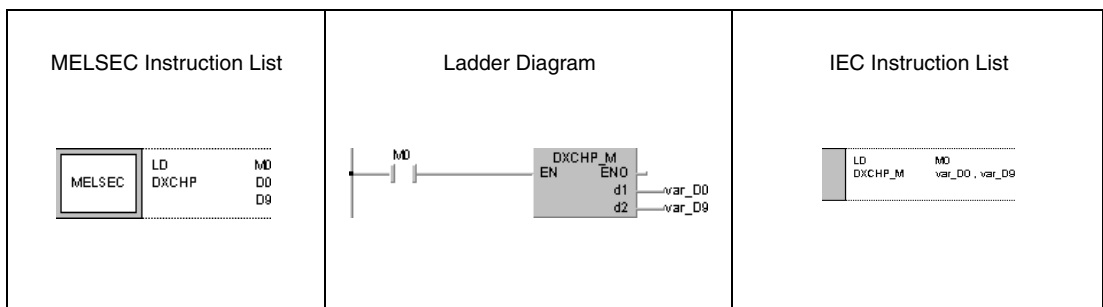
With leading edge from X10, the following program exchanges data in D0 and D1 and data in M16 through M47.



**Program Example 4**

DXCHP

With leading edge from M0, the following program exchanges data in D0 and D1 and data in D9 and D10.



**NOTE**

The program examples 3 and 4 will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.



**6.4.8 BXCH, BXCHP**

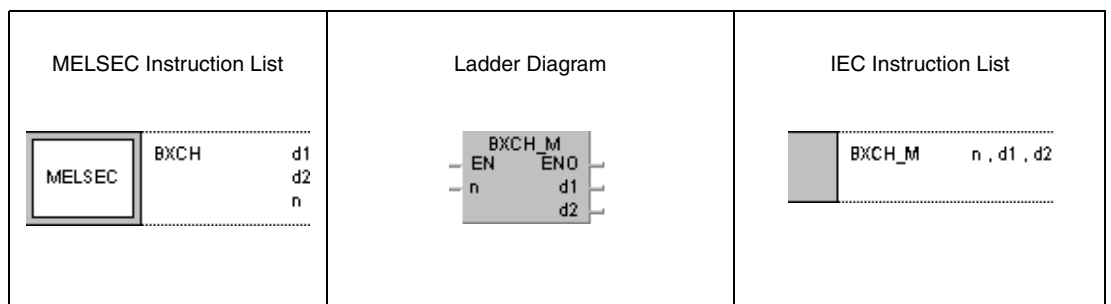
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

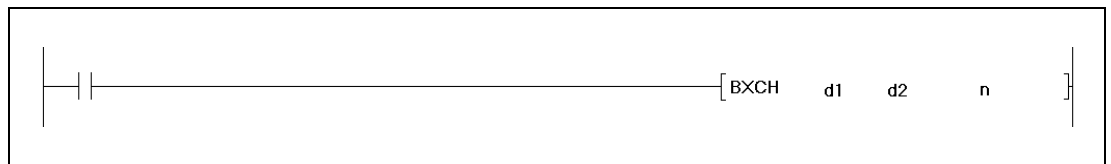
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
d1	—	●	●	—	—	—	—	—	—	SM0	4
d2	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

**GX IEC Developer**



**GX Developer**



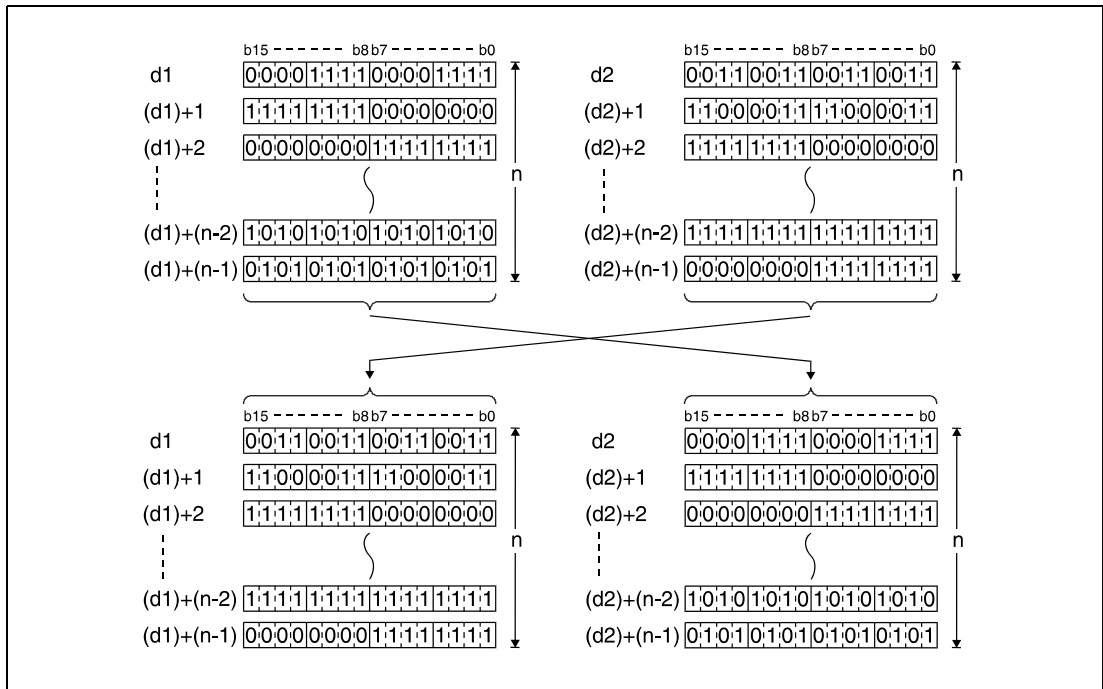
**Variables**

Set Data	Meaning	Data Type
d1	First number of device storing data to be exchanged	BIN 16-bit
d2		
n	Number of exchanges	

**Functions**     **BIN block data exchange**

**BXCH**    **BIN 16-bit block data exchange**

The BXCH instruction exchanges BIN 16-bit block data in d1 and BIN 16-bit block data in d2.



**Operation Errors**

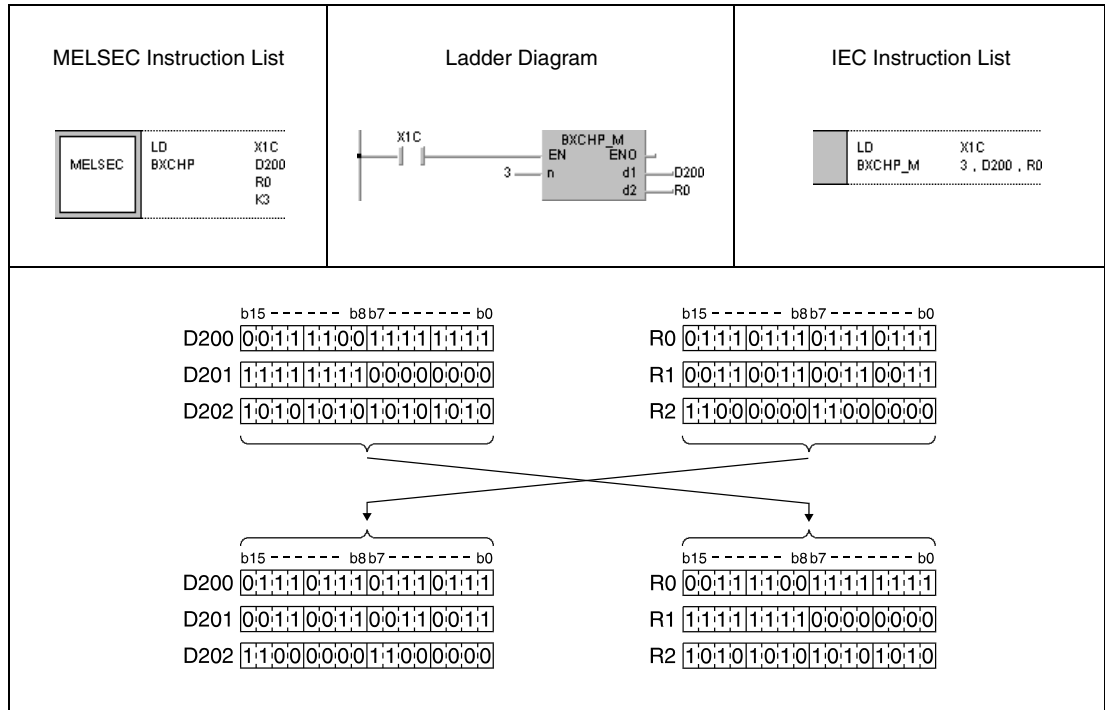
In the following cases an operation error occurs and the error flag is set:

- The number of data blocks determined by n exceeds the storage device numbers designated by d1 and d2 (error code: 4101).
- The storage device numbers designated by d1 and d2 overlap (error code: 4101)

**Program Example** BXCHP

With leading edge from X1C, the following program exchanges data blocks beginning from D200 and data blocks beginning from R0. The number of blocks (3) to be exchanged is determined by the constant K3.

The bit patterns show the structure of bits before and after the transfer.



6.4.9 SWAP, SWAPP

CPU

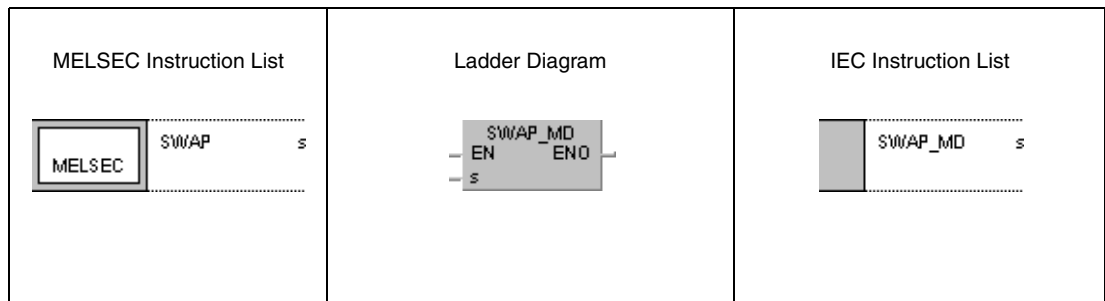
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

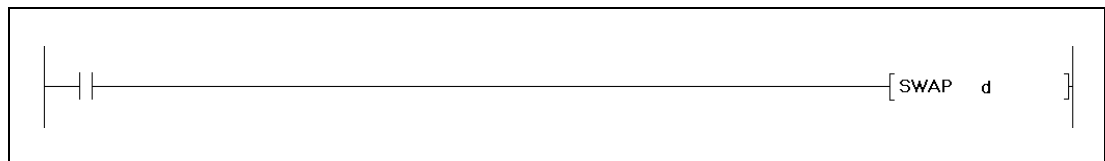
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□□		Special Function Module U□G□□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	—	3

GX IEC Developer



GX Developer



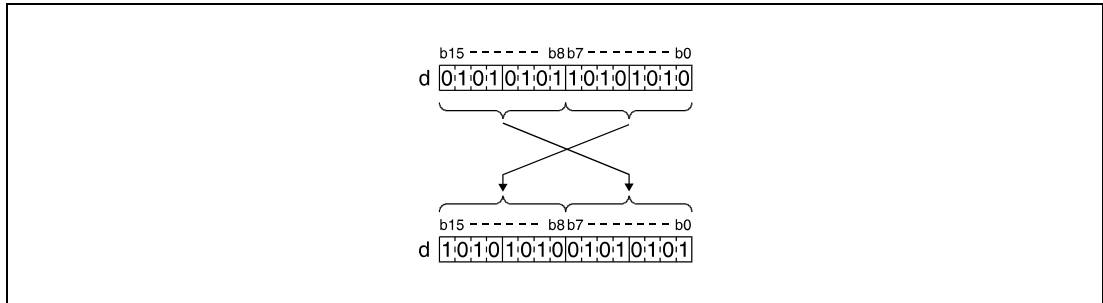
Variables

Set Data	Meaning	Data Type
s	First number of device storing data to be swapped.	BIN 16-bit

**Functions Upper and lower byte exchanges**

**SWAP Upper and lower byte exchanges**

The swap instruction exchanges the upper and lower 8 bits (upper and lower byte) of BIN 16-bit data in s.



**Program Example**

**SWAPP**

With leading edge from X10, the following program exchanges the upper and lower 8 bits in R10.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List



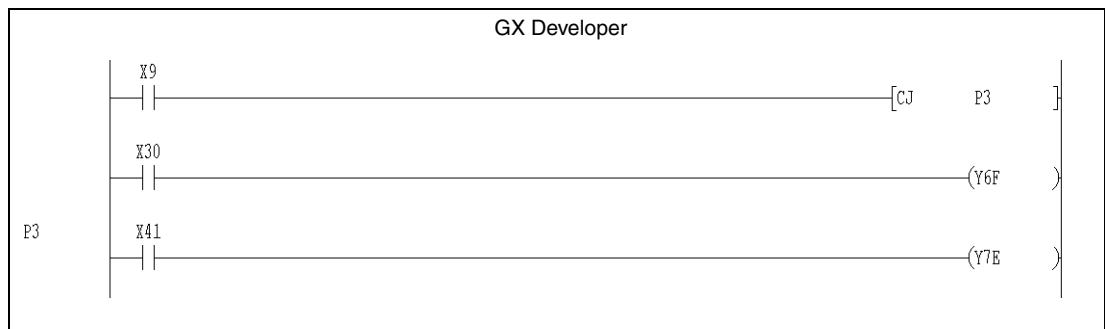
## 6.5 Program Branch Instructions

Program branch instructions include a jump destination.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Conditional Jump	CJ	CJ_M
Conditional Jump from next Scan	SCJ	SCJ_M
Jump	JMP	JMP_M
Jump to End of Program	GOEND	GOEND_M

A jump destination is designated by a pointer P (GX Developer) or a label (GX IEC Developer). For details on programming a label in GX IEC Developer see the Programming Manual for the GX IEC Developer.

GX Developer	GX IEC Developer
LD X9	LD X9
CJ P3	JMPC Label_3
LD X30	LD X30
OUT Y6F	ST Y6F
P3	
LD X41	Label_3: LD X41
OUT Y7E	ST Y7E



6.5.1 CJ, SCJ, JMP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

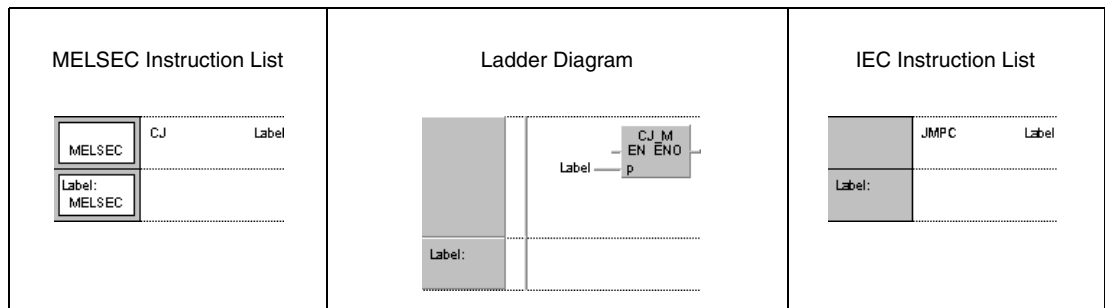
p	Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011							
	Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P	I	N		
																			●					3	1	●		●

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

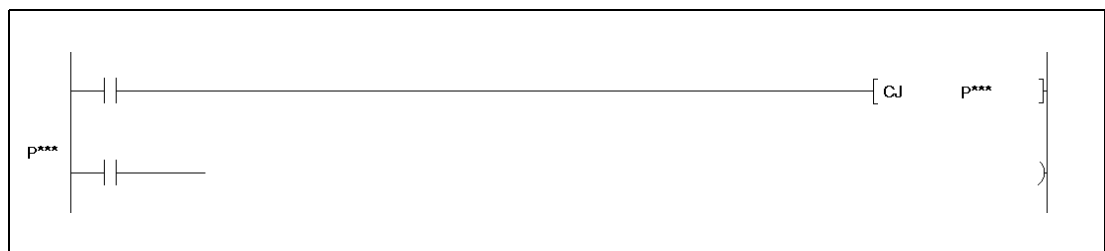
Devices  
MELSEC Q

p	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
	—	—	—	—	—	—	—	—	●	SM0	2

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
p	Jump destination	Pointer/Label



**Functions Jump instructions**

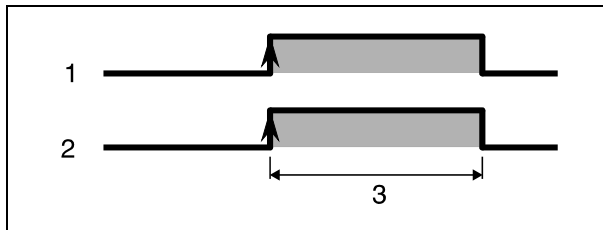
A jump instruction consists of a jump command CJ, SCJ, or JMP (**Conditional Jump, JuMP**) and a pointer (or label) P, designating the jump destination.

For the A series, the pointer (label) number has to range within P(Label)0 and P(Label)255. Here, P(Label)255 has the same meaning as an END instruction and cannot be used as jump destination.

For the Q series and the System Q, the pointer (label) number has to range within P(Label)0 and P(Label)4095. A jump destination P(Label)xx can be freely placed in a program.

**CJ Conditional jump**

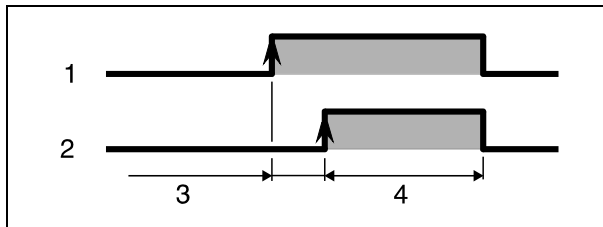
If the input condition is set, the CJ instruction executes the part of a program designated by the jump destination. If the input condition is not set, the next program step is executed.



- <sup>1</sup> Input condition
- <sup>2</sup> CJ instruction
- <sup>3</sup> Executed each scan

**SCJ Conditional jump from next program scan**

If the input condition is set, the SCJ instruction executes the part of a program designated by the jump destination from the next scan on. If the input condition is not set, the next program step is executed.



- <sup>1</sup> Input condition
- <sup>2</sup> SCJ instruction
- <sup>3</sup> One scan
- <sup>4</sup> Executed each scan

**JMP** Jump instruction

The jump instruction executes the part of a program designated by the jump destination without any input condition (unconditional jump).

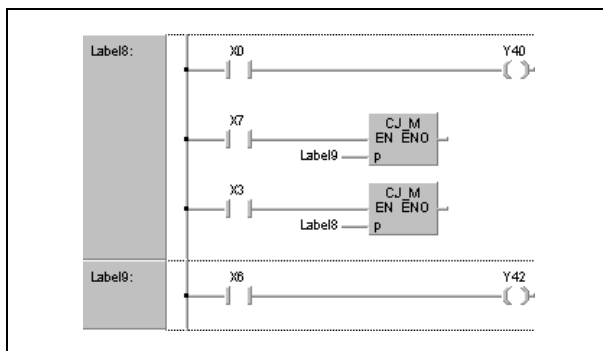
**NOTE**

*If a set timer is skipped by a CJ, SCJ, or JMC instruction it will nevertheless keep its timing accurately.*

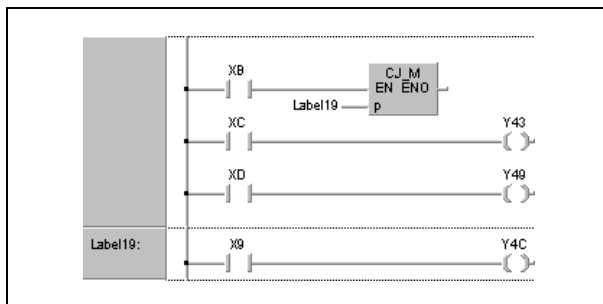
*If an OUT instruction is skipped by a jump instruction, the condition of the output remains unchanged.*

*Executing a jump instruction shortens the scan time of a program in relation to the skipped program steps (see tables in appendices).*

*The CJ, SCJ, and JMP instruction can even jump back to a lower jump destination. However, a program must exit the program loop before the watchdog timer times out (the following program example exits the loop, when X7 is set).*

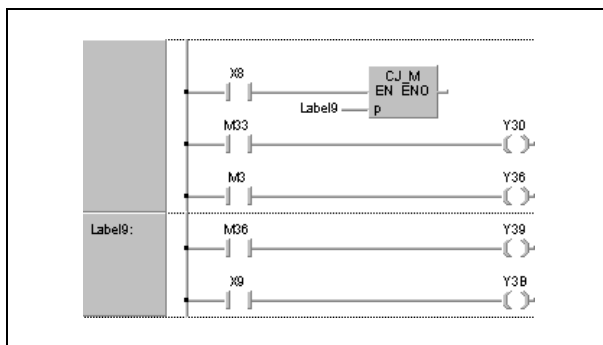


*The condition of a device skipped by a jump instruction remains unchanged. This is illustrated by the following program example:*



*After X8 is set, this program jumps to the jump destination Label19. The conditions of the outputs Y43 and Y49 even remain unchanged, if XC or XD are set or reset.*

*The jump destination Label9 occupies one program step.*



*The CJ, SCJ, or JMP instruction only jumps to destinations within one single program.*

*If a jump destination is located within the skip range during a skip operation (operation skipping parts of a program), program execution proceeds from the first available address following the jump destination.*

**Operation Errors**

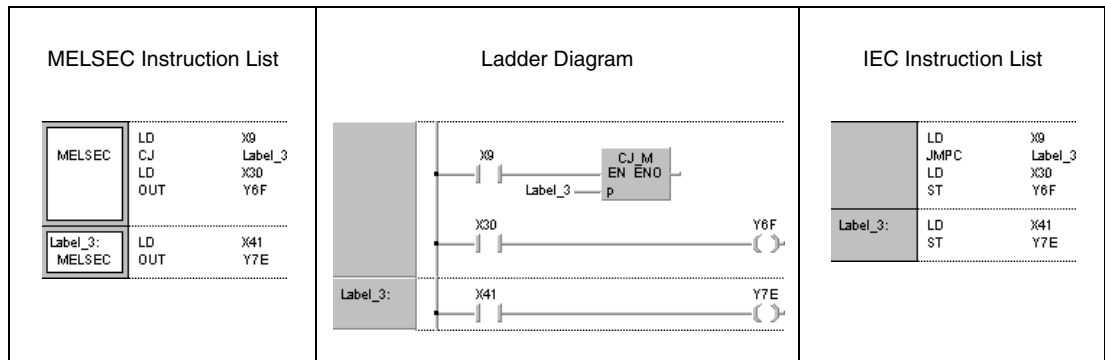
In the following cases an operation error occurs and the error flag is set:

- A common pointer has been designated. (Q series and System Q = error code 4210).
- The jump destination of the jump instruction is not defined in a program (jump destination or pointer is missing) (Q series and System Q = error code 4210).
- The jump destination is located after an END instruction. (Q series and System Q = error code 4210).

**Program Example 1**

CJ

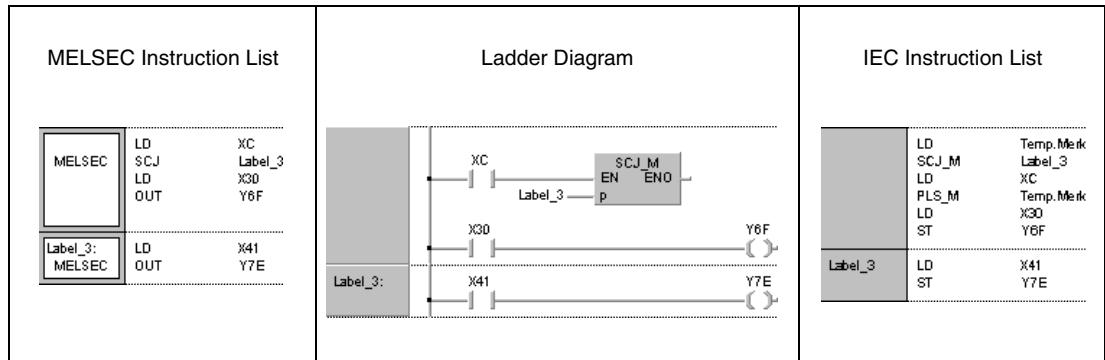
The following program jumps to the destination Label\_3 when X9 is set.



**Program Example 2**

SCJ

The following program jumps to the destination Label\_3 from the next scan when XC is set.



### 6.5.2 GOEND

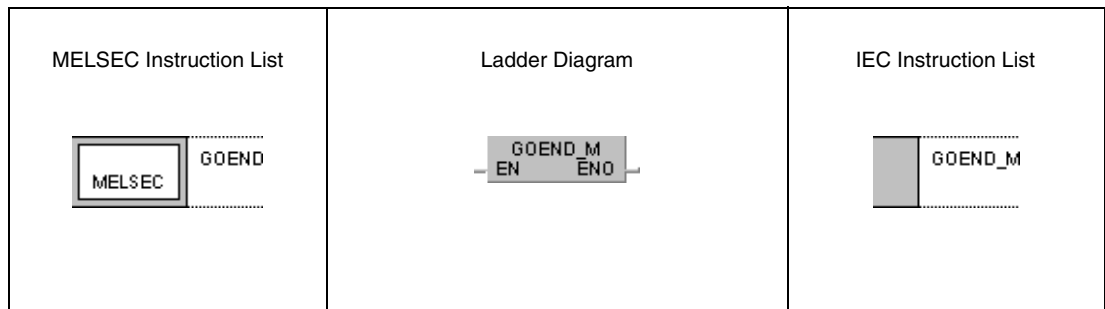
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

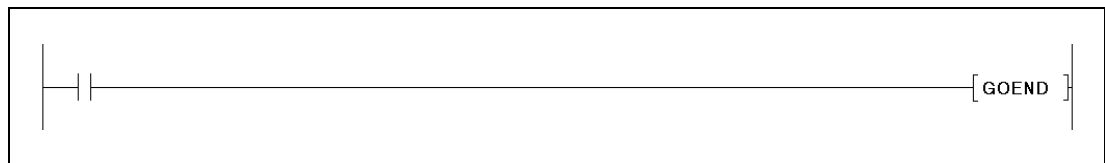
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	SM0	1

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions**      **GOEND Jump to the end of a program**

The jump destination of the GOEND instruction is the FEND or END instruction of the program.

**Operation Errors**

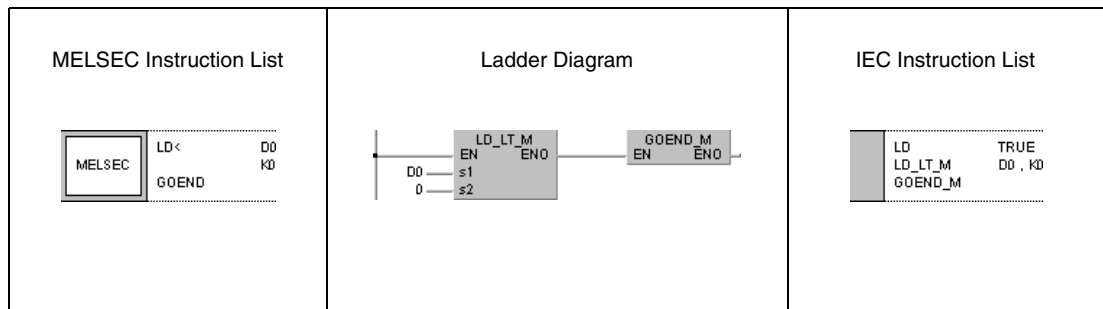
In the following cases an operation error occurs and the error flag is set:

- A GOEND instruction was executed after a CALL or ECALL instruction and before a RET instruction (error code: 4211).
- A GOEND instruction was executed after a FOR instruction and before a NEXT instruction (error code: 4200).
- A GOEND instruction was executed during an interrupt program but before an IRET instruction (error code: 4221).
- A GOEND instruction was executed between a CHKCIR and a CHKEND instruction (error code: 4230).
- A GOEND instruction was executed between an IX and an IXEND instruction (error code: 4231).

**Program Example**

**GOEND**

The following program jumps to the END instruction when data in D0 is negative.



## 6.6 Program Execution Control Instructions

Program execution control instructions invoke interrupt routines. The interrupts can be enabled or disabled individually or via bit patterns.

The following table gives an overview of these instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Interrupt disabled	DI	DI_M
Interrupt enabled	EI	EI_M
Bit pattern of execution conditions of interrupt programs	IMASK	IMASK_M
End of interrupt program	IRET	IRET_M

6.6.1 DI, EI, IMASK

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
● <sup>1</sup>	● <sup>1</sup>	●	●	●	●

<sup>1</sup> Using an AnN or AnS CPU the DI/EI instruction is only executable, if the internal relay M9053 is not set.

Devices MELSEC A

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
																					● <sup>1</sup>	

<sup>1</sup> DI and EI instruction only

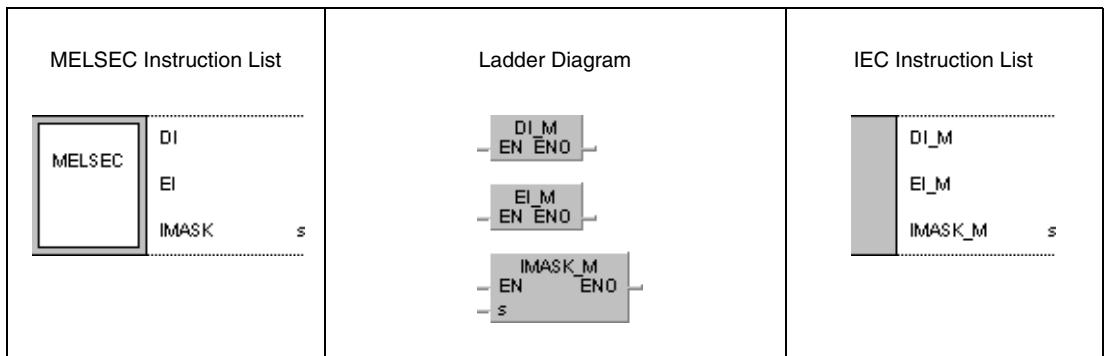
Devices MELSEC Q

s	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
—	—	● <sup>1</sup>	● <sup>1</sup>	—	● <sup>1</sup>	● <sup>1</sup>	—	—	—	—	2 ● <sup>1</sup>
—	—	—	—	—	—	—	—	—	—	—	1 ● <sup>2</sup>

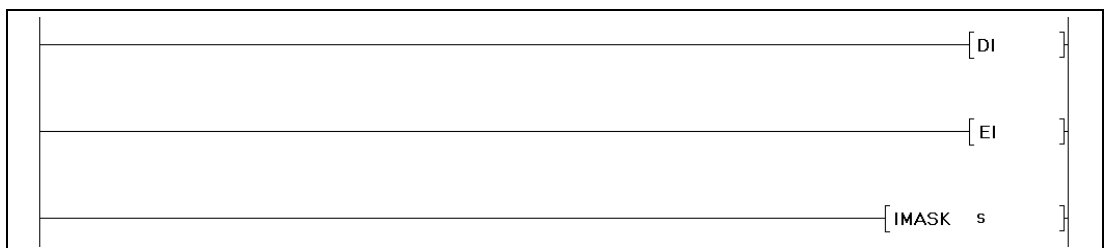
<sup>1</sup> IMASK instruction only

<sup>2</sup> DI and EI instruction only

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s	Bit pattern storing execution conditions of interrupts or first number of device storing bit pattern.	BIN 16-bit



**Functions Interrupt instructions**

An interrupt program is an inserted part of program (designated by an interrupt address lxx) that can be invoked by an external interrupt signal. The interrupt program is executed depending on the EI/DI instruction. Using an AnN CPU the meaning of the EI/DI instructions depends on the status of the internal relay M9053. Only if the relay is not set, the instructions serve as execution conditions for an interrupt program. If the internal relay M9053 is set, the instructions are used in conjunction with a link refresh (see "Refresh Instructions").

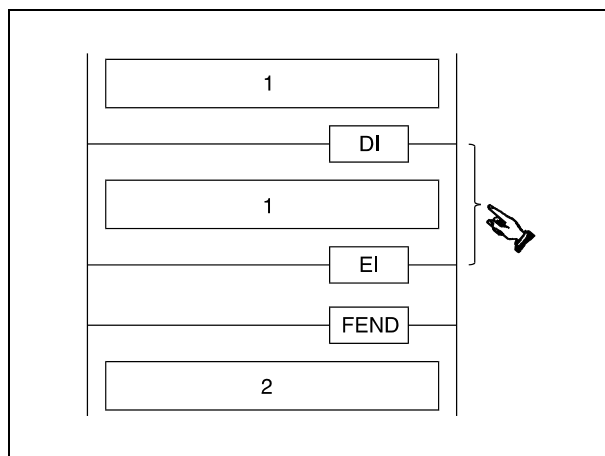
**DI Disable interrupt**

The DI instruction disables the execution of an interrupt program until an EI instruction is executed. The DI status after switching on or resetting the CPU is active.

**EI Enable interrupt**

The EI instruction enables invoking an interrupt program designated by an interrupt address lxx, or enables the execution of an IMASK instruction.

Even though an interrupt condition might be generated between the DI and EI instructions, the interrupt program is suspended until the entire cycle from DI to EI has been processed. The following diagram illustrates the execution in the GX Developer:



<sup>1</sup> Sequence program

<sup>2</sup> Interrupt program

**NOTE**

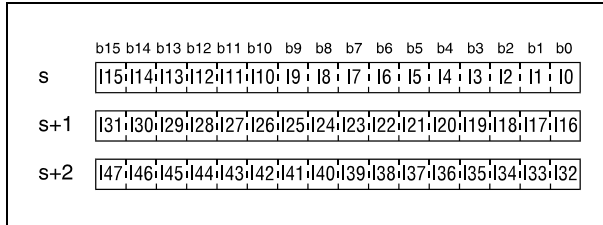
*The GX IEC Developer inserts the FEND instruction automatically. The event lxx has to be allocated to a task.*

**IMASK Bit pattern of execution conditions of interrupt programs (Q series and System Q)**

In the bit pattern designated by s a particular interrupt address is allocated to each bit. The condition of each bit determines whether the allocated interrupt can be executed. If the bit is reset (0), the interrupt program cannot be executed. If the bit is set (1), the interrupt program will be executed.

**QnACPU**

The following table shows the allocation of bits in s through s+2 to the corresponding interrupt addresses:

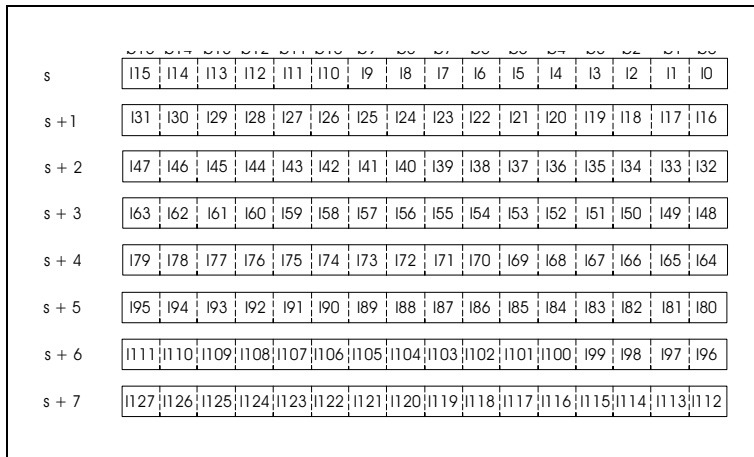


After switching on the CPU or resetting it with the RUN/STOP key switch the bits b0 to b31 (interrupt addresses I0 to I31) are set (1), i.e. these interrupt programs can be executed. The bits b32 to b47 (interrupt addresses I32 to I47) are reset (0), i.e. these interrupt programs cannot be executed.

The bit patterns designated by s through s+2 are stored in the special registers SD715 through SD717.

**System Q CPU (single processor type)**

The allocation of bits in s through s+7 to the corresponding interrupt addresses is shown below:



When the power supply of the CPU is switched on or when the CPU has been reset with the RUN/STOP switch, the execution of interrupt programs I0 through I31 is enabled.

The bit patterns designated by s through s+2 are stored in the special registers SD715 through SD717. The bit patterns designated by s+3 through s+7 are stored in the special registers SD781 through SD785.

The bit patterns are designated as s through s+7 successively although the special registers are separated (SD715 through SD717 and SD781 through SD785).

**System Q  
CPU (multi  
processor  
type)**

The allocation of bits in s through s+15 to the corresponding interrupt addresses is shown below:

	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
s	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
s + 1	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
s + 2	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
s + 3	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
s + 4	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
s + 5	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
s + 6	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
s + 7	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112
s + 8	I143	I142	I141	I140	I139	I138	I137	I136	I135	I134	I133	I132	I131	I130	I129	I128
s + 9	I159	I158	I157	I156	I155	I154	I153	I152	I151	I150	I149	I148	I147	I146	I145	I144
s + 10	I175	I174	I173	I172	I171	I170	I169	I168	I167	I166	I165	I164	I163	I162	I161	I160
s + 11	I191	I190	I189	I188	I187	I186	I185	I184	I183	I182	I181	I180	I179	I178	I177	I176
s + 12	I207	I206	I205	I204	I203	I202	I201	I200	I199	I198	I197	I196	I195	I194	I193	I192
s + 13	I223	I222	I221	I220	I219	I218	I217	I216	I215	I214	I213	I212	I211	I210	I209	I208
s + 14	I239	I238	I237	I236	I235	I234	I233	I232	I231	I230	I229	I228	I227	I226	I225	I224
s + 15	I255	I254	I253	I252	I251	I250	I249	I248	I247	I246	I245	I244	I243	I242	I241	I240

When the power supply of the CPU is switched on or when the CPU has been reset with the RUN/STOP switch, the execution of interrupt programs I0 through I31 is enabled. The execution of interrupt programs I32 through I255 is disabled.

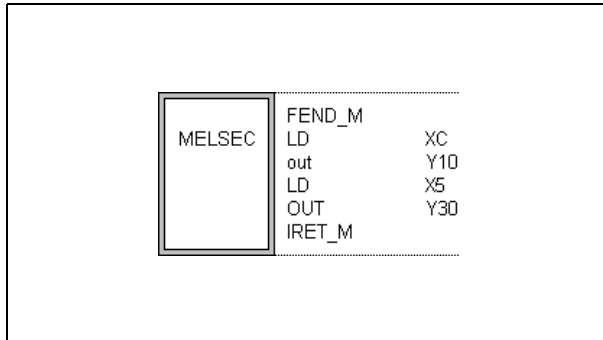
The bit patterns designated by s through s+2 are stored in the special registers SD715 through SD717. The bit patterns designated by s+3 through s+15 are stored in the special registers SD781 through SD793.

Although the special registers are separated (SD715 through SD717 and SD781 through SD793), the bit patterns are designated as s through s+15 successively.

**NOTE**

If a counter is needed within an interrupt program, there are special interrupt counters supplied for this purpose. The CPU types A3H, A3M, AnA, AnAS and AnU are not supplied with these special counters.

The interrupt address (interrupt pointer) designating the interrupt program occupies one program step.



With the GX Developer or with the GX IEC Developer in MELSEC mode the instructions FEND and IRET have to be programmed by the user.

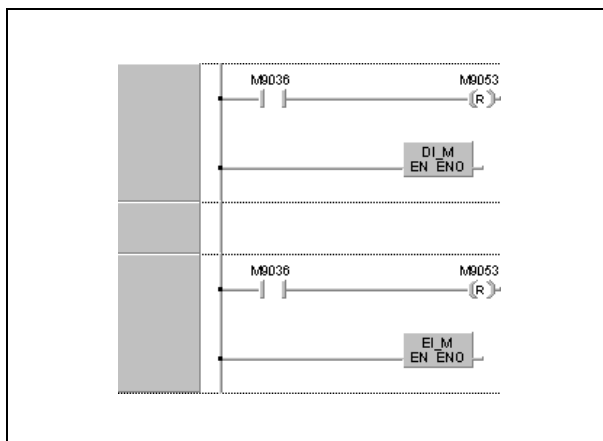
Alternatively to the MELSEC editor the IEC editor can be used. The interrupt is allocated to a task and the FEND and IRET instructions are placed automatically by the compiler of the GX IEC Developer MEDOC (see program example).

You will find a description of the interrupt conditions elsewhere in this manual.

During the execution of an interrupt program the DI status is internally set, so that no other interrupt program can be executed simultaneously. Another interrupt program can only be invoked after setting an EI instruction.

If an EI or DI instruction is placed within an MC instruction, the EI or DI instruction is executed without regard to the MC instruction.

Using an AnN or AnS CPU, the EI or DI is only executable, if the internal relay M9053 is not set. If the internal relay is set, the EI or DI instruction is the execution condition of a link refresh. With an AnN or AnS CPU the internal relay M9053 must be reset before an EI or DI instruction can be processed as a condition for an interrupt program call.



**Program Example**

EI, DI, IMASK (GX IEC Developer)

The following program enables the execution of an interrupt program, if X0 is set (1). If X0 is reset (0), the execution of the interrupt program is disabled.

The lower diagram shows the tasks to be programmed in the IEC mode. These tasks invoke the interrupt programs I1 and I2.

Interrupt\_1 (I1) and Interrupt\_2 (I2) are interrupt programs. The IRET instruction does not need to be programmed because it is placed automatically by the compiler of the GX IEC Developer.

MELSEC Instruction List	Ladder Diagram																																																												
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: center;">MELSEC</td> <td style="width: 10%; border-bottom: 1px dotted black;">LD</td> <td style="width: 10%; text-align: center;">X0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">CJ</td> <td style="border-bottom: 1px dotted black;">Label_1</td> <td style="border-bottom: 1px dotted black;">DI</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td style="border-bottom: 1px dotted black;">Label_1:</td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">X0</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td style="border-bottom: 1px dotted black;">MELSEC</td> <td style="border-bottom: 1px dotted black;">CJ</td> <td style="border-bottom: 1px dotted black;">Label_2</td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">X0</td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">MOVP</td> <td style="border-bottom: 1px dotted black;">H6</td> <td style="border-bottom: 1px dotted black;">D10</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">MOVP</td> <td style="border-bottom: 1px dotted black;">H0</td> <td style="border-bottom: 1px dotted black;">D11</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">MOVP</td> <td style="border-bottom: 1px dotted black;">H0</td> <td style="border-bottom: 1px dotted black;">D12</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">IMASK</td> <td style="border-bottom: 1px dotted black;">EI</td> <td style="border-bottom: 1px dotted black;">D10</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td style="border-bottom: 1px dotted black;">Label_2:</td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">M0</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td style="border-bottom: 1px dotted black;">MELSEC</td> <td style="border-bottom: 1px dotted black;">OUT</td> <td style="border-bottom: 1px dotted black;">Y20</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> </table>	MELSEC	LD	X0					CJ	Label_1	DI			Label_1:	LD	X0				MELSEC	CJ	Label_2	LD	X0			MOVP	H6	D10				MOVP	H0	D11				MOVP	H0	D12				IMASK	EI	D10			Label_2:	LD	M0				MELSEC	OUT	Y20				
MELSEC	LD	X0																																																											
	CJ	Label_1	DI																																																										
Label_1:	LD	X0																																																											
MELSEC	CJ	Label_2	LD	X0																																																									
	MOVP	H6	D10																																																										
	MOVP	H0	D11																																																										
	MOVP	H0	D12																																																										
	IMASK	EI	D10																																																										
Label_2:	LD	M0																																																											
MELSEC	OUT	Y20																																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="6" style="text-align: center;">IEC Instruction List</th> </tr> </thead> <tbody> <tr> <td style="width: 10%;"></td> <td style="width: 10%; border-bottom: 1px dotted black;">LD</td> <td style="width: 10%; text-align: center;">X0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">JMPC</td> <td style="border-bottom: 1px dotted black;">Label_1</td> <td style="border-bottom: 1px dotted black;">DI_M</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td style="border-bottom: 1px dotted black;">Label_1:</td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">X0</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">JMPC</td> <td style="border-bottom: 1px dotted black;">Label_2</td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">X0</td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">MOVP_M</td> <td style="border-bottom: 1px dotted black;">H6 , var_D10[10]</td> <td style="border-bottom: 1px dotted black;">MOVP_M</td> <td style="border-bottom: 1px dotted black;">H0 , var_D10[11]</td> <td style="border-bottom: 1px dotted black;">MOVP_M</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">MOVP_M</td> <td style="border-bottom: 1px dotted black;">H0 , var_D10[12]</td> <td style="border-bottom: 1px dotted black;">IMASK_M</td> <td style="border-bottom: 1px dotted black;">EI_M</td> <td style="border-bottom: 1px dotted black;">var_D10</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">EI_M</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td style="border-bottom: 1px dotted black;">Label_2:</td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">M0</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> <tr> <td style="border-bottom: 1px dotted black;">ST</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;">Y20</td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;"></td> </tr> </tbody> </table>	IEC Instruction List							LD	X0					JMPC	Label_1	DI_M			Label_1:	LD	X0					JMPC	Label_2	LD	X0			MOVP_M	H6 , var_D10[10]	MOVP_M	H0 , var_D10[11]	MOVP_M		MOVP_M	H0 , var_D10[12]	IMASK_M	EI_M	var_D10		EI_M					Label_2:	LD	M0				ST		Y20				
IEC Instruction List																																																													
	LD	X0																																																											
	JMPC	Label_1	DI_M																																																										
Label_1:	LD	X0																																																											
	JMPC	Label_2	LD	X0																																																									
	MOVP_M	H6 , var_D10[10]	MOVP_M	H0 , var_D10[11]	MOVP_M																																																								
	MOVP_M	H0 , var_D10[12]	IMASK_M	EI_M	var_D10																																																								
	EI_M																																																												
Label_2:	LD	M0																																																											
ST		Y20																																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: left;">Task Information</th> </tr> </thead> <tbody> <tr> <td style="width: 80%;"> <b>Task Attributes</b>                      Event: I1                      Interval: 0                      Priority: 31                 </td> <td style="width: 20%; text-align: center;">                     OK                      Cancel                      Comment                 </td> </tr> <tr> <td colspan="2">                     Name: Interrupt_1                      Size: 87 Bytes                      Type: TASK <input checked="" type="checkbox"/> Timer/ Output Control                      Last Change: 06.10.1997 14:59:36                      Security Level: <input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7  <input checked="" type="checkbox"/> Allow Read Access for lower Levels                 </td> </tr> </tbody> </table>	Task Information		<b>Task Attributes</b> Event: I1 Interval: 0 Priority: 31	OK Cancel Comment	Name: Interrupt_1 Size: 87 Bytes Type: TASK <input checked="" type="checkbox"/> Timer/ Output Control Last Change: 06.10.1997 14:59:36 Security Level: <input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7 <input checked="" type="checkbox"/> Allow Read Access for lower Levels		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: left;">Task Information</th> </tr> </thead> <tbody> <tr> <td style="width: 80%;"> <b>Task Attributes</b>                      Event: I2                      Interval: 0                      Priority: 31                 </td> <td style="width: 20%; text-align: center;">                     OK                      Cancel                      Comment                 </td> </tr> <tr> <td colspan="2">                     Name: Interrupt_2                      Size: 87 Bytes                      Type: TASK <input checked="" type="checkbox"/> Timer/ Output Control                      Last Change: 06.10.1997 15:00:05                      Security Level: <input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7  <input checked="" type="checkbox"/> Allow Read Access for lower Levels                 </td> </tr> </tbody> </table>	Task Information		<b>Task Attributes</b> Event: I2 Interval: 0 Priority: 31	OK Cancel Comment	Name: Interrupt_2 Size: 87 Bytes Type: TASK <input checked="" type="checkbox"/> Timer/ Output Control Last Change: 06.10.1997 15:00:05 Security Level: <input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7 <input checked="" type="checkbox"/> Allow Read Access for lower Levels																																																	
Task Information																																																													
<b>Task Attributes</b> Event: I1 Interval: 0 Priority: 31	OK Cancel Comment																																																												
Name: Interrupt_1 Size: 87 Bytes Type: TASK <input checked="" type="checkbox"/> Timer/ Output Control Last Change: 06.10.1997 14:59:36 Security Level: <input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7 <input checked="" type="checkbox"/> Allow Read Access for lower Levels																																																													
Task Information																																																													
<b>Task Attributes</b> Event: I2 Interval: 0 Priority: 31	OK Cancel Comment																																																												
Name: Interrupt_2 Size: 87 Bytes Type: TASK <input checked="" type="checkbox"/> Timer/ Output Control Last Change: 06.10.1997 15:00:05 Security Level: <input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5 <input type="radio"/> 6 <input type="radio"/> 7 <input checked="" type="checkbox"/> Allow Read Access for lower Levels																																																													

**NOTE**

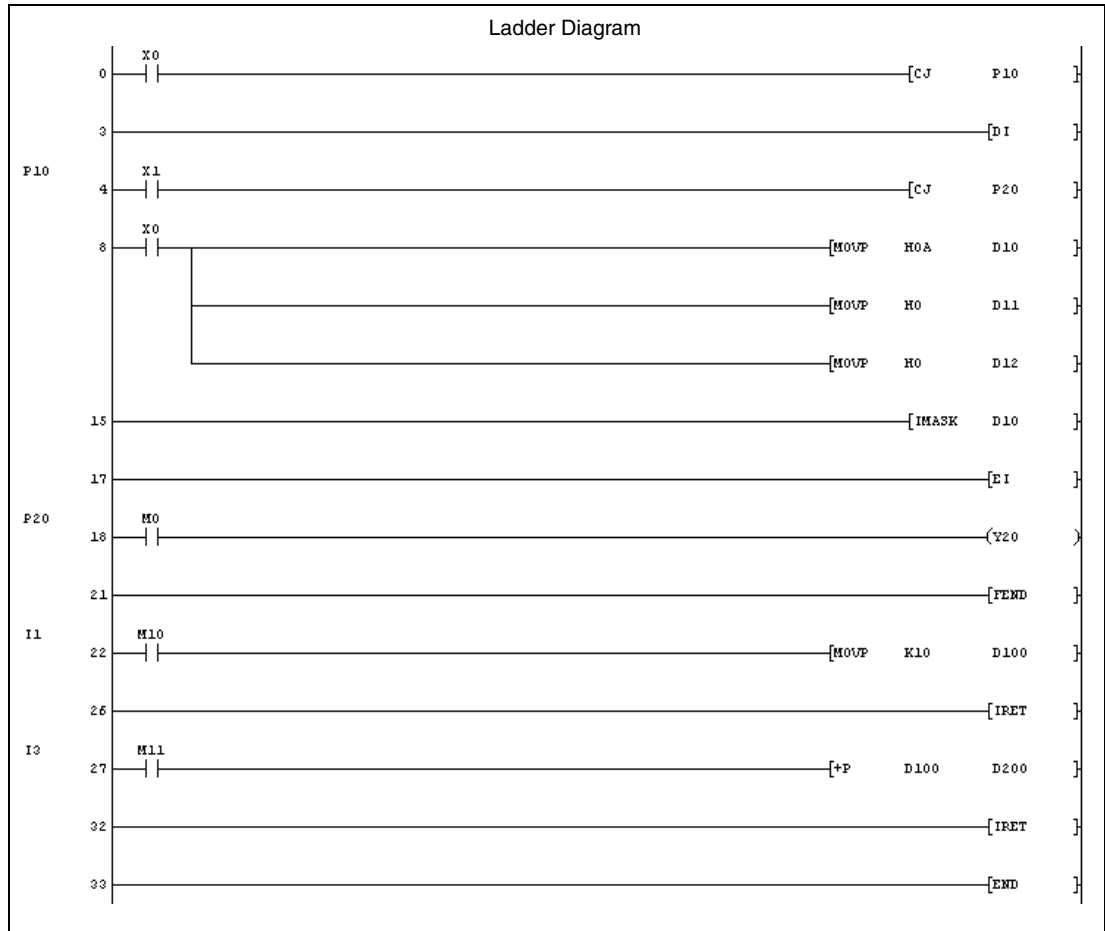
*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

**Program Example**

EI, DI, IMASK (CPU of the System Q, GX Developer)

In the following program, the execution of an interrupt program is enabled if X0 is set (1). When X0 is reset (0), the execution of the interrupt program is disabled.

I1 and I3 are interrupt programs.



		Instruction List	
0	LD	X0	
1	CJ	P10	
3	DI		
4	P10		
5	LD	X1	
6	CJ	P20	
8	LD	X0	
9	MOVP	H0A	D10
11	MOVP	H0	D11
13	MOVP	H0	D12
15	IMASK	D10	
17	EI		
18	P20		
19	LD	M0	
20	OUT	Y20	
21	FEND		
22	I1		
23	LD	M10	
24	MOVP	K10	D100
26	IRET		
27	I3		
28	LD	M11	
29	+P	D100	D200
32	IRET		
33	END		

**6.6.2 IRET**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

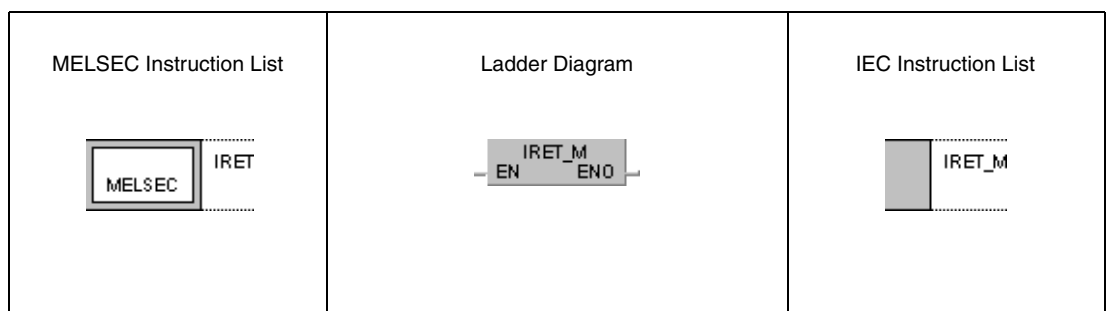
**Devices  
MELSEC A**

Usable Devices															Digit designation	Number of steps	Index	Carry Flag	Error Flag			
Bit Devices					Word Devices (16-bit)					Constant		Pointer		Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I	N	M9012	M9010 M9011
																					1	

**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

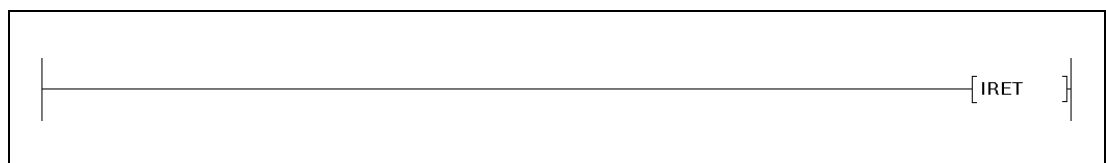
**GX IEC  
Developer**



**NOTE**

Within the IEC editors of the GX IEC Developer the IRET instruction is placed automatically in the program.

**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions**      **Return from an interrupt program to the main program**

**IRET    End of an interrupt program**

The end of an interrupt program is indicated by an IRET instruction.

Counters are processed continuously during the interrupt.

The main program is returned to after execution of the IRET instruction.

The following CPU types do not process interrupt counters: A3H, A3M, AnA, AnAS, AnU.

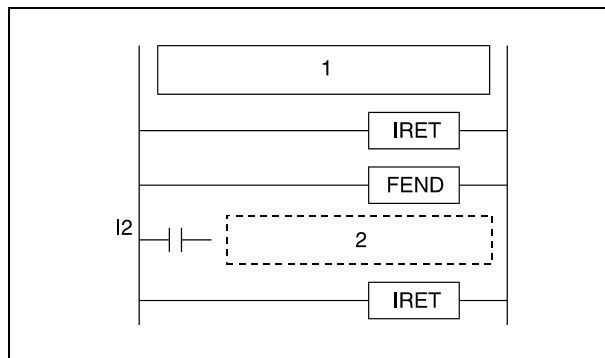
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- There is no corresponding interrupt address for the interrupt call (Q series and System Q = error code 4220).
- If the IRET instruction is placed prior to an interrupt program, the CPU quits processing at that point (Q series and System Q = error code 4223).
- An END, FEND, GOEND, or STOP instruction was placed between an interrupt call and an IRET instruction.

**NOTE**

*The following example shows a programming error!*



<sup>1</sup> Sequence program

<sup>2</sup> Interrupt program

**Program Example**

For the application of an IRET instruction in a program refer to the program examples for the EI, DI, and IMASK instructions.



## 6.7 Link Refresh Instructions

Link refresh instructions refresh data at input/output interfaces or data of transfer procedures. The following table gives an overview of these instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
I/O partial refresh (Q series and System Q)	RFS	RFS_M
	RFSP	RFSP_M
I/O partial refresh (A series)	SEG	SEG_M
Refresh instruction for link and interface data	COM	COM_M
Execution condition of refresh instruction for link and interface data	EI	EI_M
	DI	DI_M

6.7.1 RFS, RFSP

CPU

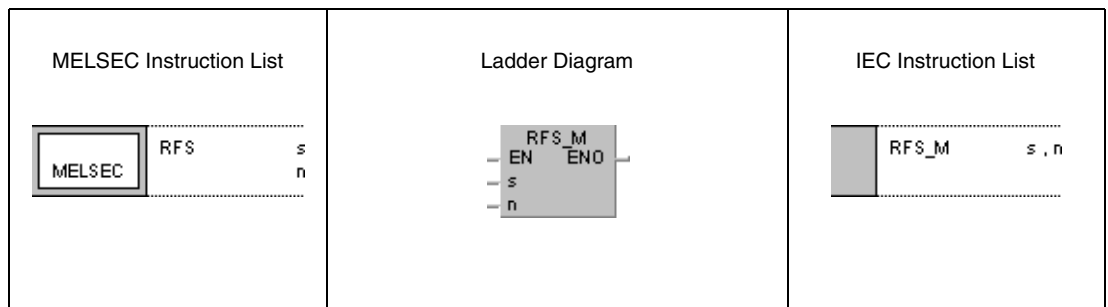
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

Devices  
MELSEC Q

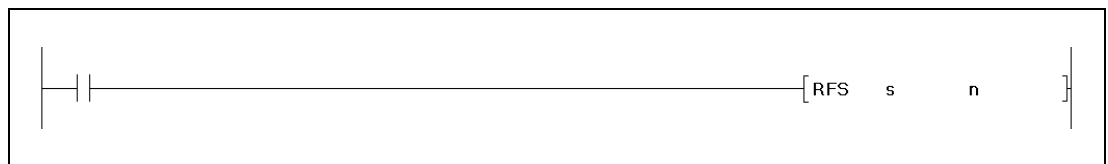
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	● <sup>1</sup>	—	—	—	—	—	—	—	—	SM0	3
n	●	●	●	●	●	●	●	●	—		

<sup>1</sup> X and Y only

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s	First number of I/O device to be refreshed.	Bit
n	Number of I/O bits to be refreshed.	BIN 16-bit

**Functions I/O partial refresh (Q series and System Q)**

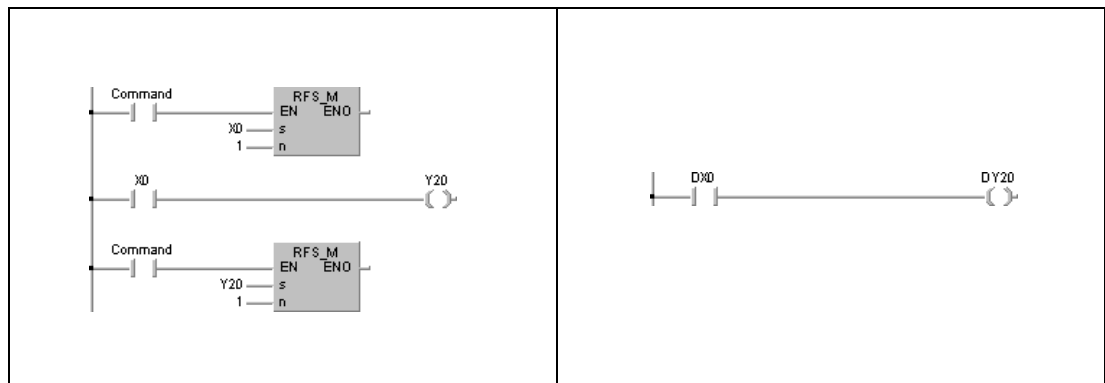
**RFS Refresh instruction**

The RFS instruction refreshes the inputs and outputs of the designated range of I/O devices during one program scan. It reads data from an external source or writes data to an output module.

Data is read from an external source or written to an external output module in a batch after executing an END instruction. Therefore, a pulse signal cannot be output during one program scan.

Executing a SEG instruction, the designated I/O addresses of inputs (X) and outputs (Y) are refreshed separately. Thus, even pulse signals can be output.

If direct access inputs/outputs (DX/DY) are used, the inputs (X) and outputs (Y) are refreshed bit by bit.



The program example on the left refreshes the input X0 and the output Y20 via an RFS instruction.

The program example on the right performs the same functions via DX and DY without a refresh instruction.

**Operation Errors**

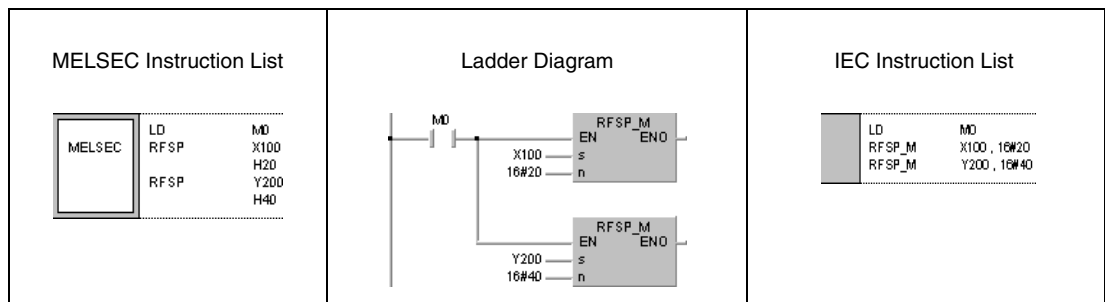
In the following cases an operation error occurs and the error flag is set:

- The number of bits determined by n exceeds the input/output device range.

**Program Example 1**

**RFSP**

With leading edge from M0, the following program refreshes the inputs X100 through X11F and the outputs Y200 through Y23F.



6.7.2 SEG

CPU

AnS	AnN	An(S)	AnU	QnA(S), Q4AR	System Q
● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>		

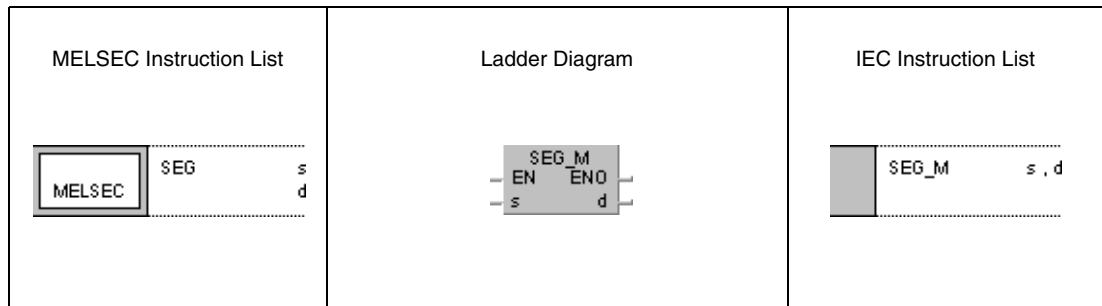
<sup>1</sup> A partial refresh is only executable, if the internal relay M9052 is set (1).

Devices  
MELSEC A

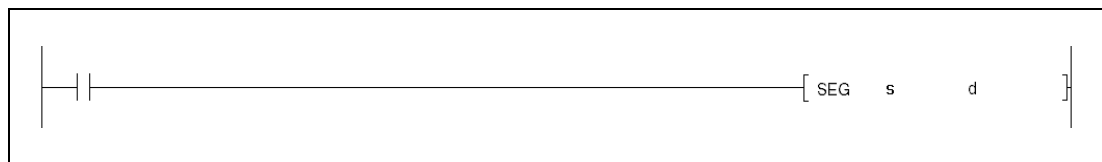
	Usable Devices															Digit designation K1 ↓ K8	Number of steps g ● <sup>1</sup>	Index ●	Carry Flag M9012	Error Flag M9010 M9011	
	Bit Devices					Word Devices (16-bit)					Constant	Pointer	Level								
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z				V	K	H (16#)
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●						

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
s	First number of I/O device to be refreshed.	Bit
d	Number of I/O bits to be refreshed.	BIN 16-bit

**Functions I/O partial refresh (A series)**

**General notes**

A partial refresh is only executable, if the internal relay M9052 is set (1). If this internal relay is not set, a SEG instruction serves as 7-segment decoder.

**SEG Partial refresh**

The SEG instruction enables refreshing a determined range of I/O devices, if the input condition is set.

Executing a partial refresh, the determined devices are refreshed during one program scan only. Input signals are still received and output signals are still passed on to output modules.

A partial refresh changes the operation condition of inputs (X) and outputs (Y) for one program scan during the I/O processing in normal refresh mode.

Applying a simple link refresh, the input and output signals are processed in a batch after execution of an END instruction. Therefore, during a program scan no pulse signals can be output. Applying a SEG instruction for a partial refresh however, the designated I/O devices at the inputs (X) and the outputs (Y) are refreshed separately. Thus, even pulse signals can be output.

**NOTE**

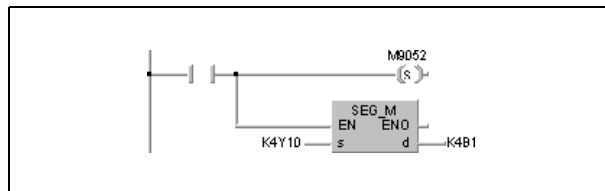
*Using an A2C CPU during a program scan, no pulse signals can be output while communicating with the I/O modules even if a partial refresh of the outputs (Y) via a SEG instruction is performed. For details, refer to the A2C CPU User Manual.*

**Execution conditions**

Program structure

First, the internal relay M9052 has to be set. The source data designation by s, following the SEG instruction determines the first number of device (inputs (X)/outputs (Y) only) to be refreshed. In addition, the number of I/O bits is determined in blocks of 8 I/O bits each.

The following diagram shows a programming pattern of the SEG instruction.



**First number of device**

The first number of device is always specified by the first device address of input or output devices (eg. X0, X10, Y20 etc.).

If a device address is set between Yn0 and Yn7 (Xn0 and Xn7), the refresh starts from address Yn0 (Xn0). If a device address is set between Yn8 and YnF (Xn8 and XnF), the refresh starts from address Yn8 (Xn8).

Number of I/O bits

The number of I/O bits available for a refresh can be set in a range of 8 to 2048. The allocation to blocks of 8 bits applies as follows:

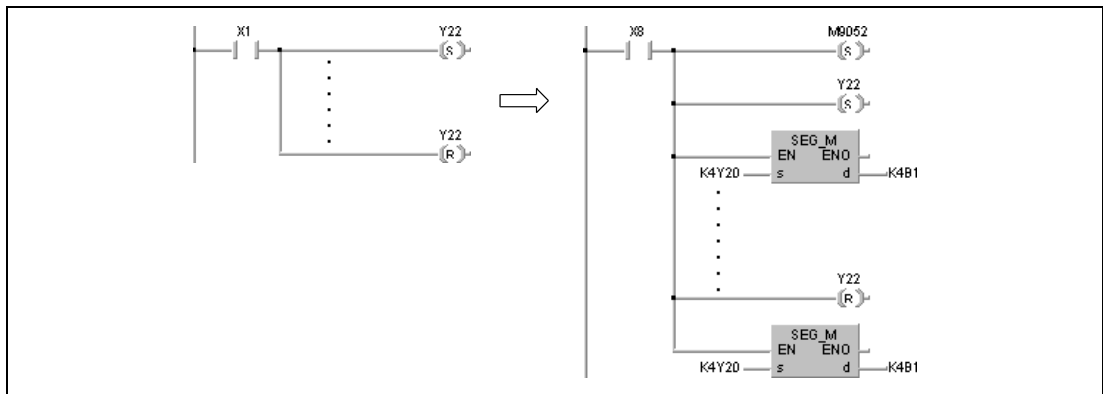
- B1 = 8 I/O bits
- B2 = 16 I/O bits
- ...
- BA = 80 I/O bits
- BB = 88 I/O bits
- ...
- B10 = 128 I/O bits
- ...
- BFF = 2048 I/O bits

Setting B0 refreshes all I/O bits in the PLC from the first number of device on.

The partial refresh is still proceeded if the SEG instruction is executed in direct mode of the CPU. However, in this case the operation conditions of inputs/outputs are not changed.

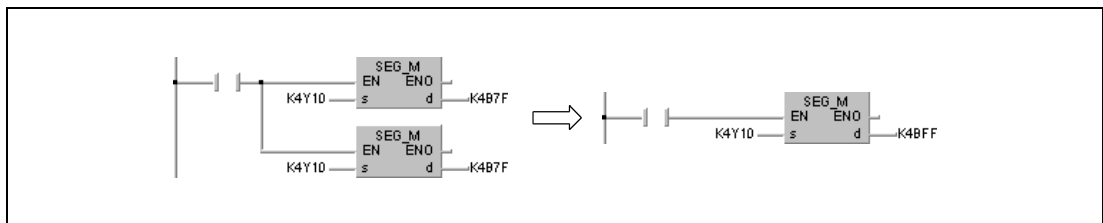
Pulse output using the SET and RST instructions in direct mode should be changed as shown below when the I/O control is changed to refresh mode.

**NOTE** *The following program cannot be processed by an A2C CPU:*



**NOTE** *Using an AnA or AnU CPU, errors in the I/O refresh might occur, if all 2048 I/O bits are refreshed via a SEG instruction. Therefore, the execution of a refresh has to be split into 2 x 1024 devices.*

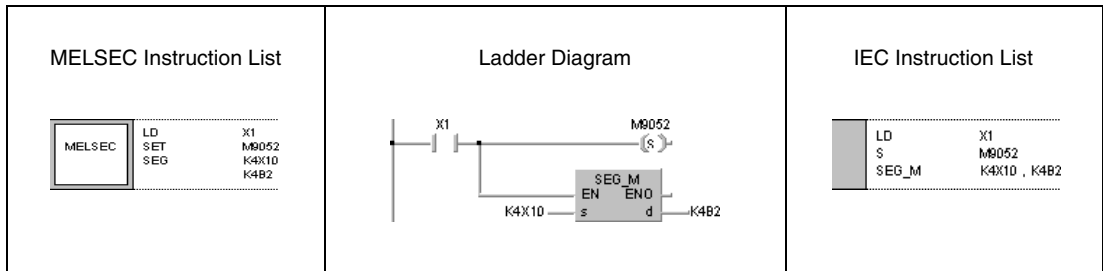
*The following program example shows the splitting of a program refreshing 2048 devices in one program scan using an AnA or AnU CPU.*



**Program Example**

SEG

The following program refreshes the inputs X10 through X1F.



6.7.3 COM

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

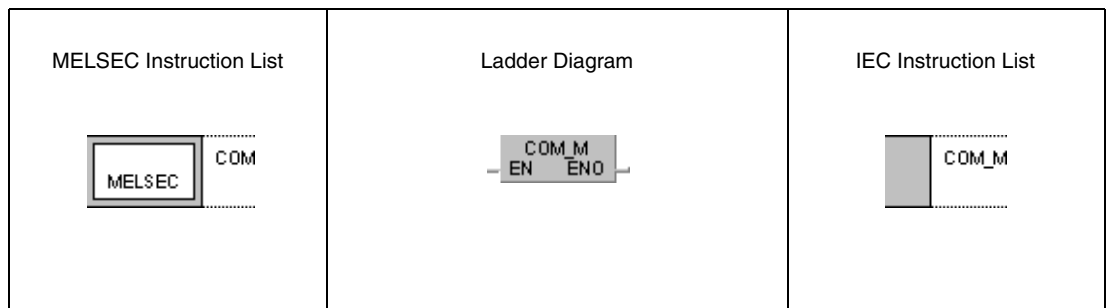
Devices  
MELSEC A

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
																	3					

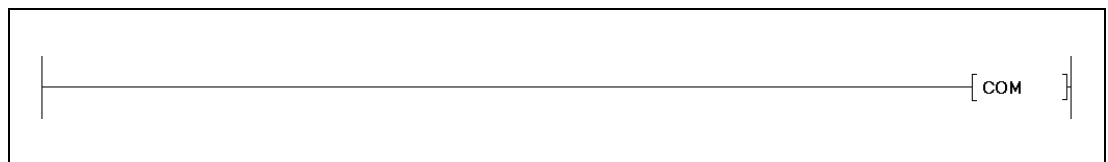
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
—	—	—



**Functions**    **Link refresh****COM**    **Refresh instruction for link and interface data**

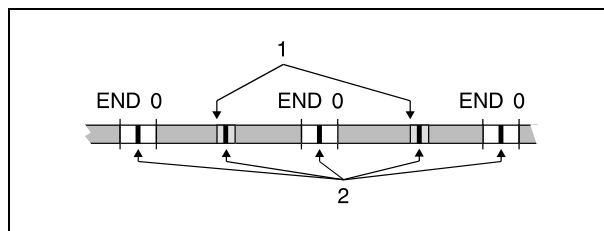
With a CPU of the Q series or the System Q the executed function of the COM instruction depends on the operation condition of the special relay SM775:

- If SM775 is not set (0), the link and interface data are refreshed (link refresh) and general data processing is performed (END processing).
- If SM775 is set (1), only general data processing is performed (END processing).

The following explanations apply to the Q series/System Q with SM775 not set (0) and to the A series:

A COM instruction is used to speed up data communication with a remote I/O station. If the scan time of a master station is longer than that of a local station, a COM instruction enables correct processing of received input and output data.

On execution of a COM instruction the CPU temporarily interrupts the sequence program, performs general data processing (END processing), and refreshes link and interface data (link refresh).



<sup>1</sup> COM instruction

<sup>2</sup> General data processing/link refresh

A COM instruction may be used any number of times in the sequence program. In this respect, note that the sequence program scan time is increased by the period of general data processing and link refresh times.

**NOTE**

*General data processing performs the following functions:*

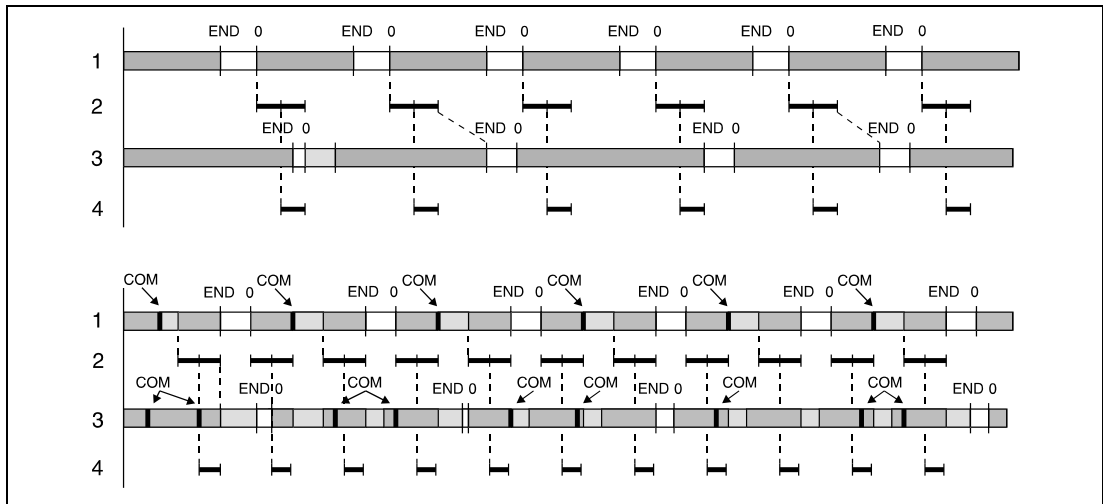
- *Communication between PLC and peripheral devices.*
- *Monitoring of other stations.*
- *Reading of buffer memory of other special function modules via a computer link module.*

*Link refresh processing performs the following functions:*

- *Refresh of CC-Link*
- *Automatic refresh of intelligent function modules*
- *Refresh of MELSECNET/10 and MELSECNET/H*
- *Automatic refresh of multi-CPU shared memory (for multi processor type CPUs of the System Q with function version B or later only)*

**Execution conditions**

The upper diagram shows data communication events without a COM instruction.  
 The lower diagram shows data communication events using a COM instruction.



- 1 Master station program
- 2 Data communication
- 3 Local station program
- 4 Remote I/O station, I/O refresh

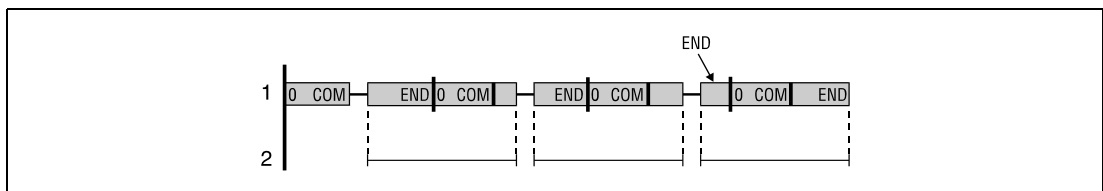
Data communication between links is speeded up in the sequence program of the master station via the COM instruction, because the number of communication events with the remote I/O station increases.

Data may not be received properly as shown above, if the scan time of the local station sequence program is longer than that of the master station. In this case, secure data communication is achieved with the COM instruction applied in the sequence program of the local station.

If a COM instruction is programmed in the sequence program of a local station, a link refresh is performed every time the local station receives the master station command between the following instructions:

- Step 0 and COM instruction
- COM instruction and COM instruction
- COM instruction and END instruction

If the link scan time of the link is longer than the sequence program scan time of the master station, data communication cannot be speeded up even if a COM instruction was programmed in the master station.



- 1 Sequence program of the master station
- 2 Link scan time in the slave station

### 6.7.4 EI, DI

**CPU**


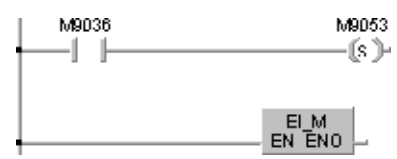
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
● <sup>1</sup>	● <sup>1</sup>				

<sup>1</sup> A link refresh can only be executed, if M9053 is set.

**Devices  
MELSEC A**

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I

**GX IEC  
Developer**

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																
<div style="border: 1px solid black; padding: 2px; display: inline-block; margin-bottom: 5px;">MELSEC</div> <table style="border-collapse: collapse; width: 100%;"> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">M9036</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">SET</td><td style="padding: 2px;">M9053</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">TRUE</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">DI</td><td style="padding: 2px;"></td></tr> </table>	LD	M9036	SET	M9053	LD	TRUE	DI			<table style="border-collapse: collapse; width: 100%;"> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">M9036</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">S</td><td style="padding: 2px;">M9053</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">TRUE</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">DI_M</td><td style="padding: 2px;"></td></tr> </table>	LD	M9036	S	M9053	LD	TRUE	DI_M	
LD	M9036																	
SET	M9053																	
LD	TRUE																	
DI																		
LD	M9036																	
S	M9053																	
LD	TRUE																	
DI_M																		
<div style="border: 1px solid black; padding: 2px; display: inline-block; margin-bottom: 5px;">MELSEC</div> <table style="border-collapse: collapse; width: 100%;"> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">M9036</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">SET</td><td style="padding: 2px;">M9053</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">TRUE</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">EI</td><td style="padding: 2px;"></td></tr> </table>	LD	M9036	SET	M9053	LD	TRUE	EI			<table style="border-collapse: collapse; width: 100%;"> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">M9036</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">S</td><td style="padding: 2px;">M9053</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">LD</td><td style="padding: 2px;">TRUE</td></tr> <tr><td style="border-right: 1px dashed black; padding: 2px;">EI_M</td><td style="padding: 2px;"></td></tr> </table>	LD	M9036	S	M9053	LD	TRUE	EI_M	
LD	M9036																	
SET	M9053																	
LD	TRUE																	
EI																		
LD	M9036																	
S	M9053																	
LD	TRUE																	
EI_M																		

**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions Execution conditions for a link refresh**

**General notes**

The executed function of a link refresh (see COM instruction) depends on the EI/DI instruction. The function of the EI/DI instruction with an AnN or A2C CPU depends on the status of the internal relay M9053. Only if this relay is set, these instructions serve as execution conditions for a link refresh. If the internal relay M9053 is not set, these instructions serve as execution conditions for an interrupt program.

**DI Disable link refresh execution**

The DI instruction disables the execution of a link refresh until an EI instruction is executed. After switching on or resetting the CPU the link refresh status is enabled.

Link refresh is always enabled during END processing.

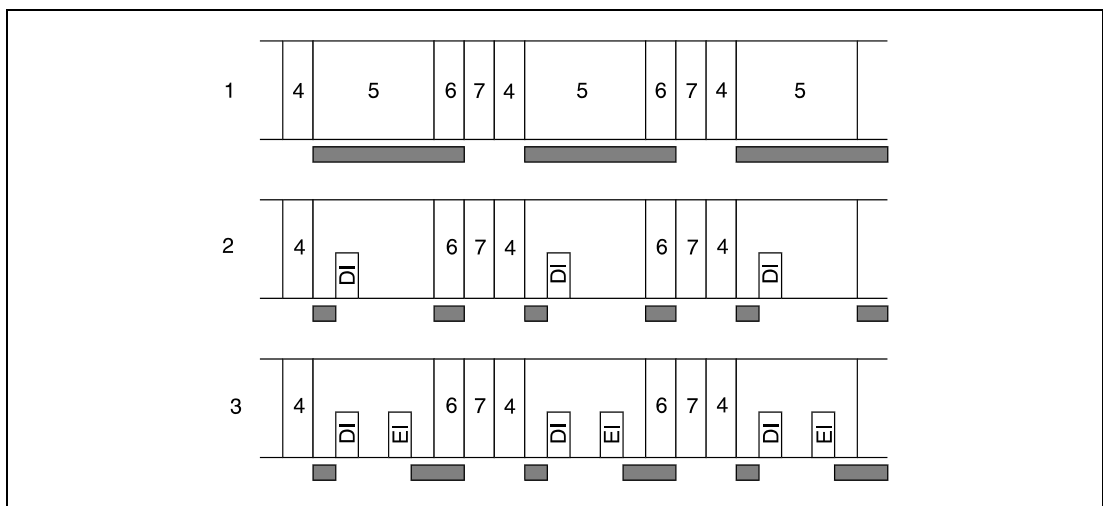
**EI Enable link refresh execution**

The execution of a link refresh is enabled after setting an EI instruction.

**Execution conditions**

The following diagram shows the execution conditions for the EI/DI instructions.

The markings indicate link data processing. There is no wait period for constant scan, if constant scan is not specified. In direct mode an I/O refresh is not possible.



- 1 Program execution without EI/DI instruction
- 2 Program execution with DI instruction
- 3 Program execution with EI/DI instruction
- 4 I/O refresh
- 5 Sequence processing
- 6 END processing
- 7 Wait period for constant scan

Processing is accomplished with fulfilled execution condition.

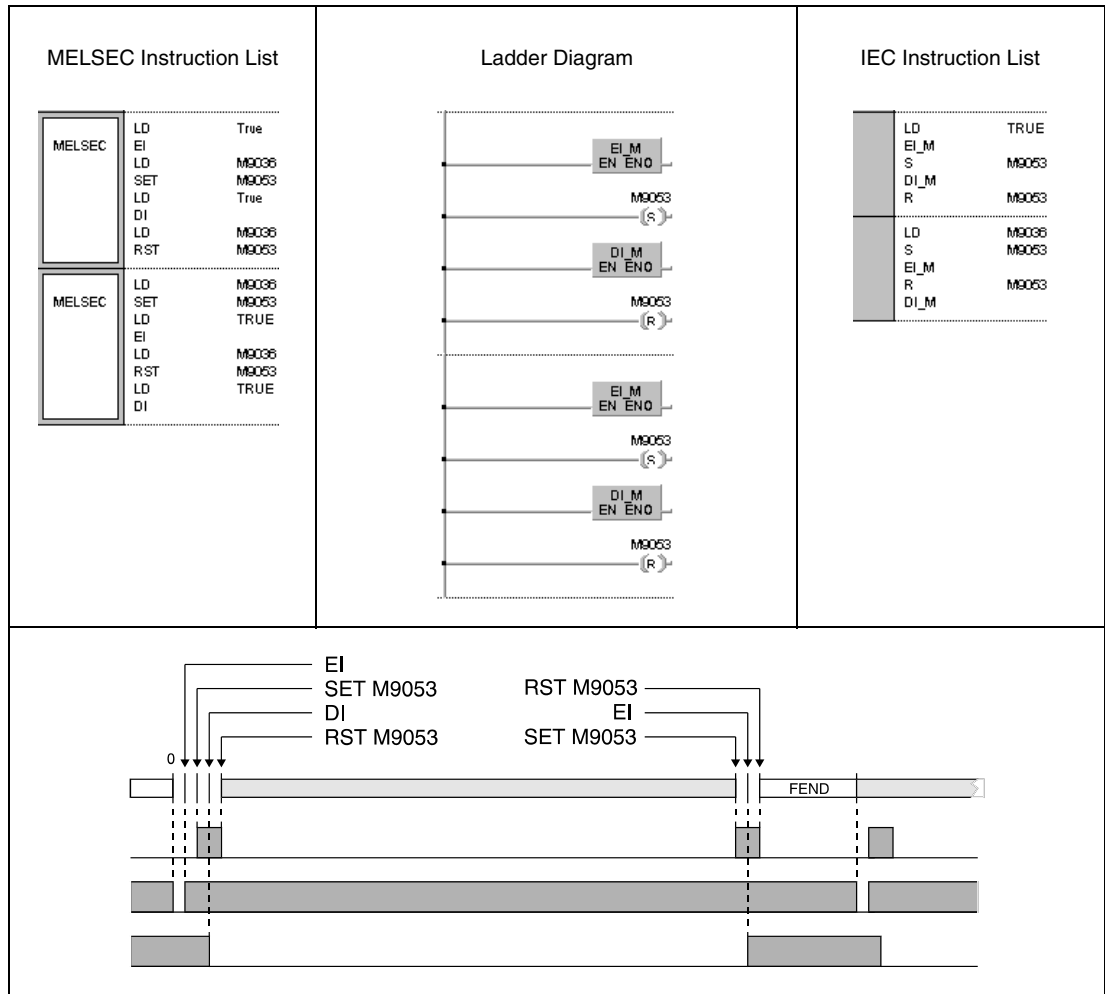
The function of the EI/DI instruction depends on the status of the internal relay M9053. After the execution of an EI/DI instruction, M9053 can be set (1) or reset (0).

If an EI or DI instruction is located within an MC instruction, it is processed independently of the execution of the MC instruction.

**Program Example**

EI

The following program disables a link refresh until the EI instruction is executed just before the FEND instruction. Invoking an interrupt program is supported at any time. The diagrams show the program execution over a period of time.



## 6.8 Other convenient Instructions

The instructions in the following table support programming of special timers and special counters, pulse counters and pulse outputs. Also included are instructions for positioning rotary tables and for building input matrices.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
1-Phase Input count-up/-down Counter	UDCNT1	UDCNT1_M
2-Phase Input count-up/-down Counter	UDCNT2	UDCNT2_M
Programmable (teaching) Timer	TTMR	TTMR_M
Special Function Timer	STMR	STMR_M
	STMRH	STMRH_M
Positioning of Rotary Tables	ROTC	ROTC_M
Ramp Signal	RAMP	RAMP_M
Pulse Counter	SPD	SPD_M
Pulse Output with set Number of Outputs	PLSY	PLSY_M
Pulse Width Modulation	PWM	PWM_M
Building of Input Matrices	MTR	MTR_M

6.8.1 UDCNT1

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

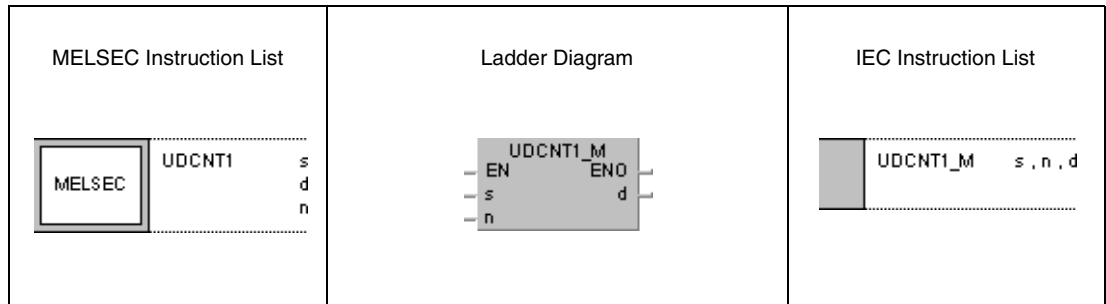
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	● <sup>1</sup>	—	—	—	—	—	—	—	—	4	
d	—	● <sup>2</sup>	—	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

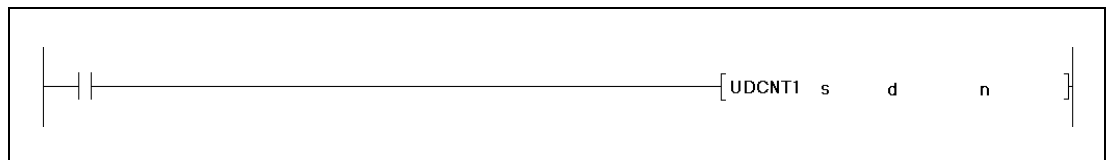
<sup>1</sup> X only

<sup>2</sup> C only

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	s+0: Input device number for count input (pulse signal, phase).	bit	Array [1..2] of BOOL
	s+1: Set count up or down 0 = count up 1 = count down		
d	Number of counter performing the UDCNT1 instruction.	BIN 16-bit (counter only)	ANY16
n	Setting	BIN 16-bit	ANY16

**Functions**    **1-phase count-up/-down counter****UDCNT1**    **Counter instruction**

When the input designated by s+0 (array\_s [0]) changes from 0 to 1 the current count of the counter designated by d is updated. Consequently, only leading edges are counted.

The counting direction is determined by the status of the input designated by s+1 (array\_s [1]):

If the input condition is 0, the pulses of the input designated by s+0 (Array\_s [0]) are added to the current count value.

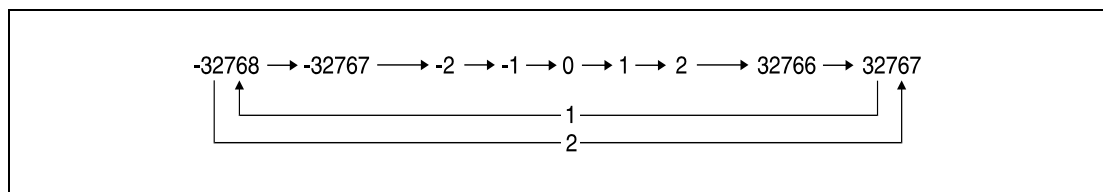
If the input condition is 1, the pulses of the input designated by s+0 (Array\_s [0]) are subtracted from the current count value.

The count processing performs as follows:

When counting up, the counter contact designated by d is set (1), if the current count value is identical to the setting value in n. The counting process continues while the counter contact is set (see program example).

When counting down, the counter contact is reset (0), if the current count value is identical to n-1 (see program example).

The counter designated by d is a ring counter. If the count reads 32767 and is increased by 1, the counter jumps to -32768. If the count reads -32768 and is decreased by 1, the counter jumps to 32767. The following diagram illustrates ring counting:



<sup>1</sup> Counting up

<sup>2</sup> Counting down

The UDCNT1 instruction is started when the execution condition is set and stopped when the execution condition is reset. If the counter is started once again, it counts on from the value before it was stopped.

An RST instruction resets the counter designated by d and the according counter contact.

**NOTE**

*The counting process of a UDCNT1 instruction is performed during a CPU interrupt (1 ms for a System Q multi processor CPU, 5 ms for a QnACPU). For this reason only pulses with set/reset times over 1ms resp. 5 ms can be counted accurately.*

*The setting value cannot be changed during the counting process (-> the input designated by s+0 (Array\_s [0]) is set). In order to change the setting, the input designated by s+0 (Array\_s [0]) has to be reset.*

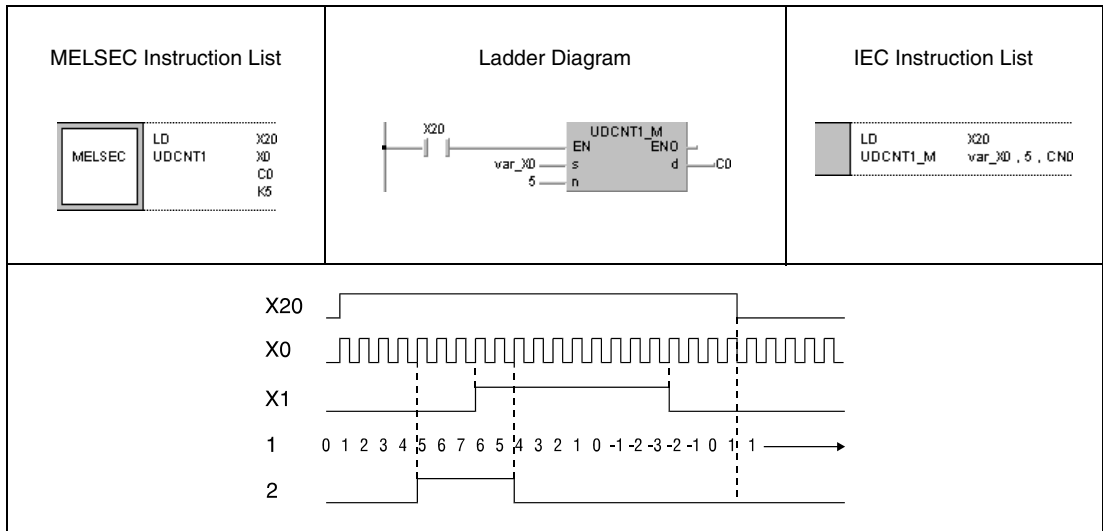
*Counters designated by a UDCNT1 instruction cannot be used by other instructions at the same time. In this case they would not return an accurate count.*



**Program Example**

UDCNT1

If X20 is set, the following program designates counter C0 (up/down counter) to count the number of leading edges from X0.



<sup>1</sup> Count

<sup>2</sup> Counter contact of counter C0

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

6.8.2 UDCNT2

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

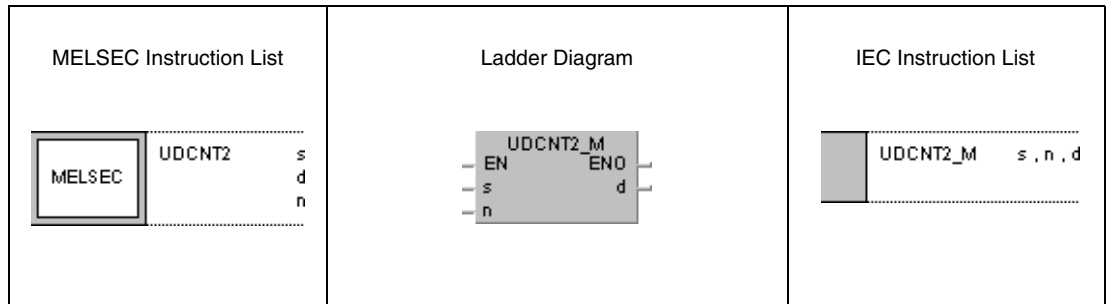
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	● <sup>1</sup>	—	—	—	—	—	—	—	—	4	
d	—	● <sup>2</sup>	—	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

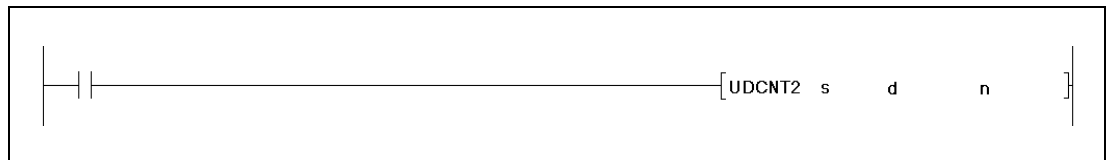
<sup>1</sup> X only

<sup>2</sup> C only

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	s+0: Input device number for count input (pulse signal, phase A)	Bit	Array [1..2] of BOOL
	s+1: Input device number of count input (pulse signal, phase B)		
d	Number of counter performing the UDCNT1 instruction	BIN 16-bit (counter only)	ANY16
n	Setting	BIN 16-bit	ANY16

**Functions**    **2-phase count-up/-down counter****UDCNT2**        **Counter instruction**

The count of the counter designated by d is changed depending on the condition of the two inputs s+0 (array\_s [0]) and s+1 (array\_s [1]).

The direction of the count is determined as follows:

If the input s+0 (array\_s[0]) is set (1) and the input s+1 (array\_s[1]) changes from 0 to 1 the current count is increased by 1.

If the input s+0 (array\_s[0]) is set (1) and the input s+1 (array\_s[1]) changes from 1 to 0 the current count is decreased by 1.

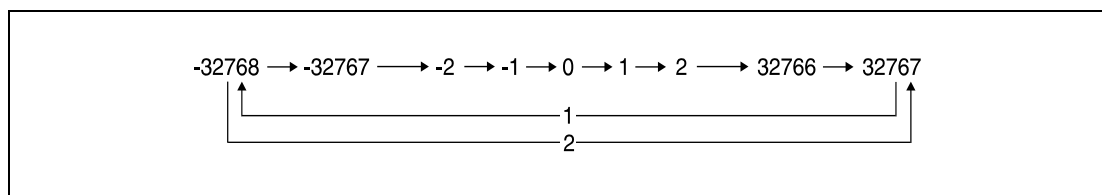
If the input s+0 (array\_s[0]) is reset (0) no counting operation is executed.

The count processing performs as follows:

When counting up, the counter contact designated by d is set (1), if the current count value is identical to the setting value in n. The counting process continues while the counter contact is set (see program example).

When counting down, the counter contact is reset (0), if the current count value is identical to n-1 (see program example).

The counter designated by d is a ring counter. If the count reads 32767 and is increased by 1, the counter jumps to -32768. If the count reads -32768 and is decreased by 1, the counter jumps to 32767. The following diagram illustrates ring counting:



<sup>1</sup> Counting up

<sup>2</sup> Counting down

The UDCNT2 instruction is started when the execution condition is set and stopped when the execution condition is reset. If the counter is started once again, it counts on from the value before it was stopped.

An RST instruction resets the counter designated by d and the according counter contact.

**NOTE**

*The counting process of a UDCNT2 instruction is performed during a CPU interrupt (1 ms for a System Q multi processor CPU, 5 ms for a QnACPU). For this reason only pulses with set/reset times over 1ms resp. 5 ms can be counted accurately.*

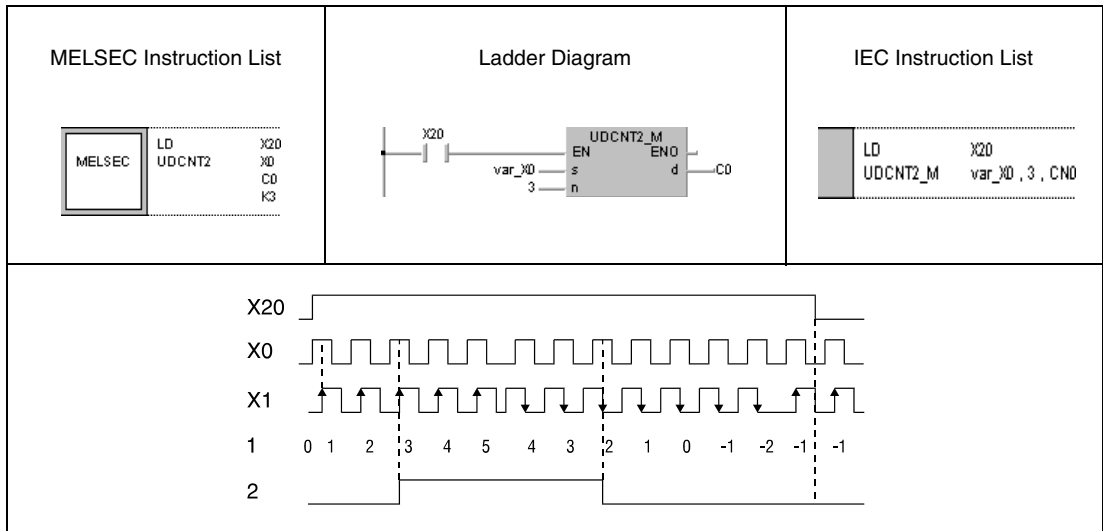
*The setting value cannot be changed during the counting process (-> the input designated by s+0 (Array\_s [0]) is set). In order to change the setting, the input designated by s+0 (Array\_s [0]) has to be reset.*

*Counters designated by a UDCNT2 instruction cannot be used by other instructions at the same time. In this case they would not return an accurate count.*

**Program Example**

UDCNT2

If X20 is set, the following program designates counter C0. The count and the count direction (up/down) depend on the conditions of X0 and X1.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 6.8.3 TTMR

**CPU**


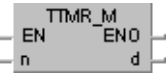
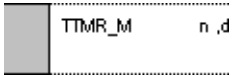
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

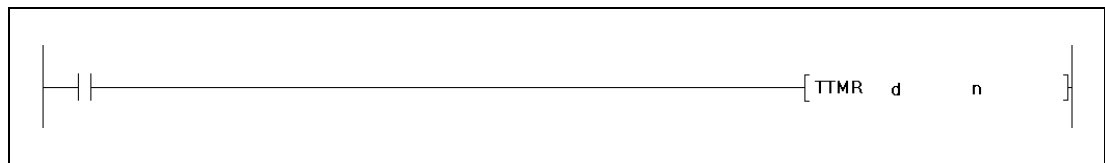
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
d	—	●	●	—	—	—	—	—	—	SM0	3
n	—	●	●	●	●	●	●	●	—		

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
d	d+0: Device storing measurement value.	BIN 16-bit	Array [1..2] of ANY16
	d+1: For internal use by the CPU.		
n	Measurement value multiplier		ANY16

**Functions**      **Programmable (teaching) Timer**

**TTMR    Timer instruction**

A timer programmed via the TTMR instruction measures the time of an input signal in seconds. The measurement value is multiplied with n and stored in d (array\_d [0]+[1]).

With leading edge from the input the devices d+0 (array\_d [0]) and d+1 (array\_d [1]) are cleared.

The multipliers designated by n are as follows:

- n = 0, multiplier 1
- n = 1, multiplier 10
- n = 2, multiplier 100

**NOTE**

*Time measurement is performed during the execution of a TTMR instruction. Applying a JMP instruction or a similar instruction to the TTMR instruction causes inaccurate time measurement.*

*The multiplier n must not be changed during the execution of a TTMR instruction. A change would cause inaccurate measurement.*

*The TTMR instruction can also be used in low speed type programs.*

*The device designated by d+1 (array\_d [1]) is used by the CPU. A change would cause inaccurate measurement.*

**Operation Errors**

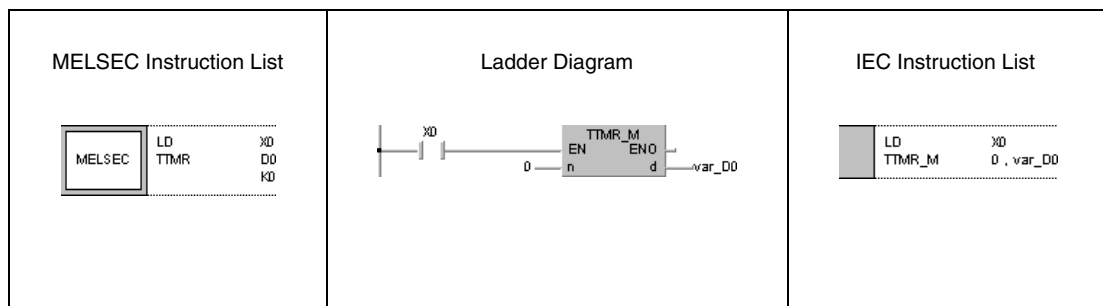
In the following cases an operation error occurs and the error flag is set:

- The value designated by n exceeds the relevant device range of 0 to 2 (error code: 4100).

**Program Example**

**TTMR**

If X0 is set, the following program measures the time in seconds (multiplier = 1). The result is stored in D0.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

**6.8.4 STMR, STMRH**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

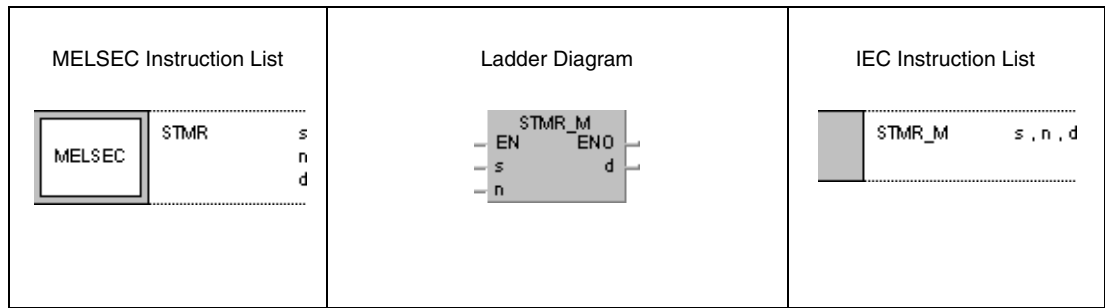
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

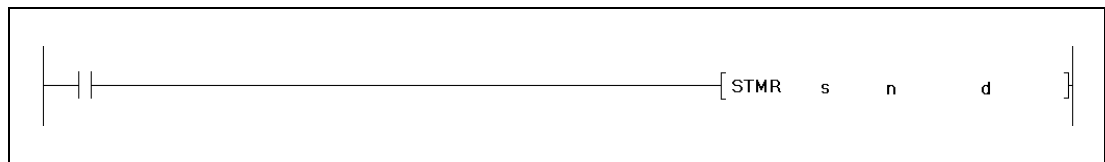
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant	Other		
	Bit	Word		Bit	Word						
s	—	● <sup>1</sup>	—	—	—	—	—	—	—	3	
n	●	—	—	—	—	—	—	—	—		
d	—	●	●	●	●	●	●	●	—		

<sup>1</sup> Can only be used by timer (T) data.

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	Number of timer.	BIN 16-bit (timer only)	ANY16
n	Time setting.	BIN 16-bit	ANY16
d	d+0: OFF delay timer output. d+1: One shot timer output after OFF (Set by trailing edge). d+2: One shot timer output after ON (Set by leading edge). d+3: ON delay timer output.	Bit	Array [1..4] of BOOL

<b>Functions</b>	<b>Special function timer</b>
	<b>STMR</b> <b>Timer instruction for low speed timers</b>
	<b>STMRH</b> <b>Timer instruction for high speed timers</b>

The STMR instruction uses outputs designated by d+0 through d+3 (array\_d [0] through array\_d [3]) to perform four different timer functions:

OFF delay timer output (d+0) (array\_d [0])

The output designated by d+0 (array\_d [0]) is set (1) with leading edge from the execution condition. With trailing edge from the execution condition and after a period of time designated by n the output is reset (0) again.

One shot timer output after OFF (Set by trailing edge, d+1 (array\_d [1]))

The output designated by d+1 (array\_d [1]) is set (1) with trailing edge from the execution condition. After a period of time designated by n or with leading edge from the execution condition the output is reset (0) again.

One shot timer output after ON (Set by leading edge, d+2 (array\_d [2]))

The output designated by d+2 (array\_d [2]) is set (1) with leading edge from the execution condition. After a period of time designated by n or with trailing edge from the execution condition the output is reset (0) again.

ON delay timer output (d+3 (Array [3]))

The output designated by d+3 (array\_d [3]) is set (1) with trailing edge from the timer coil. This corresponds to an ON delay time designated by n. The output d+3 is also set with trailing edge from the execution condition and then reset (0) after a period of time designated by n.

The timer coil of the timer designated by s is set (0) with leading edge from the execution condition and starts measuring the time designated by n.

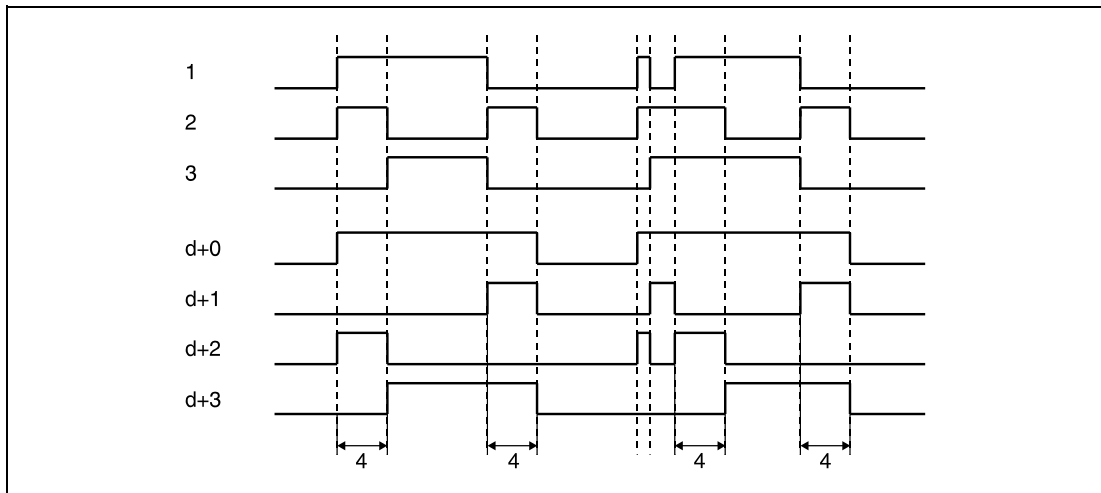
The timer coil measures time until the measurement value matches the time setting n and then drops out.

If the execution condition is reset before the time setting n has passed, the timer coil remains set and time measurement is suspended at that point.

If the execution condition is set again the measurement value is cleared to 0 and time measurement starts again.

The timer contact designated by s is either set by trailing edge from the execution condition and set timer coil or by trailing edge from the timer coil and set execution condition. The timer contact is reset by trailing edge from the execution condition and reset timer coil. The timer contact is supplied for CPU internal use only.





- <sup>1</sup> Execution condition
- <sup>2</sup> Timer coil designated by s
- <sup>3</sup> Timer contact designated by s
- <sup>4</sup> Time setting n

Time measurement is performed during the execution of an STMR instruction. Applying a JMP instruction or a similar instruction to the STMR instruction causes inaccurate time measurement.

The realtime designated by d can be calculated by multiplying the time setting n with the time unit for low speed timers (default value = 100 ms).

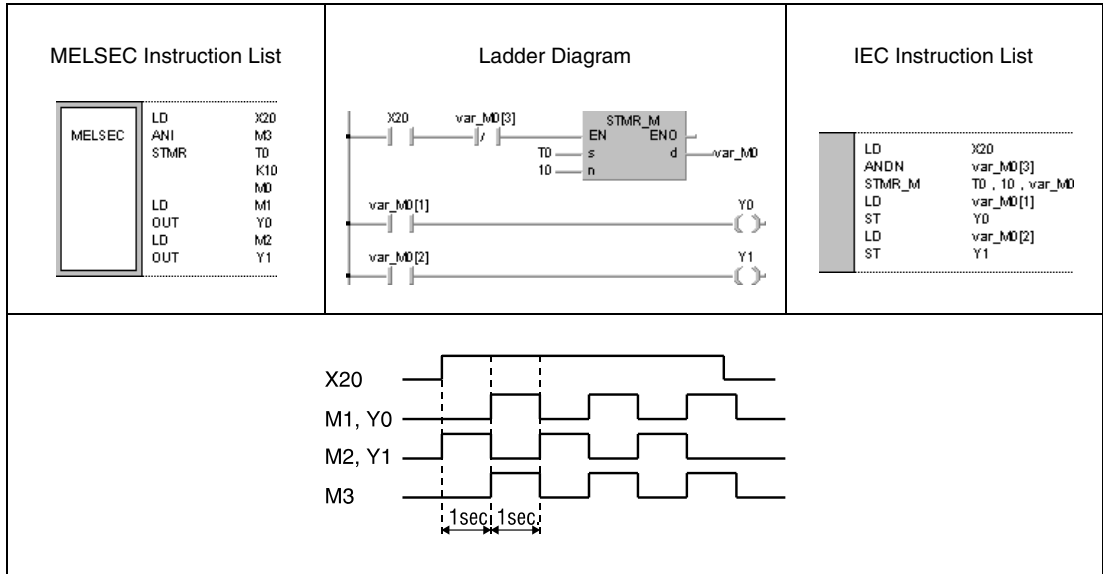
The constant n has to range within 1 and 32767.

The timer designated by s cannot be used by an OUT instruction. If an OUT instruction and an STMR instruction use the same timer, the STMR instruction cannot be performed accurately.

**Program Example**

**STMR**

If X20 is set, the following program alternately sets the outputs Y0 and Y1 for 1 second each. The used timer is a 100 ms timer. The time period of 1 second is calculated by multiplying K10 with 100 ms.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 6.8.5 ROTC

**CPU**

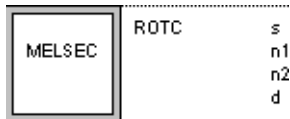
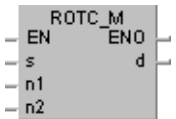
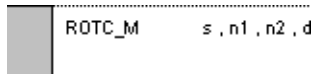
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

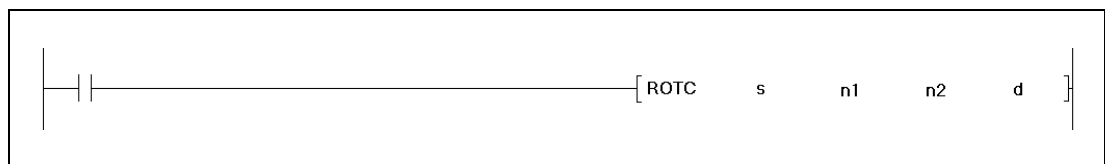
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	5
n1	●	●	●	●	●	●	●	●	—		
n2	●	●	●	●	●	●	●	●	—		
d	●	—	—	—	—	—	—	—	—		

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	--

**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	s+0: Measurement of table rpm (internal use only).	BIN 16-bit	Array [1..3] of ANY16
	s+1: Number of position.		
	s+2: Number of sector.		
n1	Number of sectors (divisions) on table (2 to 32767).		ANY16
n2	Number of low speed sectors (0 to n1).	ANY16	
d	d+0: A-phase input signal.	Bit	Array [1..8] of Bool
	d+1: B-phase input signal.		
	d+2: Zero position detection input signal.		
	d+3: High speed forward output signal (internal use only).		
	d+4: Low speed forward output signal (internal use only).		
	d+5: Stop output signal (internal use only).		
	d+6: High speed reverse output signal (internal use only).		
	d+7: Low speed reverse output signal (internal use only).		

**Functions      Positioning instruction for rotary tables**

**ROTC    Positioning instruction**

The ROTC instruction rotates a sector designated by s+2 (array\_s [2]) on a table with a specified number of sectors (divisions) designated by n1 to a specified position designated by s+1 (array\_s [1]).

The positions and sectors on the rotary table are numbered counterclockwise.

The value in s+0 (array\_s [0]) is internally used by the system to determine which sector is located where in relation to the zero position. This value must not be changed, otherwise the table will not be positioned accurately.

The value in n2 determines the number of sectors the table can be rotated by at low speed. This value must be less than that designated by n1.

The A/B-phase inputs designated by d+0 (array\_d [0]) and d+1 (array\_d [1]) detect the direction of the rotation. Both inputs receive pulses. If the A-phase input d+0 (array\_d [0]) is set, the direction of the rotation is determined by the pulse edge of the B-phase input d+1 (array\_d [1]):

If the B-phase is at leading edge at that moment the table rotates clockwise (to the right).

If the B-phase is at trailing edge at that moment the table rotates counterclockwise (to the left).

The input designated by d+2 (array\_d [2]) detects the zero position. This input is set, if sector 0 reaches position 0. If this input is set during the execution of a ROTC instruction, the value in s+0 (array\_s [0]) is reset. For accurate positioning this value in s+0 (array\_s [0]) should be reset before positioning via the ROTC instruction.

Data in d+3 (array\_d [3]) through d+7 (array\_d [7]) store output signals for operating the rotary table. Which output signal is set depends on the current operation result of the ROTC instruction.

If all operation results were 0 just before executing a ROTC instruction, the outputs designated by d+3 (array\_d [3]) through d+7 (array\_d [7]) are reset without positioning the table. After resetting the execution condition these outputs are reset either.

A ROTC instruction can only be executed once in a program. Repeated application within one program causes faulty operation of the instruction.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The value specified in s+0 (array\_s [0]) through s+2 (array\_s [2]) or in n2 is greater than that in n1 (error code: 4100).

**Program Example** ROTC

In the following program the contacts X0, X1 (incremental encoder), and X2 address the internal relays for detection of the rotating direction and zero position M0 (var\_M0 array [0]) through M2 (var\_M0 array [2]). The contact X2 is activated, if sector 0 is located at position 0 (zero position detection).

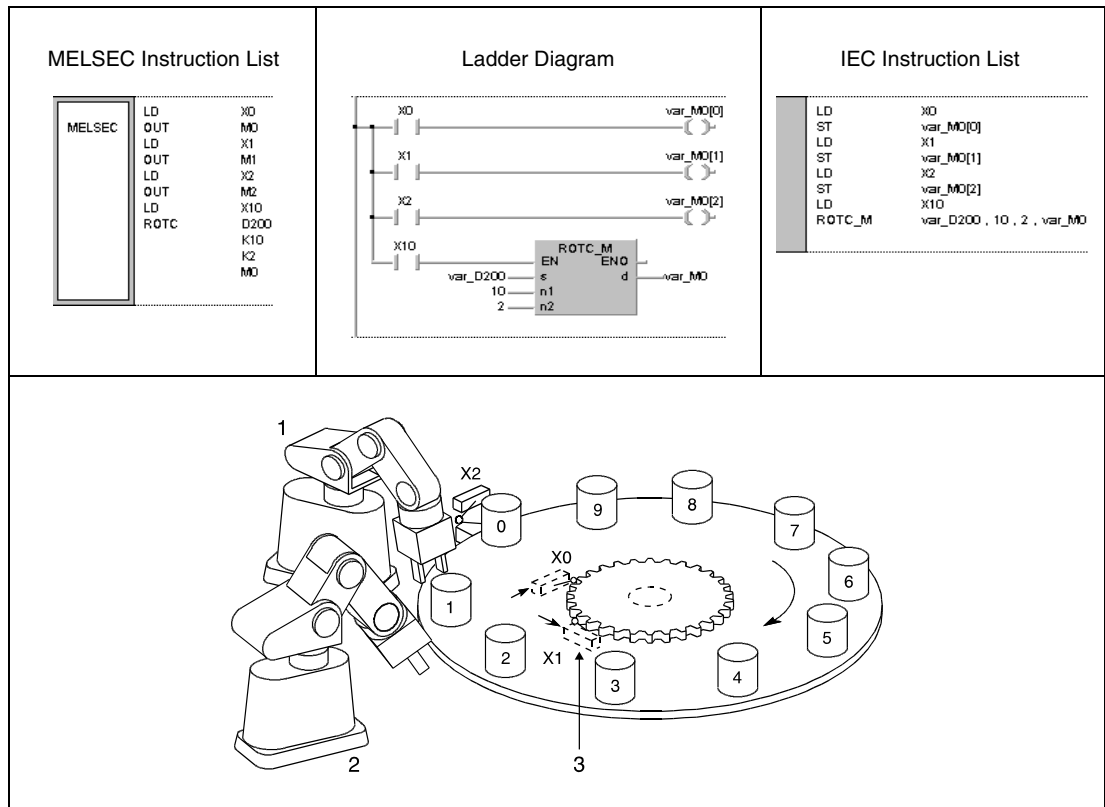
The rotary table shown below is divided into 10 sectors.

Which item (sector) will be moved to which station (position) has to be specified in D201 (var\_D200 array [1]) and D202 (var\_D200 array [2]) before the execution of the ROTC instruction.

Due to the value n1=10 the contact of the counter register outputs 10 pulses each rotation (division). The value n2=2 specifies the number of low speed divisions.

For example, if register D201 (var\_D200 array [1]) stores the value 0 and register D202 (var\_D200 [2]) stores the value 3, the rotary table moves item 3 (sector 3) to station 0 (position 0) travelling the shortest distance (clockwise). The sectors 1 through 3 rotate at low speed.

For an allocation of single registers and internal relays or array elements respectively to the corresponding functions see the table following the example.



- 1 Station 0 (position 0)
- 2 Station 1 (position 1)
- 3 Incremental encoder

Data register	Meaning	Remark
D200 (var_D200 Array [0])	Counter register	
D201 (var_D200 Array [1])	Position of station	These values are written to the data registers D201 (var_D200 array [1]) and D202 (var_D200 array [2]) via a MOV instruction.
D202 (var_D200 Array [2])	Position of item	
M0 (var_M0 Array [0])	A-phase signal	The internal relays M0 (var_M0 array [0]) through M2 (var_M0 array [2]) are addressed by the inputs X0 through X2 (see program example).
M1 (var_M0 Array [1])	B-phase signal	
M2 (var_M0 Array [2])	Zero position detection signal	
M3 (var_M0 Array [3])	High speed forward rotation	After X10 is set the ROTC instruction is activated and the internal relays M3 (var_M0 array [3]) through M7 (var_M0 array [7]) are assigned specified functions. After resetting X10 these internal relays are reset either.
M4 (var_M0 Array [4])	Low speed forward rotation	
M5 (var_M0 Array [5])	Stop signal	
M6 (var_M0 Array [6])	High speed reverse rotation	
M7 (var_M0 Array [7])	Low speed reverse rotation	

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 6.8.6 RAMP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

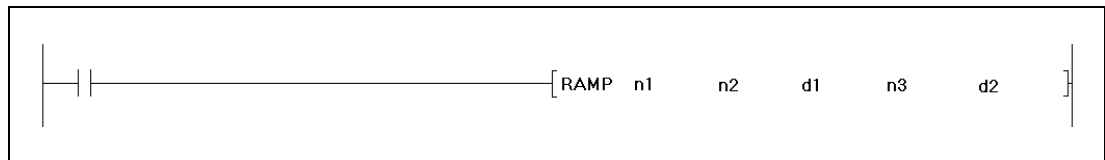
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
n1	●	●	●	●	●	●	●	●	—	—	6
n2	●	●	●	●	●	●	●	●	—	—	
d1	●	●	●	●	●	●	●	—	—	—	
n3	●	●	●	●	●	●	●	●	—	—	
d2	●	—	—	—	—	—	—	—	—	—	

**GX IEC Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>MELSEC</p> </div> <p>RAMP      n1                  n2                  d1                  n3                  d2</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>RAMP_M      n1 , n2 , n3 , d1 , d2</p> </div>
--	-----------------------	---

**GX Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
n1	Initial value of operation.	BIN 16-bit	ANY16
n2	Final value of operation.		ANY16
d1	(d1)+0: Device storing current value.		Array [1..2] of ANY16
	(d1)+1: Device storing number of executed moves (internal use only).		
n3	Number of moves to be executed.		ANY16
d2	(d2)+0: Bit to be set after completion.	Bit	Array [1..2] of Bool
	(d2)+1: Bit determining storage of operation result.		

**Functions Ramp signal**

**RAMP Instruction for changing the content of a device gradually**

A RAMP instruction changes the content in (d1)+0 (array\_d1 [0]) gradually from the initial value designated by n1 to the final value designated by n2.

The number of moves performing the gradual changes is designated by n3.

The number of moves already executed is stored in (d1)+1 (array\_d1 [1]) for internal system use.

When the operation is completed the device designated by (d2)+0 (array\_d2 [0]) is set.

The signal condition of the device (d2)+0 (array\_d2 [0]) and the content of the device (d1)+0 (array\_d1 [0]) depend on the signal condition of the device (d2)+1 (array\_d2 [1]):

If the device (d2)+1 (array\_d2 [1]) is not set, the device (d2)+0 (array\_d2 [0]) will be reset during the next scan and the RAMP instruction will begin a new move operation from the value currently stored in (d1)+0 (array\_d1 [0]).

If the device (d2)+1 (array\_d2 [1]) is set, the device (d2)+0 (array\_d2 [0]) remains set and the value in (d1)+0 (array\_d1 [0]) is not changed (storage).

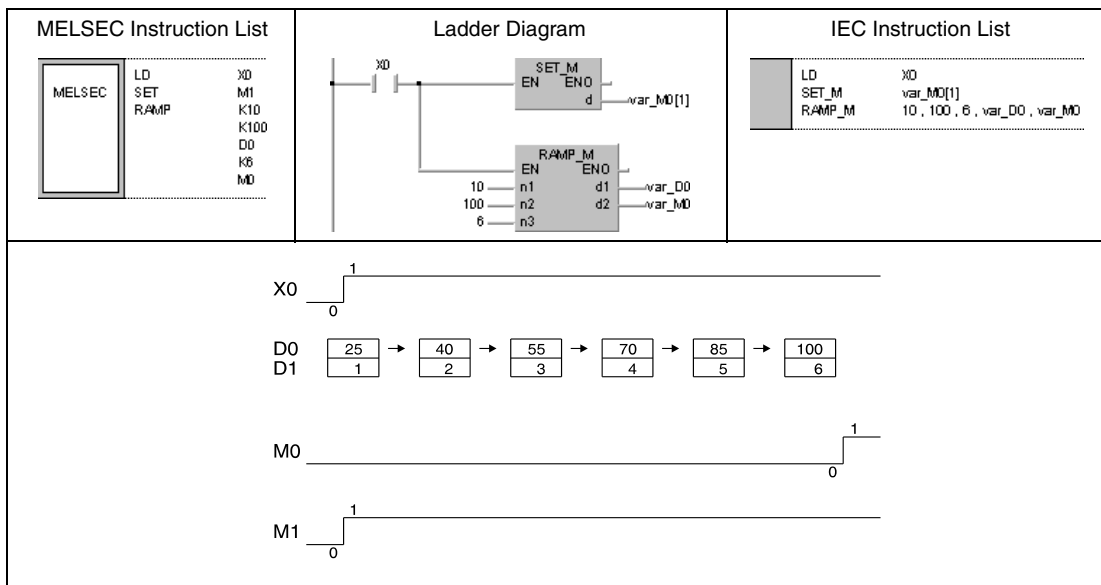
If the execution condition is reset during the operation, the content in (d1)+0 (array\_d1 [0]) does not change. If the execution condition is set once again, the RAMP instruction changes the current content in (d1)+0 (array\_d1 [0]) stored before the reset.

During the processing of the instruction the values in n1 and n2 must not be changed.

**Program Example**

**RAMP**

The following program increases the content in D0 within 6 moves from 10 to 100 and stores the content in D0 when the operation is completed.



**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



6.8.7 SPD

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

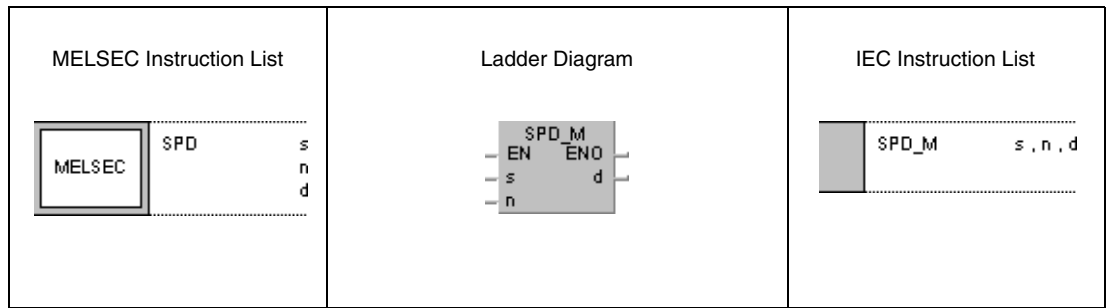
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices  
MELSEC Q

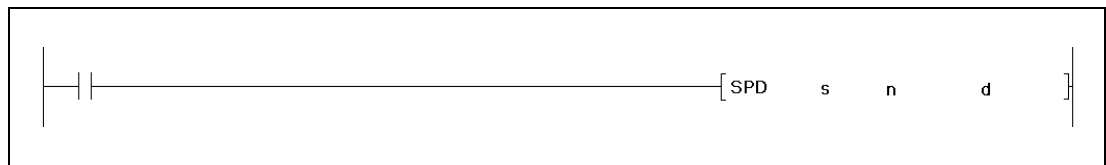
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant	Other		
	Bit	Word		Bit	Word						
s	● <sup>1</sup>	—	—	●	●	●	●	●	—	—	4
n	●	●	●	—	—	—	—	—	—	—	
d	—	●	●	●	●	●	●	●	—	—	

<sup>1</sup> Nur X

GX IEC Developer



GX Developer



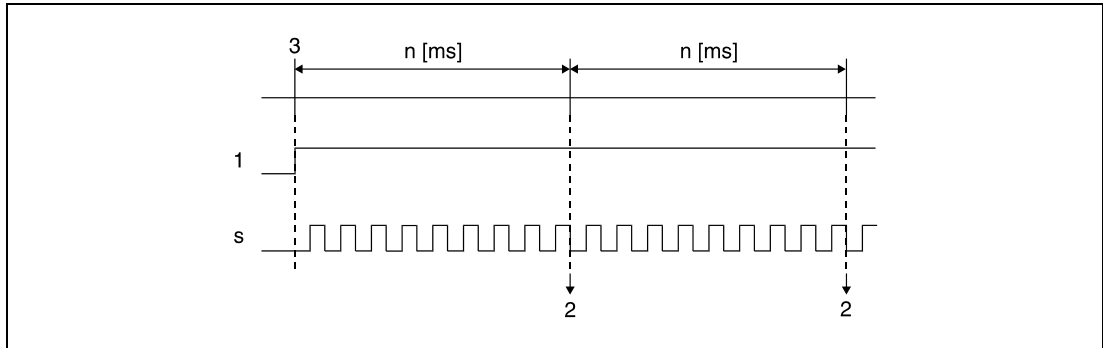
Variables

Set Data	Meaning	Data Type
s	Pulse input signal.	Bit
n	Measurement time (unit: ms).	BIN 16-bit
d	First number of device storing measurement result.	

**Functions**     **Pulse density measurement**

**SPD**     **Pulse density measurement**

The SPD instruction counts pulses at the input designated by s for a period of time specified by n. The result of the measurement is stored in d.



- <sup>1</sup> Execution condition.
- <sup>2</sup> The result of the measurement is stored in d.
- <sup>3</sup> Begin of measurement.

While the execution condition is set, the measurement begins again from 0 after the measurement time has passed. In order to stop the SPD measurement the execution condition has to be reset.

The SPD instruction stores the data from the designated devices in the CPU work area, and performs the current count operation during a 5 ms system interrupt. For this reason, the number of times the instruction can be used is limited. The SPD instructions exceeding this limit are not processed.

**Note**

The count processing for pulses used with the SPD instruction is conducted during a interrupt. Therefore, to count the pulses, it is necessary to provide their ON and OFF time as long as the interrupt time of the CPU or longer. The interrupt time is 1 ms for a System Q multi processor CPU and 5 ms for a QnA-CPU.

When a System Q CPU is used, the SPD instruction is not processed if n= 0.

When a QnA CPU is used, the SPD instruction is not processed if n= 0 or if n is not a multiple of 5.

The SPD instruction can be used as many as 6 times within all the programs being executed. The seventh and the subsequent SPD instructions are not processed.

**Program Example**

**SPD**

If X10 is set, the following program counts the pulses at X0 during a period of time of 500 ms. The result is stored in D0.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List
<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">MELSEC</div> <pre style="font-family: monospace; margin-top: 10px;"> LD      X10 SPD     X0         K500         D0                     </pre>		<pre style="font-family: monospace; margin-top: 10px;"> LD      X10 SPD_M   X0 , 500 , D0                     </pre>

### 6.8.8 PLSY

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

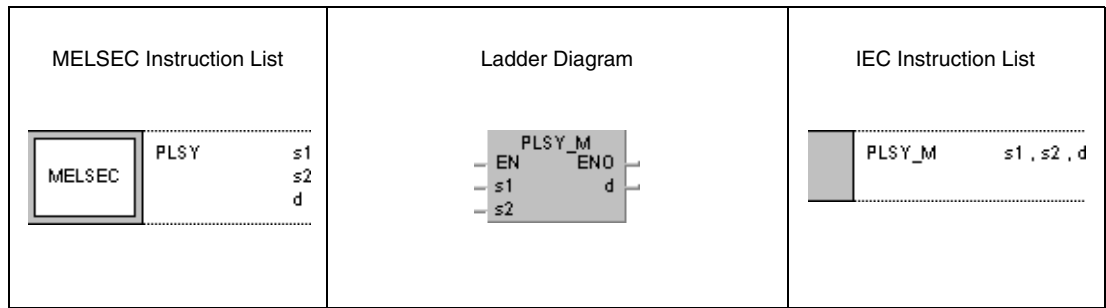
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

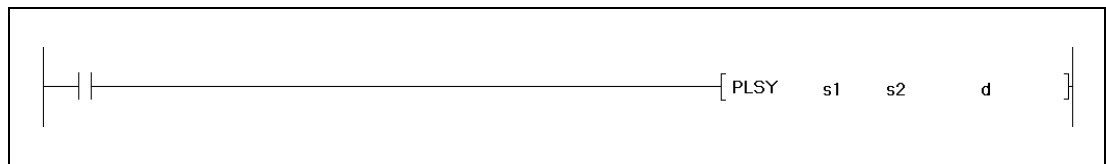
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	—	4
s2	●	●	●	●	●	●	●	●	—	—	
d	● <sup>1</sup>	—	—	—	—	—	—	—	—	—	

<sup>1</sup> Y only

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
s1	Device storing pulse frequency setting.	BIN 16-bit
s2	Device storing number of output pulses setting.	
d	Device storing output destination.	Bit

**Functions**     **Pulse output with adjustable number of pulses**

**PLSY**     **Pulse output instruction**

The PLSY instruction outputs a number of pulses specified by s2 at a frequency specified by s1 to an output designated by d.

The frequency range in s1 can be specified from 1 to 100 Hz. If the value 0 is set in s1, the PLSY instruction outputs a continuous signal.

The number of output pulses in s2 can be specified from 1 to 32767.

Only outputs corresponding to the output module can be designated by d.

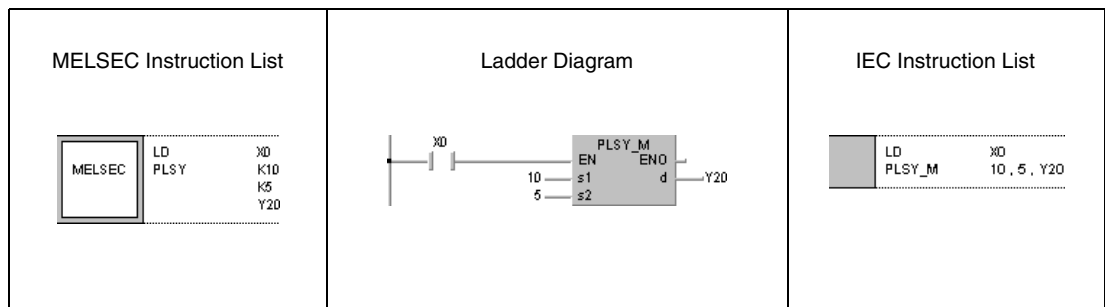
Pulse output begins with leading edge from the execution condition of the PLSY instruction. During pulse output the execution condition must not be reset. Resetting the execution condition suspends the pulse output.

The PLSY instruction stores the data from the designated devices in the CPU work area, and performs the current output operation during a system interrupt. For this reason, the PLSY instruction can only be used once in a program. The interrupt time is 1 ms for a System Q multi processor CPU and 5 ms for a QnA CPU.

**Program Example**

PLSY

If X0 is set, the following program outputs five 10 Hz pulses to Y20.



**6.8.9 PWM**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

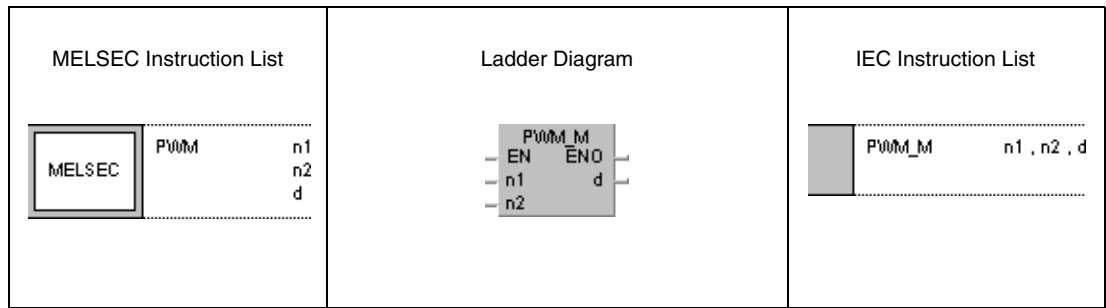
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

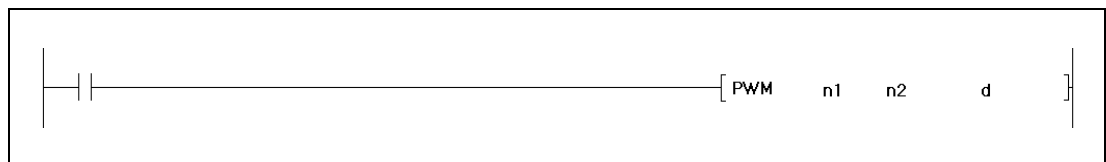
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant	Other		
	Bit	Word		Bit	Word						
n1	●	●	●	●	●	●	●	●	—	—	4
n2	●	●	●	●	●	●	●	●	—	—	
d	● <sup>1</sup>	—	—	—	—	—	—	—	—	—	

<sup>1</sup> Nur Y

**GX IEC  
Developer**



**GX  
Developer**



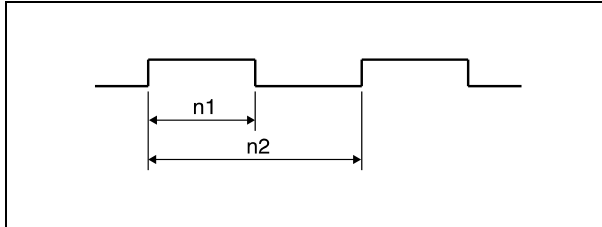
**Variables**

Set Data	Meaning	Data Type
n1	Number of device storing ON time setting.	BIN 16-bit
n2	Number of device storing cycle time setting.	
d	Number of device storing output destination.	Bit

**Functions**     **Pulse width modulation**

**PWM     Modulation instruction**

The PWM instruction outputs pulses at a cycle time specified by n2 and with an ON time specified by n1 to an output designated by d.



The times in n1 and n2 can be specified from 1 to 65535 ms when a multi processor CPU of the System Q is used. When a QnA CPU is used, the range for the values in n1 and n2 goes from 5 ms to 65535 ms. The value set in n1 has to be less than that in n2.

**Notes**

*The PWM instruction registers the data from the designated devices in the work area of the CPU, and performs the current output operation during a system interrupt (1 ms for System Q CPUs, 5 ms for QnA CPUs). For this reason, the PWM instruction can only be used once in a program.*

*The instruction is not processed in the following cases:*

- When both n1 and n2 are 0
- When n2 is smaller or equal to n1
- When n1 and n2 are not multiples of 5 (only when a QnA CPU is used)

**Program Example**

**PWM**

If X0 is set, the following program outputs pulses at a cycle time of 1 second and with an ON time of 100 ms to Y20.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: center;">MELSEC</td> <td style="width: 10%; text-align: center;">LD</td> <td style="width: 10%; text-align: center;">X0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td></td> <td style="text-align: center;">PWM</td> <td style="text-align: center;">K100</td> <td style="text-align: center;">K1000</td> <td style="text-align: center;">Y20</td> <td></td> <td></td> <td></td> </tr> </table>	MELSEC	LD	X0							PWM	K100	K1000	Y20				<p>Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%; text-align: center;">LD</td> <td style="width: 10%; text-align: center;">X0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td></td> <td style="text-align: center;">PWM_M</td> <td style="text-align: center;">100</td> <td style="text-align: center;">1000</td> <td style="text-align: center;">Y20</td> <td></td> <td></td> <td></td> </tr> </table>		LD	X0							PWM_M	100	1000	Y20			
MELSEC	LD	X0																																
	PWM	K100	K1000	Y20																														
	LD	X0																																
	PWM_M	100	1000	Y20																														

6.8.10 MTR

CPU

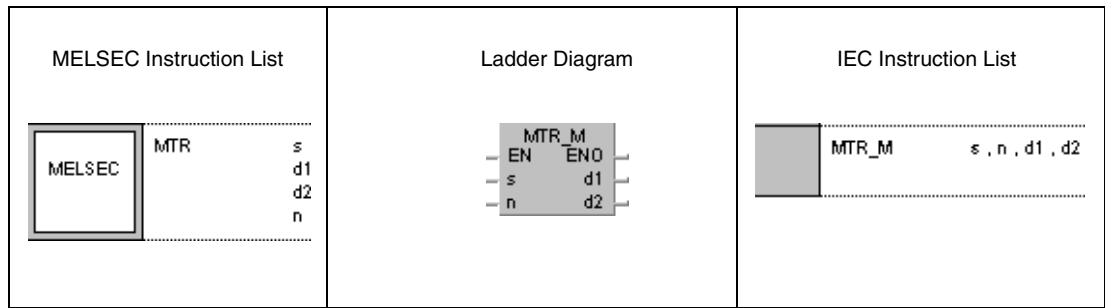
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

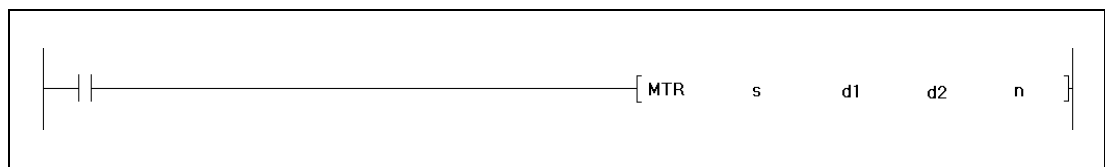
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other		
	Bit	Word		Bit	Word						
s	●	—	—	—	—	—	—	—	—	SM0	5
d1	●	—	—	—	—	—	—	—			
d2	●	—	—	—	—	—	—	—			
n	●	●	●	●	●	●	●	—			

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
s	Initial input device.	Bit
d1	Initial output device.	
d2	First number of device storing matrix input data.	
n	Number of input rows.	BIN 16-bit

**Functions**      **Building an input matrix****MTR**      **Instruction for reading n data rows into an input matrix.**

The MTR instruction reads the information of 16 bits (0/1) beginning from the device designated by s. The number of repetitions (rows) is designated by n. The conditions of read data are stored in the device designated by d2 onwards. This way, a matrix of 16 bits and n rows is built.

One row (16 bits) can be read each scan.

The reading process is continuously repeated from the first to the nth row.

Due to the format of the input matrix (16 bits x n rows) the device designated by d2 has to supply space for 16 bits x n rows either to store the data.

Each row is selected beginning with the output designated by d1. The corresponding output for each row of 16 bits to be read is set or reset by the system automatically. The number of outputs is identical with the number of rows. Thus, each single row can be addressed accurately by the system

The device numbers designated by s, d1, and d2 must be divisible by 16.

The number of rows n can be designated from 2 to 8.

Note, that the MTR instruction directly operates on current input and output data.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

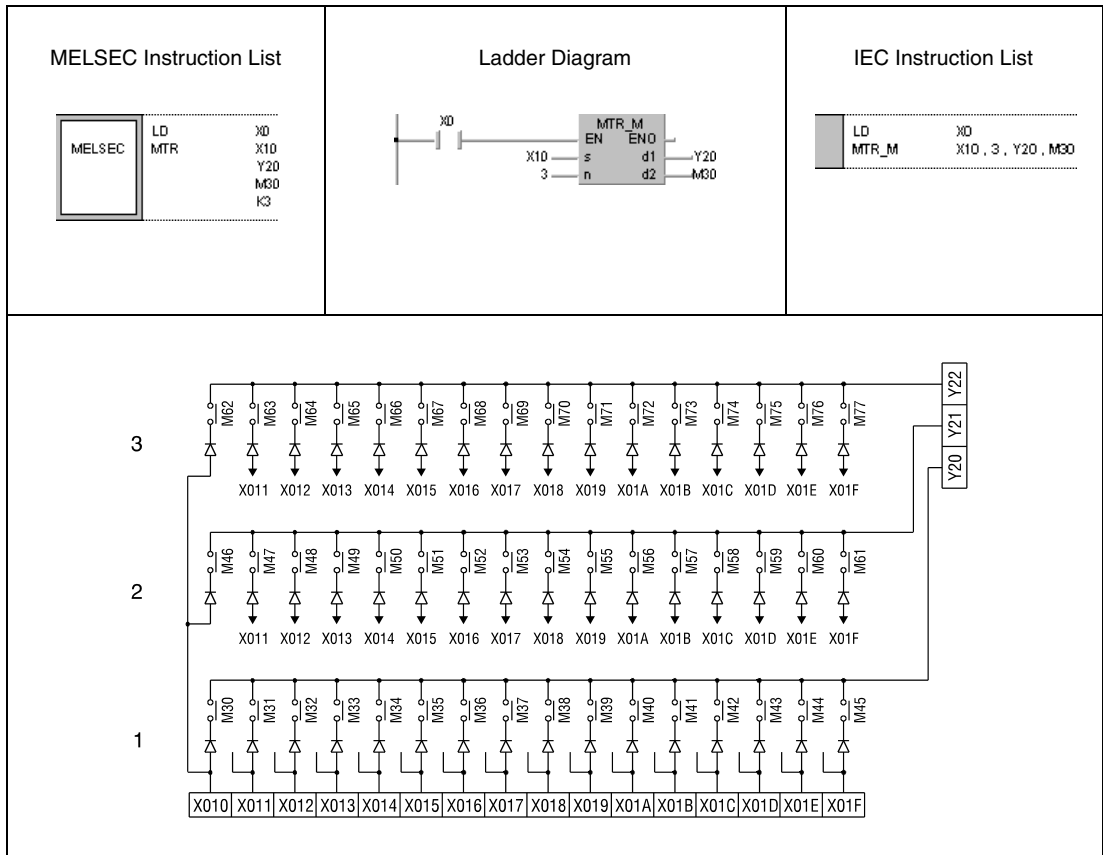
- The device numbers designated by s, d1, and d2 are not divisible by 16 (error code: 4101).
- The device designated by s exceeds the current input range (error code: 4101).
- The device designated by s exceeds the current output range (error code: 4101).
- The matrix space 16 bits x n rows exceeds the relevant device range of d2 (error code: 4101).
- The value in n does not range within 2 and 8 (error code: 4100).



**Program Example**

**MTR**

If X0 is set, the following program reads the inputs X10 through X1F three times and stores the results in M30 through M77. A matrix is built with 16 bits x 3 rows. The rows are addressed via the outputs Y20 through Y22.



- 1 1st row
- 2 2nd row
- 3 3rd row



# 7 Application Instructions, Part 2

The application instructions, part 2 are specific instructions for several special functions. The following table shows the division of these functions:

Instruction	Meaning
Logical operation instructions	Logical AND / OR, logical exclusive OR / exclusive NOR
Rotation instructions	16-bit and 32-bit data right / left rotation
Shift instructions	Shift data by bit or word
Bit processing instructions	Set, reset, and test bits
Data processing instructions	Search, encode, and decode data at specified devices Disunite and unite data
Structured program instructions	Repeated operation, subroutine program calls, subroutine calls between program files, switching between main and subprogram parts, micro computer program calls, index qualification of entire ladders, store index qualification values in data tables
Data table operation instructions	Write to and read data from a data table, delete and insert data blocks in a data table
Buffer memory access instructions	Buffer memory access of special function modules or remote modules
Display instructions	Output ASCII characters to the outputs of a module or to an LED display
Debugging and failure diagnosis instructions	Failure checks, setting and resetting status latch, sampling trace, program trace
Character string processing instructions	Character string (ASCII code) processing
Special function instructions	Trigonometrical functions, square root and exponential calculation with BCD data and floating point data
Data control instructions	Upper and lower limit control and storage of checked data
File register switching instructions	Switching between file register blocks and files
Clock instructions	Writing and reading clock data
Peripheral device instructions	Message output and key input on peripheral units
Program instructions	Select different program execution modes
Other instructions	Reset watchdog timer (WDT), set and reset carry, pulse generation, direct read from indirect access file registers, numerical key input from keyboard, batch save or recovery of index registers, write to EEPROM file registers

## 7.1 Logical operation instructions

Via the logical operation instructions logical connections such as logical sum or logical product are programmed.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
AND (logical product)	WAND	WAND_M, WAND_3_M
	WANDP	WANDP_M, WANDP_3_M
	DAND	DAND_M, DAND_3_M
	DANDP	DANDP_M, DANDP_3_M
	BKAND	BKAND_M
	BKANDP	BKANDP_M
OR (logical sum)	WOR	WOR_M, WOR_3_M
	WORP	WORP_M, WORP_3_M
	DOR	DOR_M, DOR_3_M
	DORP	DORP_M, DORP_3_M
	BKOR	BKOR_M
	BKORP	BKORP_M
Exclusive OR (XOR)	WXOR	WXOR_M, WXOR_3_M
	WXORP	WXORP_M, WXORP_3_M
	DXOR	DXOR_M, DXOR_3_M
	DXORP	DXORP_M, DXORP_3_M
	BKXOR	BKXOR_M
	BKXORP	BKXORP_M
Exclusive NOR (XNR)	WXNR	WXNR_M, WXNR_3_M
	WXNRP	WXNRP_M, WXNRP_3_M
	DXNR	DXNR_M, DXNR_3_M
	DXNRP	DXNRP_M, DXNRP_3_M
	BKXNR	BKXNR_M
	BKXNRP	BKXNRP_M

Logical instructions are processed bit by bit as binary data. The two conditions (0 and 1) are connected and the result of the connection is output to a destination address.

**NOTE**

*Within the IEC editors please use the IEC instructions.*

The following table shows the logical connection results of the conditions 0 and 1. A and B are input variables and Y is the output variable.

Logical Connection	Processing Details	Operation Expression	Example		
			A	B	Y
Logical AND	Output Y set to 1, only if both inputs A <b>and</b> B are set to 1.	$Y = A \times B$	0	0	0
			0	1	0
			1	0	0
			1	1	1
Logical OR	Output Y set to 1, if at least one of the inputs A <b>or</b> B is set to 1.	$Y = A + B$	0	0	0
			0	1	1
			1	0	1
			1	1	1
Logical exclusive OR (XOR)	Output Y set to 1, if the inputs A and B are different, and is set to 0 if A and B are equal.	$Y = \bar{A} \times B + A \times \bar{B}$	0	0	0
			0	1	1
			1	0	1
			1	1	0
Logical exclusive NOR (XNR)	Output Y set to 1, if the inputs A and B are equal, and is set to 0, if A and B are different.	$Y = (\bar{A} + B) (A + \bar{B})$	0	0	1
			0	1	0
			1	0	0
			1	1	1

7.1.1 WAND, WANDP, DAND, DANDP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag										
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level																
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N	M9012	M9010 M9011					
<b>WAND</b>																														
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							K1 ↓ K4	5 ● <sup>1</sup>	●	●			
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●													7 ● <sup>1</sup>	●	●
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							9 ↓ K8	9 ● <sup>1</sup>	●	●			
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●													
d1		●	●	●	●	●	●	●	●	●	●	●	●	●	●									1	●	●				
<b>DAND</b>																														
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							K1 ↓ K8	9 ● <sup>1</sup>	●	●		
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							1					●	●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other			
Bit	Word		Bit	Word							
<b>WAND</b>											
s	●	●	●	●	●	●	●	●	—	—	3
d	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	—	—	4
s2	●	●	●	●	●	●	●	●	—	—	
d1	●	●	●	●	●	●	●	—	—	—	
d	●	●	●	●	●	●	●	—	—	—	
<b>DAND</b>											
s	●	●	●	●	●	●	●	●	—	—	4 <sup>1)</sup>
d	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	—	—	4 <sup>2)</sup>
s2	●	●	●	●	●	●	●	●	—	—	
d	●	●	●	●	●	●	●	—	—	—	

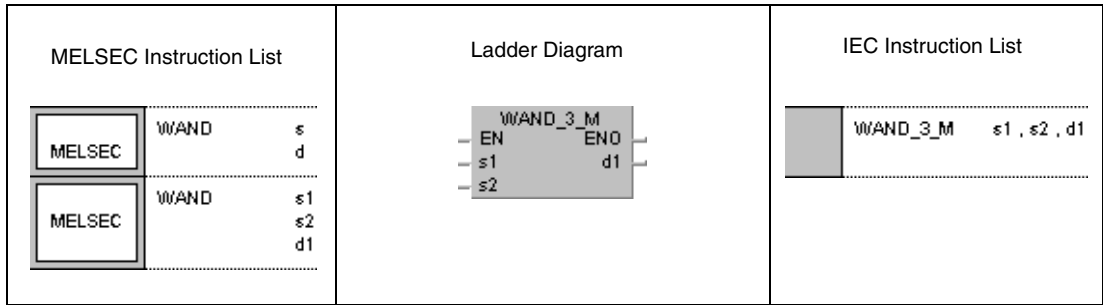
<sup>1</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU is used: 4
- If a single processor System Q CPU is used: 3
- If a multi processor System Q CPU is used with internal word devices (except for file register ZR) or constants: 6
- If a multi processor System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6
- If a multi processor System Q CPU is used with devices other than above mentioned: 4

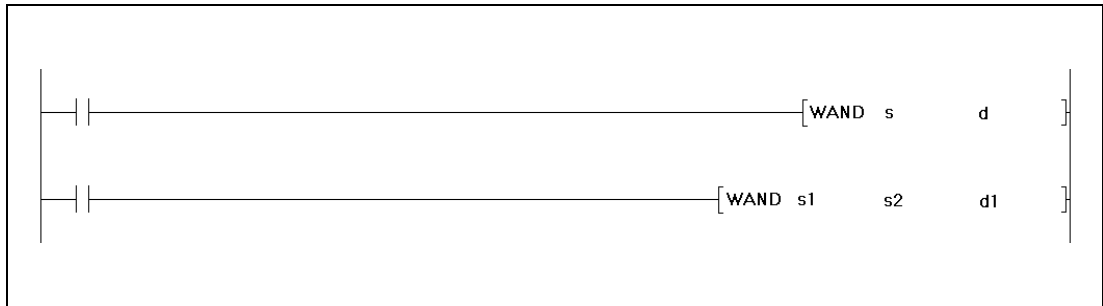
<sup>2</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU is used: 4
- If a System Q CPU is used with internal word devices (except for file register ZR) or constants: 6
- If a System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6
- If a System Q CPU is used with devices other than above mentioned: 4

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	Data for logical product, or first number of device storing such data.	BIN 16-/32-bit
d		
s1	Data for logical product, or first number of device storing such data.	
s2		
d1 (for DAND d)	First number of device storing result of logical operation.	

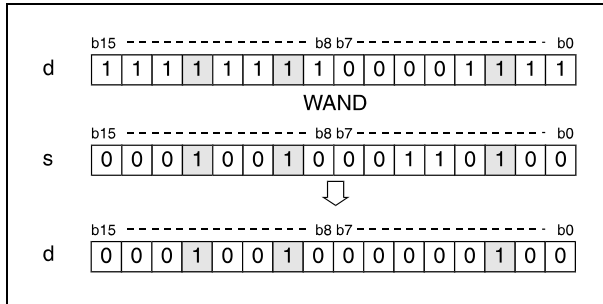
**Functions Logical AND**

**WAND 16-bit data**

The logical AND forms the logical product of two input variables.

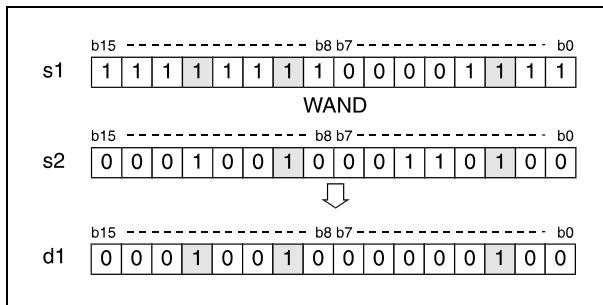
● Variation 1:

16-bit data designated by s and d form the logical product bit by bit. The result is output to the device designated by d.



● Variation 2:

16-bit data designated by s1 and s2 form the logical product bit by bit. The result is output to the device designated by d1.



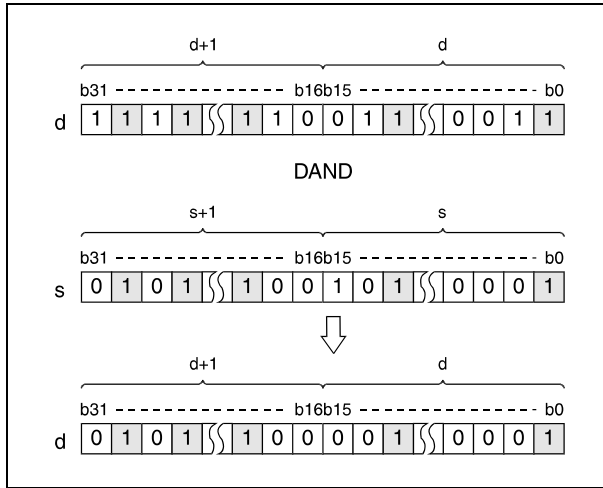
Bits exceeding the digit designation are set to 0. For example, if the digit designation is specified by K2, the higher 8 bits (b8 through b15) are processed as 0.



**DAND 32-bit data**

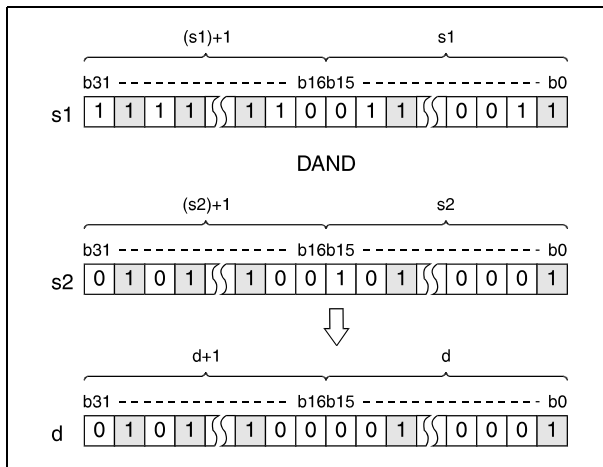
● Variation 1:

32-bit data designated by s and d form the logical product bit by bit. The result is output to the device designated by d.



● Variation 2 (Q series and System Q):

32-bit data designated by s1 and s2 form the logical product bit by bit. The result is output to the device designated by d.

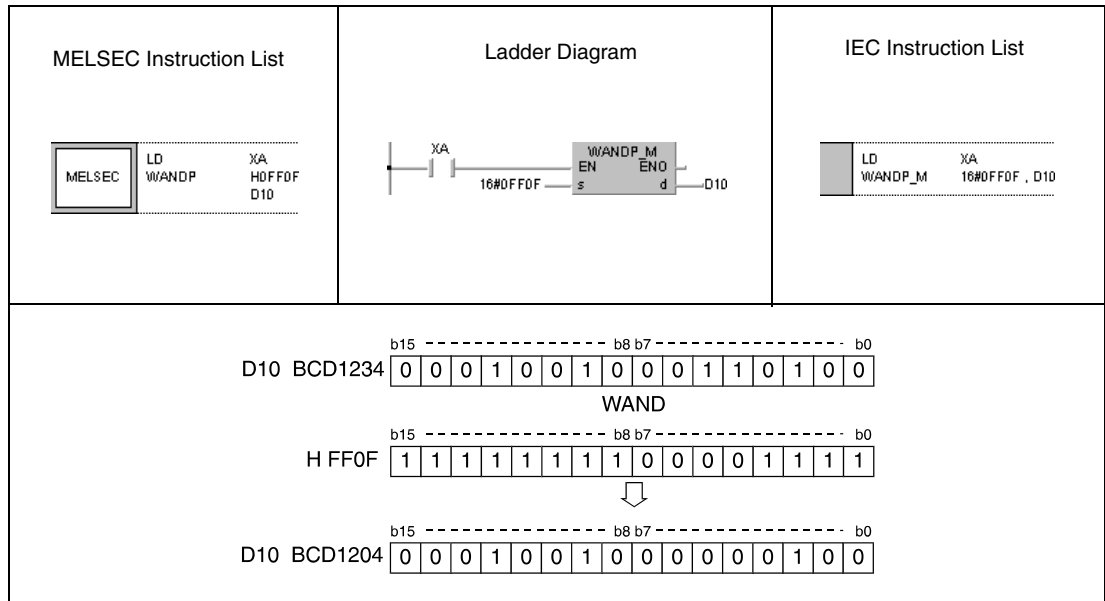


After executing the connection, all bits exceeding the digit designation are set to 0.

**Program Example 1**

WANDP (s, d)

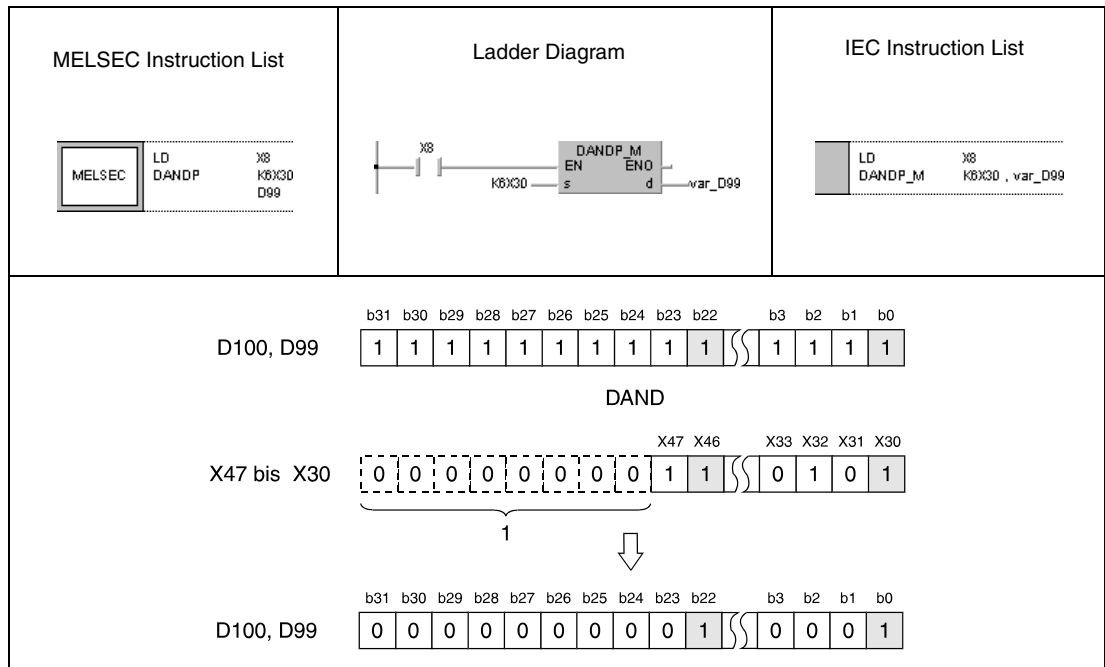
With leading edge from XA, the following program sets the digit of tens (b5-b7) in the BCD 4-digit value in D10 to 0. The result is stored again in D10.



**Program Example 2**

DANDP (s, d)

With leading edge from X8, the following program forms the logical product of 32-bit data in D99 and D100 and 24-bit data at X30 through X47. The result is stored again in D99 and D100.

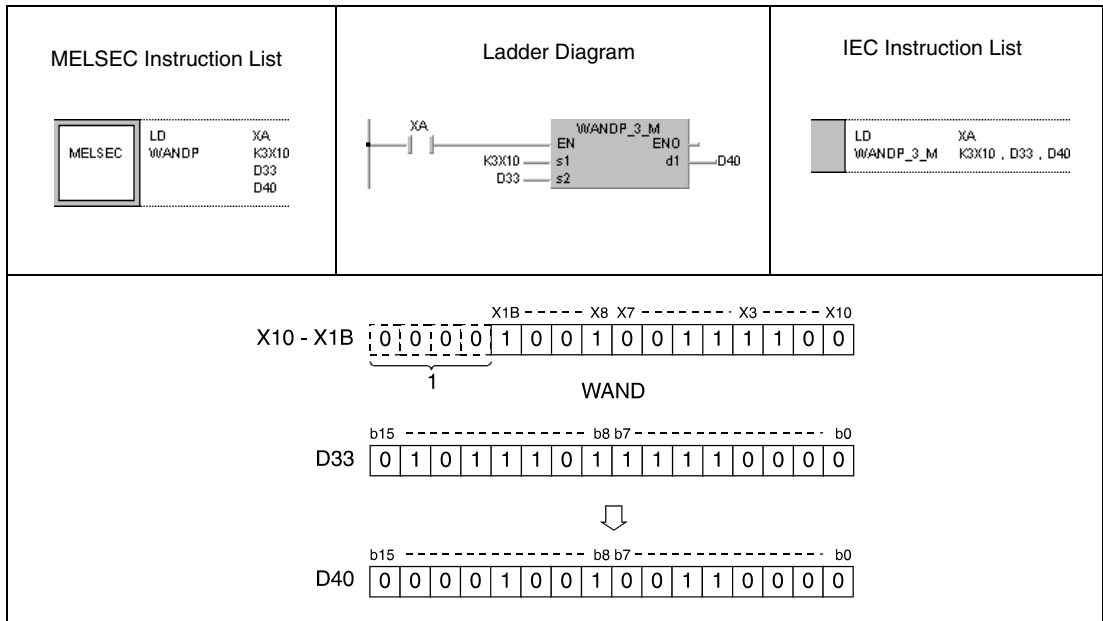


<sup>1</sup> These bits are set to 0.

**Program Example 3**

WANDP (s1, s2, d1)

With leading edge from XA, the following program forms the logical product of data in X10 through X1B and data in D33. The result is stored in D40.

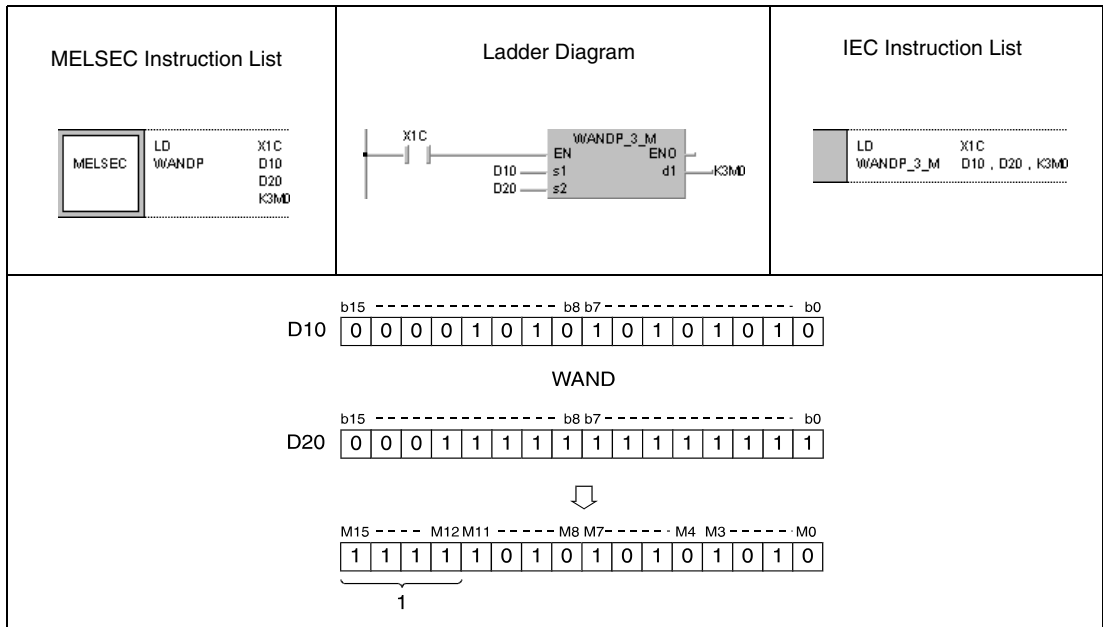


<sup>1</sup> These bits are set to 0.

**Program Example 4**

WANDP (s1, s2, d1)

With leading edge from X1C, the following program forms the logical product of data in D10 and D20. The result is stored in M0 through M11.

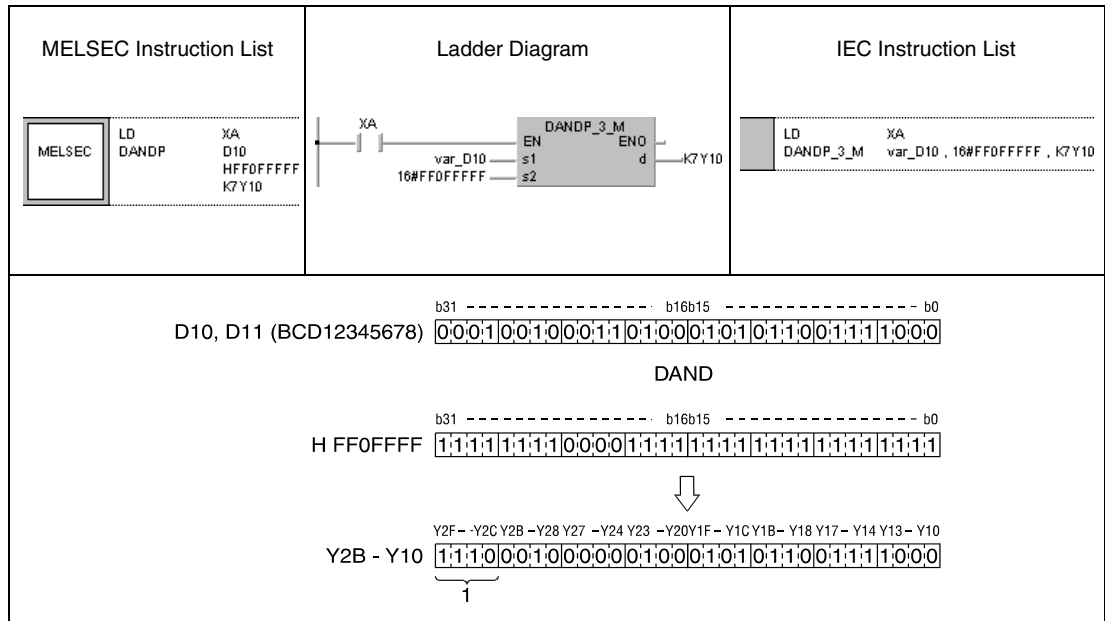


<sup>1</sup> These bits remain unchanged.

**Program Example 5**

DANDP (s1, s2, d)

With leading edge from XA, the following program sets the digit of hundreds in the BCD 4-digit value in D10 and D11 to 0. The result is output at Y10 through Y2B.



<sup>1</sup> These bits remain unchanged

**NOTE**

The program examples 2 and 5 will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

**7.1.2 BKAND, BKANDP**

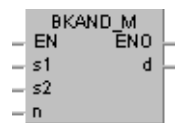
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

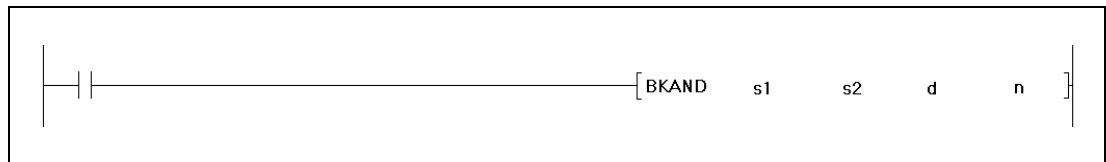
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	SM0	5	
s2	—	●	●	—	—	—	●	—			
d	—	●	●	—	—	—	—	—			
n	●	●	●	●	●	●	●	—			

**GX IEC Developer**

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">MELSEC</td> <td style="border: none;">BKAND</td> <td style="border: none;">s1 s2 d n</td> </tr> </table>	MELSEC	BKAND	s1 s2 d n	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="border: none;">BKAND_M</td> <td style="border: none;">s1, s2, n, d</td> </tr> </table>	BKAND_M	s1, s2, n, d
MELSEC	BKAND	s1 s2 d n					
BKAND_M	s1, s2, n, d						

**GX Developer**



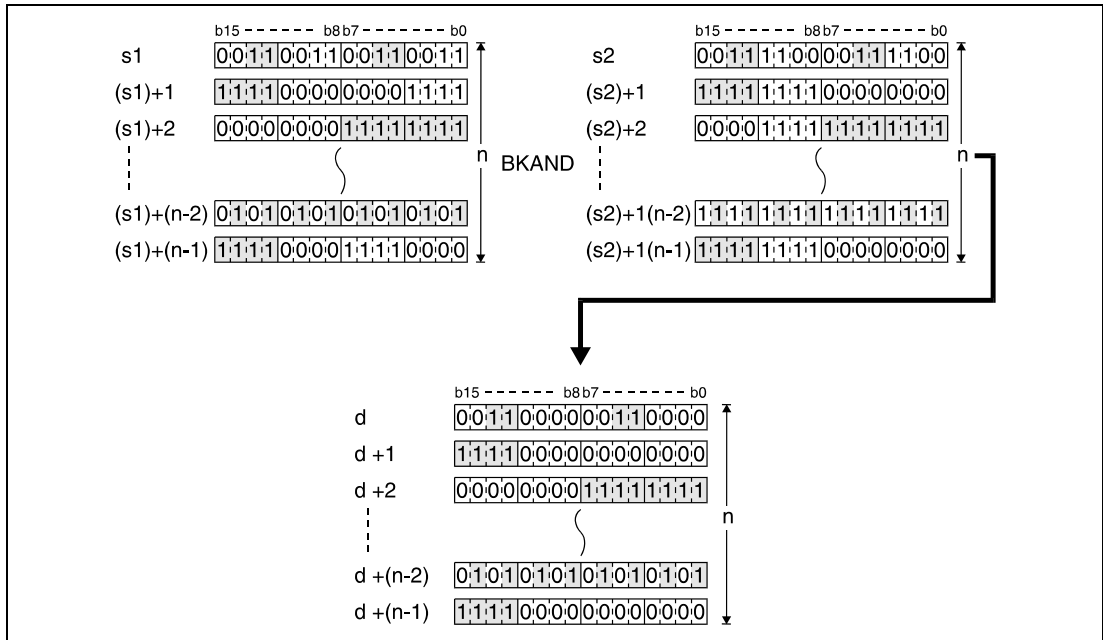
**Variables**

Set Data	Meaning	Data Type
s1	First number of device storing data for logical product.	BIN 16-bit
s2	First number of data or first number of device storing data for logical operation.	
d	First number of device storing result of logical operation.	
n	Number of data blocks forming the logical product.	

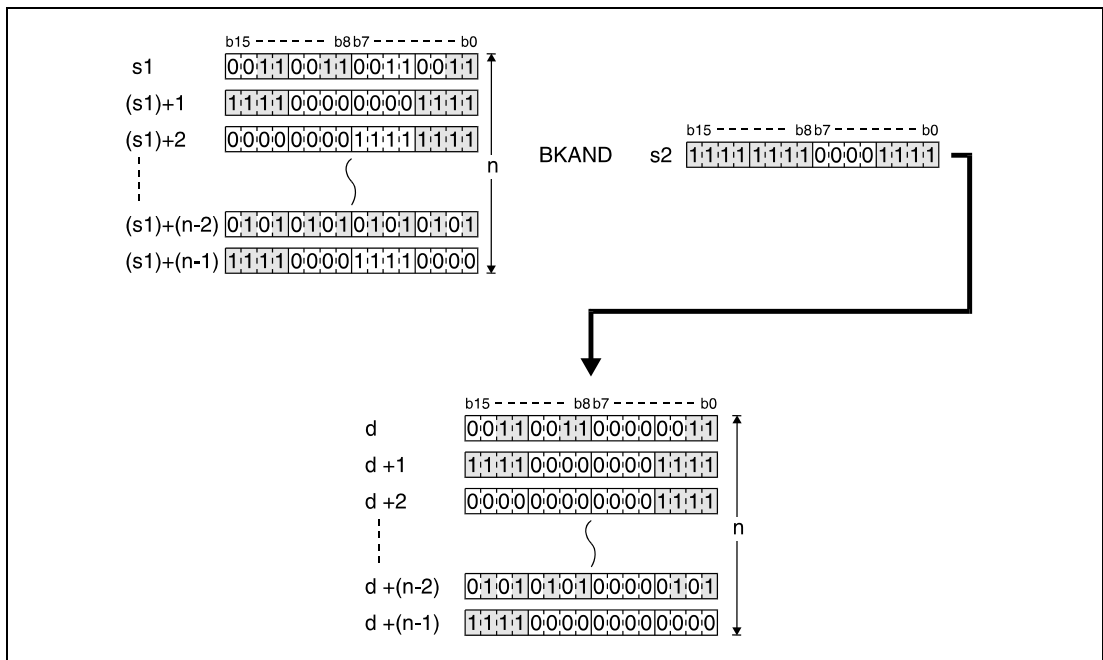
**Functions Forming a logical product with 16-bit data blocks**

**BKAND Forming a logical product with data blocks**

The BKAND instruction forms the logical product beginning with the nth 16-bit data block from s1 onwards and with the nth 16-bit data block from s2 onwards. The according 16-bit block of the result is stored beginning from device d onwards. The number of blocks to be processed is specified by n.



The constant in s2 must range within -32768 and 32767.



**Operation Errors**

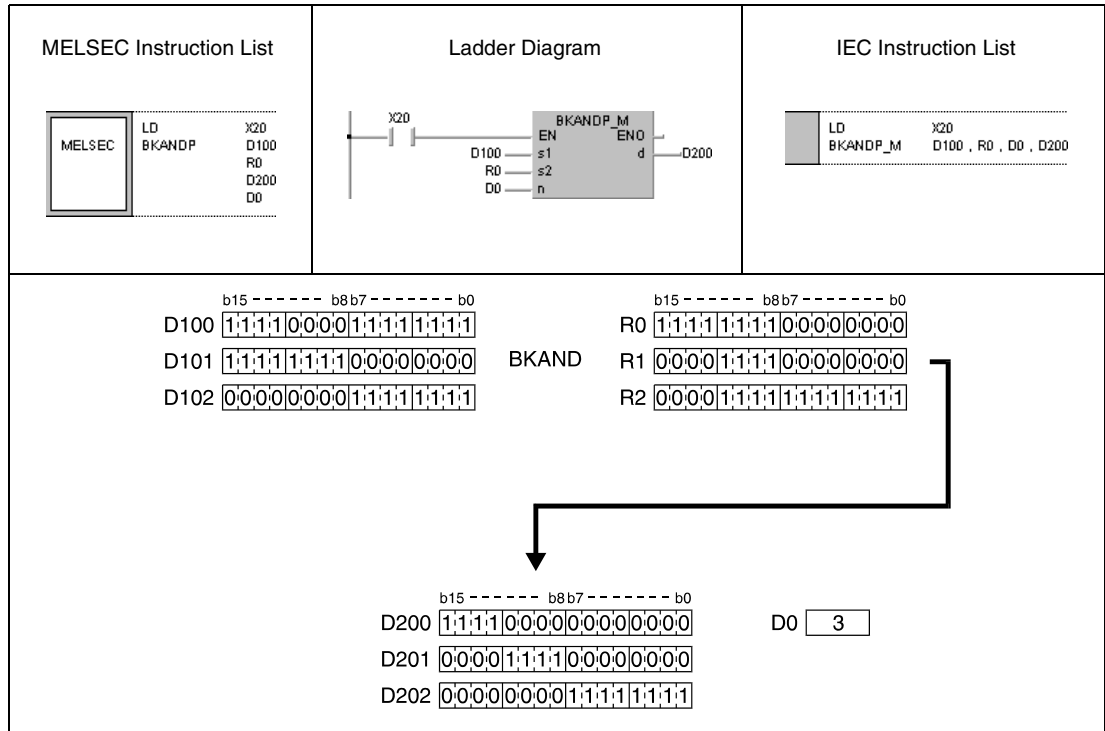
In the following cases an operation error occurs and the error flag is set:

- The number of data blocks determined by n exceeds the storage device numbers designated by s1, s2, or d (error code: 4101).
- The storage device numbers designated by s1, s2, or d overlap (error code: 4101).

**Program Example**

**BKANDP**

With leading edge from X20, the following program forms the logical product of data in registers D100 through D102 and data in registers R0 through R2. The result is stored in registers D200 through D202. The number of 16-bit data blocks (3) to be processed is stored in D0.



7.1.3 WOR, WOPR, DOR, DORP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag			
Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I	N	M9012
<b>WOR</b>																								
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K4	5	●	●
d																						1		
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K8	7	●	●
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					1		
d1																								
<b>DOR</b>																								
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K8	9	●	●
d																						1		

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

Devices  
MELSEC Q

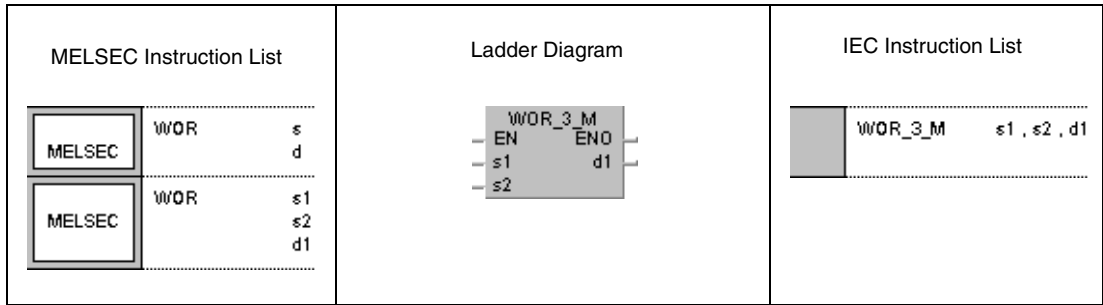
Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word				U			
<b>WOR</b>											
s	●	●	●	●	●	●	●	●	—	—	3
d	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	—	—	4
s2	●	●	●	●	●	●	●	●	—	—	
d1	●	●	●	●	●	●	●	—	—	—	
<b>DOR</b>											
s	●	●	●	●	●	●	●	●	—	—	4 <sup>1)</sup>
d	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	—	—	4 <sup>2)</sup>
s2	●	●	●	●	●	●	●	●	—	—	
d	●	●	●	●	●	●	●	—	—	—	

<sup>1</sup> The number of steps depends on the device and the type of CPU.  
 If a QnA-CPU is used: 4  
 If a single processor System Q CPU is used: 3  
 If a multi processor System Q CPU is used with internal word devices (except for file register ZR) or constants: 6  
 If a multi processor System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6  
 If a multi processor System Q CPU is used with devices other than above mentioned: 4

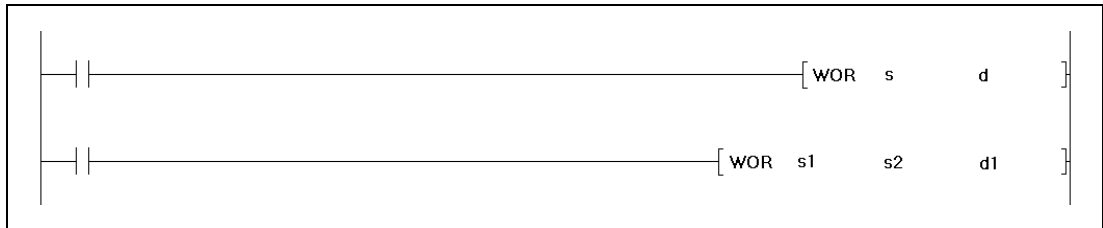
<sup>2</sup> The number of steps depends on the device and the type of CPU.  
 If a QnA-CPU is used: 4  
 If a System Q CPU is used with internal word devices (except for file register ZR) or constants: 6  
 If a System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6  
 If a System Q CPU is used with devices other than above mentioned: 4



**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	Data for logical sum, or first number of device storing such data.	BIN 16-/32-bit
d		
s1	Data for logical sum, or first number of device storing such data.	
s2		
d1 (for DOR d)	First number of device storing result of logical operation.	

**Functions**

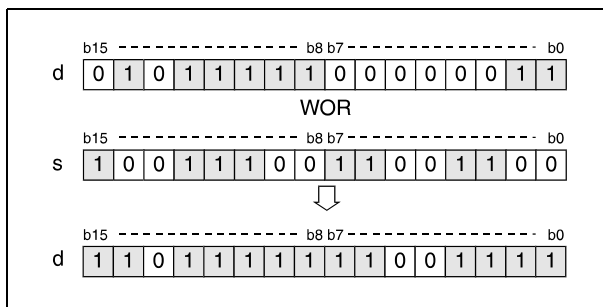
**Logical OR**

**WOR 16-bit data**

The logical OR forms the logical sum of two input variables.

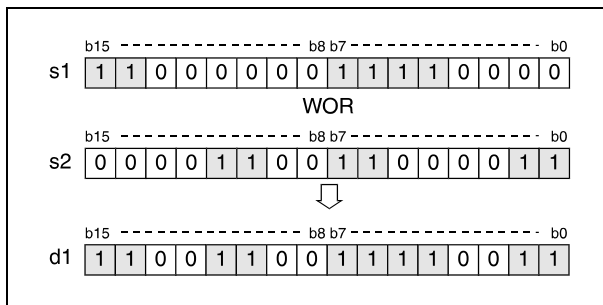
● Variation 1:

16-bit data designated by s and d are added bit by bit. The result is output to the device designated by d.



● Variation 2:

16-bit data designated by s1 and s2 are added bit by bit. The result is output to the device designated by d1.

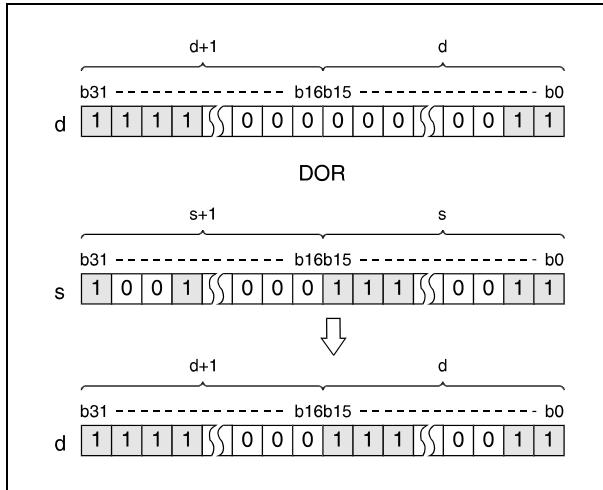


Bits exceeding the digit designation are set to 0. For example, if the digit designation is specified by K2, the higher 8 bits (b8 through b15) are processed as 0.

**DOR 32-bit data**

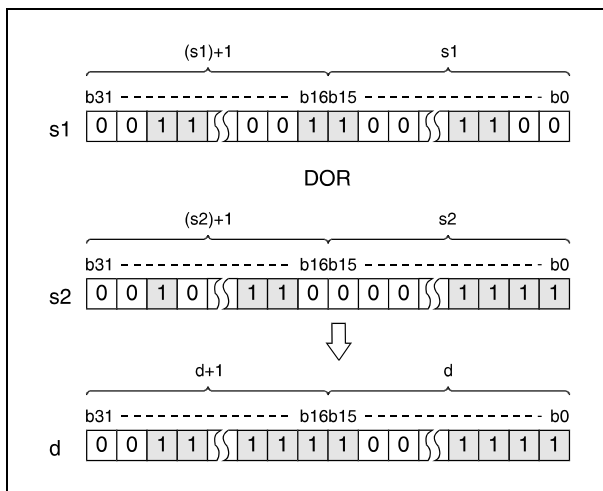
● Variation 1:

32-bit data designated by s and d are added bit by bit. The result is output to the device designated by d.



● Variation 2 (Q series and System Q):

32-bit data designated by s1 and s2 are added bit by bit. The result is output to the device designated by d.

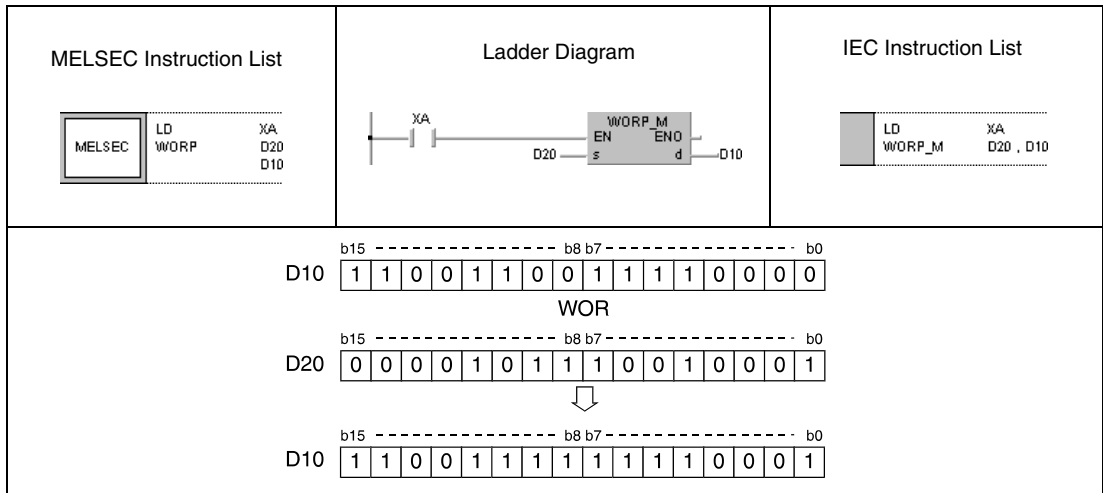


After executing the connection, all bits exceeding the digit designation are set to 0.

**Program Example 1**

WORP (s, d)

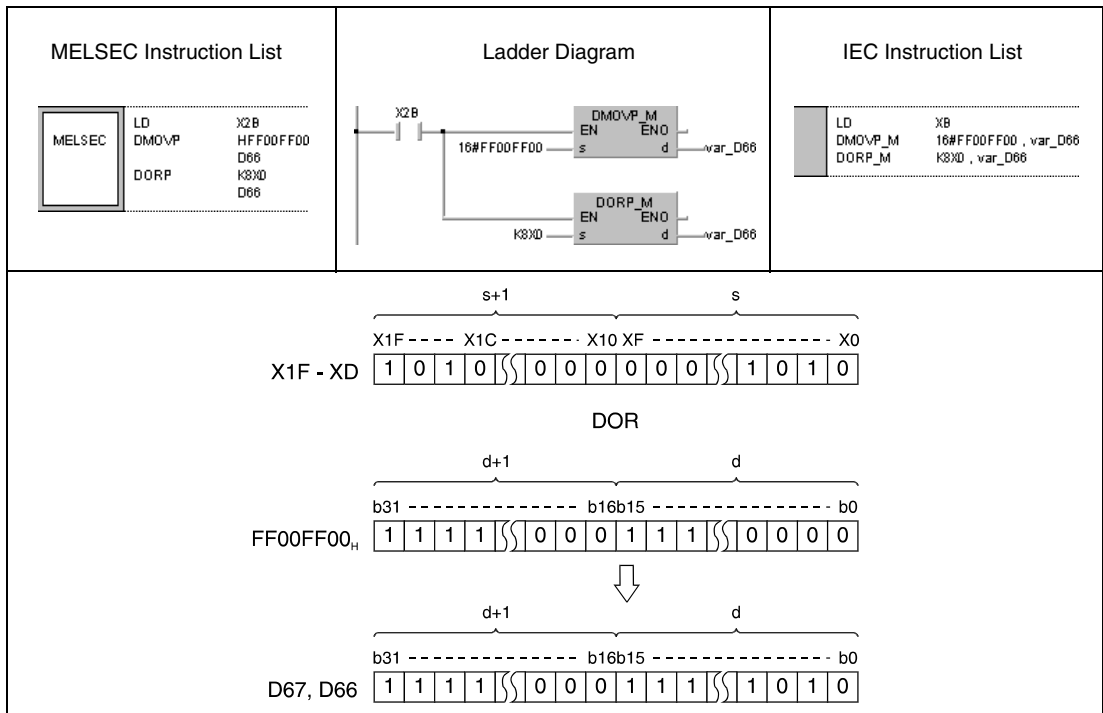
With leading edge from XA, the following program adds data in D10 to data in D20. The result is stored in D10.



**Program Example 2**

DORP (s, d)

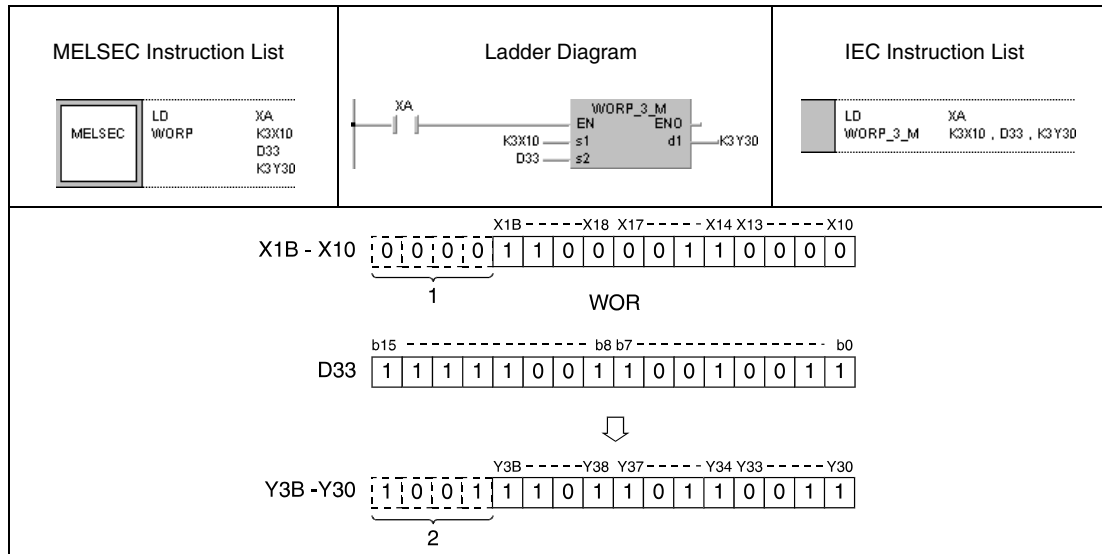
With leading edge from X2B, the following program adds data at the inputs X0 through X1F to a hexadecimal value FF00FF00. The result is stored in D66 and D67.



**Program Example 3**

WORP (s1, s2, d1)

With leading edge from XA, the following program adds data at the inputs X10 through X1B to data in D33. The result is output to the outputs Y30 through Y3B.

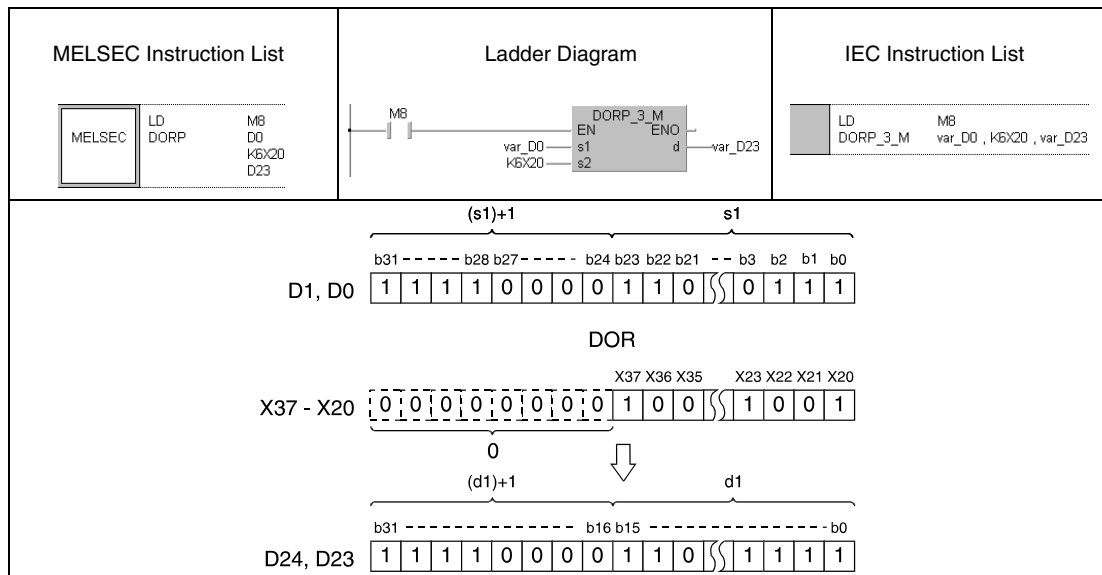


- 1 These bits are set to 0
- 2 These bits remain unchanged

**Program Example 4**

DORP (s1, s2, d)

With leading edge from M8, the following program adds 32-bit data in D0 and D1 to 24-bit data at the inputs X20 through X37. The result is stored in D23 and D24.



**NOTE**

The program examples 2 and 4 will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

7.1.4 BKOR, BKORP


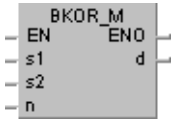
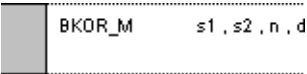
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

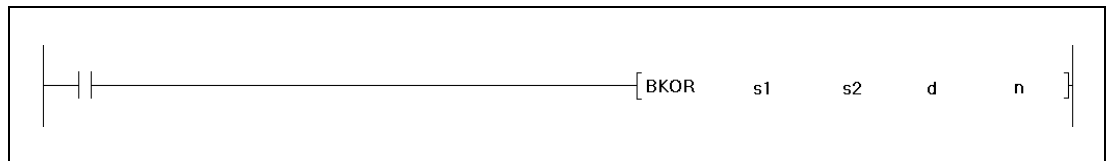
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	5
s2	—	●	●	—	—	—	—	●	—		
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	---

GX Developer



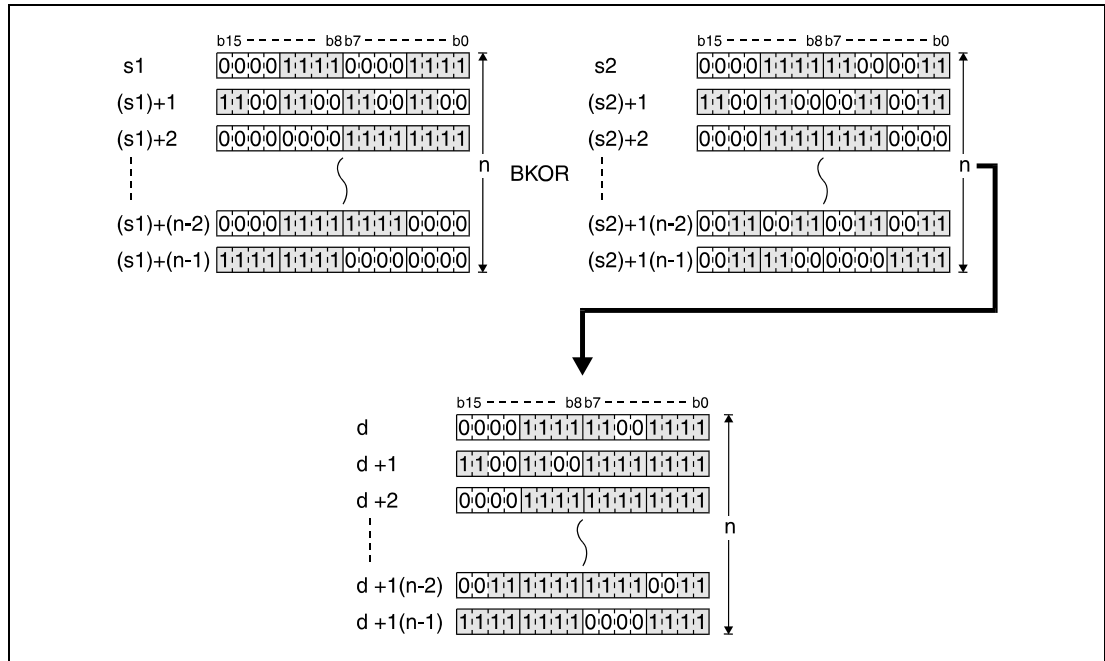
Variables

Set Data	Meaning	Data Type
s1	First number of device storing data for logical sum.	BIN 16-bit
s2	First number of data, or first number of device storing data for logical sum.	
d	First number of device storing result of logical operation.	
n	Number of data blocks forming the logical sum.	

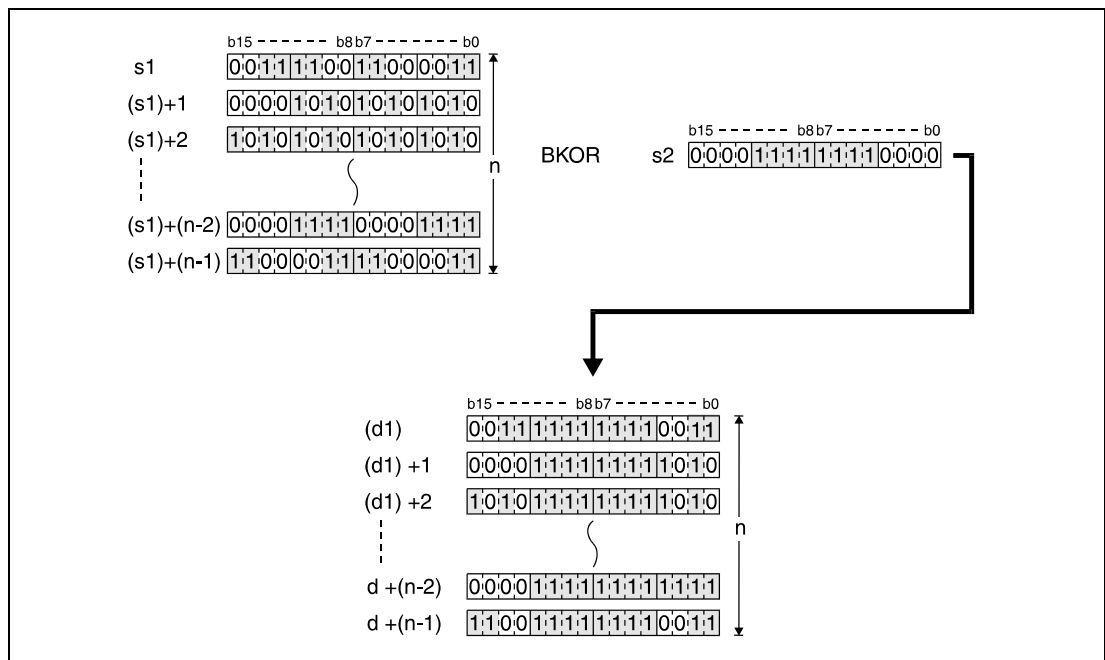
**Functions Forming a logical sum with 16-bit data blocks**

**BKOR Forming a logical sum with data blocks**

The BKOR instruction forms the logical sum beginning with the nth 16-bit data block from s1 onwards and with the nth 16-bit data block from s2 onwards. The according 16-bit block of the result is stored beginning from device d onwards. The number of blocks to be processed is specified by n.



The constant in s2 must range within -32768 and 32767.



**Operation Errors**

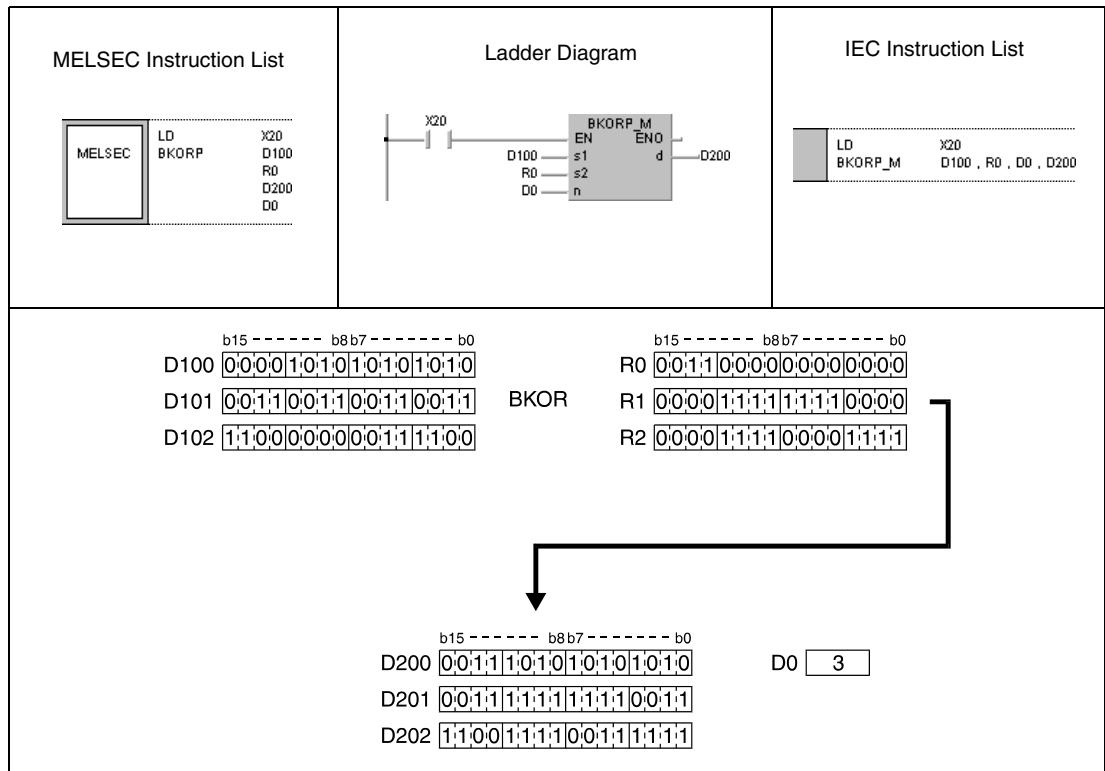
In the following cases an operation error occurs and the error flag is set:

- The number of data blocks determined by n exceeds the storage device numbers designated by s1, s2, or d (error code: 4101).
- The storage device numbers designated by s1, s2, or d overlap (error code: 4101).

**Program Example**

**BKORP**

With leading edge from X20, the following program forms the logical sum of data in registers D100 through D102 and data in registers R0 through R2. The result is stored in registers D200 through D102. The number of 16-bit data blocks (3) to be processed is stored in D0.





7.1.5 WXOR, WXORP, DXOR, DXORP

CPU

AnS	AnN	AnA(S)	AnU	QnAS, Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag					
Bit Devices				Word Devices (16-bit)								Constant	Pointer	Level											
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N		
<b>WXOR</b>																									
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K4	5 ↓ 7	●	●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●										
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
d1		●	●	●	●	●	●	●	●	●	●	●	●	●	●										
<b>DXOR</b>																									
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1 ↓ K8	9 ↓ 1	●	●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●										

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps	
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other				
Bit	Word		Bit	Word								
<b>WXOR</b>												
s	●	●	●	●	●	●	●	●	●	—	—	3
d	●	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	●	—	—	4
s2	●	●	●	●	●	●	●	●	●	—	—	
d1	●	●	●	●	●	●	●	●	—	—	—	
<b>DXOR</b>												
	●	●	●	●	●	●	●	●	●	—	—	4 <sup>1)</sup>
	●	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	●	—	—	4 <sup>2)</sup>
s2	●	●	●	●	●	●	●	●	●	—	—	
d	●	●	●	●	●	●	●	●	—	—	—	

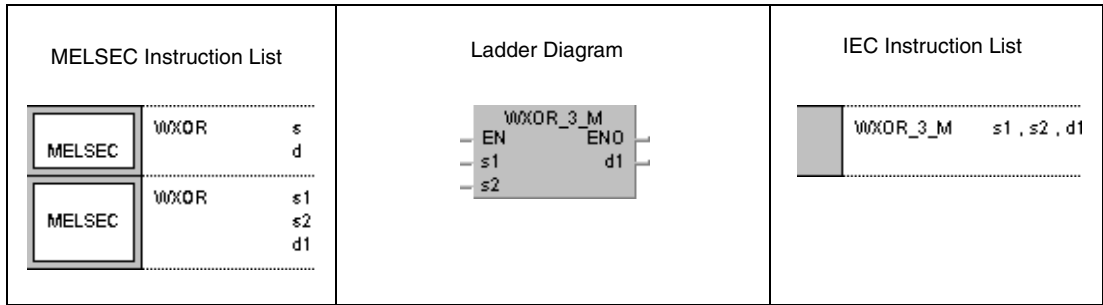
<sup>1</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU is used: 4
- If a single processor System Q CPU is used: 3
- If a multi processor System Q CPU is used with internal word devices (except for file register ZR) or constants: 6
- If a multi processor System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6
- If a multi processor System Q CPU is used with devices other than above mentioned: 4

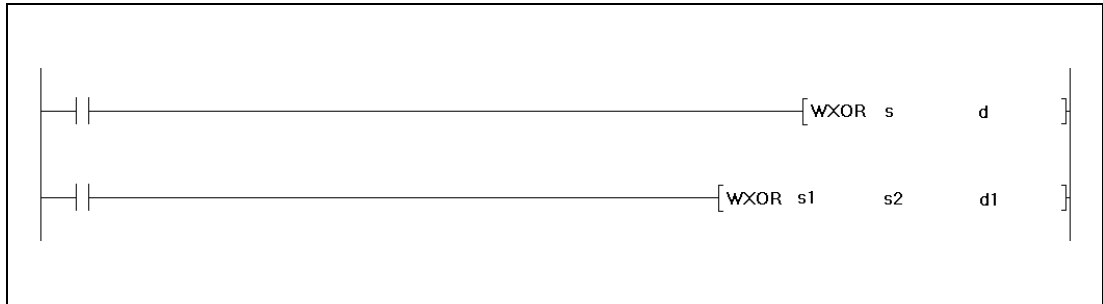
<sup>2</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU is used: 4
- If a System Q CPU is used with internal word devices (except for file register ZR) or constants: 6
- If a System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6
- If a System Q CPU is used with devices other than above mentioned: 4

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s	Data for exclusive OR operation, or first number of device storing such data.	BIN 16-/32-bit
d		
s1	Data for exclusive OR operation, or first number of device storing such data.	
s2		
d1 (for DXOR d)	First number of device storing result of logical operation.	

Functions

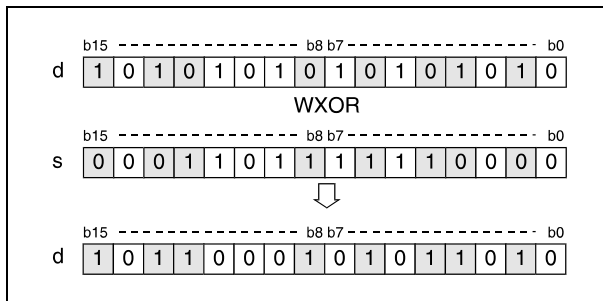
**Logical exclusive OR**

**WXOR 16-bit data**

The logical exclusive OR forms the logical sum of two input variables (  $Y = (\bar{A}xB) + (A\bar{x}B)$  ).

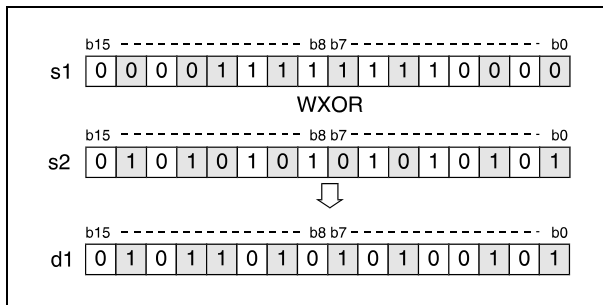
● Variation 1:

16-bit data designated by s and d form a logical exclusive OR connection. The result is output to the device designated by d.



● Variation 2:

16-bit data designated by s1 and s2 form a logical exclusive OR connection. The result is output to the device designated by d.

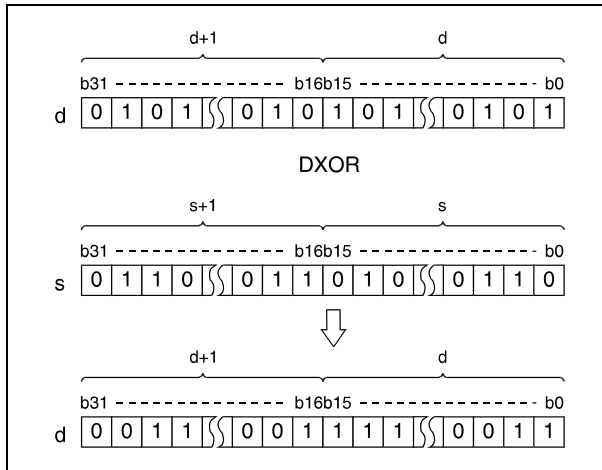


Bits exceeding the digit designation are set to 0. For example, if the digit designation is specified by K2, the higher 8 bits (b8 through b15) are processed as 0.

**DXOR 32-bit data**

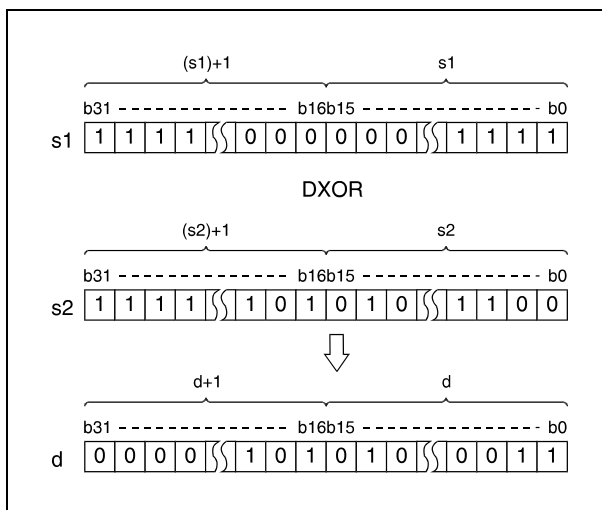
● Variation 1:

32-bit data designated by s and d form a logical exclusive OR connection. The result is output to the device designated by d.



● Variation 2 (Q series and System Q):

32-bit data designated by s1 and s2 form a logical exclusive OR connection. The result is output to the device designated by d.



After executing the connection, all bits exceeding the digit designation are set to 0.

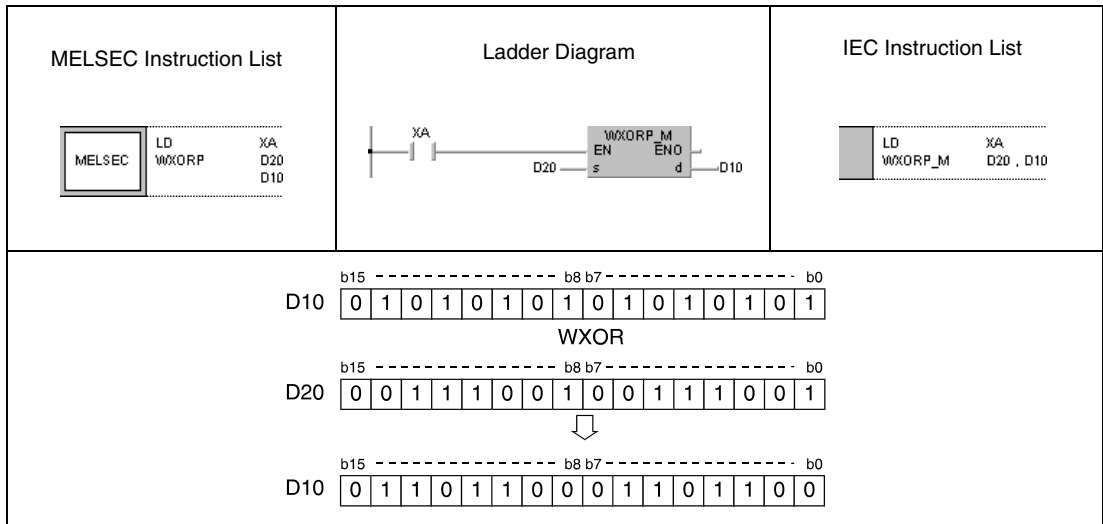
**NOTE**

For variation 1 (s, d) no operation errors are associated with the WXOR, WXORP, DXOR, and DXORP instructions, provided that index qualification is not applied.

**Program Example 1**

WXORP (s, d)

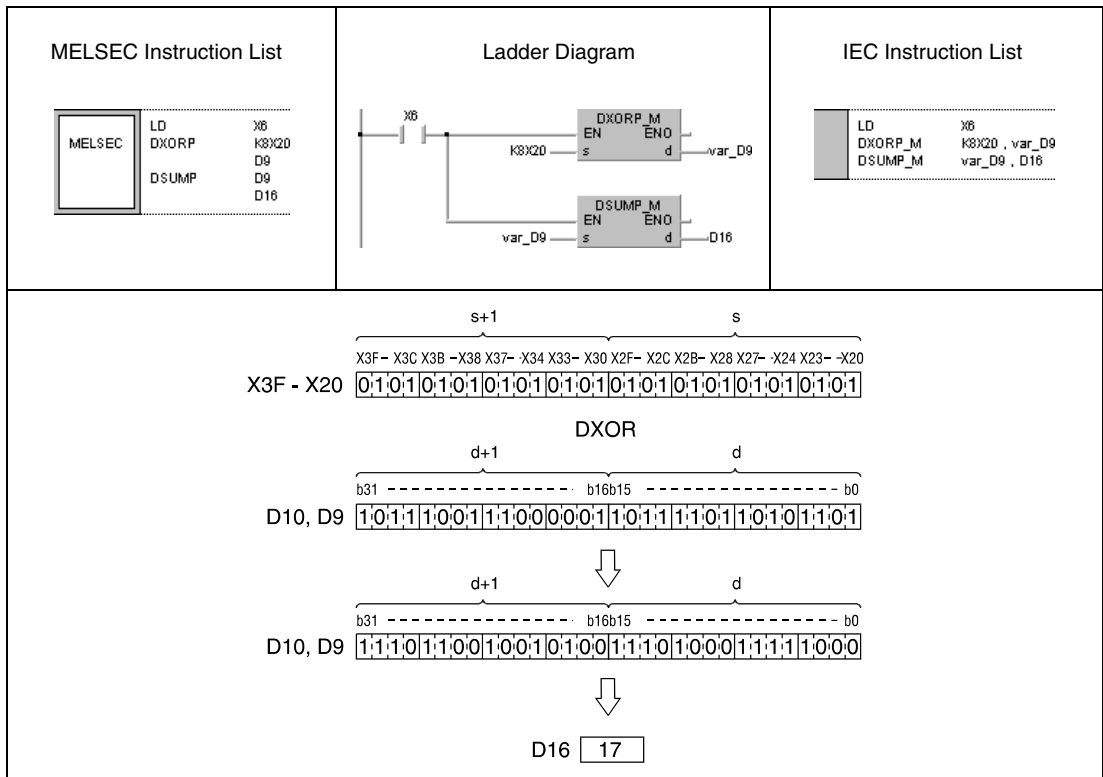
With leading edge from XA, the following program connects data in D10 with data in D20. The result is stored again in D10.



**Program Example 2**

DXORP (s, d)

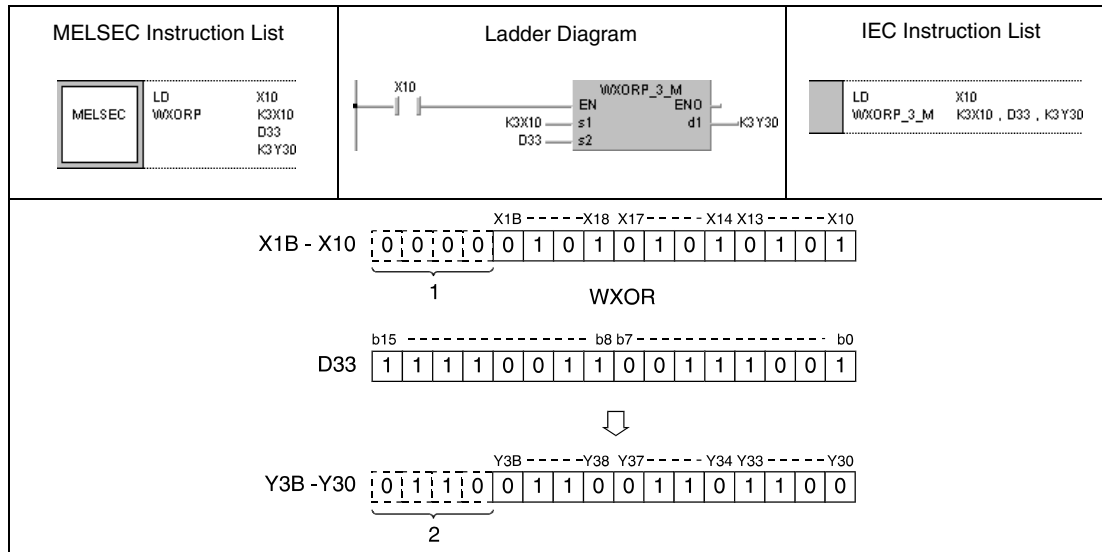
With leading edge from X6, the following program compares 32-bit data at the inputs X20 through X3F to the bit pattern in data registers D9 and D10. The result is stored again in D9 and D10. The number of set bits in D9 and D10 is stored in D16.



**Program Example 3**

WXORP (s1, s2, d1)

With leading edge from X10, the following program forms an exclusive OR connection of input data X10 through X1B with data in D33. The result is stored in D33 and output to Y30 through Y3B.

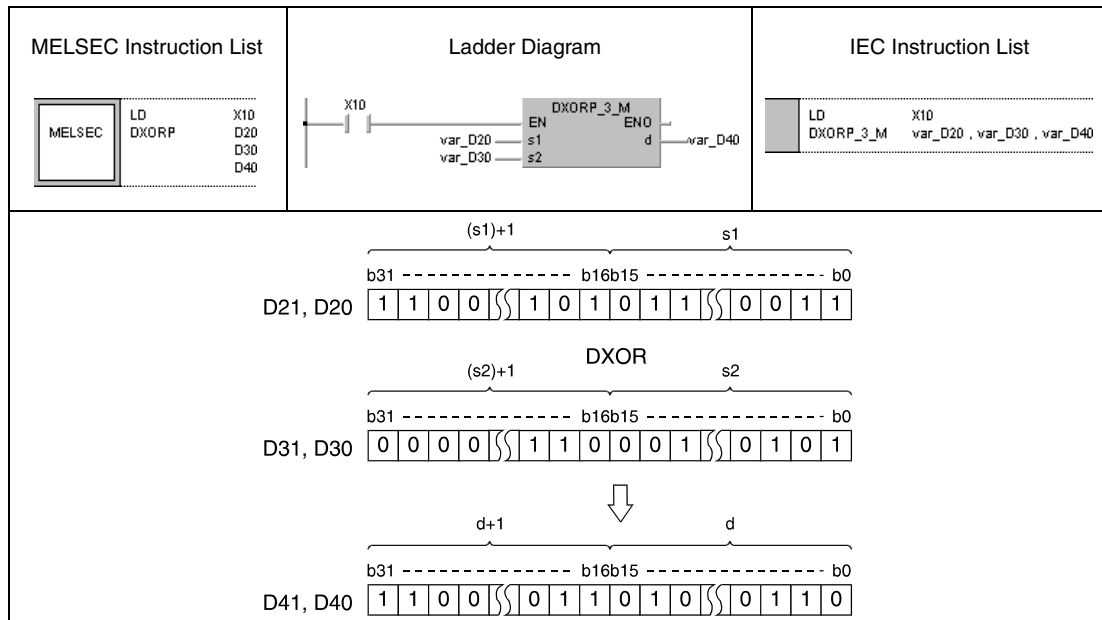


- <sup>1</sup> These bits are set to 0
- <sup>2</sup> These bits remain unchanged

**Program Example 4**

DXORP (s1, s2, d)

With leading edge from X10, the following program forms an exclusive OR connection of data in D20 and D21 with data in D30 and D31. The result is stored in D40 and D41.



**NOTE**

The program examples 2 and 4 will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

**7.1.6 BKXOR, BKXORP**


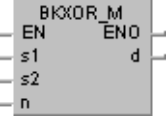
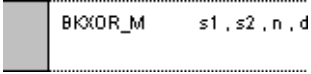
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

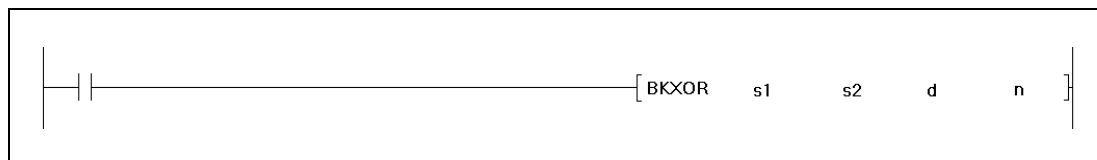
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	5
s2	—	●	●	—	—	—	—	●	—		
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	---

**GX Developer**



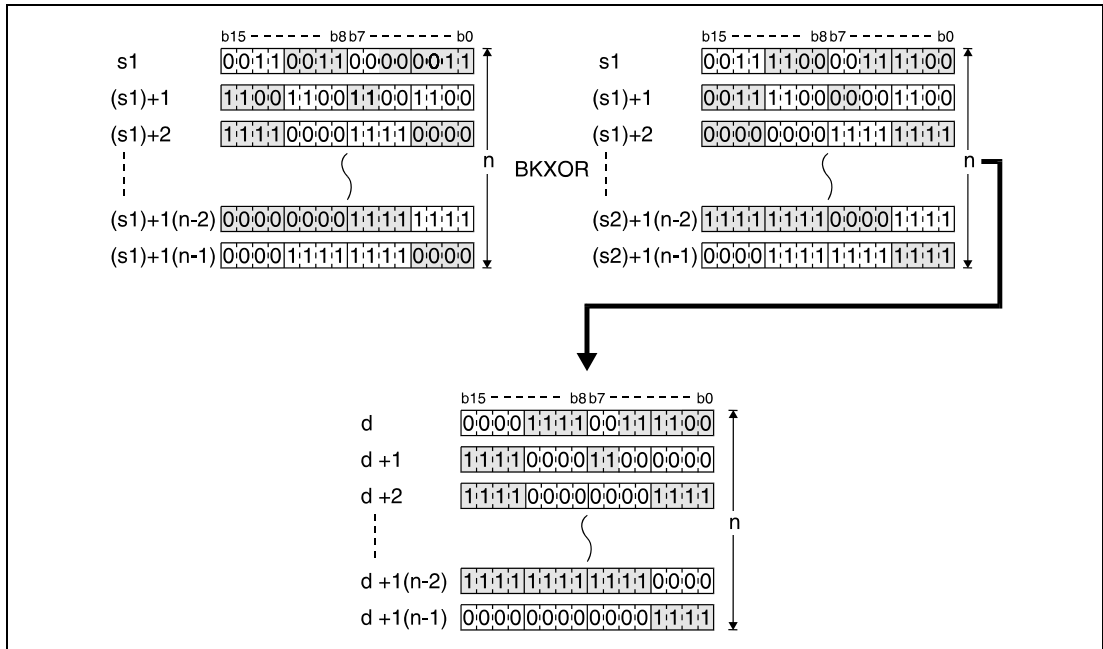
**Variables**

Set Data	Meaning	Data Type
s1	First number of device storing data for logical operation.	BIN 16-bit
s2	First number of data, or first number of device storing data for logical operation.	
d	First number of device storing result of operation.	
n	Number of data blocks forming the exclusive OR operation.	

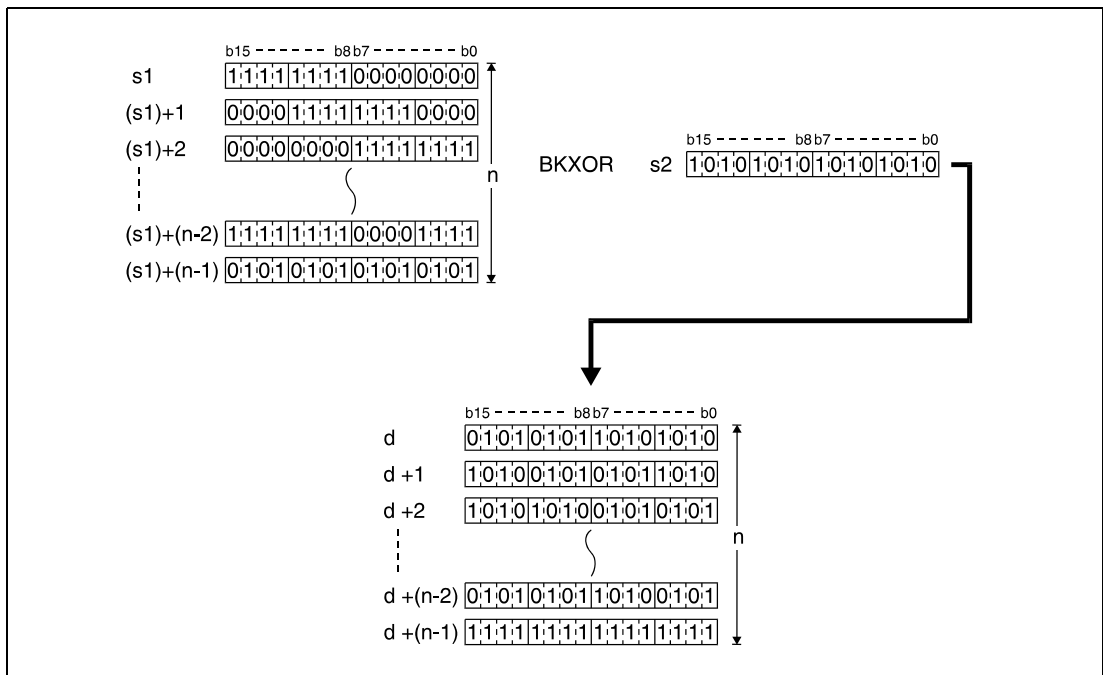
**Functions Exclusive OR operations with 16-bit data blocks**

**BKXOR Exclusive OR operations with data blocks**

The BKXOR instruction performs an exclusive OR operation beginning with the nth 16-bit data block from s1 onwards and with the nth 16-bit data block from s2 onwards. The according 16-bit block of the result is stored beginning from device d onwards. The number of blocks to be processed is specified by n.



The constant in s2 must range within -32768 and 32767.





**Operation Errors**

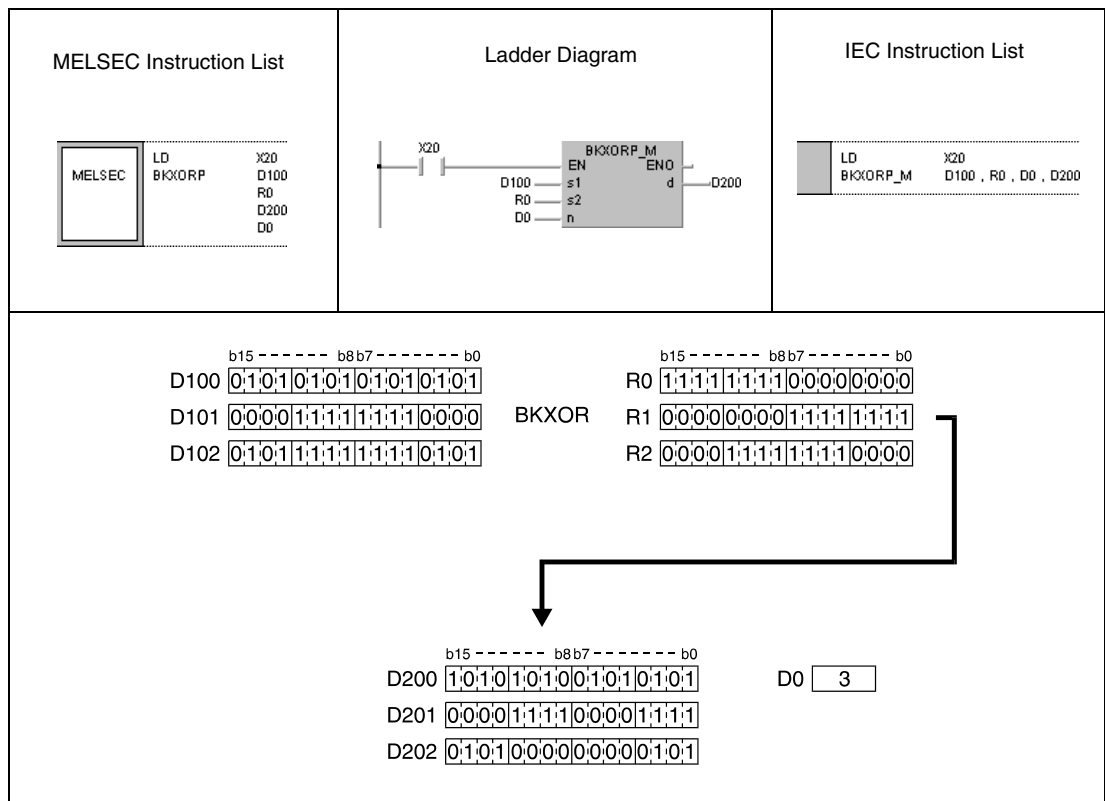
In the following cases an operation error occurs and the error flag is set:

- The number of data blocks determined by n exceeds the storage device numbers designated by s1, s2, or d (error code: 4101).
- The storage device numbers designated by s1, s2, or d overlap (error code: 4101).

**Program Example**

**BKXORP**

With leading edge from X20, the following program performs an exclusive OR operation with data in registers D100 through D102 and data in registers R0 through R2. The result is stored in registers D200 through D202. The number of 16-bit data blocks (3) to be processed is stored in D0.



7.1.7 WXNR, WXNRP, DXNR, DXNRP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag				
Bit Devices				Word Devices (16-bit)						Constant	Pointer	Level	M9012	M9010 M9011										
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N	
<b>WXNR</b>																								
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					5 ↓ 1	●	●
d																								
s1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					7 ↓ 1	●	●
s2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							
d1		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							
<b>DXNR</b>																								
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					9 ↓ 1	●	●
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
<b>WXNR, WXNRP</b>											
s	●	●	●	●	●	●	●	●	—	—	3
d	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	—	—	4
s2	●	●	●	●	●	●	●	●	—	—	
d	●	●	●	●	●	●	●	—	—	—	
<b>DXNR, DXNRP</b>											
s	●	●	●	●	●	●	●	●	—	—	4 <sup>1)</sup>
d	●	●	●	●	●	●	●	—	—	—	
s1	●	●	●	●	●	●	●	●	—	—	4 <sup>2)</sup>
s2	●	●	●	●	●	●	●	●	—	—	
d1	●	●	●	●	●	●	●	—	—	—	

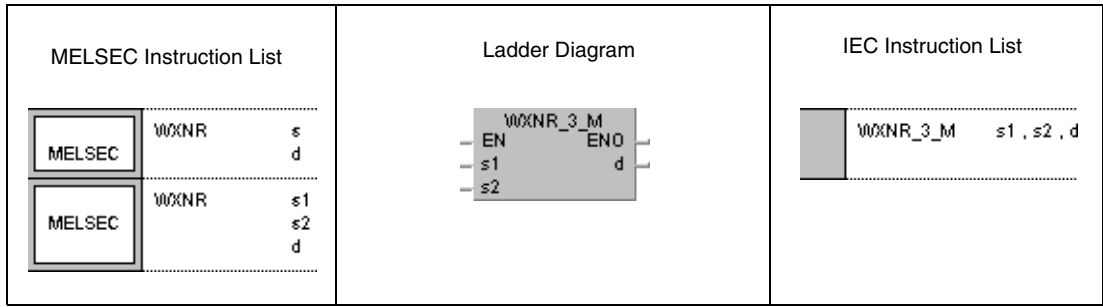
<sup>1</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU is used: 4
- If a single processor System Q CPU is used: 3
- If a multi processor System Q CPU is used with internal word devices (except for file register ZR) or constants: 6
- If a multi processor System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6
- If a multi processor System Q CPU is used with devices other than above mentioned: 4

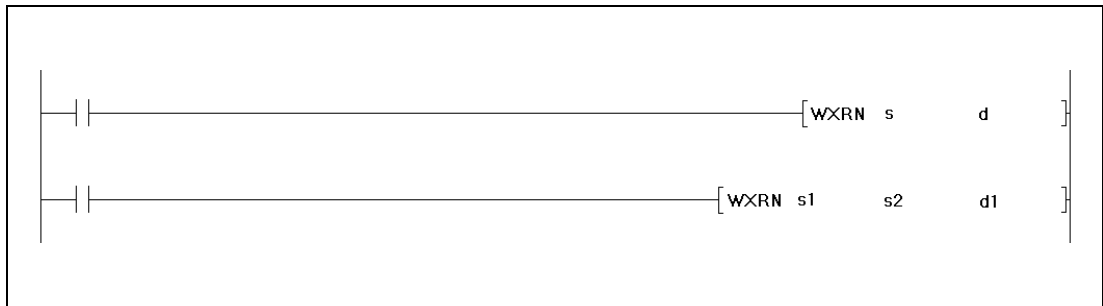
<sup>2</sup> The number of steps depends on the device and the type of CPU.

- If a QnA-CPU is used: 4
- If a System Q CPU is used with internal word devices (except for file register ZR) or constants: 6
- If a System Q CPU is used with Bit Devices, whose device numbers are multiples of 16, whose digit designation is K4, and which use no index qualification: 6
- If a System Q CPU is used with devices other than above mentioned: 4

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	Data for exclusive NOR operation, or first number of device storing such data.	BIN 16-/32-bit
d		
s1	Data for exclusive NOR operation, or first number of device storing such data.	
s2		
d (d1 for WXNRP)	First number of device storing result of logical operation	

**Functions**

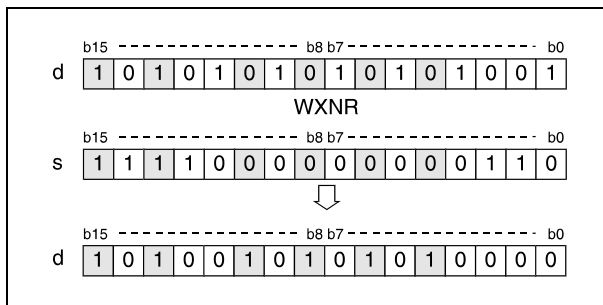
**Logical exclusive NOR**

**WXNR 16-bit data**

The logical exclusive NOR forms the logical product of the logical sum of two input variables ( $Y = (\bar{A} + B) \times (A + \bar{B})$ ).

● Variation 1:

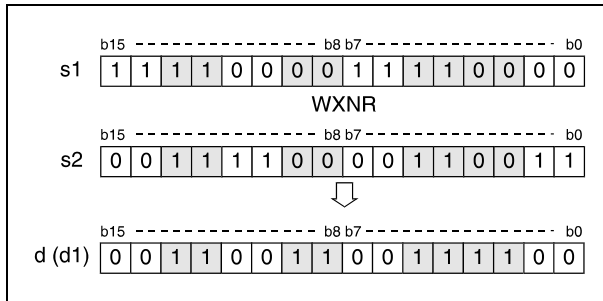
16-bit data designated by s and d form a logical exclusive NOR connection. The result is output to the device designated by d.



● Variation 2:

16-bit data designated by s1 and s2 form a logical exclusive NOR connection. The result is output to the device designated by d.

The WXNRP operation instruction outputs the result to the device designated by d1.

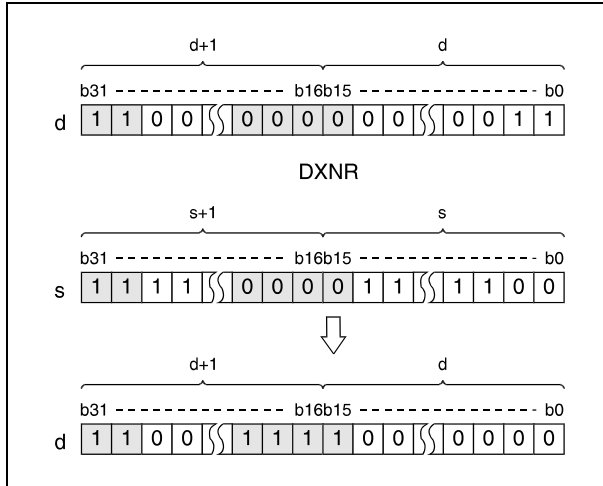


Bits exceeding the digit designation are set to 0. For example, if the digit designation is specified by K2, the higher 8 bits (b8 through b15) are processed as 0.

**DXNR 32-bit data**

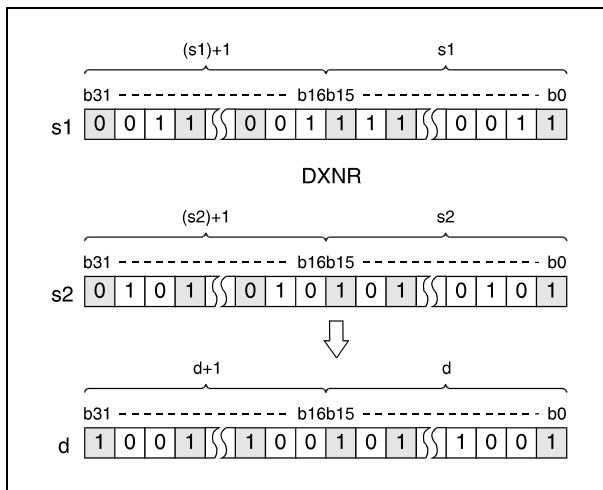
● Variation 1:

32-bit data designated by s and d form a logical exclusive NOR connection. The result is output to the device designated by d.



● Variation 2 (Q series and System Q):

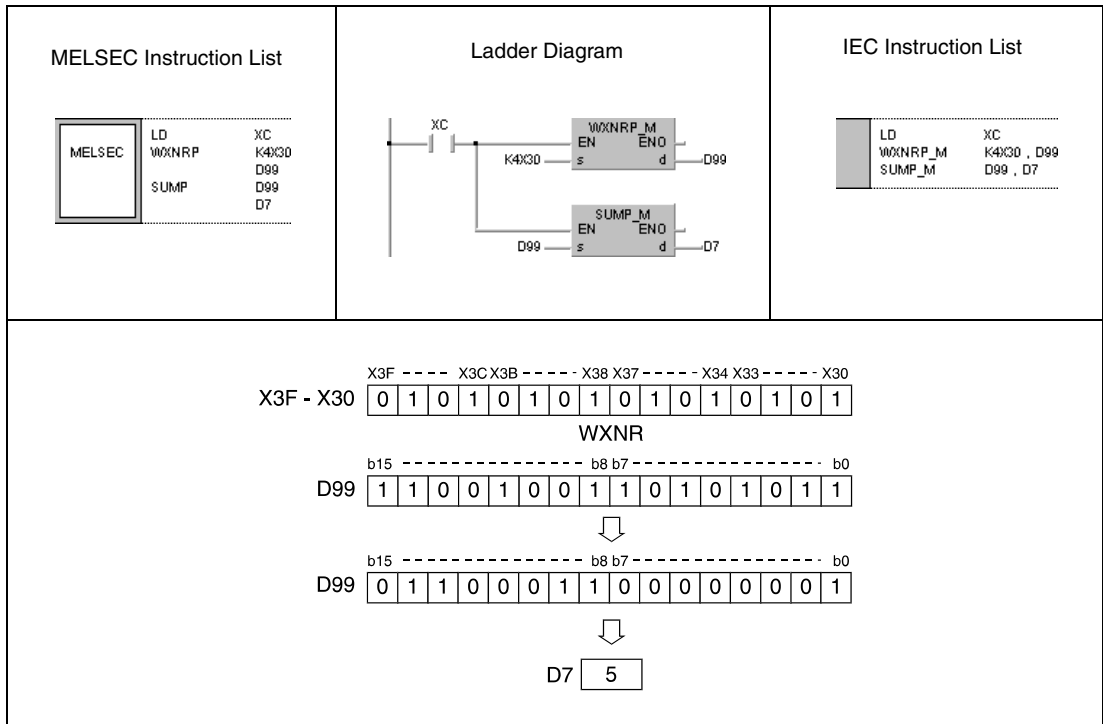
16-bit data designated by s1 and s2 form a logical exclusive NOR connection. The result is output to the device designated by d.



After executing the connection, all bits exceeding the digit designation are set to 0.

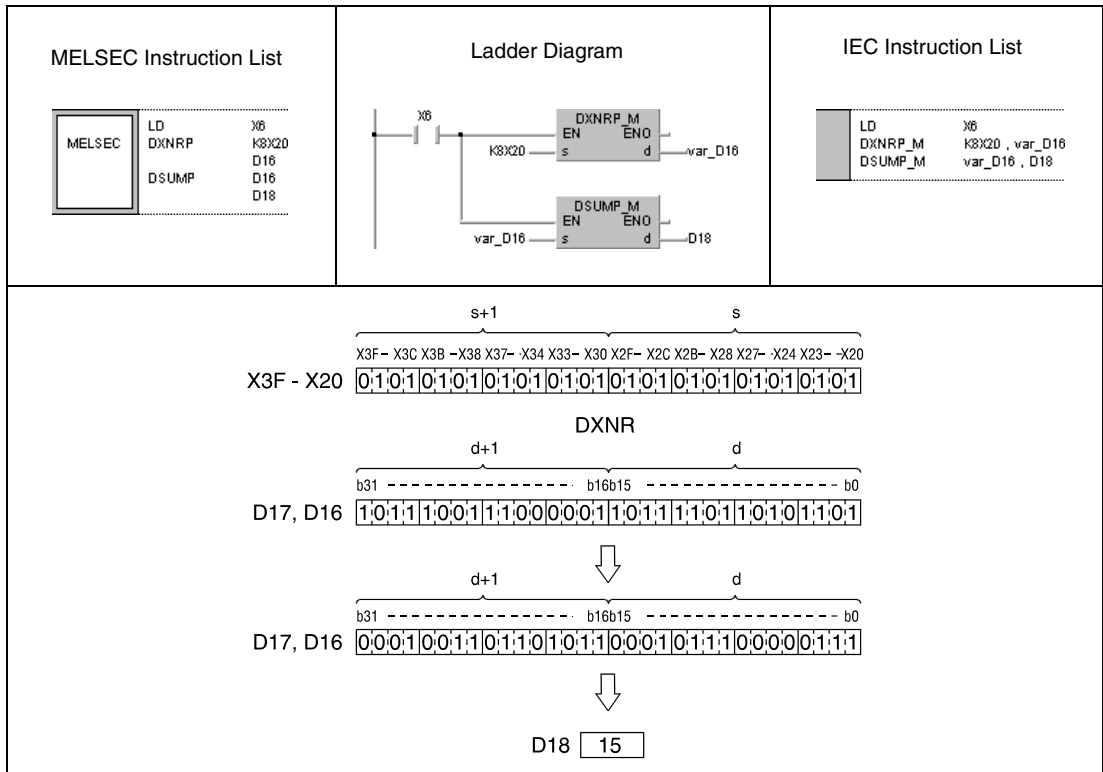
**Program Example 1** WXNRP (s, d)

With leading edge from XC, the following program compares the bit pattern of the 16-bit data value at the inputs X30 through X3F to the data value in D99. The result of the operation is stored again in D99. The number of set bits is stored in D7.



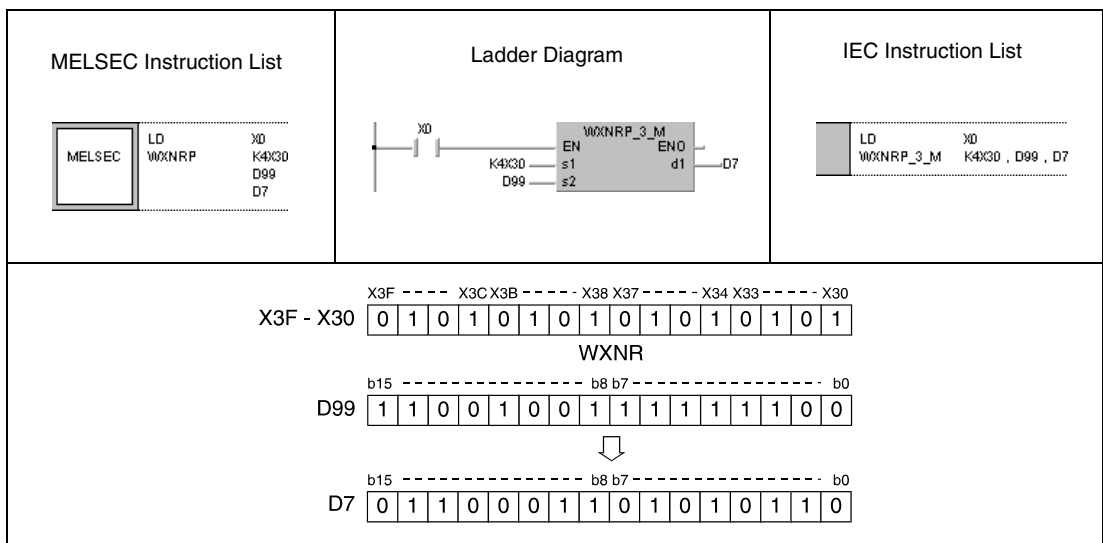
**Program Example 2** DXNRP (s, d)

With leading edge from X6, the following program compares the bit pattern of the 32-bit data value at the inputs X20 through X3F to data in D16 and D17. The result of the operation is stored again in D16 and D17. The number of set bits is stored in D18.



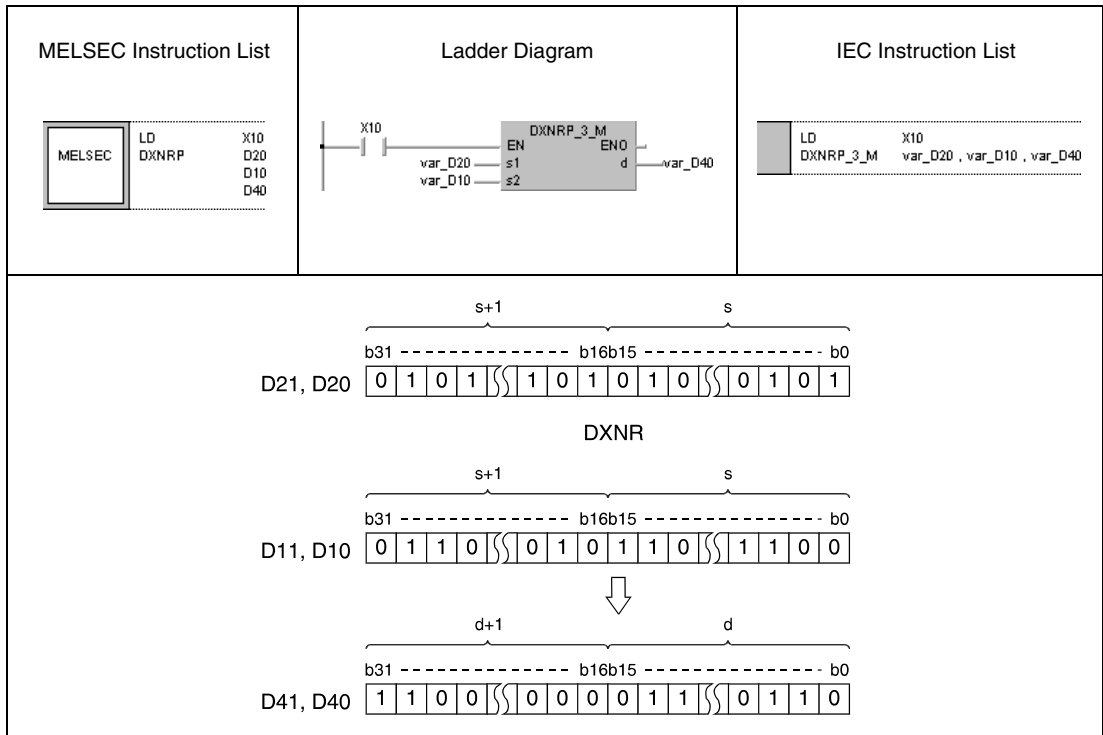
**Program Example 3** WXNRP (s1, s2, d1)

With leading edge from X0, the following program performs an exclusive NOR operation with 16-bit data at the inputs X30 through X3F and data in D99. The result of the operation is stored in D7.



**Program Example 4** DXNRP (s1, s2, d)

With leading edge from X10, the following program performs an exclusive NOR operation with 32-bit data in the registers D20 and D21 and with data in D10 and D11. The result of the operation is stored in D40 and D41.



**NOTE** The program examples 2 and 4 will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.



**7.1.8 BKXNR, BKXNRP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

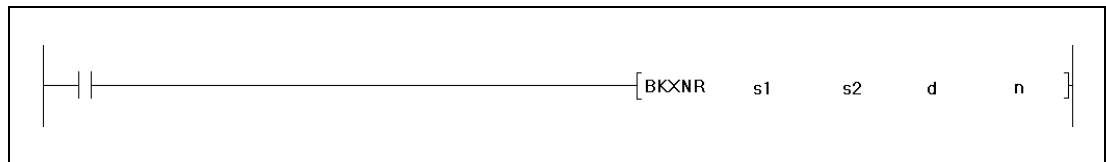
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	5
s2	—	●	●	—	—	—	—	●	—		
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">BKXNR</td> <td style="padding: 2px;">s1 s2 d n</td> </tr> </table> </div>	MELSEC	BKXNR	s1 s2 d n	<p>Ladder Diagram</p> <div style="text-align: center; margin-top: 10px;"> </div>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="padding: 2px;">BKXNR_M</td> <td style="padding: 2px;">s1, s2, n, d</td> </tr> </table> </div>		BKXNR_M	s1, s2, n, d
MELSEC	BKXNR	s1 s2 d n						
	BKXNR_M	s1, s2, n, d						

**GX Developer**



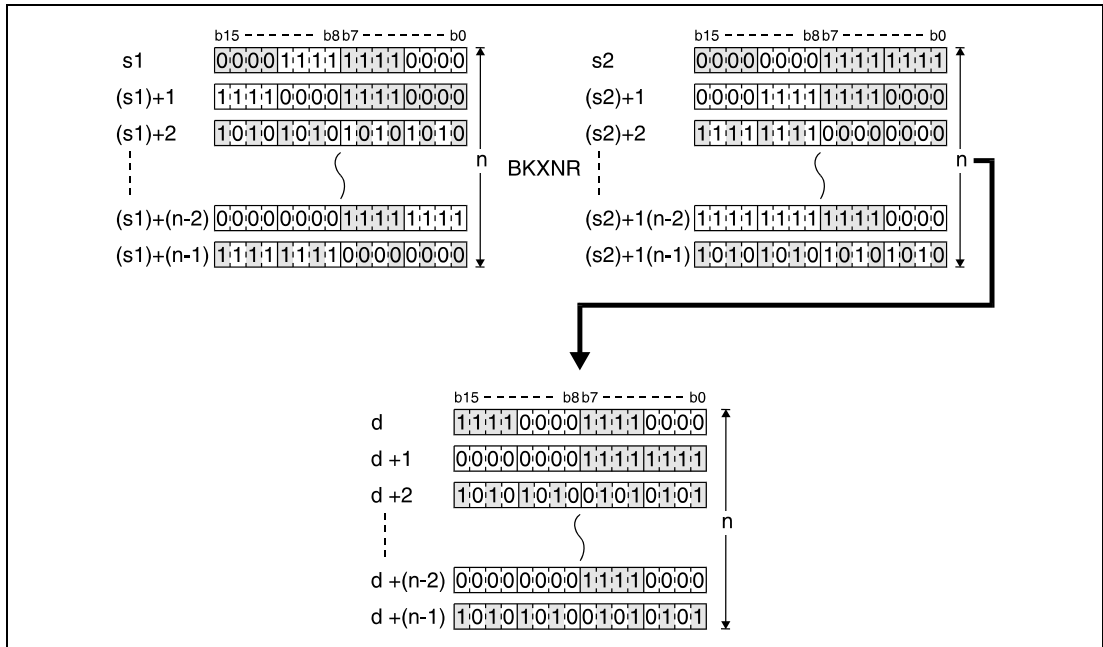
**Variables**

Set Data	Meaning	Data Type
s1	First number of device storing data for logical operation.	BIN 16-bit
s2	First number of data, or first number of device storing data for logical operation	
d	First number of device storing result of logical operation.	
n	Number of data blocks to be processed.	

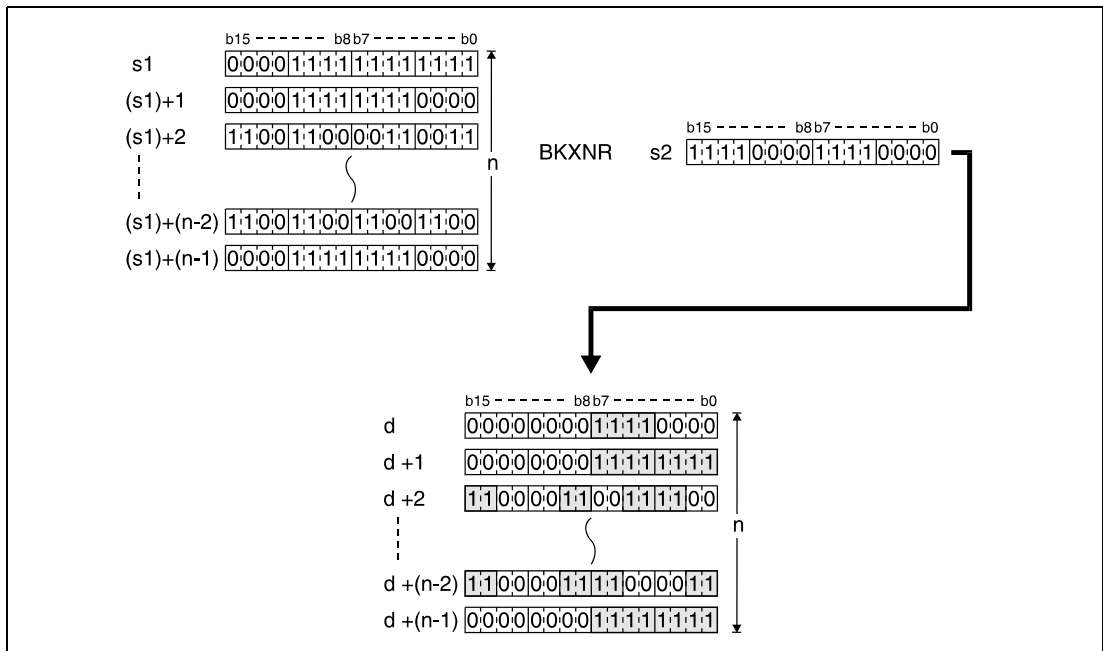
**Functions Exclusive NOR operations with 16-bit data blocks**

**BKXNR Exclusive NOR operations with data blocks**

The BKXNR instruction performs an exclusive NOR operation beginning with the nth 16-bit data block from s1 onwards and with the nth 16-bit data block from s2 onwards. The according 16-bit block of the result is stored beginning from device d onwards. The number of blocks to be processed is specified by n.



The constant in s2 must range within -32768 and 32767.



**Operation Errors**

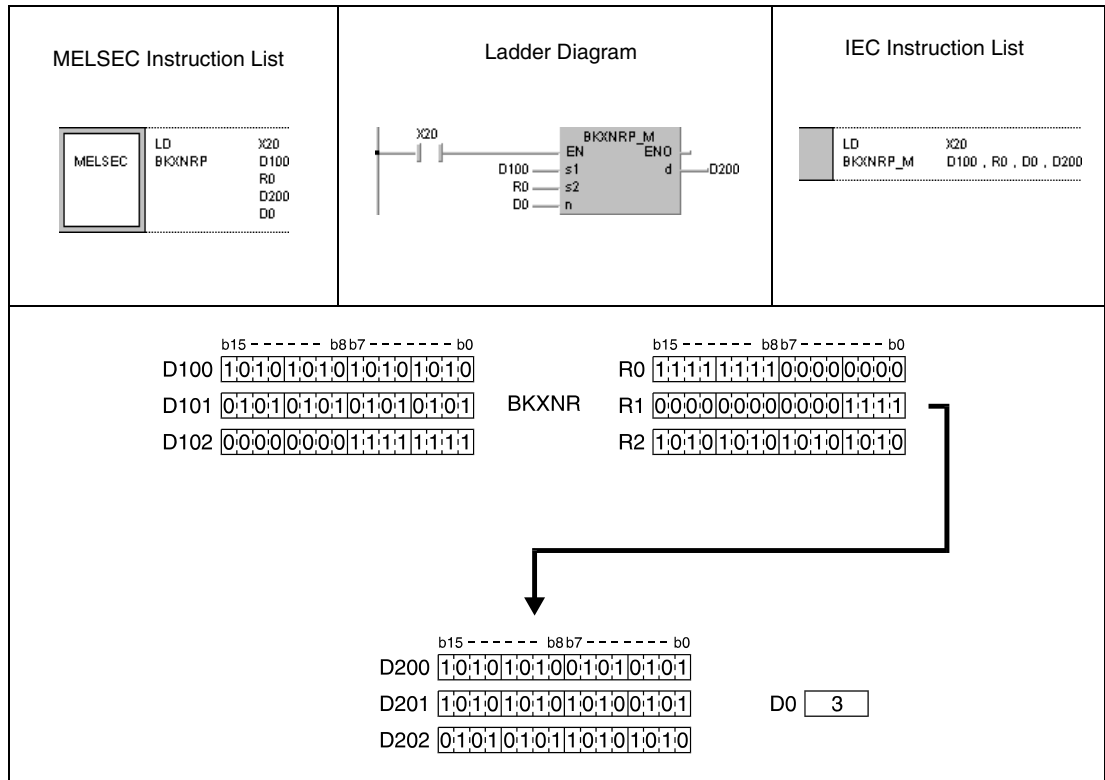
In the following cases an operation error occurs and the error flag is set:

- The number of data blocks determined by n exceeds the storage device numbers designated by s1, s2, or d (error code: 4101).
- The storage device numbers designated by s1, s2, or d overlap (error code: 4101).

**Program Example**

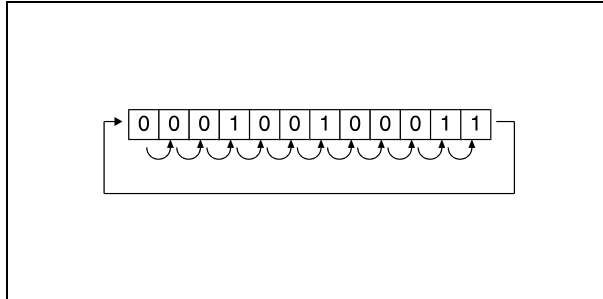
**BKXNRP**

With leading edge from X20, the following program performs an exclusive NOR operation with data in registers D100 through D102 and with data in registers R0 through R2. The result of the operation is stored in the registers D200 through D202. The number of 16-bit blocks (3) to be processed is stored in D0.



## 7.2 Data rotation instructions

The following rotation instructions rotate data stored in accumulators and registers bit by bit. Data can be rotated to the right as well as to the left.



Rotation instructions can alternatively be applied with or without carry flag. The rotation instructions are suitable for 16-bit and 32-bit data. In total, 16 different rotation instructions are supplied:

Function	MELSEC Instruction in MELSEC Editor	IEC Instruction in IEC Editor
Data rotation to the right (16-bit)	ROR	ROR_M
	RORP	RORP_M
	RCR	RCR_M
	RCRP	RCRP_M
Data rotation to the left (16-bit)	ROL	ROL_M
	ROLP	ROLP_M
	RCL	RCL_M
	RCLP	RCLP_M
Data rotation to the right (32-bit)	DROR	DROR_M
	DRORP	DRORP_M
	DRCR	DRCR_M
	DRCRP	DRCRP_M
Data rotation to the left (32-bit)	DROL	DROL_M
	DROLP	DROLP_M
	DRCL	DRCL_M
	DRCLP	DRCLP_M

**NOTE** *Within the IEC editors please use the IEC instructions.*

7.2.1 ROR, RORP, RCR, RCRP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

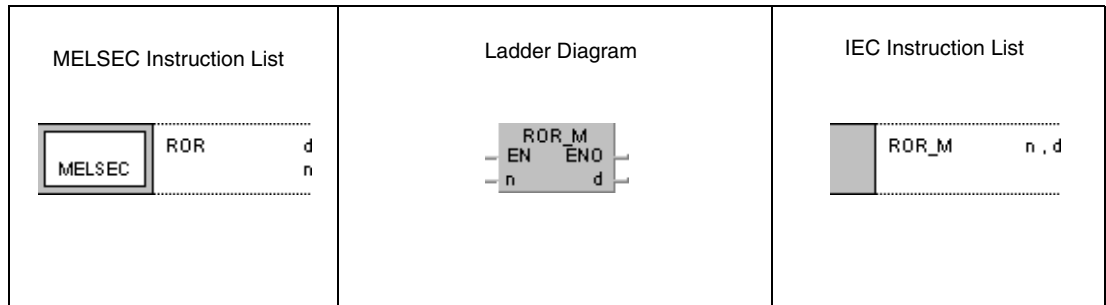
Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
n																●	●				●	●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

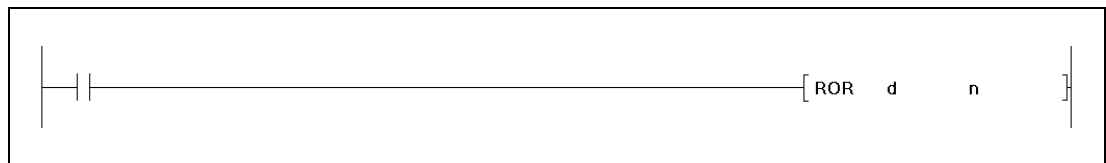
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
d	●	●	●	●	●	●	—	—	—	3	
n	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



NOTE

The A series always rotates data in register A0. For this reason, there is no device d available when programming this operation instruction for the A series.

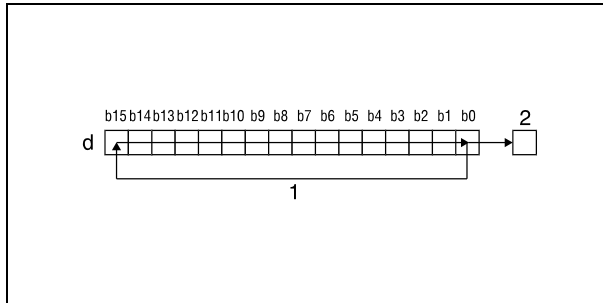
Variables

Set Data	Meaning	Data Type
d	First number of device performing data rotation. A series: device A0 only.	BIN 16-bit
n	Number of rotations (0 to 15).	

**Functions Data rotation to the right (16-bit)**

**ROR Rotation instruction without carry flag**

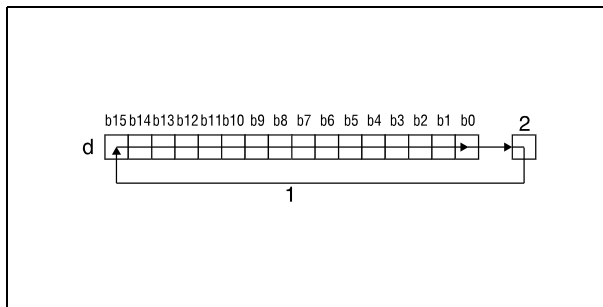
The ROR instruction rotates data bits in the device designated by d (A0) by n bits to the right. The carry flag (A series = M9012, Q series and System Q = SM700) is not included. It retains the condition of the latest bit rotated from b0 to b15.



- <sup>1</sup> Rotation by n bits
- <sup>2</sup> Carry flag

**RCR Rotation instruction with carry flag**

The RCR instruction rotates data bits in the device designated by d (A0) by n bits to the right, including the carry flag. The carry flag (A series = M9012, Q series and System Q = SM700) retains the condition of the bit rotated by n bits. The condition of the carry flag (0 or 1) prior to the rotation is moved to the right within d (A0) by n bits beginning from b15.



- <sup>1</sup> Rotation by n bits
- <sup>2</sup> Carry flag

**NOTE Q series and System Q only:**

If a bit device is designated by d, the rotation operation is performed with a device supplying the specified number of digits. The number of digits the bits are rotated by is determined by the remainder of the following quotient:

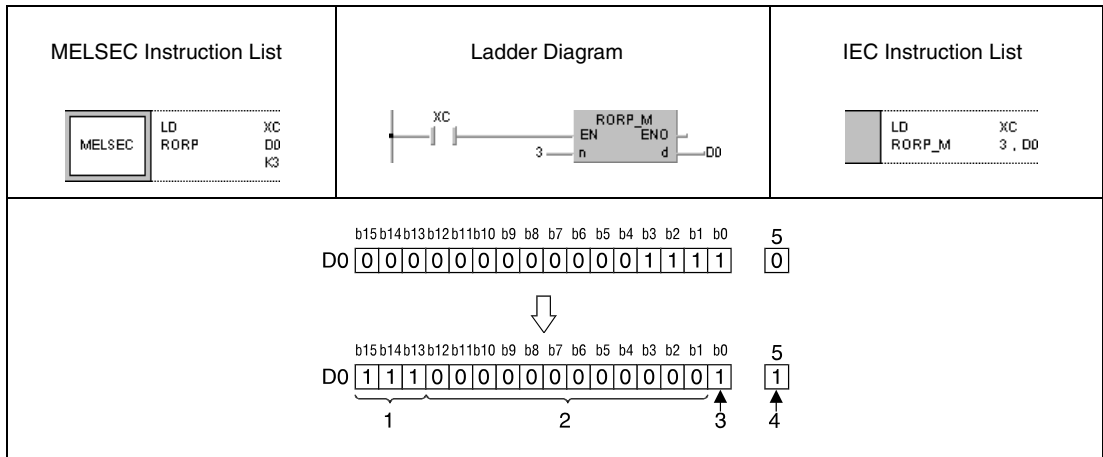
Number of rotations n / number of bits

For example, 16 rotations of 12 bits correspond to a rotation by 4 bits, since the remainder of the quotient 16/12 equals 4. The reason for this is that a bit x in 12 bits after 12-fold rotation again reaches the same position prior to the rotation.

For this reason, specify a value in the range from 0 to 15 as n.

**Program Example 1** RORP (Q series and System Q)

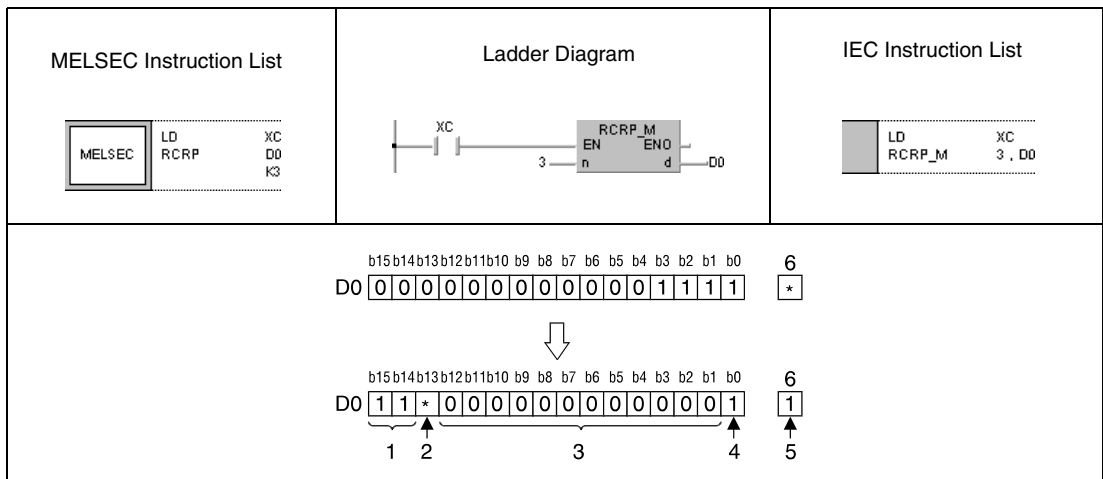
With leading edge from XC, the following program rotates the contents of D0 by 3 bits to the right.



- <sup>1</sup> Contents of bits b0 - b2 before the rotation
- <sup>2</sup> Contents of bits b4 - b15 before the rotation
- <sup>3</sup> Contents of bit b3 before the rotation
- <sup>4</sup> Contents of bit b2 before the rotation
- <sup>5</sup> Carry flag

**Program Example 2** RCRP (Q series and System Q)

With leading edge from XC, the following program rotates the contents of D0 by 3 bits to the right; the carry flag SM700 is included. The condition of SM700 (0/1) prior to the rotation is moved to the right by 3 digits.



- <sup>1</sup> Contents of bits b1 and b0 before the rotation
- <sup>2</sup> Contents of carry flag before the rotation
- <sup>3</sup> Contents of bits b4 - b15 before the rotation
- <sup>4</sup> Contents of bit b3 before the rotation
- <sup>5</sup> Contents of bit b2 before the rotation
- <sup>6</sup> Carry flag

7.2.2 ROL, ROLP, RCL, RCLP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

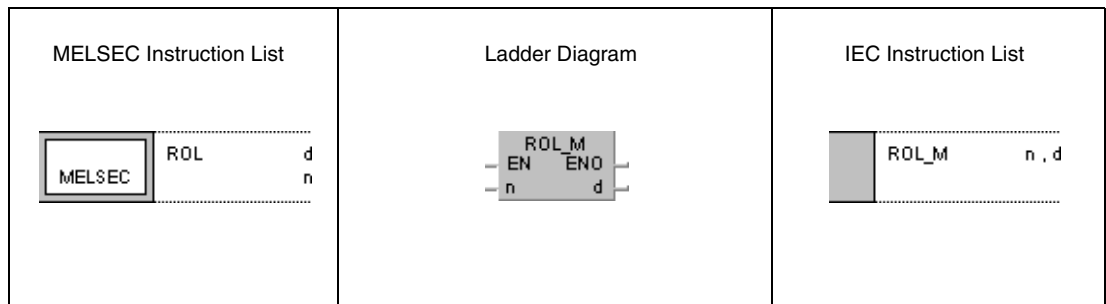
Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level	
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
n																●	●				●	●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

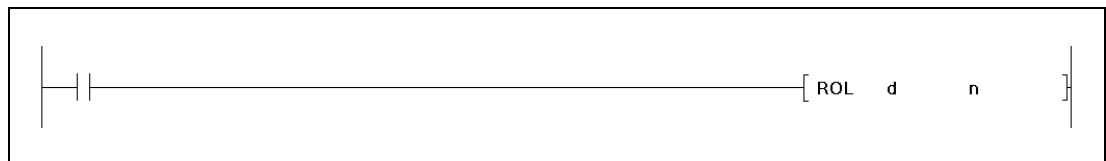
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
d	●	●	●	●	●	●	—	—	—	—	3
n	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



NOTE

The A series always rotates data in register A0. For this reason, there is no device d available when programming this operation instruction for the A series.

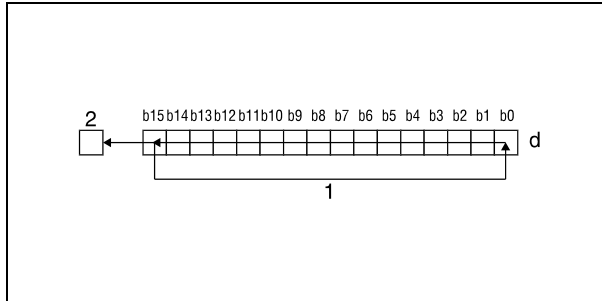
Variables

Set Data	Meaning	Data Type
d	First number of device performing data rotation. A series: device A0 only.	BIN 16-bit
n	Number of rotations (0 to 15).	



**Functions**    **Data rotation to the left (16-bit)****ROL**    **Rotation instruction without carry flag**

The ROL instruction rotates data bits in the device designated by d (A0) by n bits to the left. The carry flag (A series = M9012, Q series and System Q = SM700) is not included. It retains the condition of the latest bit rotated from b0 to b15.

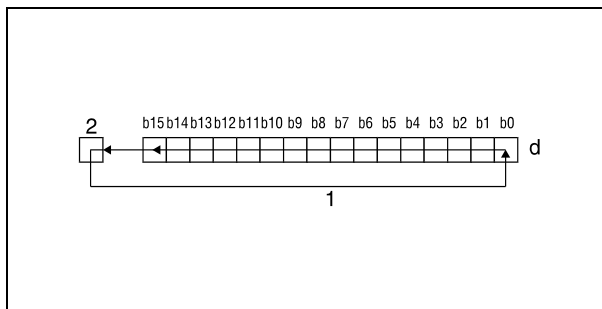


<sup>1</sup> Rotation by n bits

<sup>2</sup> Carry flag

**RCL**    **Rotation instruction with carry flag**

The RCL instruction rotates data bits in the device designated by d (A0) by n bits to the left, including the carry flag. The carry flag (A series = M9012, Q series and System Q = SM700) retains the condition of the bit rotated by n bits. The condition of the carry flag (0 or 1) prior to the rotation is moved to the left within d (A0) by n bits beginning from b15.



<sup>1</sup> Rotation by n bits

<sup>2</sup> Carry flags

**NOTE**

*Q series and System Q only:*

*If a bit device is designated by d, the rotation operation is performed with a device supplying the specified number of digits. The number of digits the bits are rotated by is determined by the remainder of the following quotient:*

*Number of rotations n / number of bits*

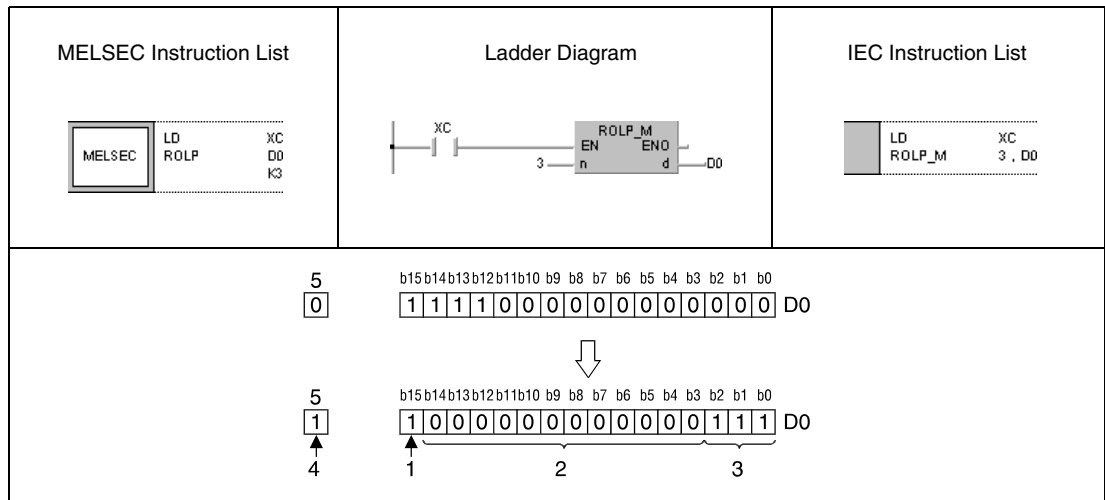
*For example, 16 rotations of 12 bits correspond to a rotation by 4 bits, since the remainder of the quotient 16/12 equals 4. The reason for this is that a bit x in 12 bits after 12-fold rotation again reaches the same position prior to the rotation.*

*For this reason, specify a value in the range from 0 to 15 as n.*

**Program Example 1**

ROLP (Q series and System Q)

With leading edge from XC, the following program rotates the contents of D0 by 3 bits to the left.

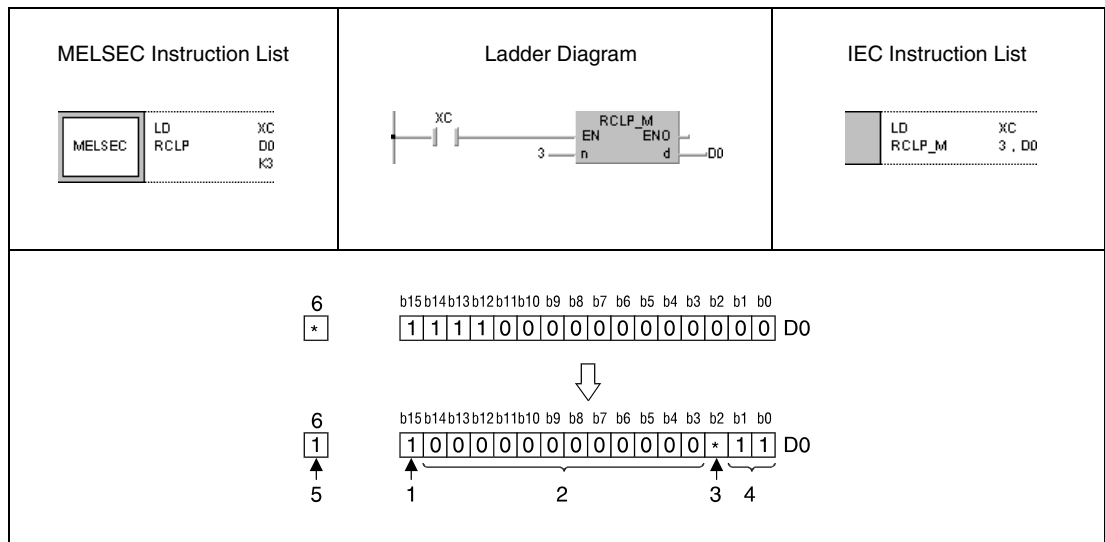


- 1 Contents of bit b12 before the rotation
- 2 Contents of bits b11-b0 before the rotation
- 3 Contents of bits b15-b13 before the rotation
- 4 Contents of bit b12 before the rotation
- 5 Carry flag

**Program Example 2**

RCLP (Q series and System Q)

With leading edge from XC, the following program rotates the contents of D0 by 3 bits to the left; the carry flag SM700 is included. The condition of SM700 (0/1) prior to the rotation is moved to the left by 3 digits.



- 1 Contents of bit b12 before the rotation
- 2 Contents of bits b11-b0 before the rotation
- 3 Contents of carry flag
- 4 Contents of bits b14 and b15 before the rotation
- 5 Contents of carry flag before the rotation
- 6 Carry flag

**7.2.3 DROR, DRORP, DRCR, DRCRP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices MELSEC A**

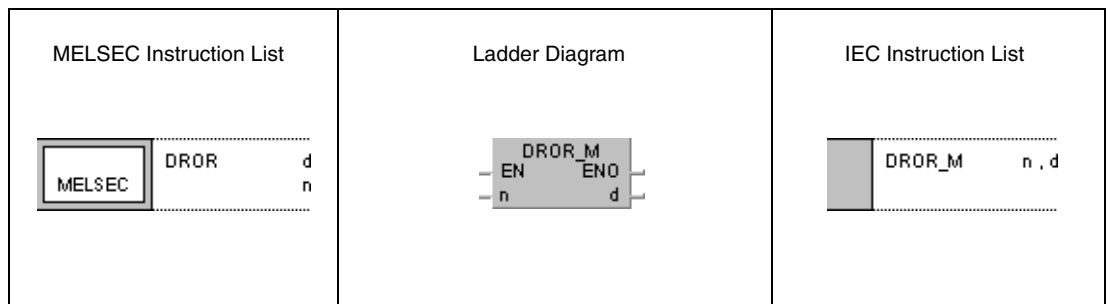
Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag			
Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level		
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N
n																●	●					●	●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

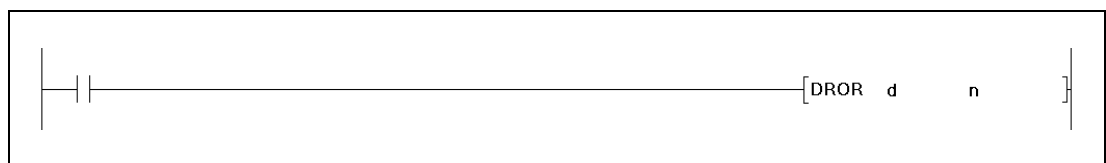
**Devices MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
d	●	●	●	●	●	●	—	—	—	3	
n	●	●	●	●	●	●	●	—	—		

**GX IEC Developer**



**GX Developer**



**NOTE**

The A series always rotates data in registers A0 and A1. For this reason, there is no device d available when programming this operation instruction for the A series.

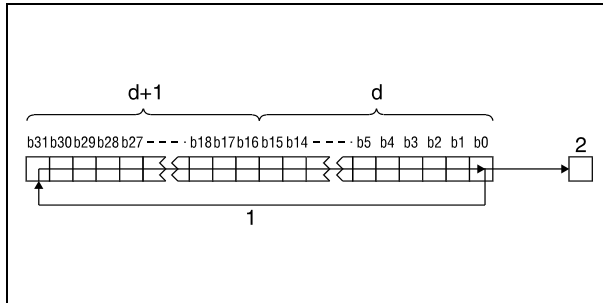
**Variables**

Set Data	Meaning	Data Type
d	First number of device performing data rotation. A series: device A0 and A1 only.	BIN 32-bit
n	Number of rotations (0 to 31).	BIN 16-bit

**Functions**     **Data rotation to the right (32-bit)**

**DROR    Rotation instruction without carry flag**

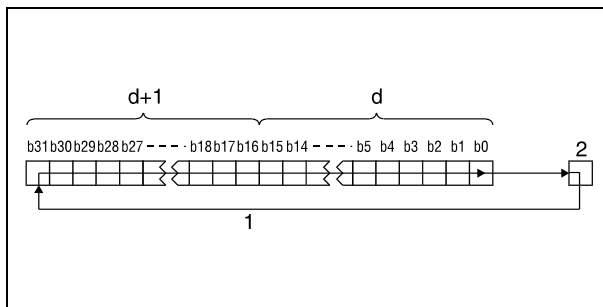
The DROR instruction rotates data bits in the device designated by d (A0, A1) by n bits to the right. The carry flag (A series = M9012, Q series and System Q = SM700) is not included. It retains the condition of the latest bit rotated from b0 to b31.



- <sup>1</sup> Rotation by n bits
- <sup>2</sup> Carry flag

**DRCR    Rotation instruction with carry flag**

The DRCR instruction rotates data bits in the device designated by d (A0, A1) by n bits to the right, including the carry flag. The carry flag (A series = M9012, Q series and System Q = SM700) retains the condition of the bit rotated by n bits. The condition of the carry flag (0 or 1) prior to the rotation is moved to the right within d (A0, A1) by n bits beginning from b31.



- <sup>1</sup> Rotation by n bits
- <sup>2</sup> Carry flag

**NOTE**     *Q series and System Q only:*

*If a bit device is designated by d, the rotation operation is performed with a device supplying the specified number of digits. The number of digits the bits are rotated by is determined by the remainder of the following quotient:*

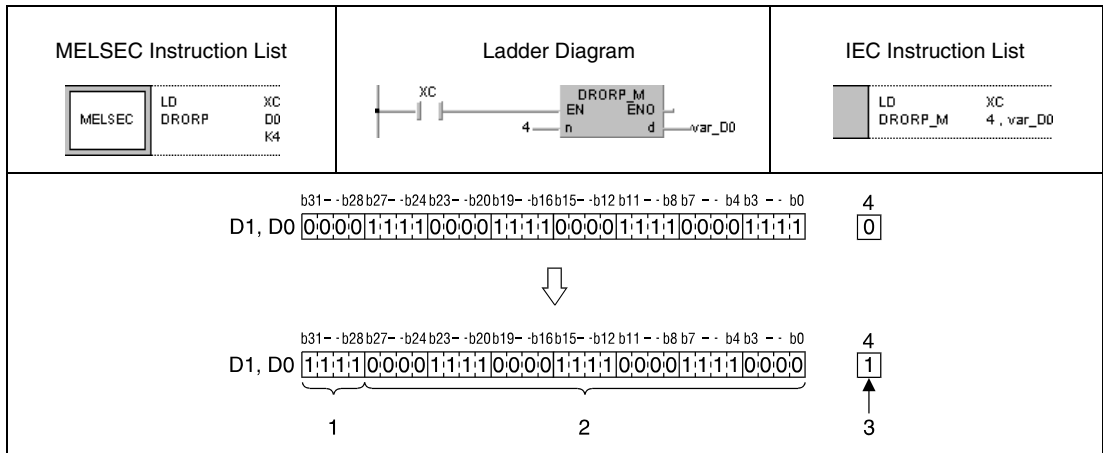
*Number of rotations n / number of bits*

*For example, 31 rotations of 24 bits correspond to a rotation by 7 bits, since the remainder of the quotient 31/24 equals 7. The reason for this is that a bit x in 24 bits after 24-fold rotation again reaches the same position prior to the rotation.*

*For this reason, specify a value in the range from 0 to 31 as n.*

**Program Example 1** DRORP (Q series and System Q)

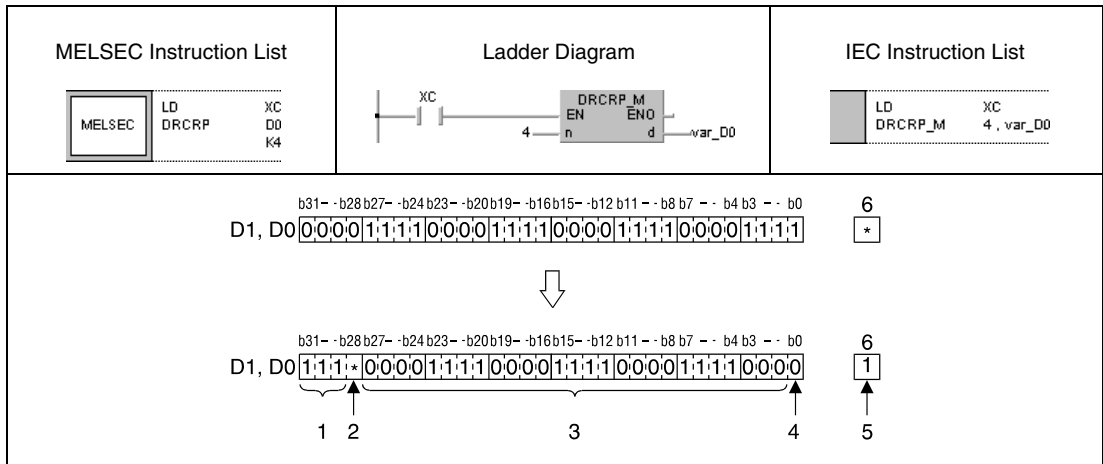
With leading edge from XC, the following program rotates the contents of D0 and D1 by 4 bits to the right.



- 1 Contents of bits b3-b0 before the rotation
- 2 Contents of bits b31-b4 before the rotation
- 3 Contents of bit b3 before the rotation
- 4 Carry flag

**Program Example 2** DRCRP (Q series and System Q)

With leading edge from XC, the following program rotates the contents of D0 and D1 by 4 bits to the right; the carry flag SM700 is included. The condition of SM700 (0/1) prior to the rotation is moved to the right by 4 digits.



- 1 Contents of bits b2-b0 before the rotation
- 2 Contents of carry flag before the rotation
- 3 Contents of bits b5-b31 before the rotation
- 4 Contents of bit b4 before the rotation
- 5 Contents of bit b3 before the rotation
- 6 Carry flag

**NOTE** These programs will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

7.2.4 DROL, DROLP, DRCL, DRCLP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

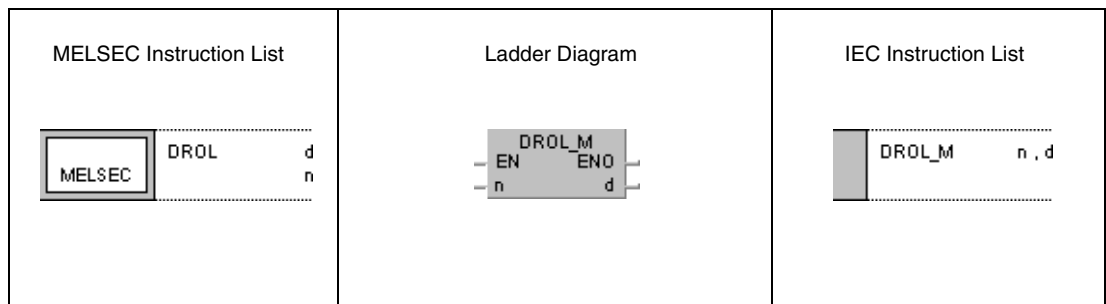
Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag					
Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level				
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N	M9012	M9010 M9011
n																●	●					●	●	●	●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this Programming Manual for the according number of steps.

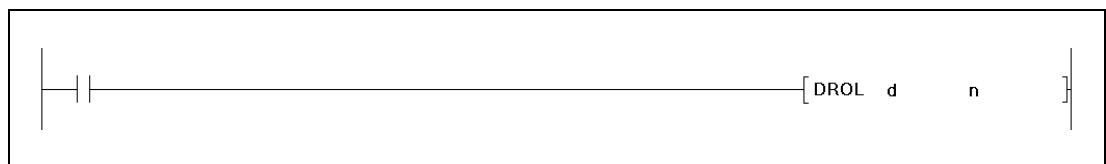
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant E	Other			
Bit	Word		Bit	Word							
d	●	●	●	●	●	●	—	—	—	3	
n	●	●	●	●	●	●	●	—	—		

GX IEC  
Developer



GX  
Developer



NOTE

The A series always rotates data in registers A0 and A1. For this reason, there is no device d available when programming this operation instruction for the A series.

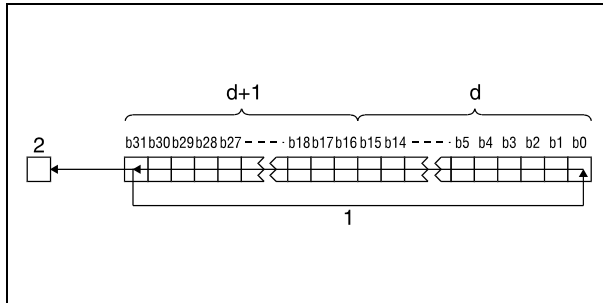
Variables

Set Data	Meaning	Data Type
d	First number of device performing data rotation. A series: device A0 and A1 only.	BIN 32-bit
n	Number of rotations (0 to 31).	BIN 16-bit

**Functions Data rotation to the left (32-bit)**

**DROL Rotation instruction without carry flag**

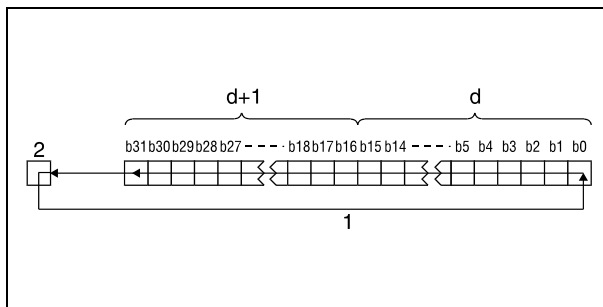
The DROL instruction rotates data bits in the device designated by d (A0, A1) by n bits to the left. The carry flag (A series = M9012, Q series and System Q = SM700) is not included. It retains the condition of the latest bit rotated from b31 to b0.



- <sup>1</sup> Rotation by n bits
- <sup>2</sup> Carry flag

**DRCL Rotation instruction with carry flag**

The DRCL instruction rotates data bits in the device designated by d (A0, A1) by n bits to the left, including the carry flag. The carry flag (A series = M9012, Q series and System Q = SM700) retains the condition of the bit rotated by n bits. The condition of the carry flag (0 or 1) prior to the rotation is moved to the left within d (A0, A1) by n bits beginning from b31.



- <sup>1</sup> Rotation by n bits
- <sup>2</sup> Carry flag

**NOTE**

*Q series and System Q only:*

*If a bit device is designated by d, the rotation operation is performed with a device supplying the specified number of digits. The number of digits the bits are rotated by is determined by the remainder of the following quotient:*

*Number of rotations n / number of bits*

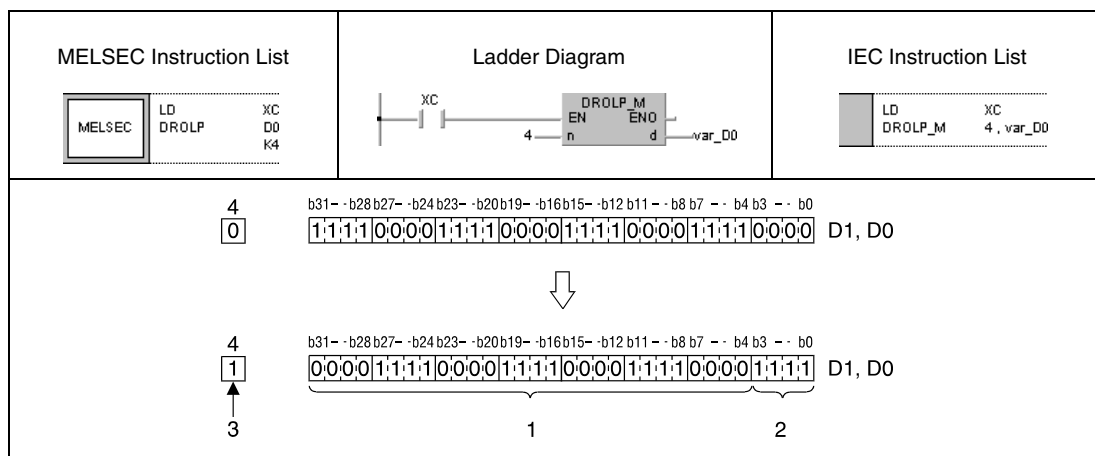
*For example, 31 rotations of 24 bits correspond to a rotation by 7 bits, since the remainder of the quotient 31/24 equals 7. The reason for this is that a bit x in 24 bits after 24-fold rotation again reaches the same position prior to the rotation.*

*For this reason, specify a value in the range from 0 to 31 as n.*

**Program Example 1**

DROLP (Q series and System Q)

With leading edge from XC, the following program rotates the contents of D0 and D1 by 4 bits to the left.

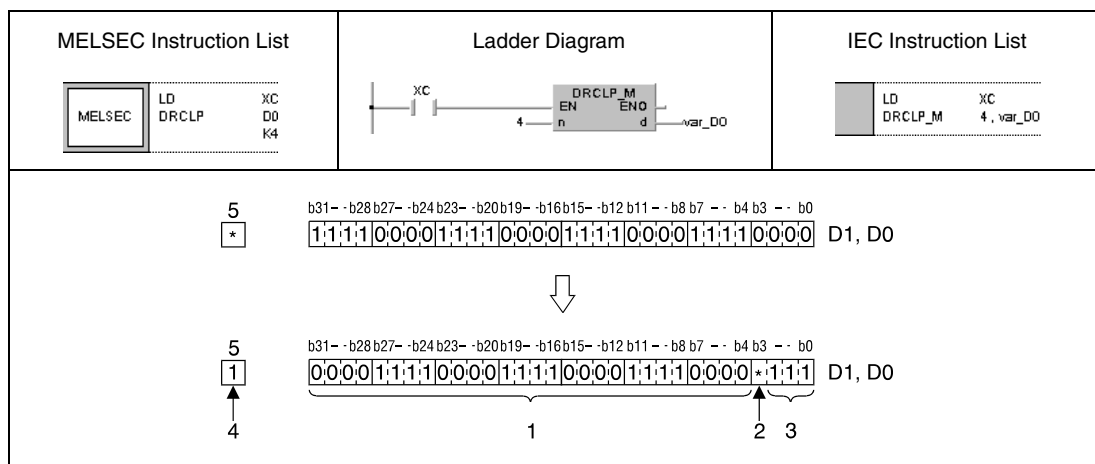


- <sup>1</sup> Contents of bits b27–b0 before the rotation
- <sup>2</sup> Contents of bits b31–b28 before the rotation
- <sup>3</sup> Contents of bit b28 before the rotation
- <sup>4</sup> Carry flag

**Program Example 2**

DRCLP (Q series and System Q)

With leading edge from XC, the following program rotates the contents of D0 and D1 by 4 bits to the left; the carry flag (SM700) is included. The condition of SM700 (0/1) prior to the rotation is moved to the left by 4 digits.



- <sup>1</sup> Contents of bits b27–b0 before the rotation
- <sup>2</sup> Contents of carry flag before the rotation
- <sup>3</sup> Contents of bits b31–b29 before the rotation
- <sup>4</sup> Contents of bit b28 before the rotation
- <sup>5</sup> Carry flag

**NOTE**

*These programs will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



## 7.3 Data shift instructions

The shift instructions move data by bits or blocks of data within one data word. Data can be shifted to the right as well as to the left.

In total, 12 different shift instructions are supplied:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Shift a 16-bit data word by n bits	SFR	SFR_M
	SFRP	SFRP_M
	SFL	SFL_M
	SFLP	SFLP_M
Shift n bit devices by 1 bit	BSFR	BSFR_M
	BSFRP	BSFRP_M
	BSFL	BSFL_M
	BSFLP	BSFLP_M
Shift n word devices by one digit	DSFR	DSFR_M
	DSFRP	DSFRP_M
	DSFL	DSFL_M
	DSFLP	DSFLP_M

**NOTE**

*Within the IEC editors please use the IEC instructions.*

7.3.1 SFR, SFRP, SFL, SFLP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

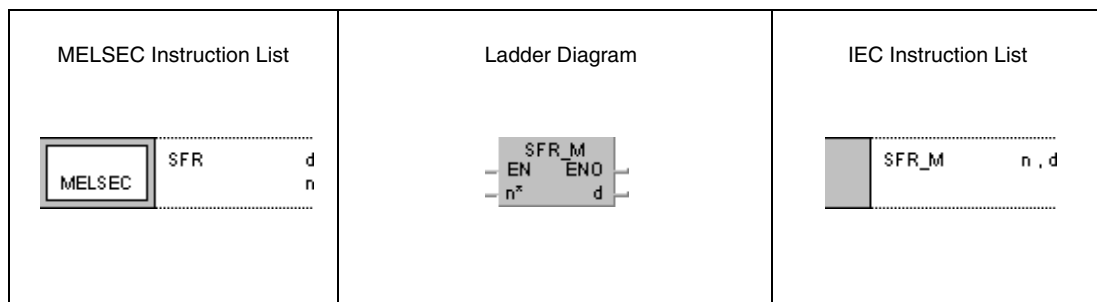
Usable Devices																			Digit designation K1 ↓ K4	Number of steps 3 <sup>1</sup>	Index ●	Carry Flag M9012	Error Flag M9010 M9011
Bit Devices						Word Devices (16-bit)						Constant		Pointer		Level							
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
d	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
n																●	●						

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

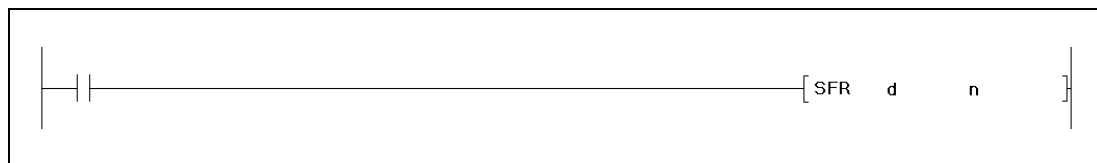
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
d	●	●	●	●	●	●	—	—	—	3	
n	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



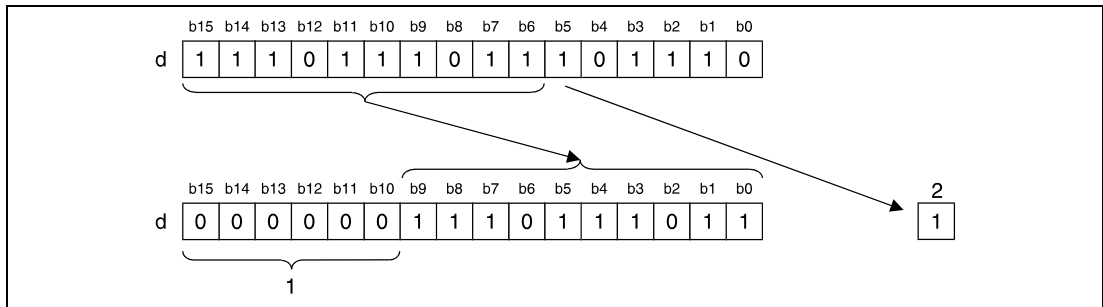
Variables

Set Data	Meaning	Data Type
d	First number of device storing data to be shifted.	BIN 16-bit
n	Number of shiftings (0 to 15).	

**Functions     Shifting a 16-bit data word by n bits**

**SFR     Shifting to the right**

The SFR instruction shifts the 16-bit data word designated by d by n bits to the right.



<sup>1</sup> These bits are set to 0

<sup>2</sup> Carry flag

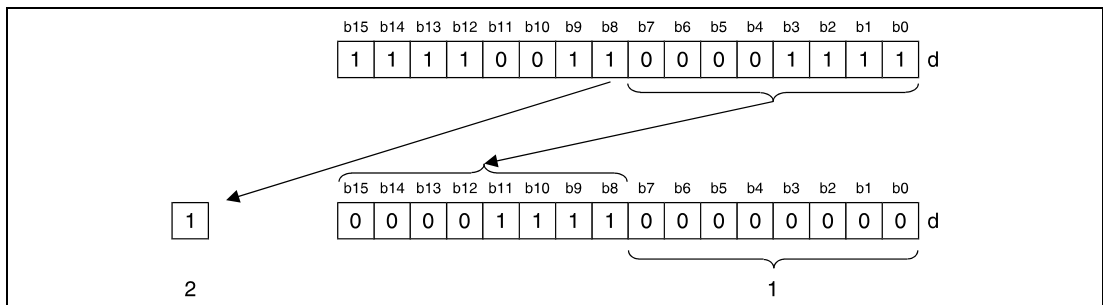
The most significant n bits beginning from bit b15 on are set to 0. The nth bit (b(n-1)) to be shifted is moved to the carry flag (A series = M9012, Q series and System Q = SM700).

For timers and counters, the actual value (count) is shifted. The setting value cannot be shifted.

For bit devices, shifting within a device with a specified number of bits is feasible (see program example 1).

**SFL     Shifting to the left**

The SFL instruction shifts the 16-bit data word designated by d by n bits to the left.



<sup>1</sup> These bits are set to 0

<sup>2</sup> Carry flag

The least significant n bits beginning from bit b0 on are set to 0. The nth bit (b(15-n)) to be shifted is moved to the carry flag (A series = M9012, Q series and System Q = SM700).

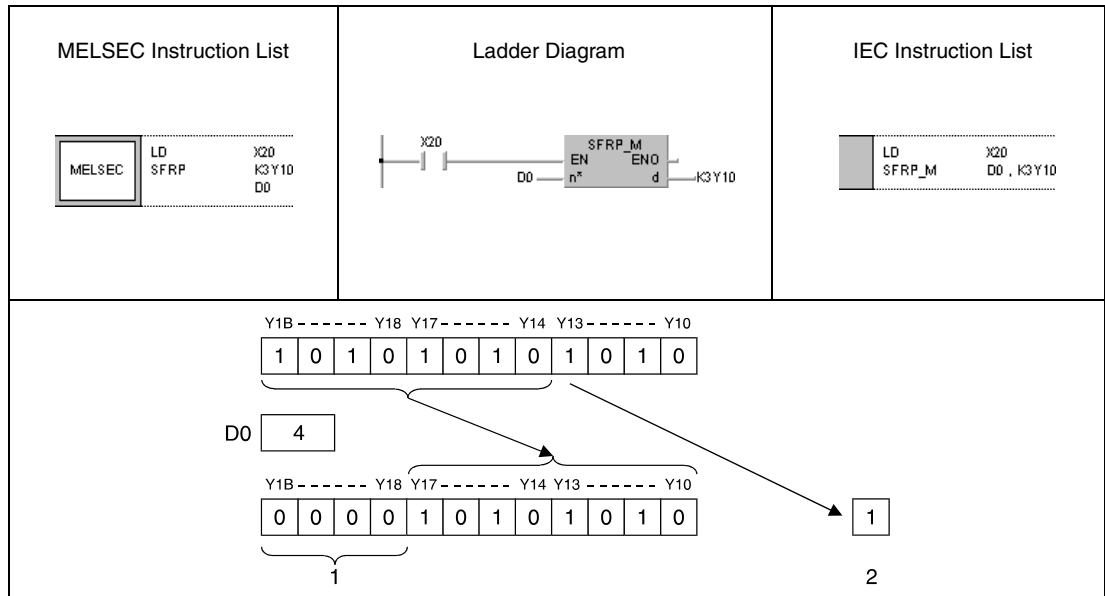
For timers and counters, the actual value (count) is shifted. The setting value cannot be shifted.

For bit devices, shifting within a device with a specified number of bits is feasible (see program example 1).

**Program Example 1**

**SFRP**

With leading edge from X20, the following program shifts the content of Y10 through Y1B by the number of bits specified by D0 to the right. The condition of bit Y13 is stored in the carry flag (A series = M9012, Q series and System Q = SM700).

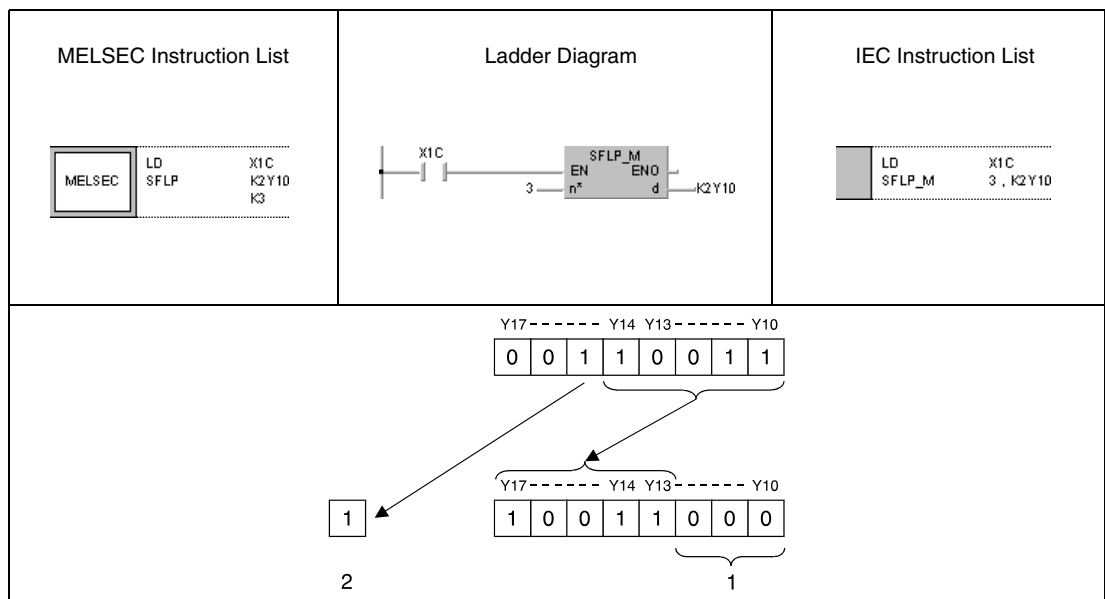


- <sup>1</sup> These bits are set to 0
- <sup>2</sup> Carry flag

**Program Example 2**

**SFLP**

With leading edge from X1C, the following program shifts the content of Y10 through Y18 by 3 bits to the left. The condition of Y15 is stored in the carry flag (A series = M9012, Q series and System Q = SM700).



- <sup>1</sup> These bits are set to 0
- <sup>2</sup> Carry flag

**7.3.2 BSFR, BSFRP, BSFL, BSFLP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices MELSEC A**

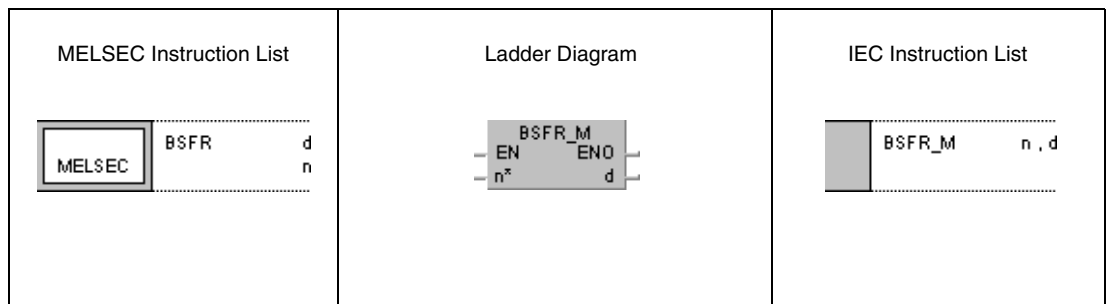
	Usable Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices							Word Devices (16-bit)							Constant		Pointer		Level					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
d	●	●	●	●	●	●																		
n																●	●				● <sup>1</sup>	●		

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

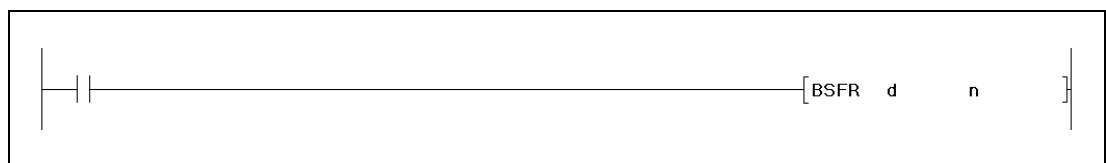
**Devices MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E	Other U		
	Bit	Word		Bit	Word						
d	●	—	—	—	—	—	—	—	—	SM0	3
n	—	●	●	●	●	●	●	●	—		

**GX IEC Developer**



**GX Developer**



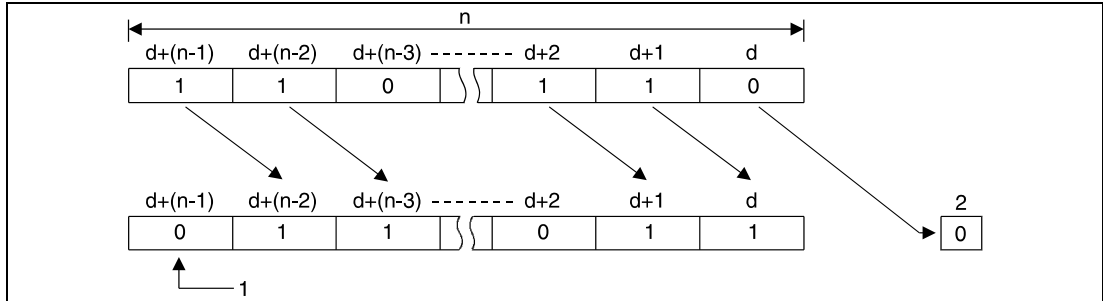
**Variables**

Set Data	Meaning	Data Type
d	First number of device to be shifted.	Bit
n	Number of devices to be shifted.	BIN 16-bit

**Functions**     **Shifting n bit devices by 1 bit**

**BSFR     Shifting to the right**

The BSFR instruction shifts the contents of specified bit devices by 1 bit to the right. The shift operation starts from the address of the device designated by d and is proceeded for the following n addresses.

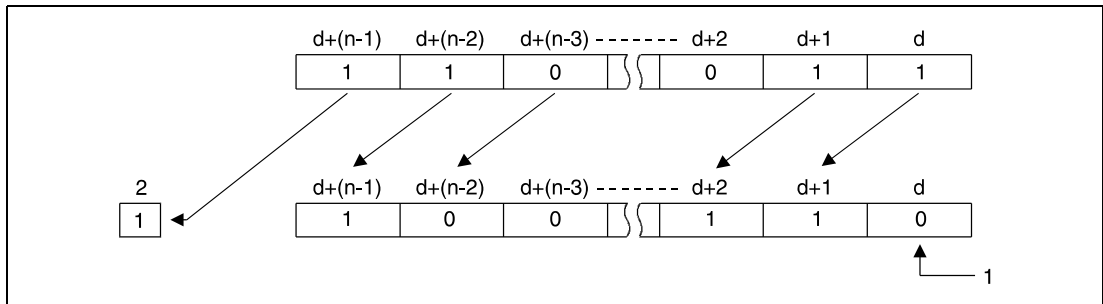


<sup>1</sup> This bit is set to 0

<sup>2</sup> Carry flag

**BSFL     Shifting to the left**

The BSFL instruction shifts the contents of specified bit devices by 1 bit to the left. The shift operation starts from the address of device designated by d and is proceeded for the following n addresses.



<sup>1</sup> This bit is set to 0

<sup>2</sup> Carry flag

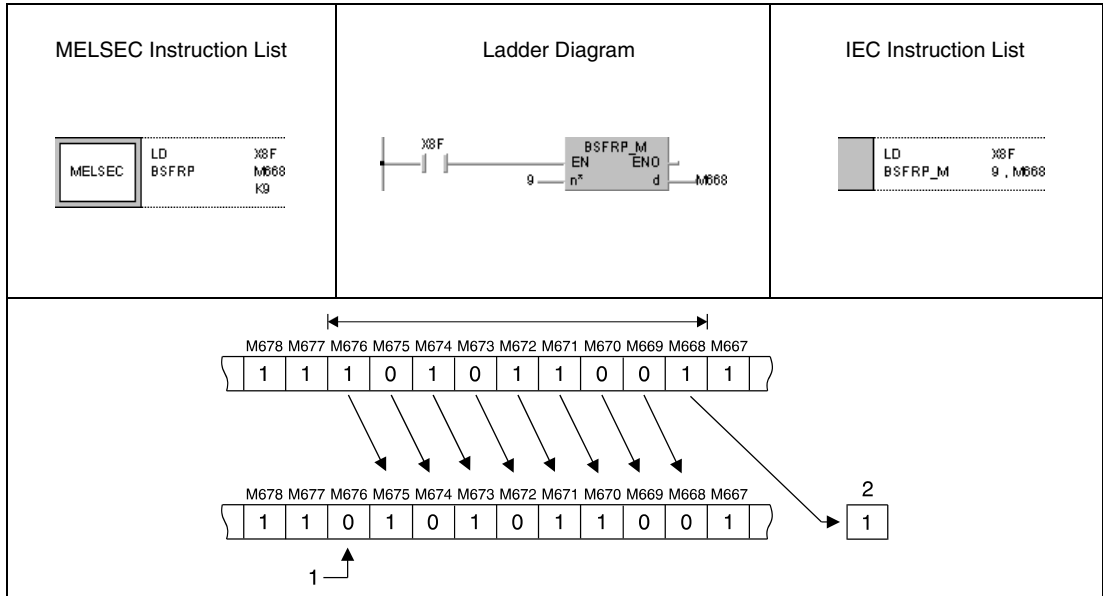
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The value in n is negative.
- The value in n exceeds the available number of bits in the device designated by d (Q series and System Q = error code 4101).

**Program Example 1** BSFRP

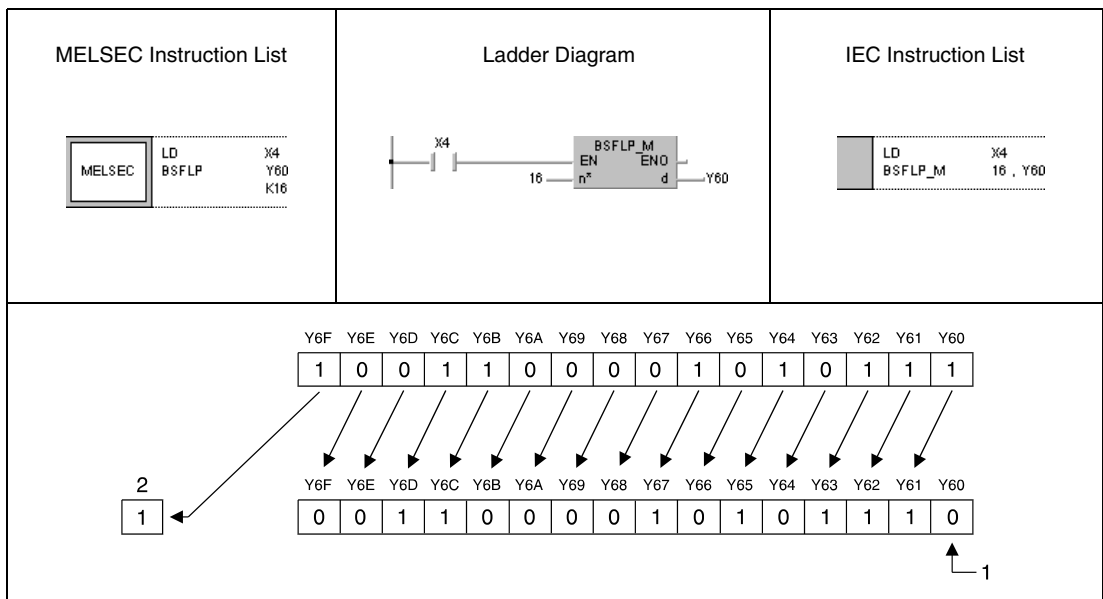
With leading edge from X8F, the following program shifts data of the internal relays M668 through M676 by one bit to the right. M668 retains the value of M669, M669 that of M670 etc. The contents of the first device (M668) is written to the carry flag (A series = M9012, Q series and System Q = SM700), and the last device (M676) retains the value 0.



- <sup>1</sup> This bit is set to 0
- <sup>2</sup> Carry flag

**Program Example 2** BSFLP

With leading edge from X4, the following program shifts the contents of the outputs Y60 through Y6F by one device to the left. The contents of the last output (Y6F) is stored in the carry flag (A series = M9012, Q series and System Q = SM700), and the first output (Y60) is reset to 0.



- <sup>1</sup> This bit is set to 0
- <sup>2</sup> Carry flag

7.3.3 DSFR, DSFRP, DSFL, DSFLP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices MELSEC A

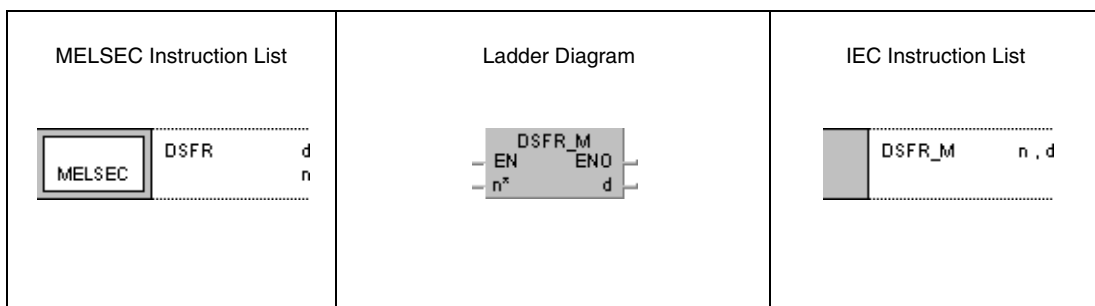
	Usable Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices							Word Devices (16-bit)							Constant		Pointer		Level					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
d							●	●	●	●	●													
n																●	●				7 <sup>1</sup>	●		

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

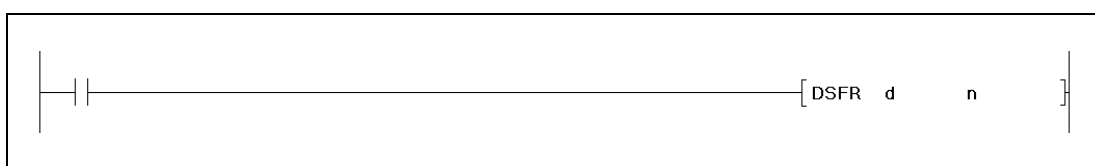
Devices MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
d	—	●	●	—	—	—	—	—	—	SM0	3
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
d	First number of device to be shifted.	BIN 16-bit
n	Number of devices to be shifted.	BIN 16-bit



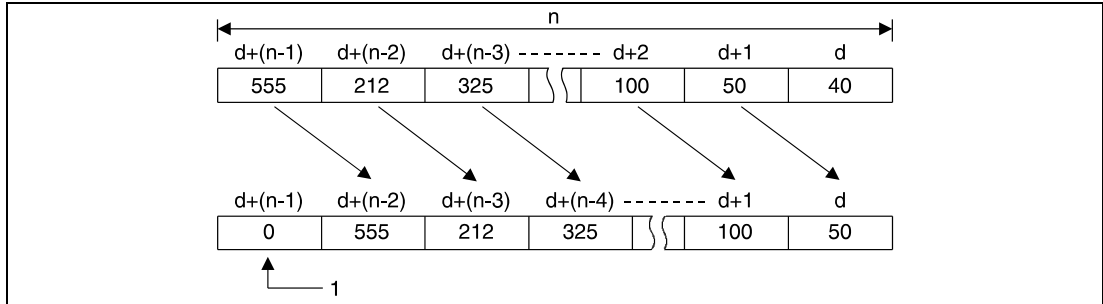
**Functions Shifting n word devices by 1 address**

**DSFR Shifting to the right**

The DSFR instruction shifts the contents of specified word devices by one address to the right. The shift operation starts from the address designated by d and is proceeded for the following n addresses.

The contents of the most significant device is reset to 0 after the shifting.

For timers and counters, the actual value (count) is shifted. The setting value cannot be shifted.



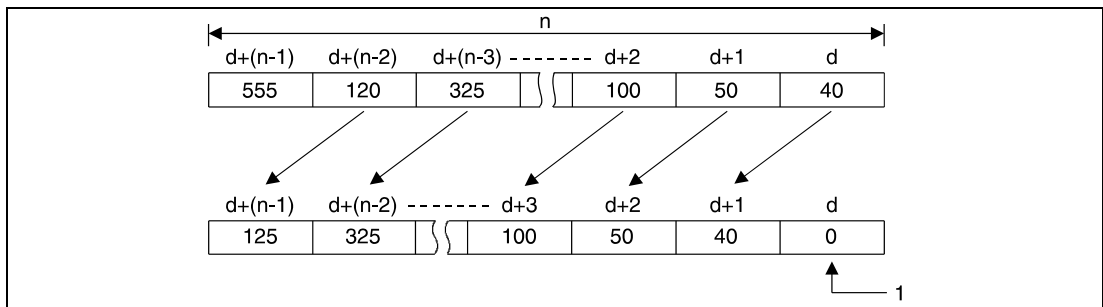
<sup>1</sup> This bit is set to 0

**DSFL Shifting to the left**

The DSFL instruction shifts the contents of specified word devices by one address to the left. The shift operation starts from the address designated by d and is proceeded for the following n addresses.

The contents of the least significant device is reset to 0 after the shifting.

For timers and counters, the actual value (count) is shifted. The setting value cannot be shifted.



<sup>1</sup> This bit is set to 0

**Operation Errors**

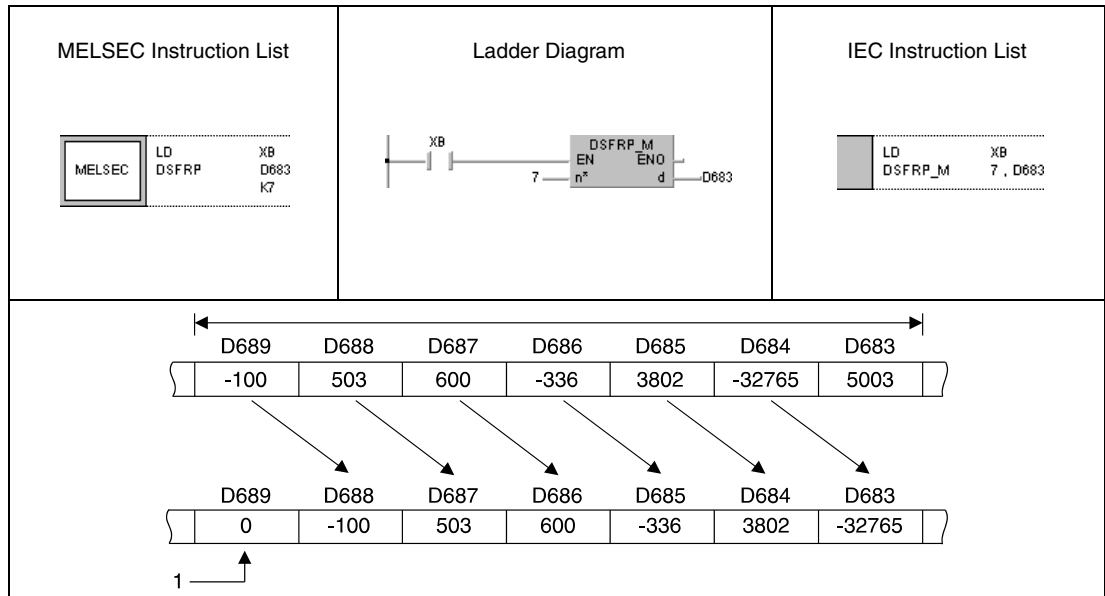
In the following cases an operation error occurs and the error flag is set:

- The value in n is negative.
- The value in n exceeds the available number of bits in the device designated by d (Q series and System Q = error code 4101).

**Program Example 1**

DSFRP

With leading edge from XB, the following program shifts data in the data registers D683 through D689 by one address to the right. D683 retains the value of D684, D684 that of D685 etc. The contents of the last data register (D689) retains the value 0.

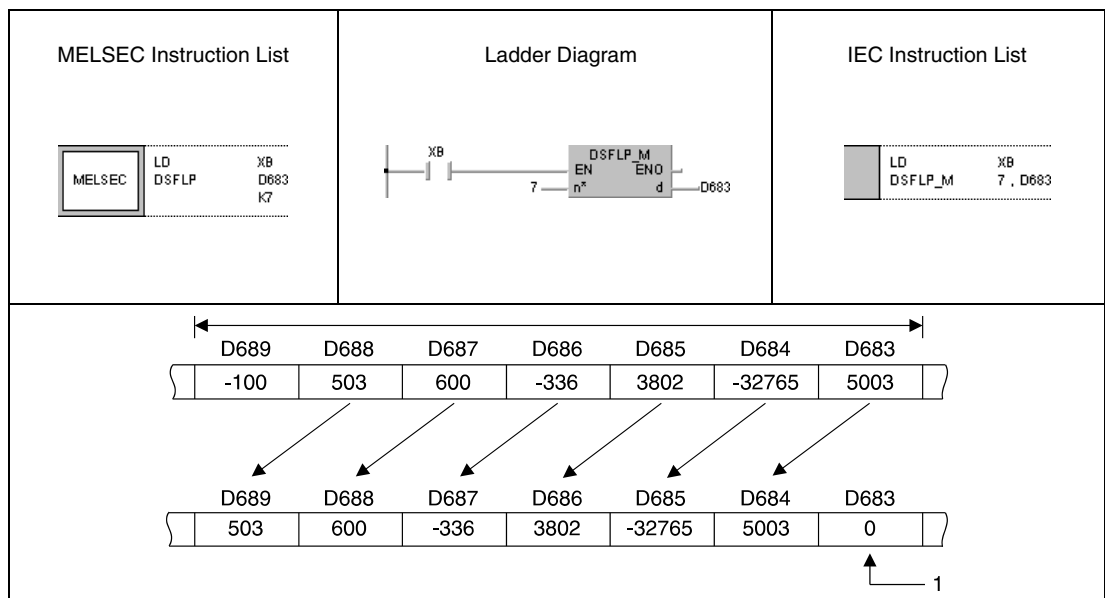


<sup>1</sup> This bit is set to 0

**Program Example 2**

DSFLP

With leading edge from XB, the following program shifts data in the data registers D683 through D689 by one address to the left. D689 retains the value of D688, D688 that of D687 etc. The contents of the first data registers (D683) retains the value 0.



<sup>1</sup> This bit is set to 0

## 7.4 Bit processing instructions

The bit processing instructions change the condition (set and reset) of single bits or entire sections of bits. The condition of bits in data words can as well be tested with the bit processing instructions.

In total, 10 bit processing instructions are supplied:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Set / reset single bits	BSET	BSET_M
	BSETP	BSETP_M
	BRST	BRST_M
	BRSTP	BRSTP_M
Test condition of single bits in 16-/32-bit data words	TEST	TEST_MD
		TEST_K_MD
	TESTP	TEST_P_MD
		TEST_K_P_MD
	DTEST	DTEST_MD
		DTEST_K_MD
	DTESTP	DTEST_P_MD
		DTEST_K_P_MD
Reset sections of bits in a batch	BKRST	BKRST_M
	BKRSTP	BKRSTP_M

7.4.1 BSET, BSETP, BRST, BRSTP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

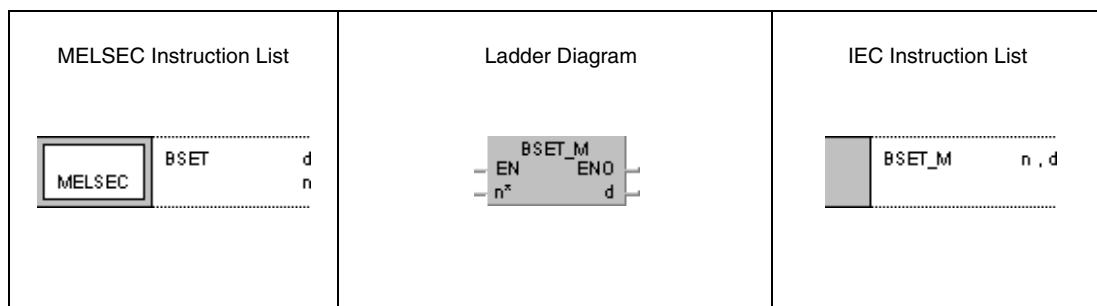
Usable Devices																				Digit designation	Number of steps	Index	Carry Flag	Error Flag	
Bit Devices								Word Devices (16-bit)								Constant		Pointer							Level
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I				N	M9012	M9010 M9011
d							●	●	●	●	●	●	●	●								7 <sup>1</sup>	●		●
n																●	●							●	

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

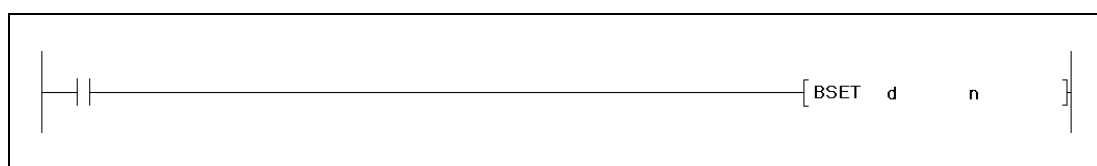
Devices  
MELSEC Q

Usable Devices									Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
Bit	Word		Bit	Word						
d	●	●	●	●	●	●	—	—	—	3
n	●	●	●	●	●	●	●	—	—	

GX IEC  
Developer



GX  
Developer



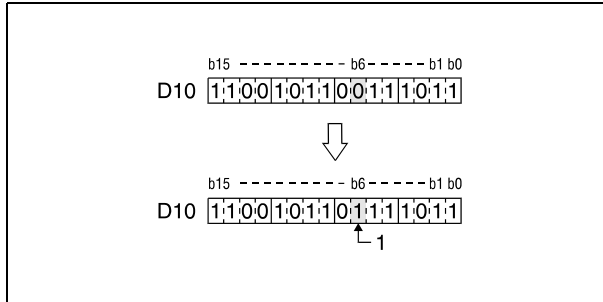
Variables

Set Data	Meaning	Data Type
d	Device storing bits to be set or reset.	BIN 16-bit
n	Number of bit to be set or reset.	

**Functions    Setting / resetting single bits**

**BSET    Setting single bits of a word device**

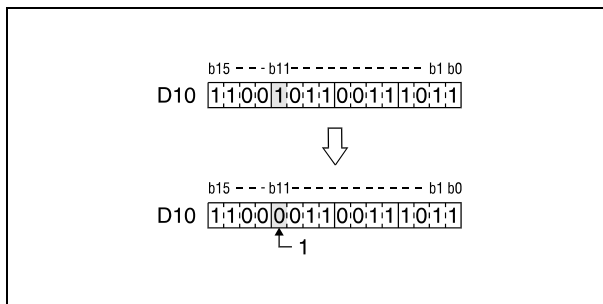
The BSET instruction sets the nth bit of a word device to 1. For n, a value between 0 and 15 (b0 to b15) can be specified. The word device is designated by d. If the value in n exceeds 15, the BSET instruction is executed within the lower 4 bits (b0 to b3). In the following diagram n is set to 6, so bit b6 is set.



<sup>1</sup> This bit is set

**BRST    Resetting single bits in a word device**

The BRST instruction resets the nth bit of a word device to 0. For n, a value between 0 and 15 (b0 to b15) can be specified. The word device is designated by d. If the value in n exceeds 15, the BRST instruction is executed within the lower 4 bits (b0 to b3). In the following diagram n is set to 11, so bit b11 is reset.



<sup>1</sup> This bit is reset

**Program Example**

**BRSTP/BSETP**

With leading edge from XB, the following program sets the bit (b3) in D8 to 1. With leading edge from the NC contact XB, the bit (b8) is reset to 0.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																																																		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 15%;">MELSEC</td> <td style="width: 15%;">LDI</td> <td style="width: 15%;">XB</td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> </tr> <tr> <td></td> <td>BRSTP</td> <td>D8</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td>K8</td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>XB</td> <td></td> <td></td> </tr> <tr> <td></td> <td>BSETP</td> <td>D8</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td>K3</td> <td></td> <td></td> </tr> </table>	MELSEC	LDI	XB				BRSTP	D8					K8				LD	XB				BSETP	D8					K3				<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"></td> <td style="width: 15%;">LDN</td> <td style="width: 15%;">XB</td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> </tr> <tr> <td></td> <td>BRSTP_M</td> <td>8, D8</td> <td></td> <td></td> </tr> <tr> <td></td> <td>LD</td> <td>XB</td> <td></td> <td></td> </tr> <tr> <td></td> <td>BSETP_M</td> <td>3, D8</td> <td></td> <td></td> </tr> </table>		LDN	XB				BRSTP_M	8, D8				LD	XB				BSETP_M	3, D8		
MELSEC	LDI	XB																																																		
	BRSTP	D8																																																		
		K8																																																		
	LD	XB																																																		
	BSETP	D8																																																		
		K3																																																		
	LDN	XB																																																		
	BRSTP_M	8, D8																																																		
	LD	XB																																																		
	BSETP_M	3, D8																																																		
<table style="margin: auto;"> <tr> <td></td> <td style="text-align: center;">b15</td> <td style="text-align: center;">-----</td> <td style="text-align: center;">b8</td> <td style="text-align: center;">-----</td> <td style="text-align: center;">b3</td> <td style="text-align: center;">--</td> <td style="text-align: center;">b0</td> </tr> <tr> <td>D8</td> <td colspan="7" style="text-align: center;">0011010111110001</td> </tr> </table> <table style="margin: auto;"> <tr> <td></td> <td style="text-align: center;">b15</td> <td style="text-align: center;">-----</td> <td style="text-align: center;">b8</td> <td style="text-align: center;">-----</td> <td style="text-align: center;">b3</td> <td style="text-align: center;">--</td> <td style="text-align: center;">b0</td> </tr> <tr> <td>D8</td> <td colspan="7" style="text-align: center;">0011010011111001</td> </tr> </table>				b15	-----	b8	-----	b3	--	b0	D8	0011010111110001								b15	-----	b8	-----	b3	--	b0	D8	0011010011111001																								
	b15	-----	b8	-----	b3	--	b0																																													
D8	0011010111110001																																																			
	b15	-----	b8	-----	b3	--	b0																																													
D8	0011010011111001																																																			

**NOTE**

Single bits in bit devices can be set or reset via a SET or an RST instruction as well. In this case the bits in the data words must be specified for addressing the registers. For example, the bit (b8) in data word D8 is addressed as D8.8.

**7.4.2 TEST, TESTP, DTEST, DTESTP**

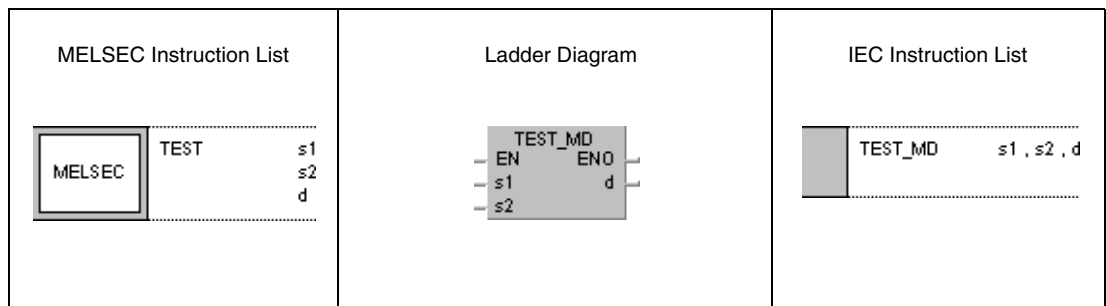
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

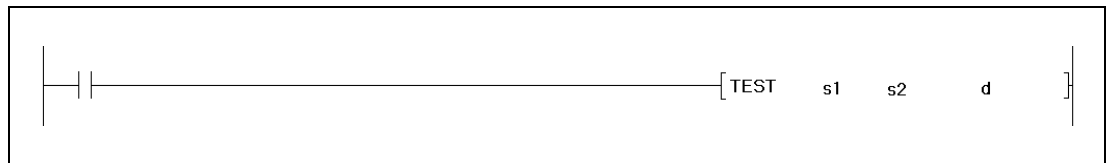
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	—	—	—	4	
s2	●	●	●	●	●	●	●	—	—		
d	●	—	—	●	—	—	—	—	—		

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s1	Number of device storing bits to be tested.	Word
s2	Number of bit to be tested.	Word
d	Number of bit device storing condition of tested bit.	Bit

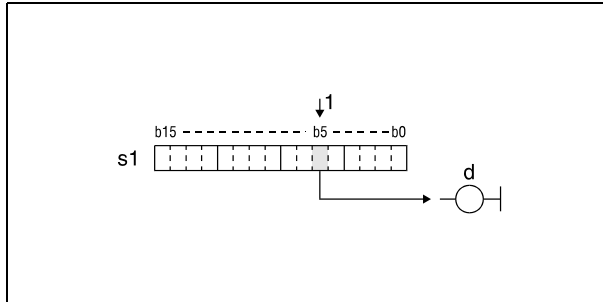
**Functions Test of single bits in 16- / 32-bit data words**

**TEST Bit test 16-bit**

The TEST instruction checks the condition of a bit s2 in a word device s1. The test result is stored in a bit device designated by d.

The device designated by d is set, if the tested bit is in condition 1, and reset, if the tested bit is in condition 0.

The bit specified by s2 can be any bit between b0 and b15 in a 16-bit data word. In the following diagram s2 is set to 5, so the condition of bit b5 in s1 is tested.



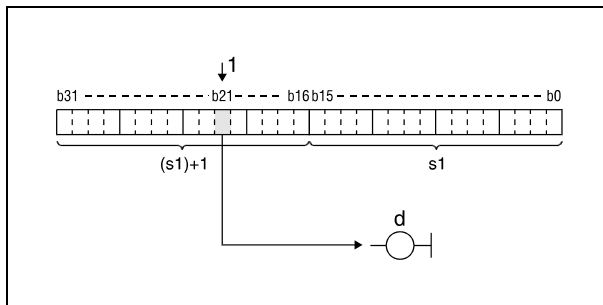
<sup>1</sup> Tested bit

**DTEST Bit test 32-bit**

The DTEST instruction checks the condition of a bit s2 in a word device s1 and (s1)+1. The test result is stored in a bit device designated by d.

The device designated by d is set, if the tested bit is in condition 1, and reset, if the tested bit is in condition 0.

The bit specified by s2 can be any bit between b0 and b31 in a 32-bit data word. In the following diagram s2 is set to 21, so the condition of bit b21 in s1 is tested.

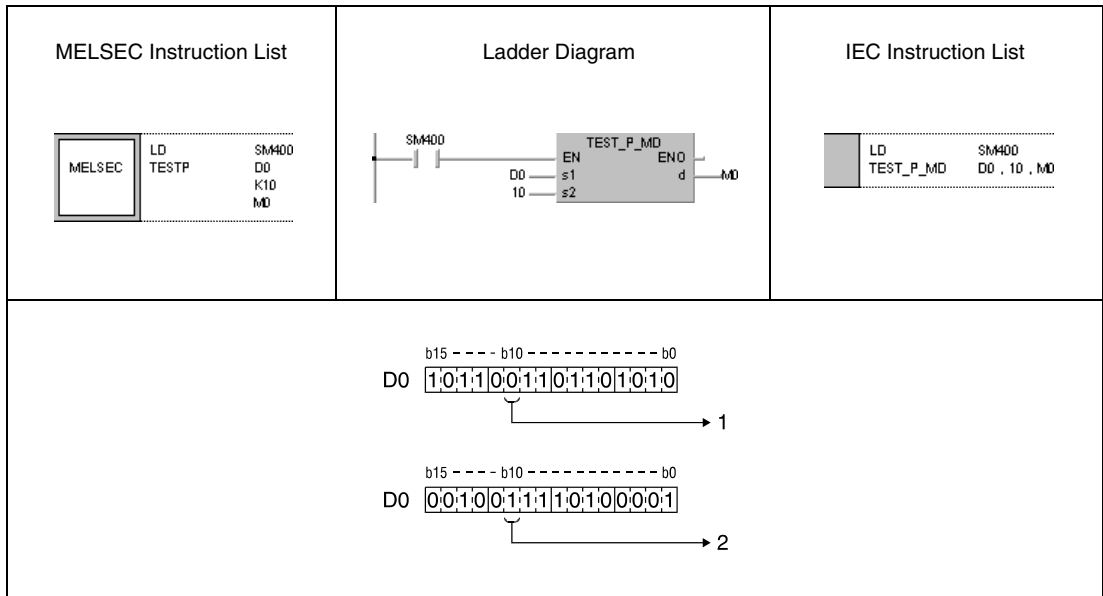


<sup>1</sup> Tested bit



**Program Example 1** TESTP

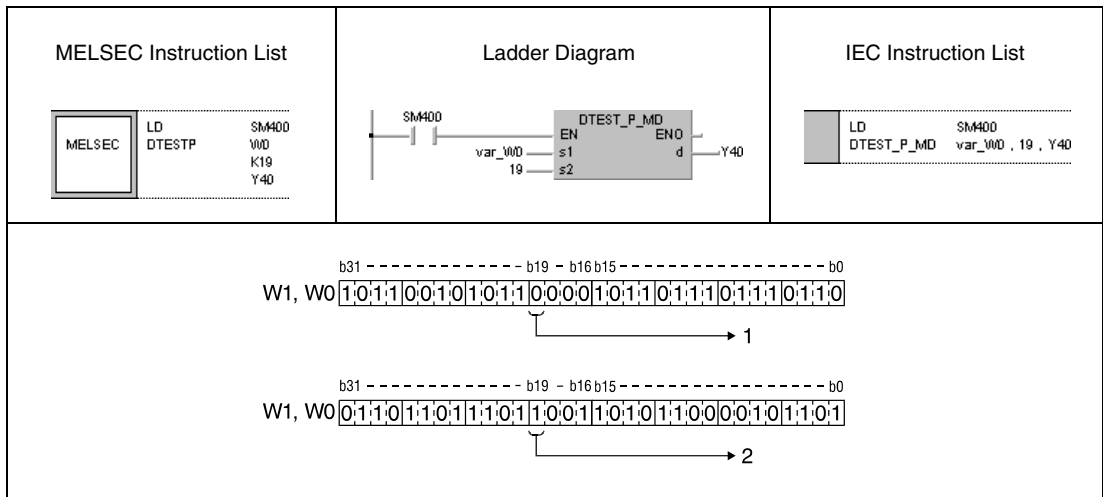
With leading edge from SM400 and depending on the test result of the bit (b10) in the 16-bit data word in D0, the following program either resets or sets relay M0.



- 1 Reset
- 2 Set

**Program Example 2** DTESTP

With leading edge from SM400 and depending on the test result of the bit (b19) in the 32-bit data word in W0 and W1, the following program either resets or sets output Y40.



- 1 Reset
- 2 Set

**NOTE**

*The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

*Instead of applying the TEST instruction, a bit to be tested can also be specified as an input contact (see diagram).*



**7.4.3 BKRST, BKRSTP**

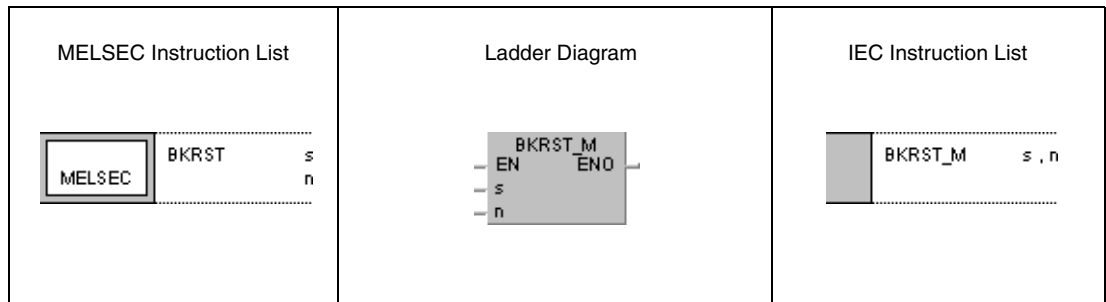
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

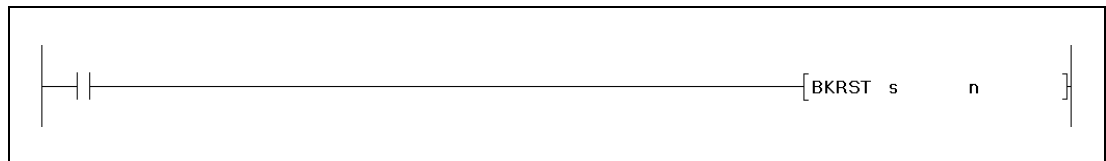
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	—	—	—	—	—	SM0	3	
n	●	●	●	●	●	●	●	—			

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	First number of device to be reset.	Bit
n	Number of devices to be reset.	BIN 16-bit

**Functions Batch reset of bits**

**BKRST Reset instruction**

The BKRST instruction resets n bits in the device designated by s.

For annunciators (F), the number n of annunciators stored in s is reset and the contents of the registers SD64 through SD79 is cleared according to the reset annunciators. The remaining data are shifted forward. Moreover, the number of annunciator entries in registers SD64 through SD79 is stored in register SD63.

For timers (T) and counters (C), after the execution of this instruction the setting values of n timers and counters are reset to 0 and the coil contacts are reset.

For all other bit devices the number n of coils or contacts in the device designated by s are reset.

If the according device is already reset, its condition remains unchanged after execution of the instruction.

**Operation Errors**

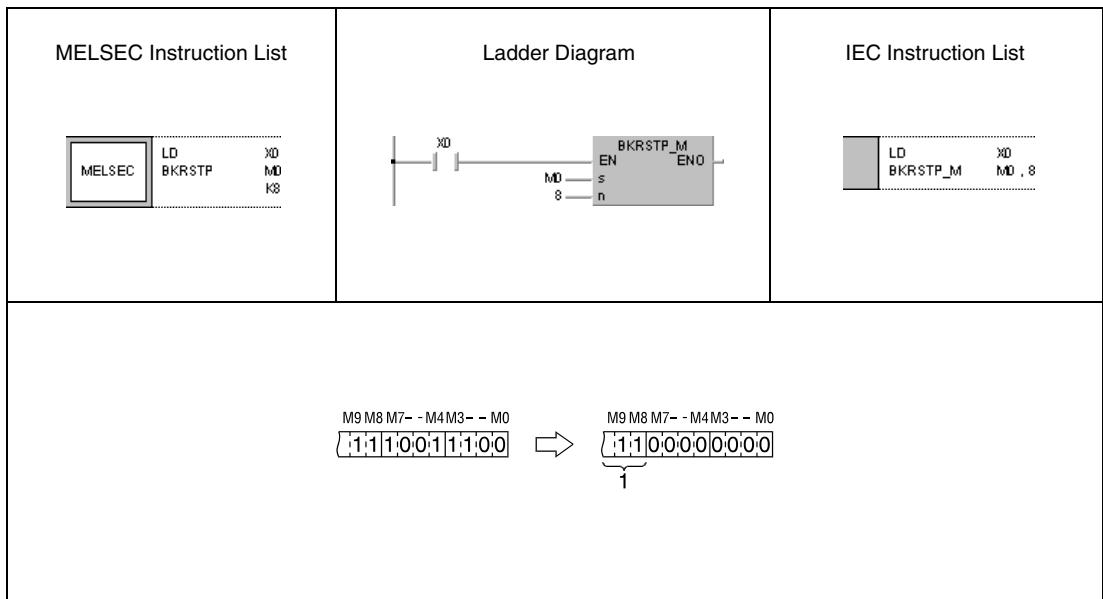
In the following cases an operation error occurs and the error flag is set:

- The value in n exceeds the number of bits of the devices designated by s (error code 4101).

**Program Example 1**

**BKRSTP**

With leading edge from X0, the following program resets the relays M0 through M7.



<sup>1</sup> These bits remain unchanged

**Program Example 2**

**BKRSTP**

With leading edge from X20, the following program resets bits from the bit (b2) in D10 to the bit (b1) in D11.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																														
<table border="1" style="border-collapse: collapse; margin: auto;"> <tr> <td style="padding: 5px;">MELSEC</td> <td style="padding: 5px;">LD</td> <td style="padding: 5px;">X20</td> </tr> <tr> <td style="padding: 5px;"></td> <td style="padding: 5px;">BKRSTP</td> <td style="padding: 5px;">D10.2 K16</td> </tr> </table>	MELSEC	LD	X20		BKRSTP	D10.2 K16		<table border="1" style="border-collapse: collapse; margin: auto;"> <tr> <td style="padding: 5px;">LD</td> <td style="padding: 5px;">X20</td> </tr> <tr> <td style="padding: 5px;">BKRSTP_M</td> <td style="padding: 5px;">D10.2, 16</td> </tr> </table>	LD	X20	BKRSTP_M	D10.2, 16																				
MELSEC	LD	X20																														
	BKRSTP	D10.2 K16																														
LD	X20																															
BKRSTP_M	D10.2, 16																															
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%; text-align: center;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b2b1b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D10</td> <td style="text-align: center;">1·1·1·0·0·0·0·1·1·1·1·1·1·1·0</td> <td></td> </tr> </table> </td> <td style="width: 33%; text-align: center; vertical-align: middle;">➔</td> <td style="width: 33%; text-align: center;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b2b1b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D10</td> <td style="text-align: center;">0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0</td> <td></td> </tr> </table> </td> </tr> <tr> <td style="text-align: center;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b1 b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D11</td> <td style="text-align: center;">0·0·0·1·0·1·0·0·1·1·1·1·1·1·1·1</td> <td></td> </tr> </table> </td> <td></td> <td style="text-align: center;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b1 b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D11</td> <td style="text-align: center;">0·0·0·1·0·1·0·0·1·1·1·1·1·1·0·0</td> <td></td> </tr> </table> </td> </tr> </table>			<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b2b1b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D10</td> <td style="text-align: center;">1·1·1·0·0·0·0·1·1·1·1·1·1·1·0</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b2b1b0		D10	1·1·1·0·0·0·0·1·1·1·1·1·1·1·0		➔	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b2b1b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D10</td> <td style="text-align: center;">0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b2b1b0		D10	0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0		<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b1 b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D11</td> <td style="text-align: center;">0·0·0·1·0·1·0·0·1·1·1·1·1·1·1·1</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b1 b0		D11	0·0·0·1·0·1·0·0·1·1·1·1·1·1·1·1			<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b1 b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D11</td> <td style="text-align: center;">0·0·0·1·0·1·0·0·1·1·1·1·1·1·0·0</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b1 b0		D11	0·0·0·1·0·1·0·0·1·1·1·1·1·1·0·0	
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b2b1b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D10</td> <td style="text-align: center;">1·1·1·0·0·0·0·1·1·1·1·1·1·1·0</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b2b1b0		D10	1·1·1·0·0·0·0·1·1·1·1·1·1·1·0		➔	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b2b1b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D10</td> <td style="text-align: center;">0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b2b1b0		D10	0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0																			
	b15 ----- b8b7 ----- b2b1b0																															
D10	1·1·1·0·0·0·0·1·1·1·1·1·1·1·0																															
	b15 ----- b8b7 ----- b2b1b0																															
D10	0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0																															
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b1 b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D11</td> <td style="text-align: center;">0·0·0·1·0·1·0·0·1·1·1·1·1·1·1·1</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b1 b0		D11	0·0·0·1·0·1·0·0·1·1·1·1·1·1·1·1			<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b15 ----- b8b7 ----- b1 b0</td> <td style="width: 33%;"></td> </tr> <tr> <td style="text-align: center;">D11</td> <td style="text-align: center;">0·0·0·1·0·1·0·0·1·1·1·1·1·1·0·0</td> <td></td> </tr> </table>		b15 ----- b8b7 ----- b1 b0		D11	0·0·0·1·0·1·0·0·1·1·1·1·1·1·0·0																			
	b15 ----- b8b7 ----- b1 b0																															
D11	0·0·0·1·0·1·0·0·1·1·1·1·1·1·1·1																															
	b15 ----- b8b7 ----- b1 b0																															
D11	0·0·0·1·0·1·0·0·1·1·1·1·1·1·0·0																															

## 7.5 Data processing instructions

Data processing instructions search data in specified devices, check the number of set bits, encode and decode data (e.g. for 7-segment displays), disunite and unite data, search maximum and minimum values, sort data, and calculate the totals of 16-/32-bit BIN data blocks.

In total, 41 different data processing instructions are supplied:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Search 16-/32-bit data	SER	SER_M
	SERP	SERP_M
	DSER	DSER_M
	DSERP	DSERP_M
Check data bits (16-/32-bit)	SUM	SUM_M
	SUMP	SUMP_M
	DSUM	DSUM_M
	DSUMP	DSUMP_M
Encode/decode data	DECO	DECO_M
	DECOP	DECOP_M
	ENCO	ENCO_M
	ENCOP	ENCOP_M
7-segment decoding	SEG	SEG_M
Disunite/unite 16-bit data words (4-bit units)	DIS	DIS_M
	DISP	DISP_M
	UNI	UNI_M
	UNIP	UNIP_M
Disunite/unite 16-bit data values (variable bit units)	NDIS	NDIS_M
	NDISP	NDISP_M
	NUNI	NUNI_M
	NUNIP	NUNIP_M
Disunite/unite 16-bit data values (byte units)	WTOB	WTOB_MD
		WTOB_K_MD
	WTOBP	WTOB_P_MD
		WTOB_K_P_MD
	BTOW	BTOW_MD
		BTOW_K_MD
	BTOWP	BTOW_P_MD
		BTOW_K_P_MD
Search maximum values in 16-/32-bit data	MAX	MAX_M
	MAXP	MAXP_M
	DMAX	DMAX_M
	DMAXP	DMAXP_M
Search minimum values in 16-/32-bit data	MIN	MIN_M
	MINP	MINP_M
	DMIN	DMIN_M
	DMINP	DMINP_M

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Sort 16-/32-bit data	SORT	SORT_M
	SORTP	SORTP_M
	DSORT	DSORT_M
	DSORTP	DSORTP_M
Calculate totals of 16-/32-bit BIN data blocks	WSUM	WSUM_M
	WSUMP	WSUMP_M
	DWSUM	DWSUM_M
	DWSUMP	DWSUMP_M

7.5.1 SER, SERP, DSER , DSERP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

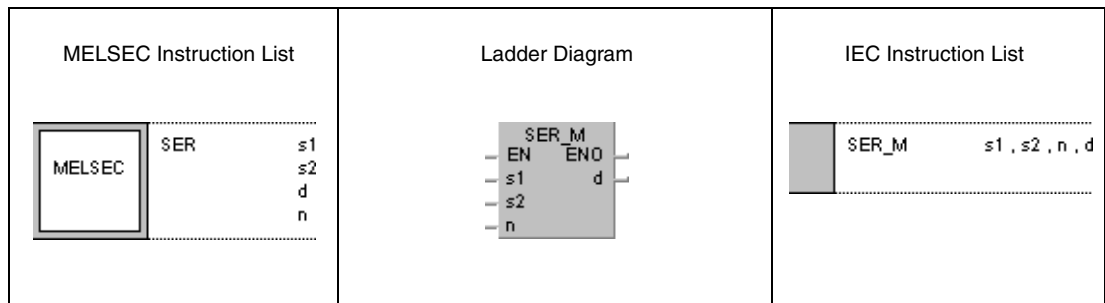
	Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V					
s1							●	●	●	●	●	●	●	●	●	●	●				
s2							●	●	●	●	●										
n																●	●				

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

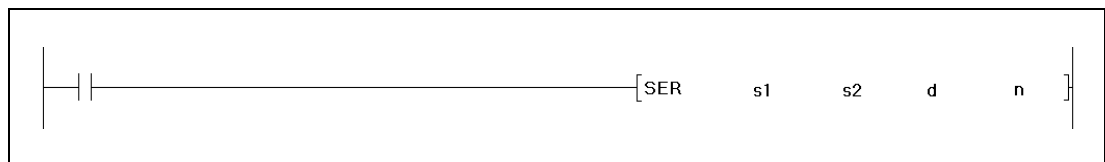
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other U
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—		
s2	—	●	●	—	—	—	—	—	—		
d	—	●	●	—	●	●	●	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC  
Developer



GX  
Developer





**NOTE** *The A series always stores the search results in registers A0 and A1. For this reason, there is no device d available when programming this operation instruction for the A series.*

**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s1	Data value to be searched, or first number of device storing this value.	Word	ANY16
s2	Data to be searched through, or first number of device storing such data.		ANY16/ANY32
d	First number of device storing result of search. A series: device A0 and A1 only.		Array [1..2] of ANY16/ANY32
n	Number of devices to be searched through.		ANY16

**Functions**

**Search data**

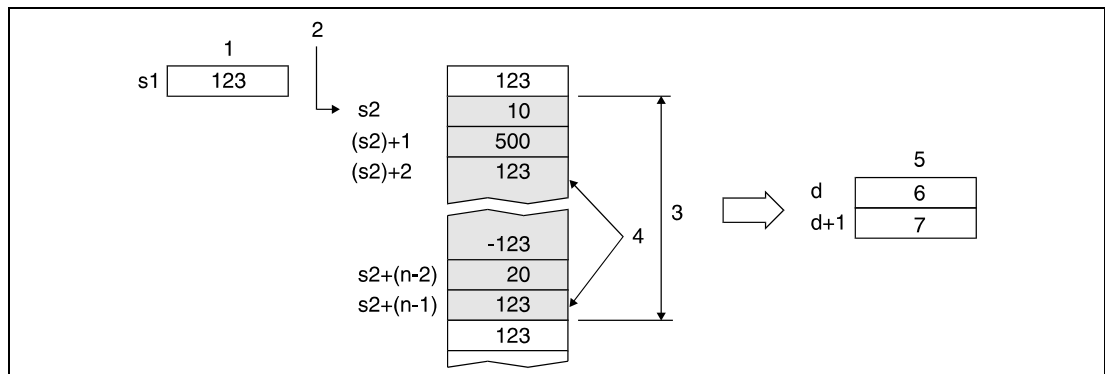
**SER (A and Q series/System Q) / SERP (Q series and System Q) Search 16-bit data**

The SER instruction enables searching specified data in a specified search range. The search operation starts from the first number of device designated by s2. The entry code being searched for is specified by s1. The digit designation, i.e. the number of devices is specified by n.

A CPU of the Q series or the System Q stores the result of the search in d and d+1 as array [1..2] of ANY16.

After finishing the search operation the position of the first device storing the data value is stored in array[1] in d. Array[2] in d+1 stores the number of data values matching the entry code.

The A series stores the position of the first device storing the matching data value in register A0. The number of matches is stored in register A1.



- 1 Entry code
- 2 Start of search
- 3 Search range (n blocks)
- 4 Matching data
- 5 Search results
- 6 Position of match
- 7 Number of matches

If the value in n is less than or equal to 0, the search operation will not be executed. If no matching data is found, the content of d and d+1 (A series = A0 and A1) is 0.

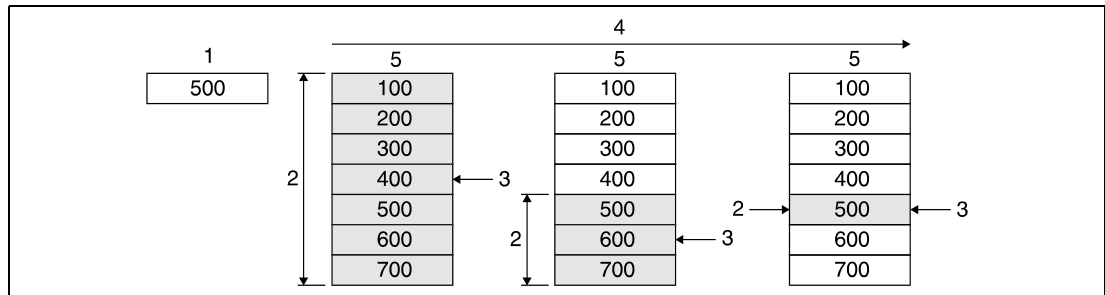
**NOTE**

*Q series and System Q*

*Provided the data to be searched through is stored in ascending order, the searching time can be shortened by setting the special relay SM702.*

SM702 ON:

*The search range is halved and the size of the entry code determines in what half the code must be stored. This half is divided once again for another decision. This operation is proceeded until the matching value is found.*



- 1 Entry code
- 2 Search range
- 3 Comparison to entry code
- 4 Processing sequence
- 5 Search data

SM702 OFF:

*The data search comparing the entry code to each data value starts from the beginning of the search range.*

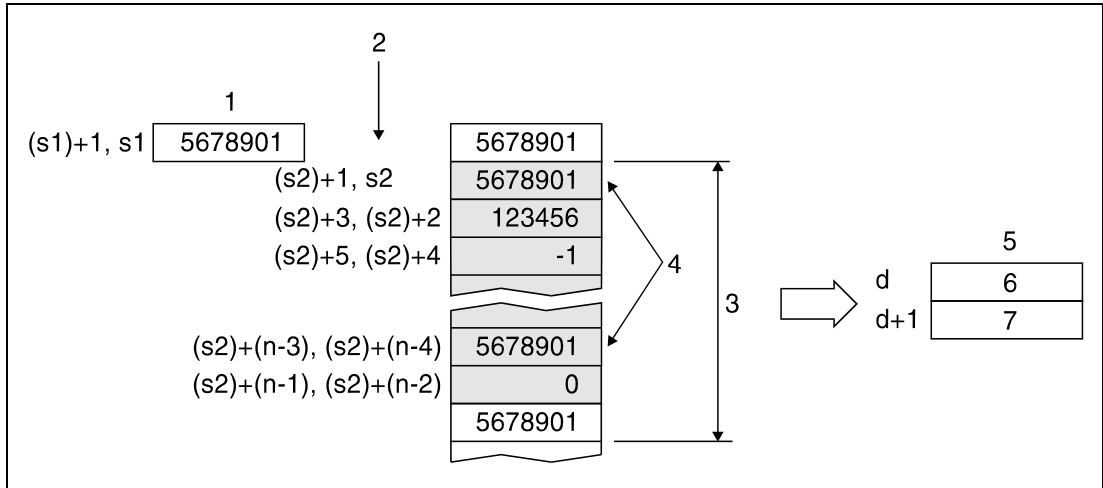
*If the search range is not sorted in ascending order, there will be no accurate result with SM702 set.*

**DSER / DSERP (Q series and System Q) Search 32-bit data**

The DSER instruction enables searching specified data in a specified search range. The search operation starts from the first number of device designated by s2 (2 x n-devices). The entry code being searched for is specified by s1 and (s1)+1. The digit designation, i.e. the number of devices is specified by n.

The result of the search is stored in d and d1 as array [1..2] of ANY16.

After finishing the search operation the position of the first device storing the data value is stored in the least significant array (d). The most significant array (d+1) stores the number of data values matching the entry code.



- 1 Entry code
- 2 Start of search
- 3 Search range (2 x n)
- 4 Matching data
- 5 Search results
- 6 Position of match
- 7 Number of matches

If the value in n is less than or equal to 0, the search operation will not be executed. If no matching data is found, the content of d and d+1 is 0.

**Operation Errors**

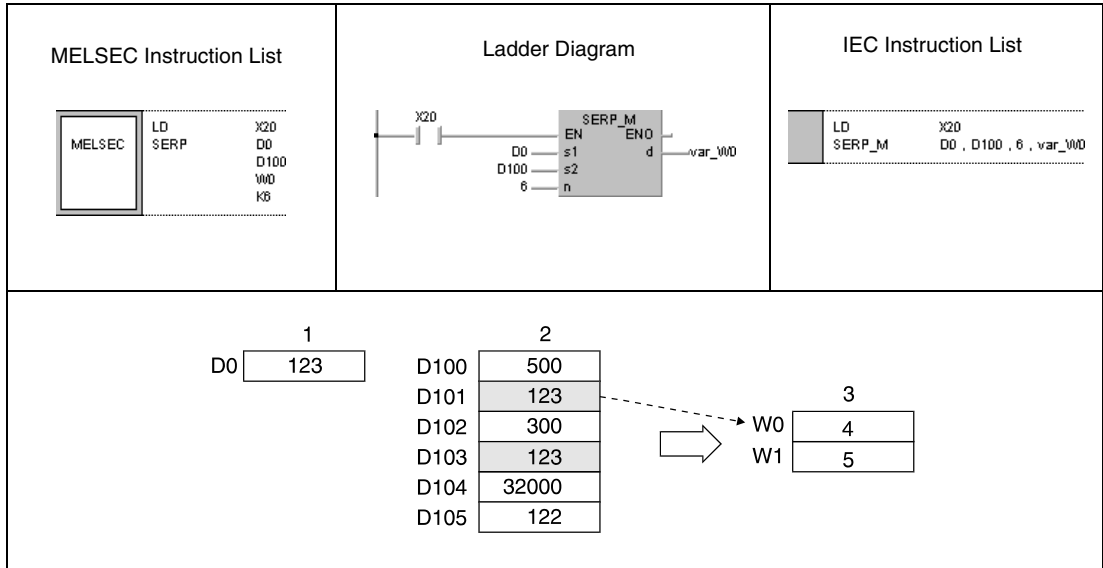
In the following cases an operation error occurs and the error flag is set:

- The search range designated by n beginning from s2 exceeds the relevant device range (Q series and System Q = error code 4101)

For details on index qualification refer to chapter 3.6.

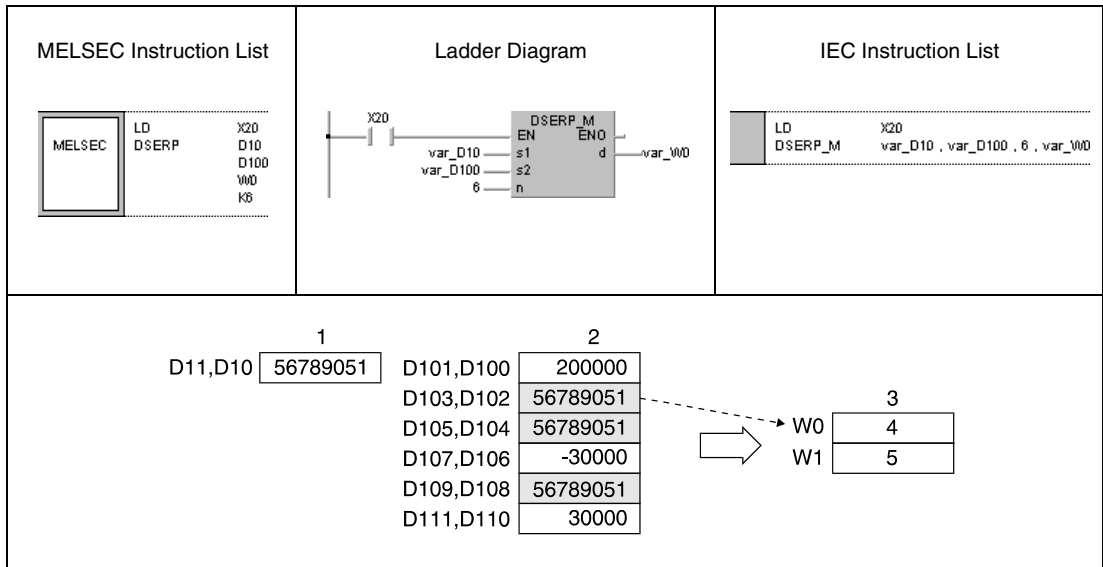
**Program Example 1** SERP (Q series and System Q)

With leading edge from X20, the following program compares data in D100 through D105 to the data value in D0. The first matching position is stored in W0. The number of matches is stored in W1.



**Program Example 2** DSERP (Q series and System Q)

With leading edge from X20, the following program compares data in D100 through D111 to the data value in D11 and D10. The first matching position is stored in W0. The number of matches is stored in W1.



- 1 Entry code
- 2 Search range
- 3 Search results
- 4 Position of first match
- 5 Number of matches

**NOTE** *These programs will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.5.2 SUM, SUMP, DSUM, DSUMP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

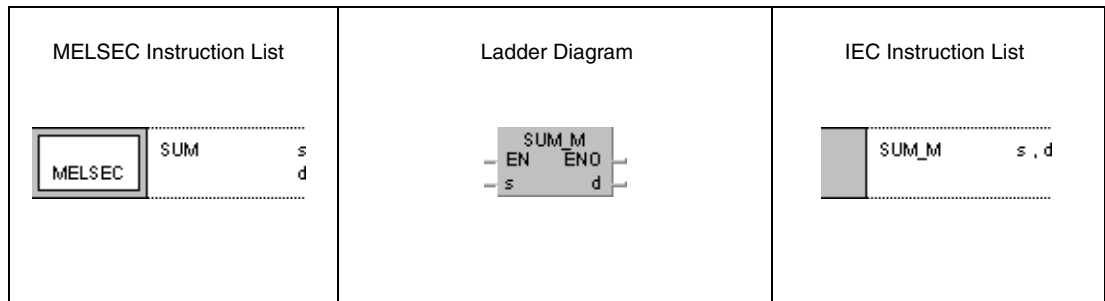
Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag				
Bit Devices					Word Devices (16-bit)						Constant		Pointer		Level										
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I	N	K1 ↓ K4	3 ↓ 1	M9012	M9010 M9011	
SUM																									
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							●	●		●
DSUM																									
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							●	●		●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

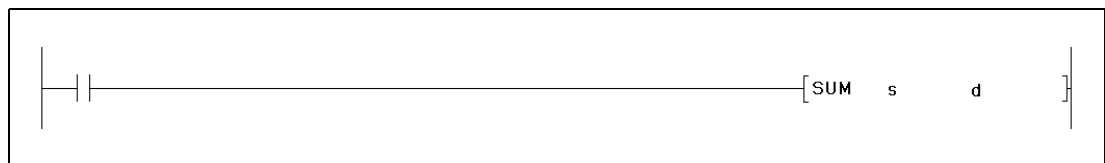
Devices  
MELSEC Q

Usable Devices								Error Flag	Number of steps	
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	3	
d	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



NOTE

The A series always stores the number of set bits in register A0. For this reason, there is no device d available when programming this operation instruction for the A series.

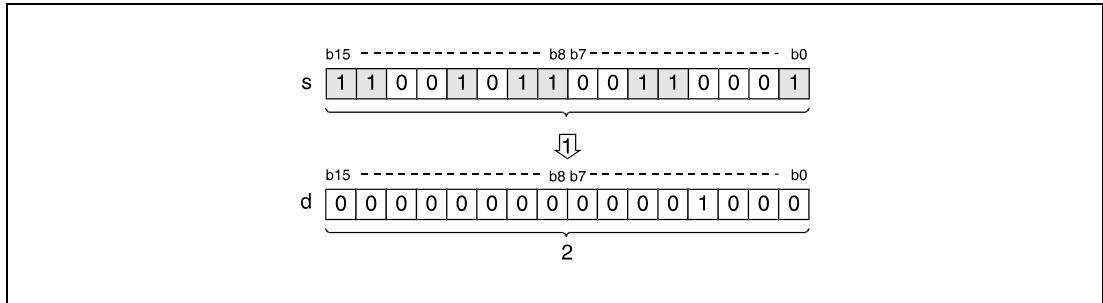
Variables

Set Data	Meaning	Data Type
s	First number of device storing data of which set bits are counted.	BIN 16-/32-bit
d	First number of device storing number of set bits. A series: device A0 only.	

**Functions Check data bits**

**SUM 16-bit**

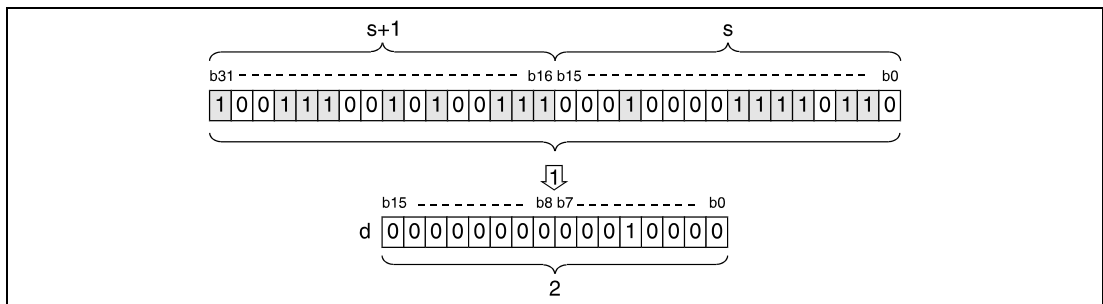
The SUM instruction determines the number of bits set in a 16-bit data word. The device range to be checked is specified by s. The number of set bits is stored in d (A0).



- <sup>1</sup> Counting set bits
- <sup>2</sup> Binary coded number of bits

**DSUM 32-bit**

The DSUM instruction determines the number of bits set in a 32-bit data word. The device range to be checked is specified by s. The number of set bits is stored in d (A0).

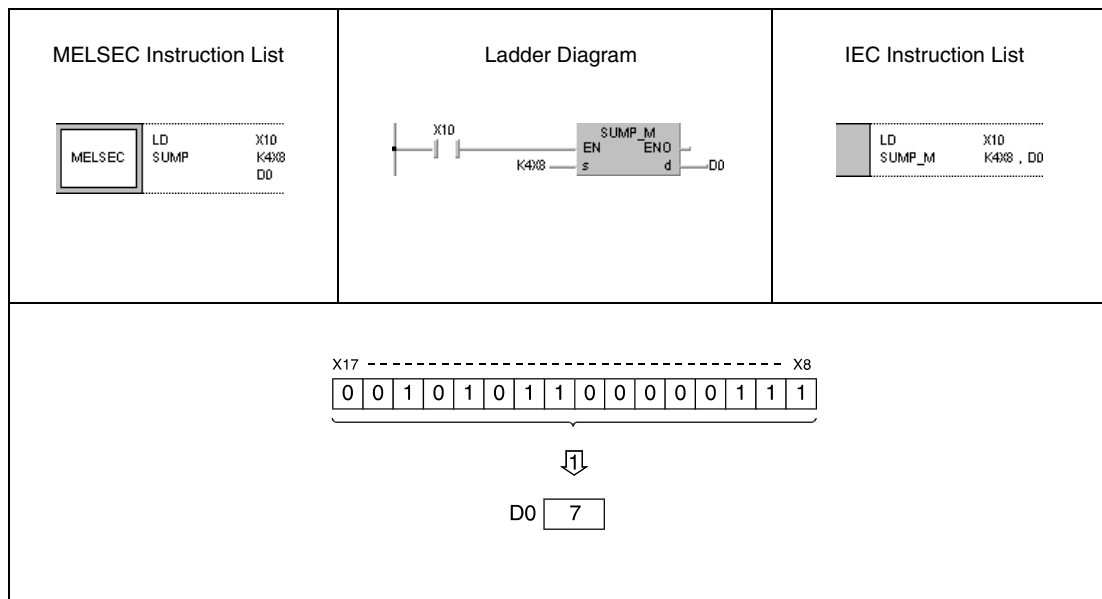


- <sup>1</sup> Counting set bits
- <sup>2</sup> Binary coded number of bits

**Program Example 1**

SUMP (Q series and System Q)

With leading edge from X10, the following program determines the number of set inputs within X8 through X10. The result is stored in D0.

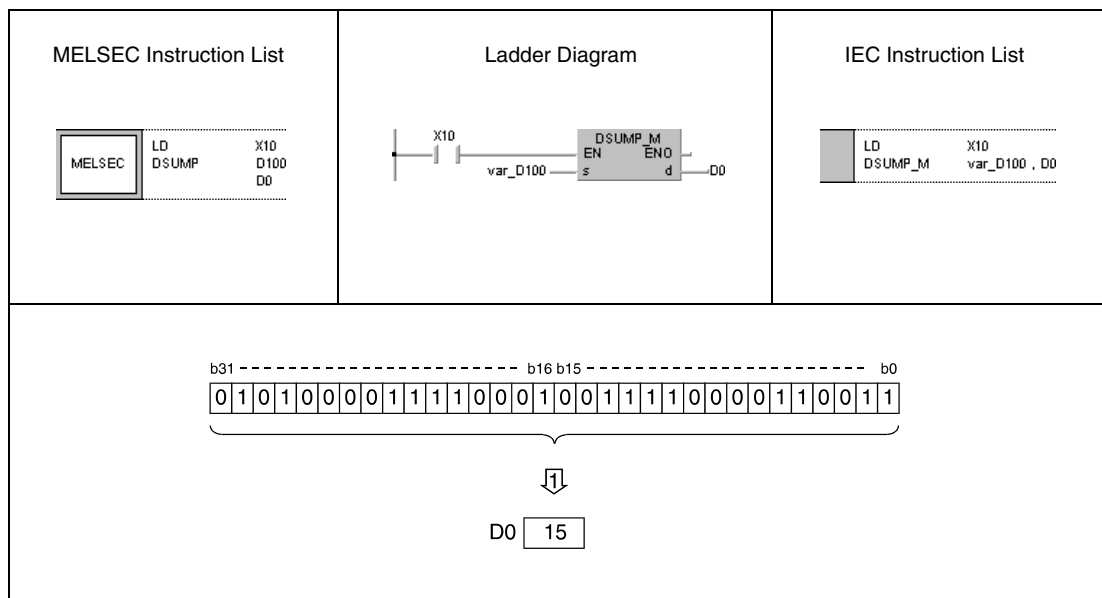


<sup>1</sup> Storing the number of set bits in D0

**Program Example 2**

DSUMP (Q series and System Q)

With leading edge from X10, the following program determines the number of set bits in D100 and D101. The result is stored in D0.



<sup>1</sup> Storing the number of set bits in D0

**NOTE**

The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.



### 7.5.3 DECO, DECOP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

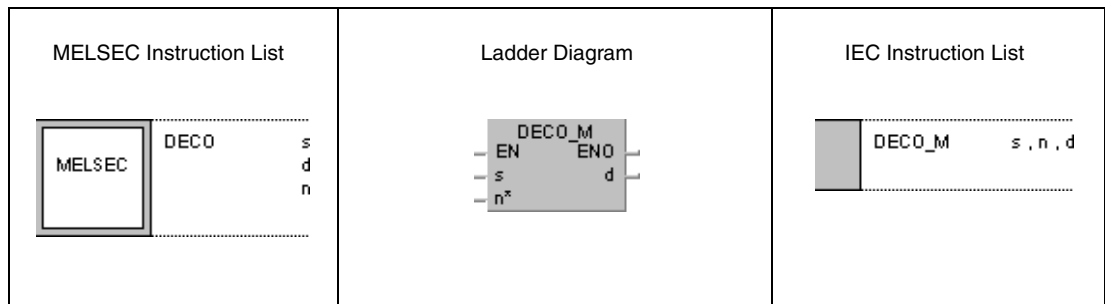
	Usable Devices																		Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011		
	Bit Devices						Word Devices (16-bit)						Constant		Pointer		Level								
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)						P	I
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					9	●		●
d		●	●	●	●	●	●	●	●	●															
n																●	●								

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

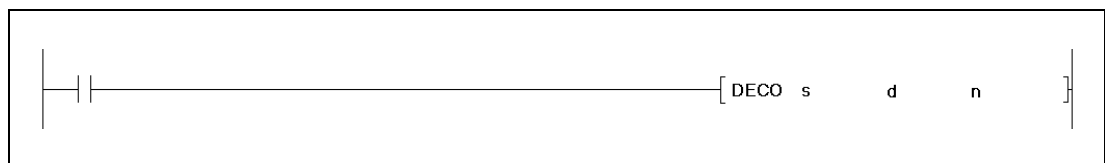
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	4
d	●	●	●	—	—	—	—	—			
n	●	●	●	●	●	●	●	●			

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
s	Coded data or device storing such data.	BIN 16-bit
d	First number of device storing decoded value.	Address
n	Number of bits containing coded data.	BIN 16-bit

**Functions**     **Decoding from 8 to 256 bits**

**DECO    Decoding data**

The DECO instruction decodes data in a device specified by s. The binary coded data is decoded as decimal number. This decimal number ( $\leq 256$ ) indicates bit x (bx), according to the  $2^x$ -th bit to be set of a device specified by d. The number of device addresses in s containing the coded data is specified by n.

The variable n must be set between 0 and 8.

If n = 0, the instruction is not executed and the specified device addresses remain unchanged.

A bit device is processed as single bit and a word device as 16-bit data value.

**Operation Errors**

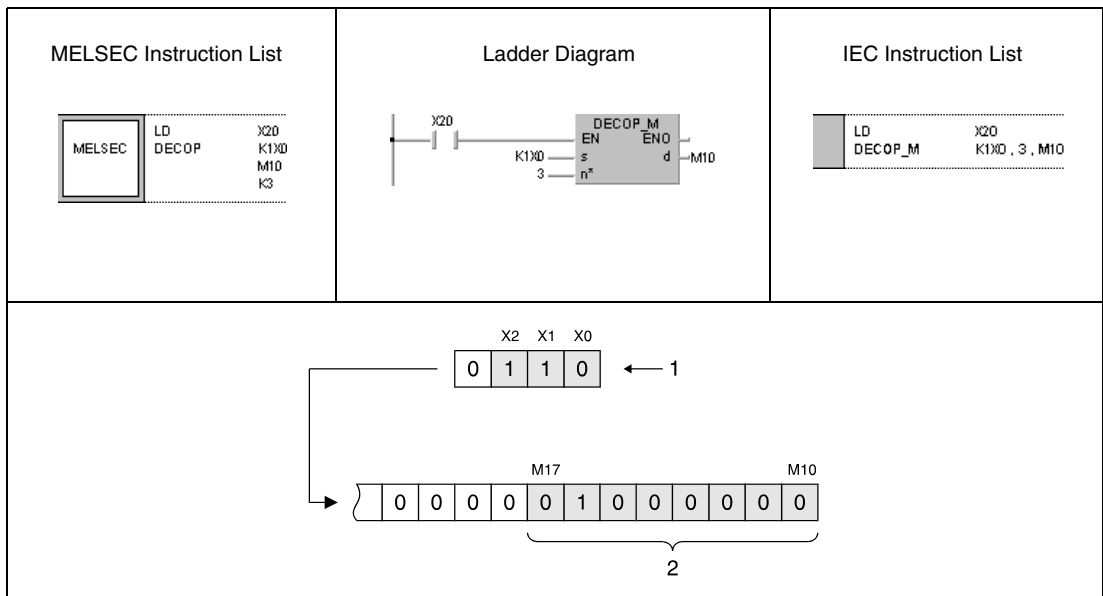
In the following cases an operation error occurs and the error flag is set:

- The variable n is not set between 0 and 8 (Q series and System Q = error code 4100).
- The bit x exceeds the relevant device range (Q series and System Q = error code 4100).

**Program Example**

DECOP

With leading edge from X20, the following program decodes data at X0 through X2. The result is stored in M10 through M17. The binary coded number 6 is contained in X0 through X2, so bit b6 (M16) in M10 through M17 is set.



<sup>1</sup> Binary coded value 6

<sup>2</sup> If the binary coded value is specified 4 bits, 8 bits are occupied

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 7.5.4 ENCO, ENCO\_P

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices MELSEC A**

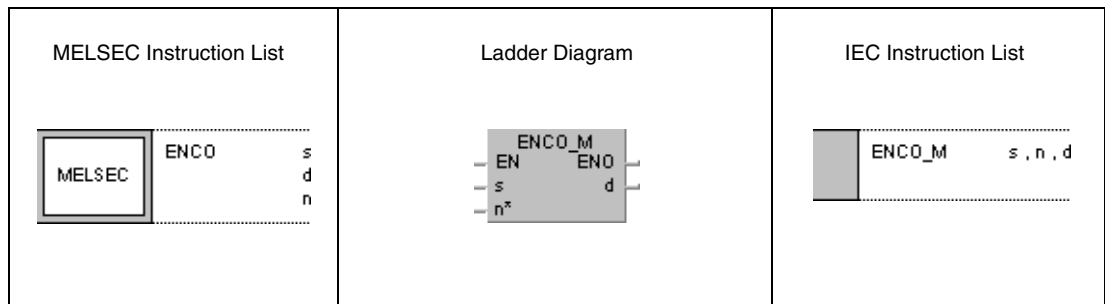
	Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011				
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level										
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P	I
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							9 1	●		●
d							●	●	●	●	●	●	●	●											
n																●	●								

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

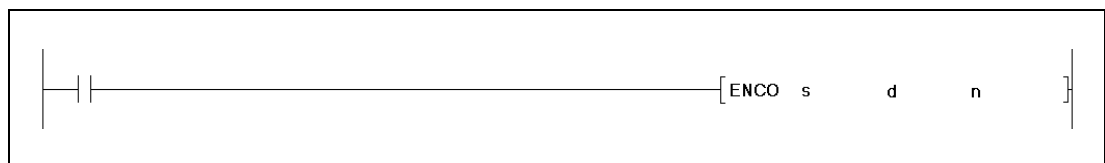
**Devices MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	—	—	—	—	●	—	SM0	4
d	●	●	●	●	●	●	●	—	—		
n	●	●	●	●	●	●	●	●	—		

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	Decoded data or device storing such data.	BIN 16-bit
d	First number of device storing coded data.	
n	Number of bits containing coded value.	

**Functions**      **Encoding from 256 to 8 bits**

**ENCO    Encoding data**

The ENCO instruction encodes data of a data record of up to 256 bits to a binary 8-bit data sequence. The initial number of device storing data to be encoded is specified by s. The bit x specified by s indicates the decimal value that will be stored binary encoded in d. The number of bits in d containing the encoded data is specified by n.

The variable n must be set between 0 and 8.

If n = 0, the instruction is not executed and the specified device addresses remain unchanged.

A bit device is processed as single bit and a word device as 16-bit data value.

If more than one bit is set processing starts with the highest bit.

**Operation Errors**

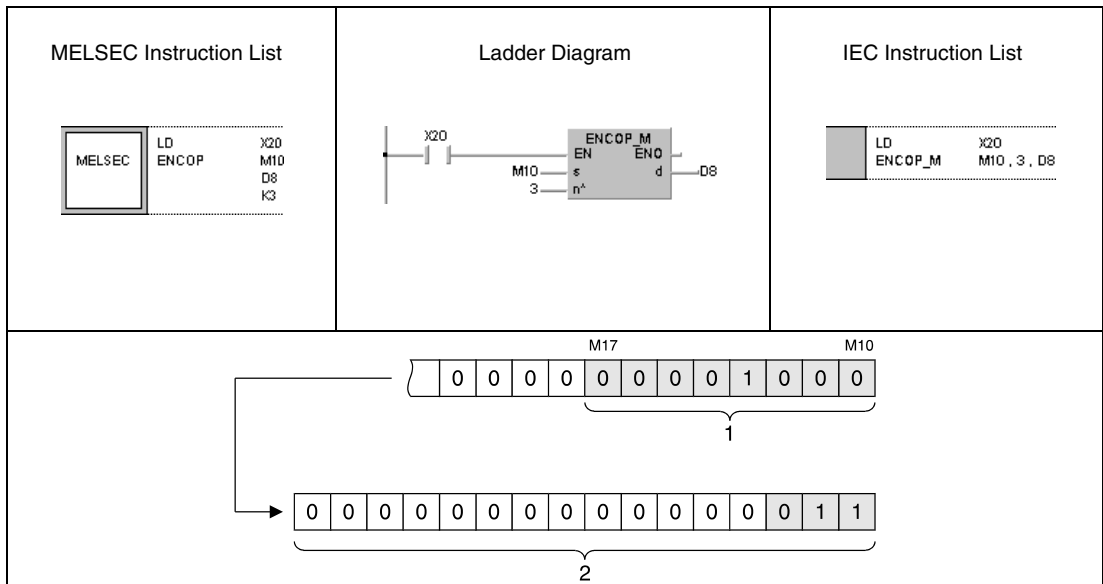
In the following cases an operation error occurs and the error flag is set:

- The variable n is not set between 0 and 8 (Q series and System Q = error code 4100).
- All bits in s up to bit x are 0 when executing the ENCO instruction.
- The value x of the set bit x in s exceeds the range that can be represented binary with 0 to 8 bits (Q series and System Q = error code 4101).
- All bits in s up to bit x are identical to d (Q series and System Q = error code 4100).

**Program Example**

ENCOP

With leading edge from X20, the following program reads data in M10 through M17 and stores it binary encoded in D8.



<sup>1</sup> If the set bit is binary encoded with 4 bits, a range of 8 bits can be represented

<sup>2</sup> Binary encoded number 3 for set bit 3 (M13)

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 7.5.5 SEG, SEGP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	●

<sup>1</sup> The SEG instruction only serves for 7-segment decoding, if the internal relay M9052 is NOT set. If the internal relay M9052 is set, the SEG instruction serves for partial refresh.

**Devices  
MELSEC A**

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag					
	Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level				
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P	I	N
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					K1	7	●		
d		●	●	●	●	●	●	●	●	●	●	●	●	●	●							● <sup>1</sup>	● <sup>2</sup>			

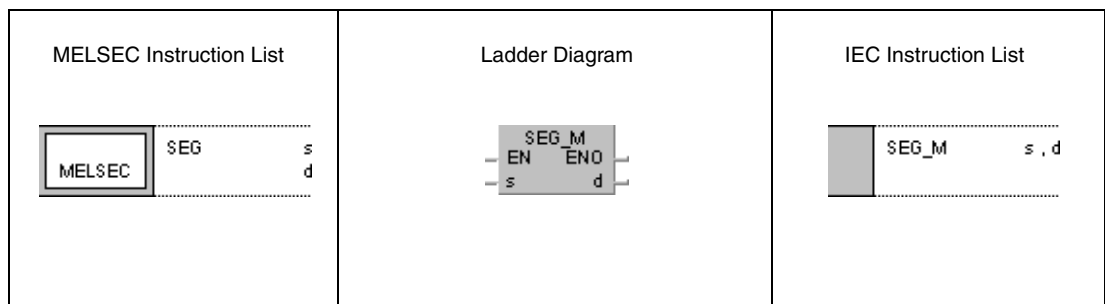
<sup>1</sup> With an A3H, A3M, or AnN CPU the digit designation can be set between K1 and K4. On all other CPUs the preset digit designation is ignored and K2 (8 bits) is processed automatically.

<sup>2</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

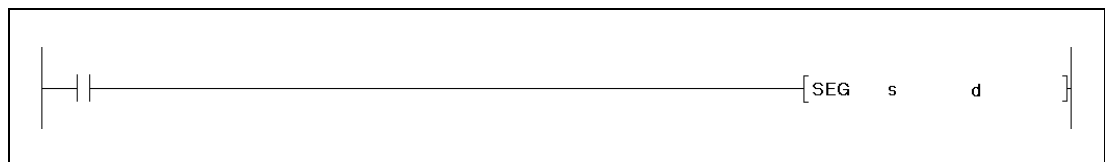
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	—	3
d	●	●	●	●	●	●	●	—	—	—	

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
s	Decoded data, or first number of device storing such data.	BIN 16-bit
d	First number of device storing 7-segment data.	

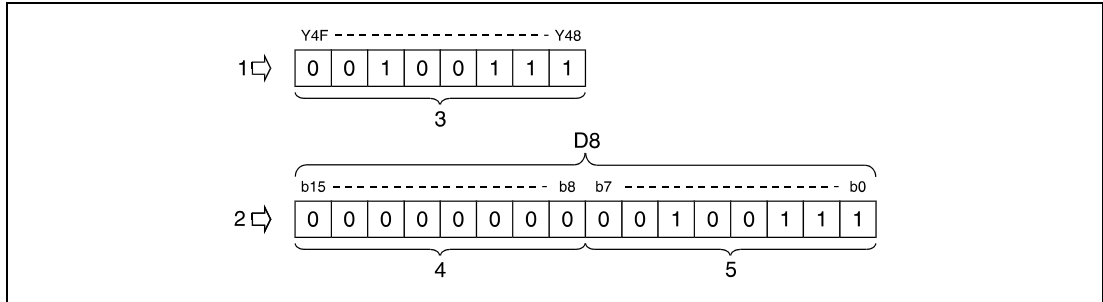
**Functions 7-Segment decoding**

**SEG (A and Q series/System Q) / SEGP (Q series and System Q)  
Decoding a 4-digit binary value**

The SEG instruction converts a 4-digit binary value into 7-segment code in order to display the values 0 to F. The data value or the initial number of data to be encoded is specified by s. The 7-segment data is stored in d.

If the encoded 7-segment data are output to bit devices, the initial device number and the digit designation must always be specified in d. If a word device is specified by d, only the device number is required.

Storage of data in several bit devices or in a word device applies to the following scheme:



- <sup>1</sup> Bit device
- <sup>2</sup> Word device
- <sup>3</sup> 8 bits
- <sup>4</sup> These bits are always reset to 0
- <sup>5</sup> 7-segment data

**7-segment data**

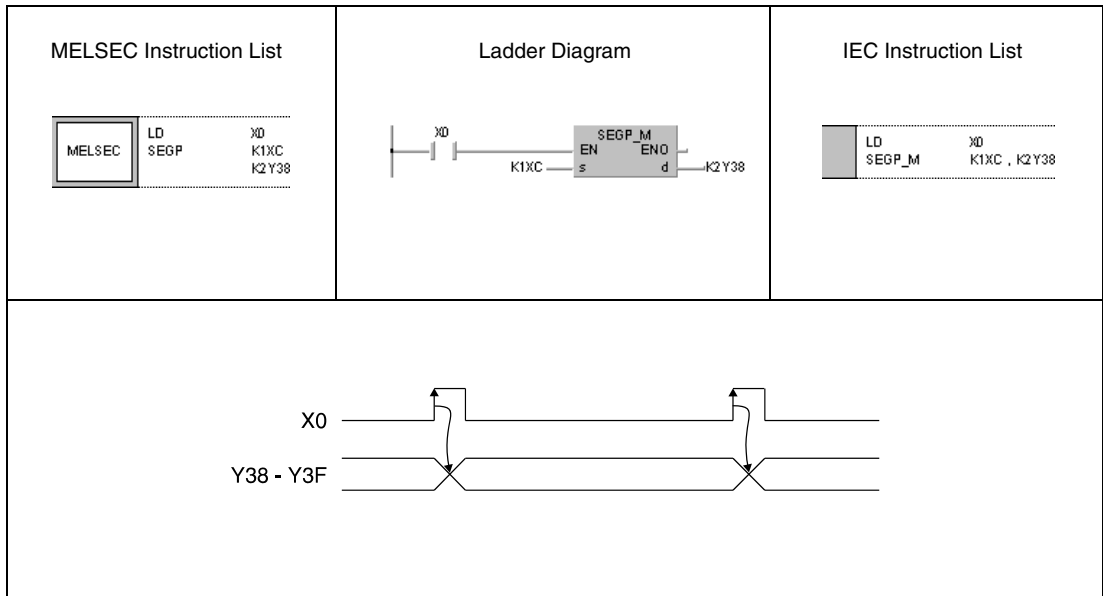
The following table contains an overview of 7-segment data in relation to the bit pattern of the source data. The first bit (b0) of 7-segment data either represents the status of the first bit device or the status of the least significant bit in a word device respectively.

s		Assignment of Segments	d								Display
HEX	Bit Pattern		B7	B6	B5	B4	B3	B2	B1	B0	
0	0000		0	0	1	1	1	1	1	1	0
1	0001		0	0	0	0	0	1	1	0	1
2	0010		0	1	0	1	1	0	1	1	2
3	0011		0	1	0	0	1	1	1	1	3
4	0100		0	1	1	0	0	1	1	0	4
5	0101		0	1	1	0	1	1	0	1	5
6	0110		0	1	1	1	1	1	0	1	6
7	0111		0	0	1	0	0	1	1	1	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	0	1	1	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	1	1	1	1	1	0	0	B
C	1100		0	0	1	1	1	0	0	1	C
D	1101		0	1	0	1	1	1	1	0	D
E	1110		0	1	1	1	1	0	0	1	E
F	1111		0	1	1	1	0	0	0	1	F

**Program Example**

**SEGP (Q series and System Q)**

With leading edge from X0, the following program outputs the condition of inputs XC through XF as 7-segment code to the outputs Y38 through Y3F. The conditions of outputs Y38 through Y3F are maintained until they are overwritten with new data.





7.5.6 DIS, DISP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

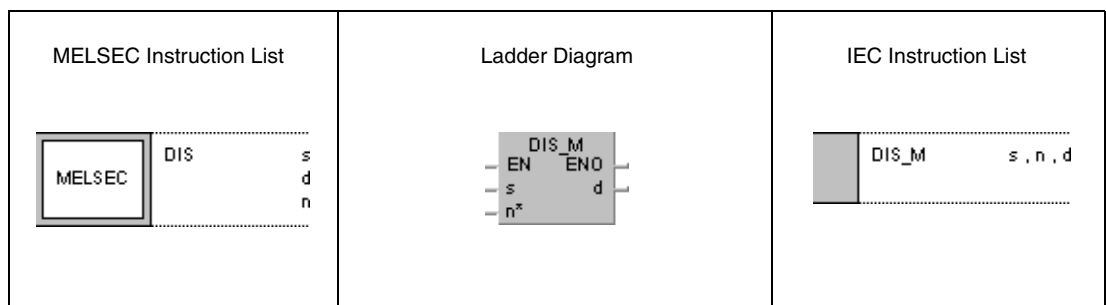
	Usable Devices																Digit designation K1 ↓ K4	Number of steps g	Index ●	Carry Flag M9012	Error Flag M9010 M9011
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)
s							●	●	●	●	●	●	●	●	●	●	●				
d							●	●	●	●											
n																●	●				

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

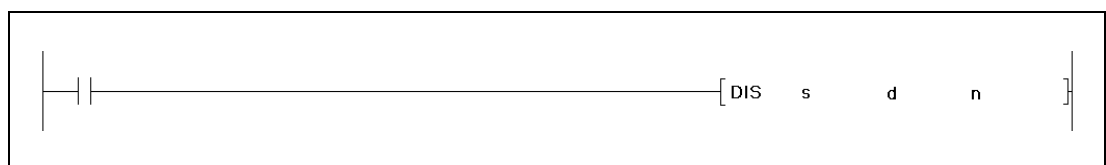
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—		
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer



GX Developer

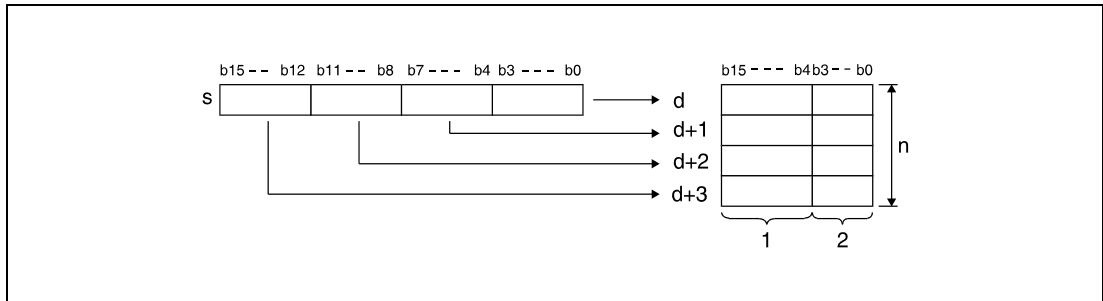


Variables

Set Data	Meaning	Data Type
s	First number of device storing data to be disunitied.	BIN 16-bit
d	First number of device storing disunitied data.	
n	Number of 4-bit groupings to be disunitied. No processing for n = 0.	

**Functions**    **Disuniting 16-bit data****DIS**    **Disuniting 16-bit data values**

The DIS instruction disunits a 16-bit data value to groupings of 4 bits and stores their conditions successively in up to 4 destination devices. For this instruction, the data value to be disuniting in *s*, the number of 4-bit groupings in *n*, and the first number of destination device in *d* must be specified. Further 4-bit groupings are stored in *d+n*.



<sup>1</sup> These bits are reset to 0.

The upper 12 bits of the destination devices beginning from device number in *d*, are reset to 0.

The variable *n* can be set from 1 to 4 (corresponding 4 to 16 bits).

For *n* = 0 no operation is executed and the specified number of device remains unchanged.

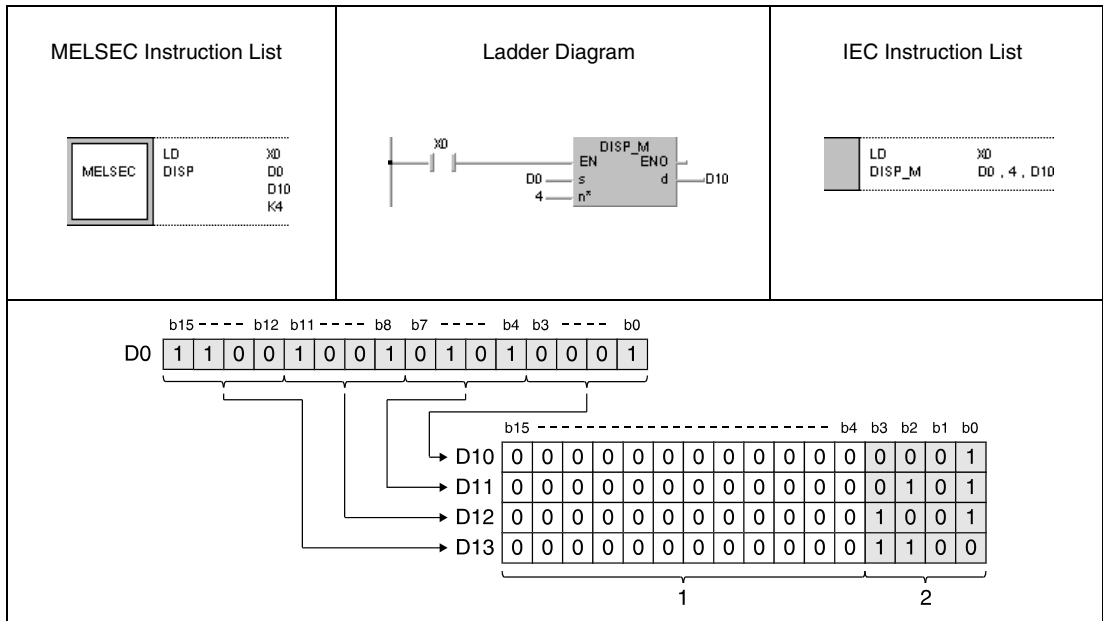
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The value in *n* is not set between 0 and 4 (Q series and System Q = error code 4100).
- The storage range *d* specified by *n* exceeds the relevant device range (Q series and System Q = error code 4101).

**Program Example** DISP

With leading edge from X0, the following program disunites the 16-bit data value in D0 and stores the bit pattern in groupings of 4 bits in series in D10 through D13.



<sup>1</sup> These bits are reset to 0

<sup>2</sup> Storage range

7.5.7 UNI, UNIP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

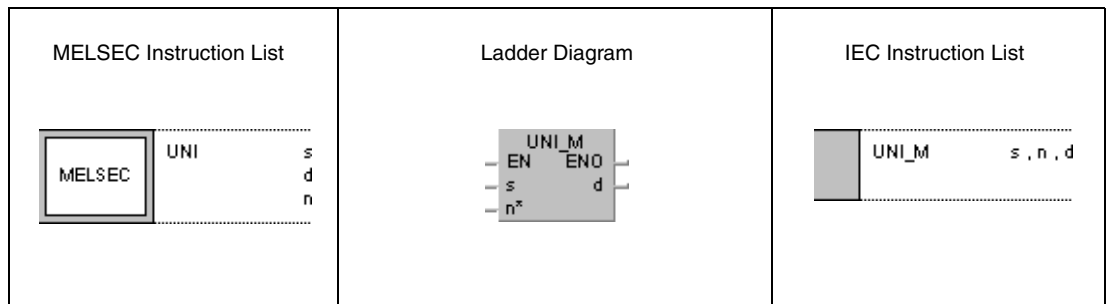
	Usable Devices																	Digit designation K1 ↓ K4	Number of steps 9 1	Index	Carry Flag M9012	Error Flag M9010 M9011	
	Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I
s							●	●	●	●													
d							●	●	●	●	●	●	●	●	●								●
n																●	●						

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

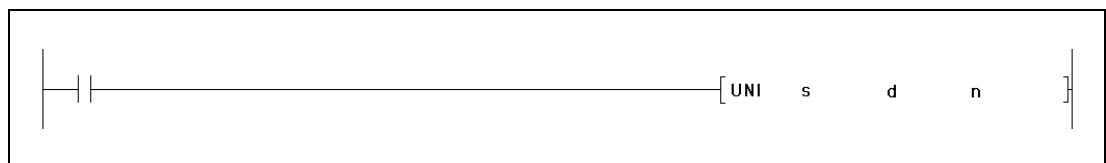
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	—	—	—	—	—	—	SM0	4
d	—	●	●	●	●	●	—	—			
n	●	●	●	●	●	●	●	—			

GX IEC  
Developer



GX  
Developer

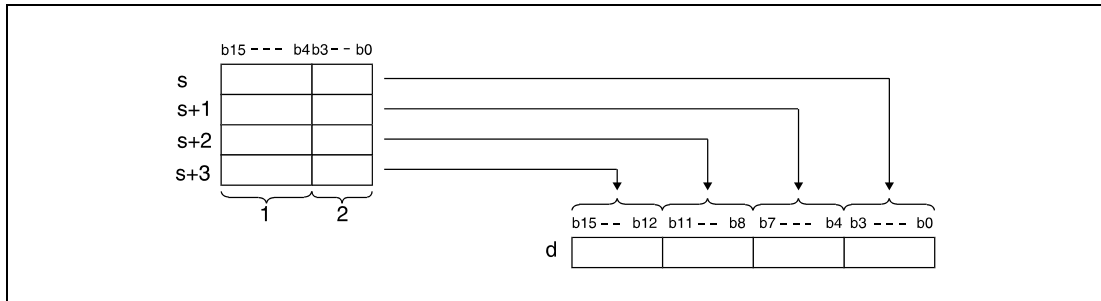


Variables

Set Data	Meaning	Data Type
s	First number of device, storing data to be united.	BIN 16-bit
d	First number of device, storing united data.	
n	Number of 4-bit groupings to be united. No processing for n = 0.	

**Functions**     **Uniting 16-bit data****UNI**     **Uniting 16-bit data values**

The UNI instruction separates each 4 lowest bits of up to four 16-bit data values and unites their conditions in one 16-bit data value. For this instruction, the first number of device storing the data values in *s* to be united, the number of successive devices *n*, and the destination address in *d* must be specified.



<sup>1</sup> These bits are ignored

<sup>2</sup> 4-bit groupings to be stored in *d*

The lower 4 bits of the source devices beginning from device number in *d*, are reset to 0.

The variable *n* can be set from 1 to 4.

For *n* = 0 no operation is executed and the specified number of device remains unchanged.

**Operation Errors**

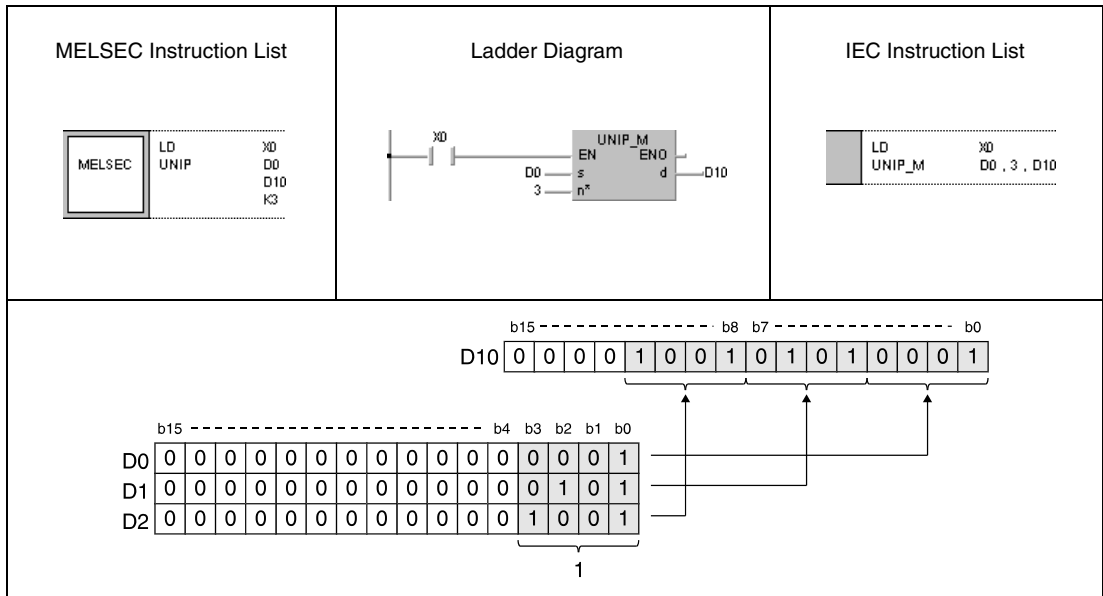
In the following cases an operation error occurs and the error flag is set:

- The value in *n* is not set within 0 and 4 (Q series and System Q = error code 4100).
- The storage range *s* specified by *n* exceeds the relevant device range (Q series and System Q = error code 4101).

**Program Example**

UNIP

With leading edge from X0, the following program unites each lowest 4 bits (b0 through b3) of data registers D0 through D2 successively to one 16-bit data value (the highest 4 digits are "0") in D10.



<sup>1</sup> 4-bit groupings to be stored in D10

**7.5.8 NDIS, NDISP, NUNI, NUNIP**

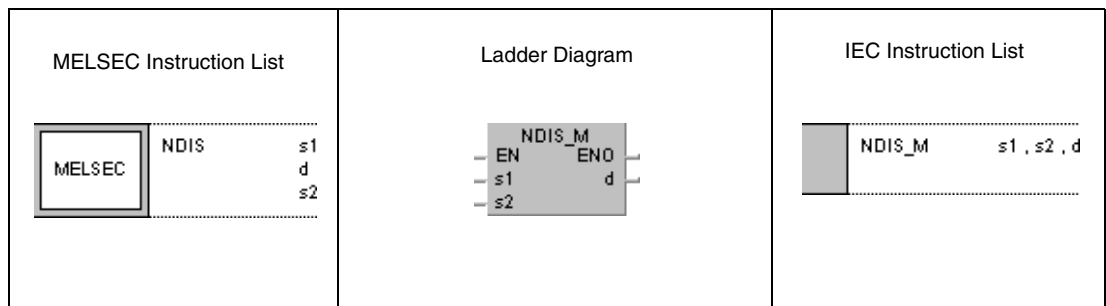
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

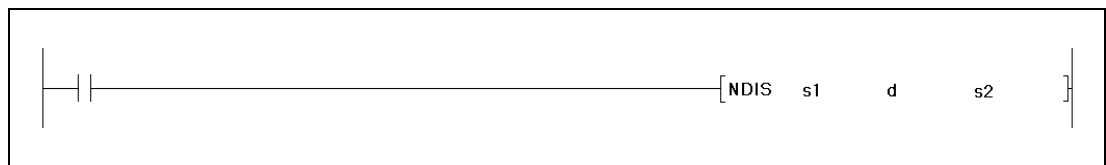
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	SM0	4	
d	—	●	●	—	—	—	—	—			
s2	—	●	●	—	—	—	—	—			

**GX IEC Developer**



**GX Developer**



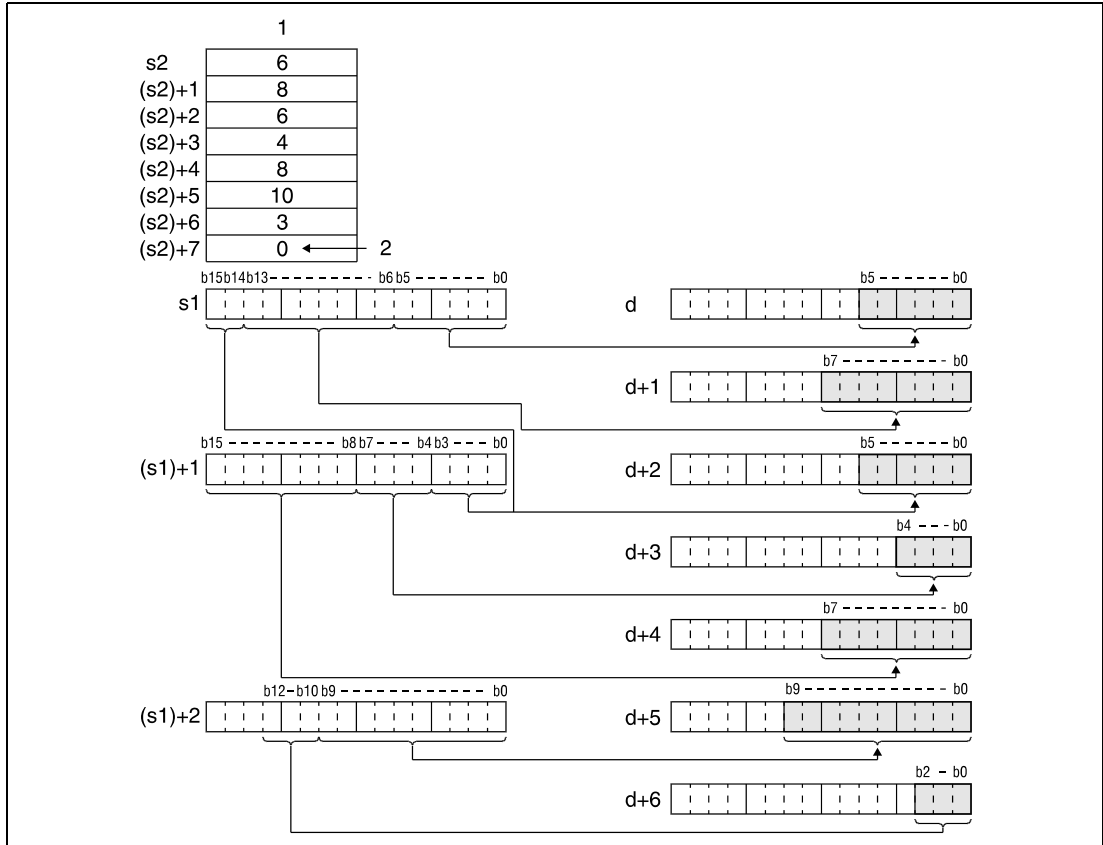
**Variables**

Set Data	Meaning	Data Type
s1	First number of device storing data to be disunited/united.	BIN 16-bit
d	First number of device storing disunited/united data.	
s2	Number of bits to be disunited/united in bit groupings.	

**Functions Disuniting or uniting of data in random bit groupings**

**NDIS Disuniting data**

The NDIS instruction disunits data in devices specified from s1 on to bit groupings with a number of bits specified by s2. The disuniting bit groupings are stored separately in the device specified by d onwards.



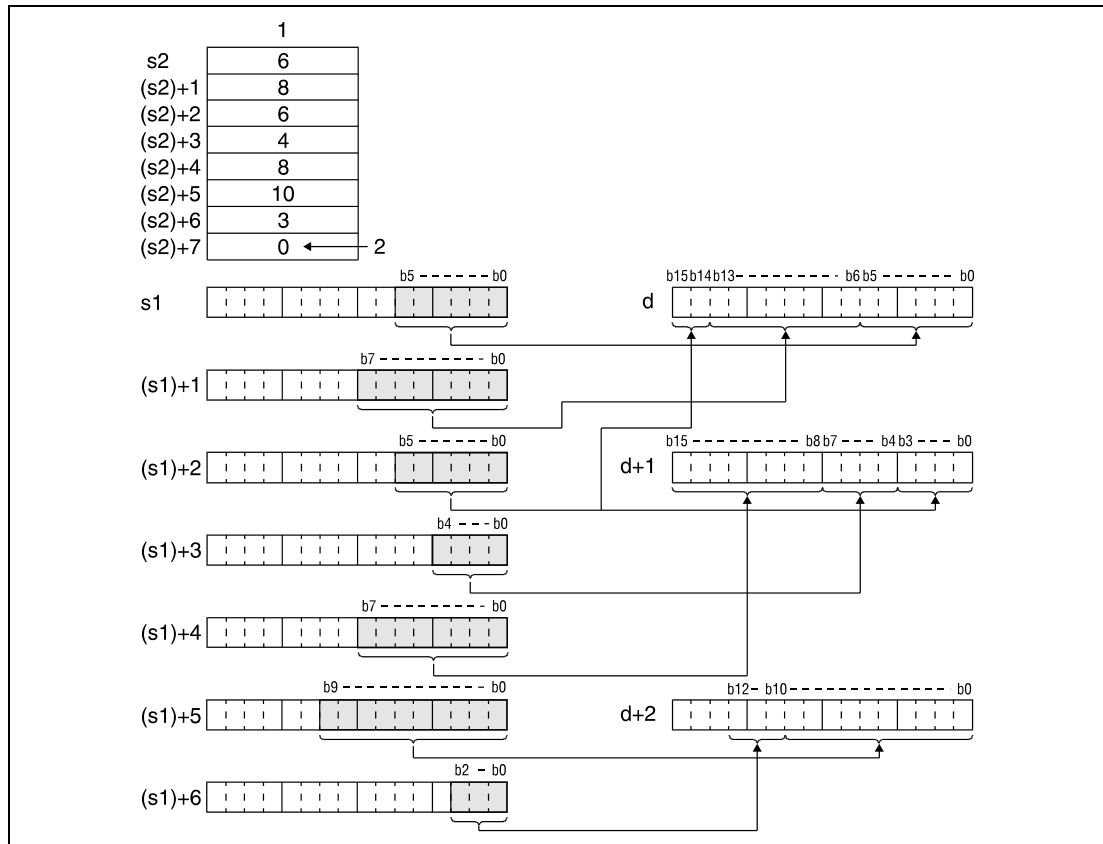
<sup>1</sup> Size of bit grouping  
<sup>2</sup> The 0 indicates the end of processing

The size of bit groupings specified by s2 can be set within 1 and 16 bits.  
 Values in s2 are processed from the first device address in s2 on and up to the address with the entry 0.



**NUNI Uniting data**

The NUNI instruction separates bit groupings of a size specified by s2 from devices specified by s1 and unites these bit groupings in one data value. The bit groupings are stored successively from the device specified by d on.



<sup>1</sup> Size of bit groupings

<sup>2</sup> The 0 indicates the end of processing

The size of bit groupings specified by s2 can be set within 1 and 16 bits.

Values in s2 are processed from the first device address in s2 on and up to the address with the entry 0.

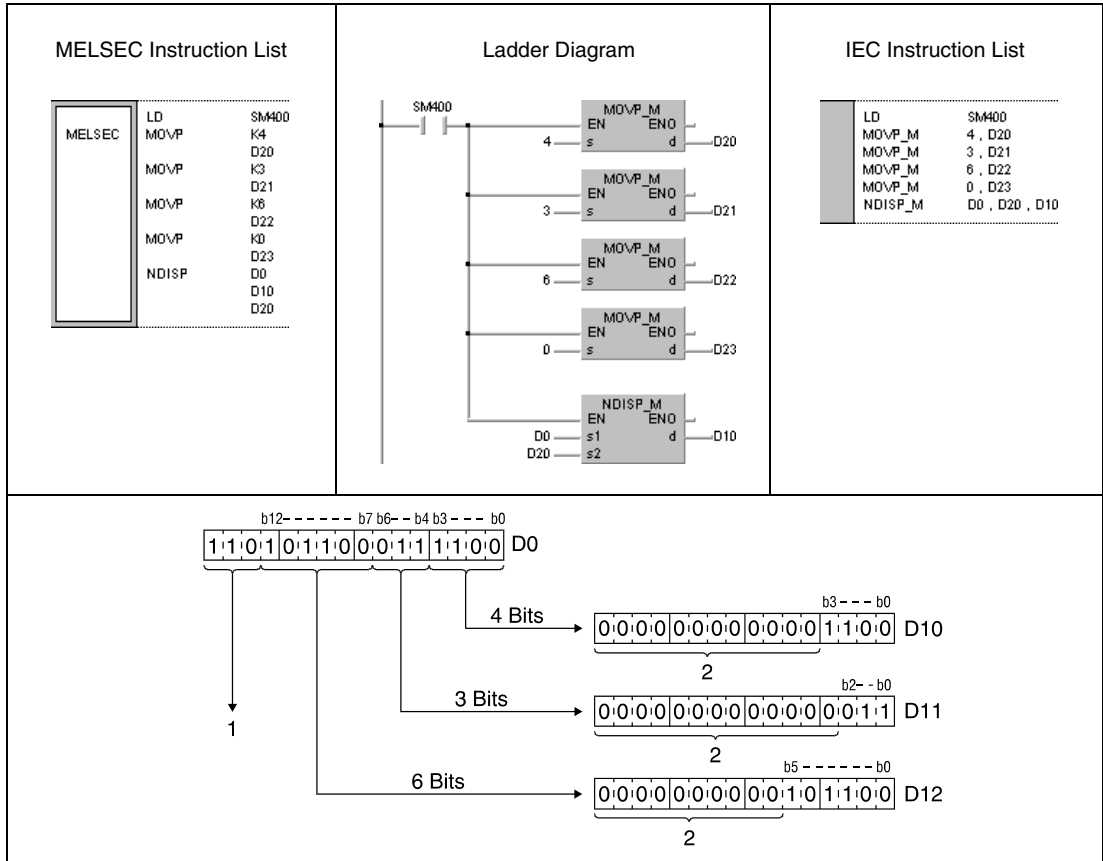
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The bit groupings of a size specified by s2 in the devices specified by s1 or d exceed the relevant storage device range (error code 4101).
- The size of bit groupings specified by s2 exceed the valid range of 1 to 16 bits (error code 4100).

**Program Example 1** NDISP

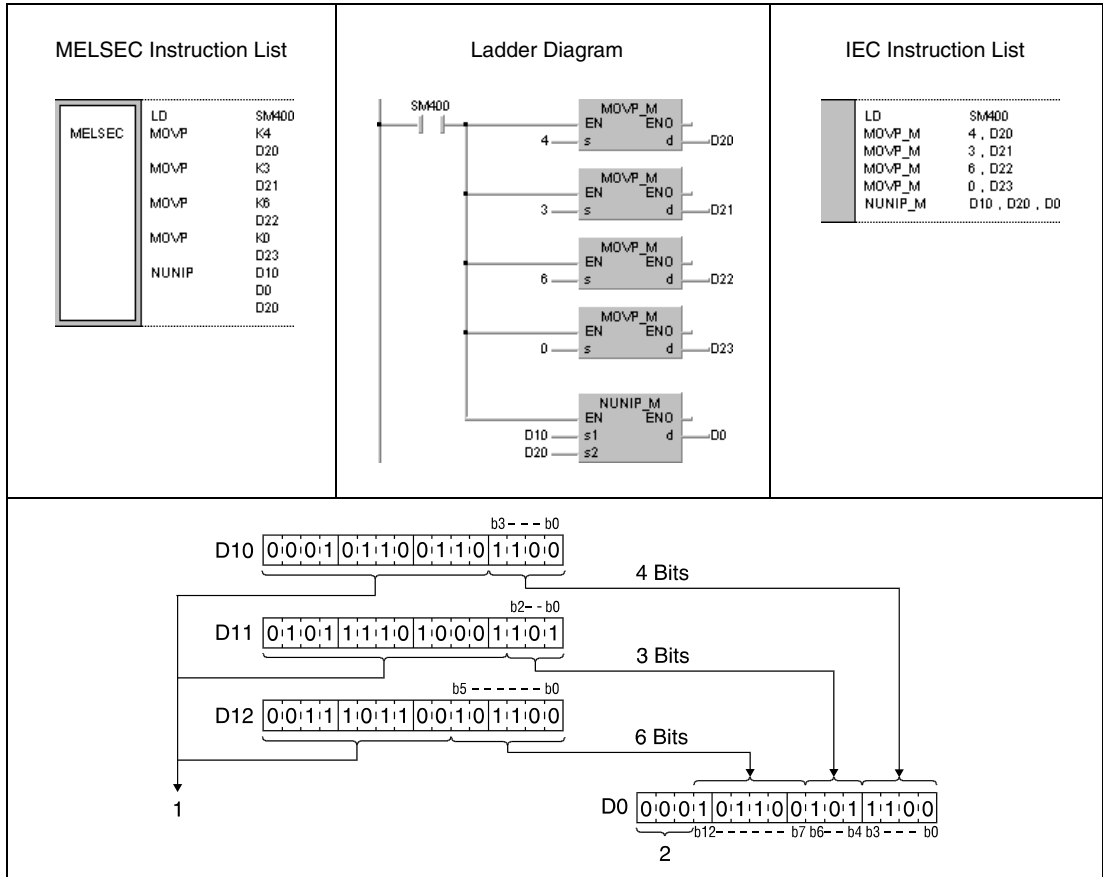
With leading edge from SM400, the following program separates the bit groupings b0 - b3 (4), b4 - b6 (3), and b 7- b12 (6) from D0 and stores each single bit grouping beginning from bit grouping b0 - b3 in D10 through D12. The values in brackets indicate the size of bit groupings in D20 through D22. D23 must store the value 0 (see functions).



<sup>1</sup> These bits are ignored  
<sup>2</sup> These bits are reset to 0

**Program Example 2** NUNIP

With leading edge from SM400, the following program separates the bit groupings b0 - b3 (4), b0 - b2 (3), and b0 - b5 (6) from D10 through D12 and stores the bit groupings successively in D0 beginning from bit grouping b0 - b3. The values in brackets indicate the size of bit groupings in D20 through D22. D23 must store the value 0 (see functions).



<sup>1</sup> These bits are ignored  
<sup>2</sup> These bits are reset to 0

7.5.9 WTOB, WTOBP, BTOW, BTOWP

CPU

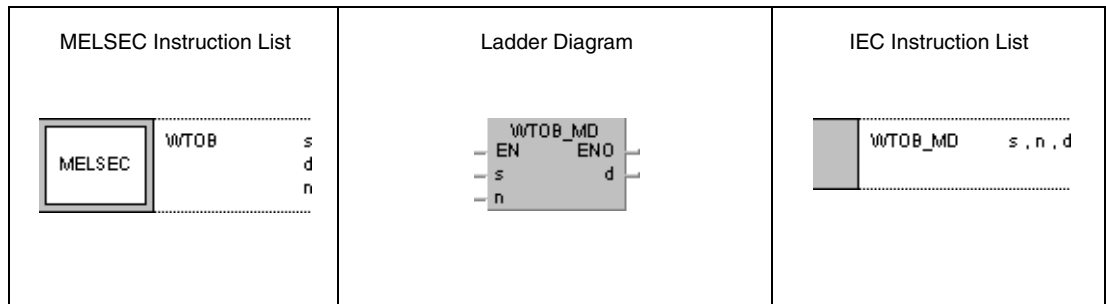
AnS	AnN	AnA, AnAS	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

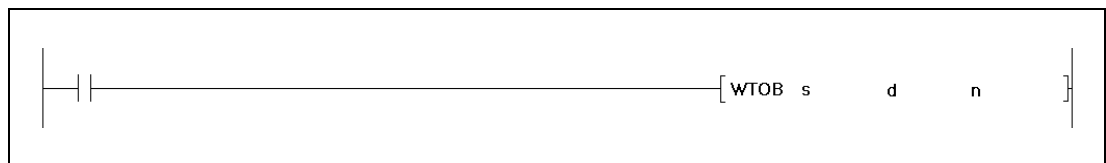
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer



GX Developer



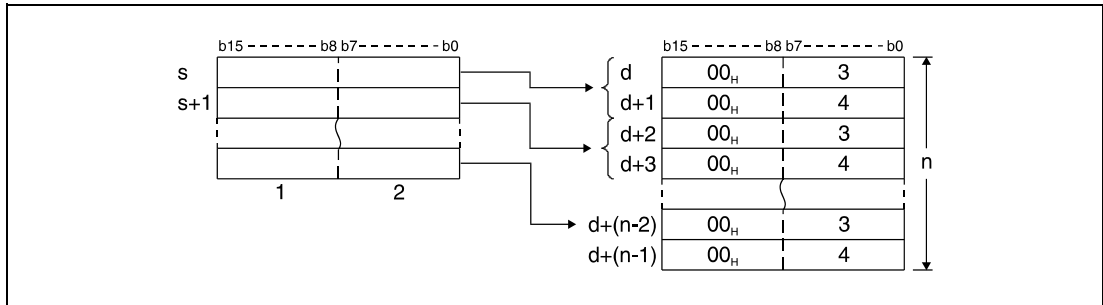
Variables

Set Data	Meaning	Data Type
s	First number of device storing data to be disunited/united in byte units.	BIN 16-bit
d	First number of device storing disunited/inited bytes.	
n	Number of byte units to be disunited/united.	

**Functions Disuniting and uniting data in byte units**

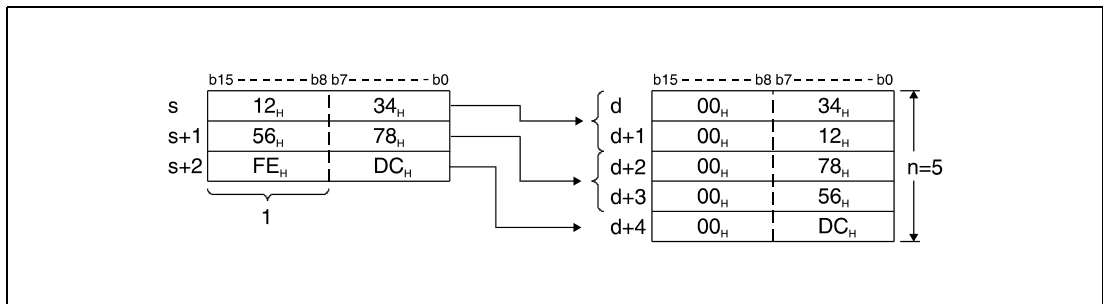
**WTOB Disuniting data**

The WTOB instruction disunits a 16-bit data value to byte units and stores their conditions successively in destination devices. For this instruction the data values in *s* to be disuniting, the number of byte units in *n*, and the first number of destination device in *d* must be specified. Further byte units are stored in *d+n*. For storage only the lowest bytes of the devices specified by *d* are used.



- <sup>1</sup> Highest bytes
- <sup>2</sup> Lowest bytes
- <sup>3</sup> Data of the according lowest bytes
- <sup>4</sup> Data of the according highest bytes

For example, if *n* = 5, 5 bytes are disuniting from the device specified by *s* through *s+2* and stored successively in the lowest bytes of the devices specified by *d* through *d+4*.

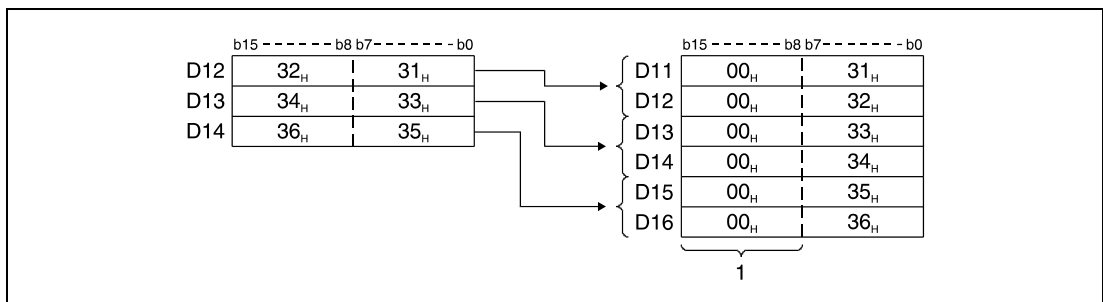


- <sup>1</sup> These bytes are ignored

The number of byte units specified by *n* automatically determines the range of 16-bit data in *s* and the storage range of the byte units in *d*.

If *n* = 0, the instruction is not executed and the specified device addresses remain unchanged.

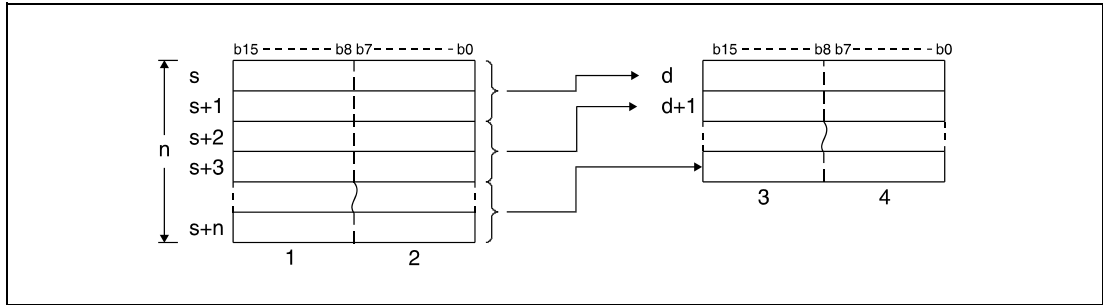
The highest bytes in the devices specified by *d* are set to the value "00H".



- <sup>1</sup> These bytes are set to "00H"

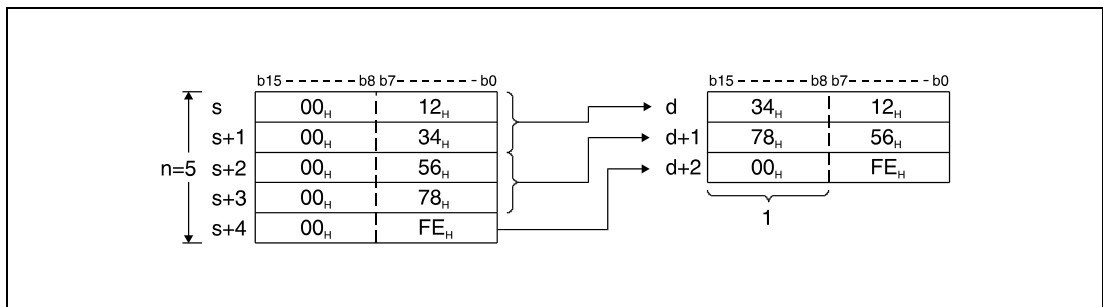
**BTOW Uniting data**

The BTOW instruction separates any lowest bytes of 16-bit data values and stores their conditions in 16-bit data values. For this instruction, the initial number of data value in s to be united, the number of byte units n, and destination device in d must be specified.



- <sup>1</sup> These bytes are ignored
- <sup>2</sup> Data of 1st through nth byte
- <sup>3</sup> Data of 2nd, 4th, and nth byte
- <sup>4</sup> Data of 1st, 3rd, and (n-1)th byte

For example, if n = 5, the 5 lowest bytes are disunitied from the device specified by s through s+4 and stored successively in the devices specified by d through d+2.



- <sup>1</sup> This byte is set to "00H"

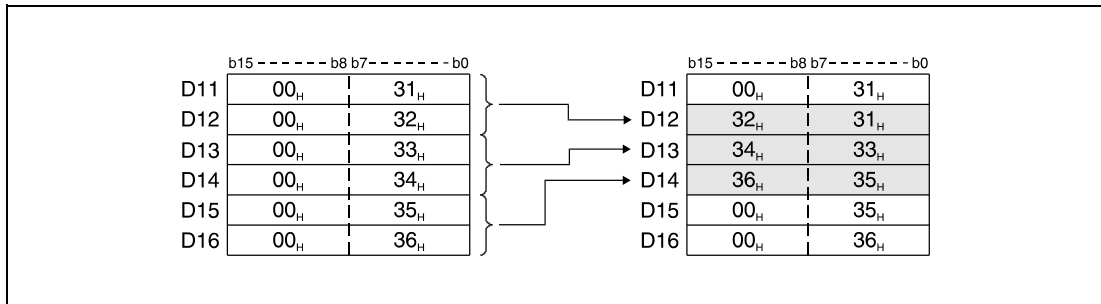
The number of byte units specified by n automatically determines the range of byte data in s and the storage range of the byte data in d.

If n = 0, the instruction is not executed and the specified device addresses remain unchanged.

The highest bytes in the devices specified by s are ignored on processing.

The operation is even processed correctly in cases where the storage ranges of s through s+n and d through d+n overlap.

The following diagram shows a case where the lowest bytes are separated from D11 through D16 and stored again successively in D12 through D14.



### Operation Errors

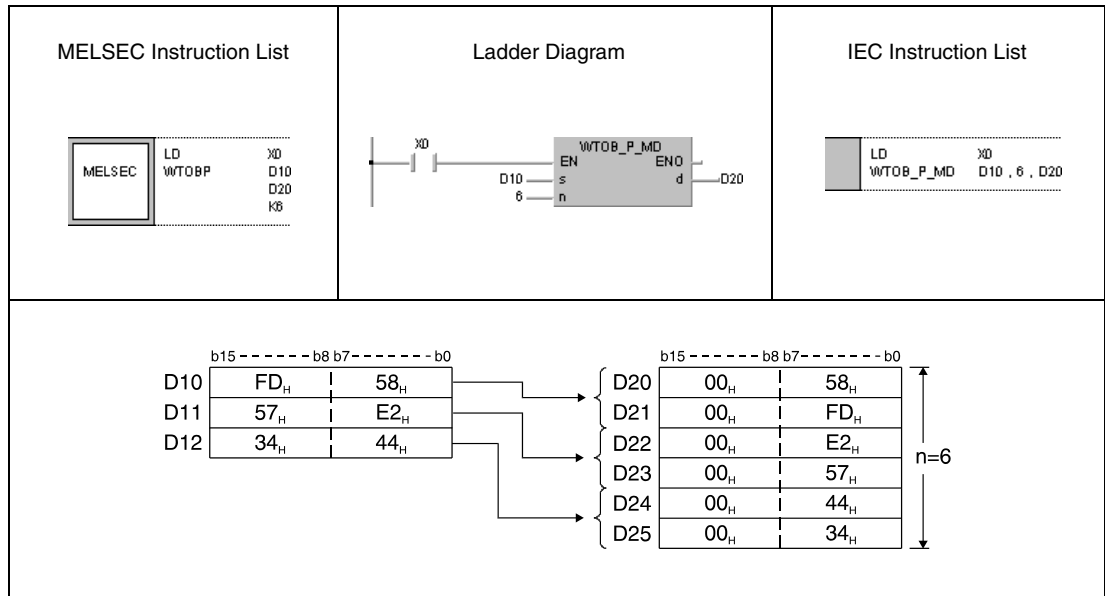
In the following cases an operation error occurs and the error flag is set:

- The number of byte units specified by n, that are stored in the device specified by s, exceeds the relevant storage device range (Q series and System Q = error code 4101).
- The number of byte units specified by n, that are stored in the device specified by d, exceeds the relevant storage device range (Q series and System Q = error code 4101).

**Program Example 1**

WTOBP

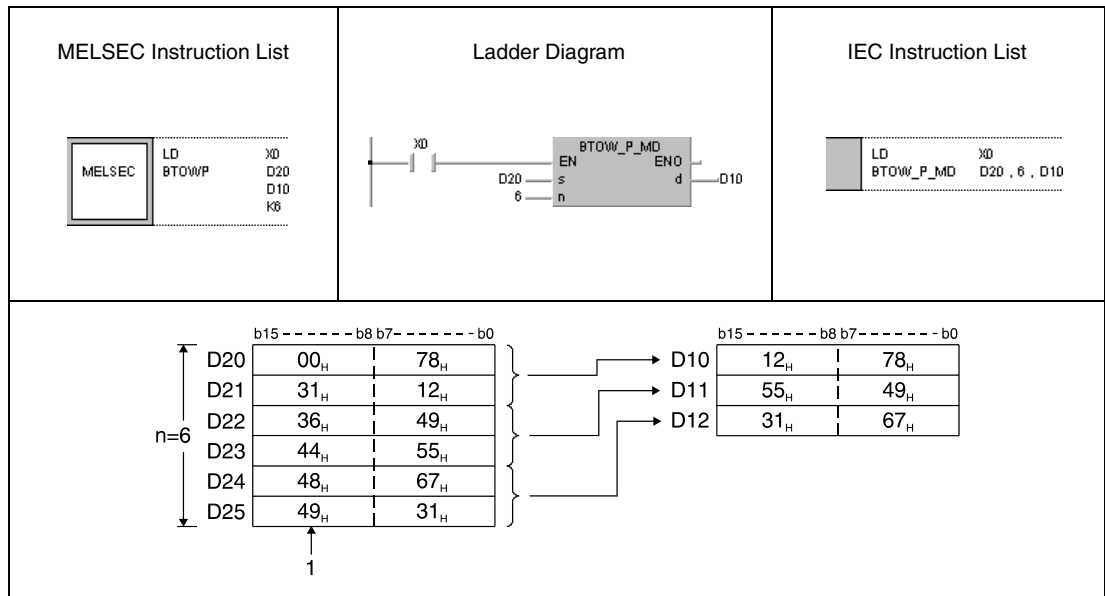
With leading edge from X0, the following program separates 6 bytes in D10 through D12 successively and stores these bytes in the lowest bytes in D20 through D25.



**Program Example 2**

BTOWP

With leading edge from X0, the following program separates the 6 lowest bytes in registers D20 through D25 and unites these bytes successively in D10 through D12.



<sup>1</sup> These bytes are ignored



**7.5.10 MAX, MAXP, DMAX, DMAXP**

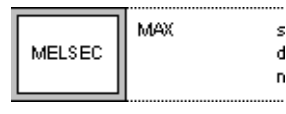
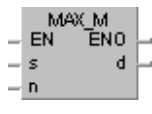
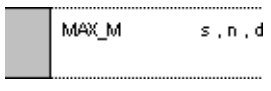
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

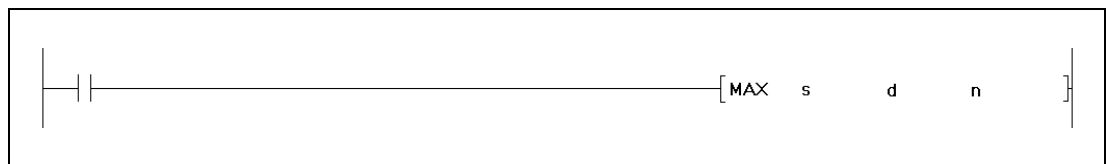
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	SM0	4	
d	—	●	●	—	—	—	—	—			
n	●	●	●	●	●	●	●	—			

**GX IEC Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

**GX Developer**



**Variables**

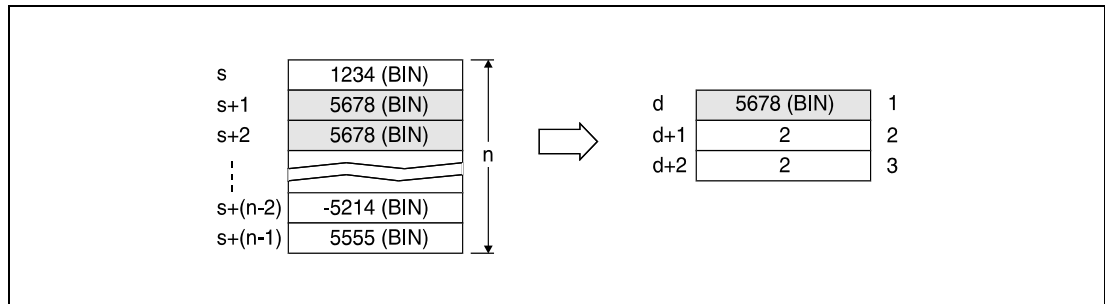
Set Data	Meaning	Data Type
s	First number of device storing data to be searched through for maximum values.	BIN 16-/32-bit
d	First number of device storing search result.	
n	Number of data blocks to be searched through.	BIN 16-bit

**Functions**     **Searching maximum values in 16-/32-bit data**

**MAX     Searching maximum values in 16-bit data**

The MAX instruction searches for maximum values in 16-bit data blocks. The number of data blocks to be searched through is specified by n. The greatest value found in s through s+(n-1) is stored in d.

The first position in s through s+(n-1) where the maximum value is found is counted beginning from s = 1 and stored in d+1. The number of existing identical maximum values is stored in d+2.

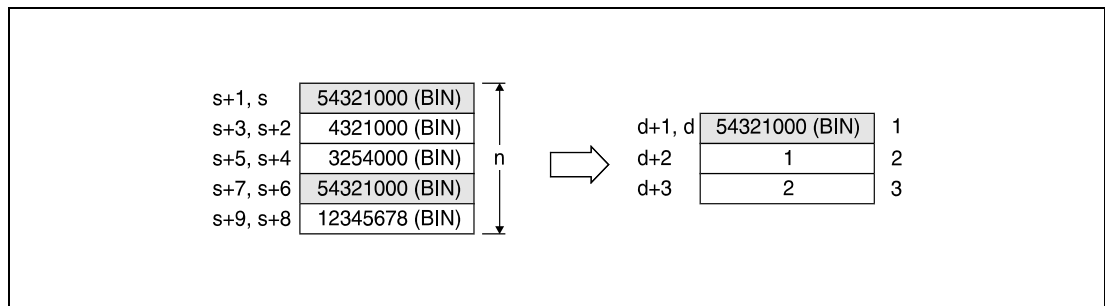


- <sup>1</sup> Found maximum value
- <sup>2</sup> First position the value has been found at
- <sup>3</sup> Number of identical maximum values

**DMAX     Searching maximum values in 32-bit data**

The DMAX instruction searches for maximum values in 32-bit data blocks. The number of data blocks to be searched through is specified by n. The greatest value found in s through s+(n-1) is stored in d.

The first position in s through s+(n-1) where the maximum value is found is counted beginning from s = 1 and stored in d+2. The number of existing identical maximum values is stored in d+3.



- <sup>1</sup> Found maximum value
- <sup>2</sup> First position the value has been found at
- <sup>3</sup> Number of identical maximum values

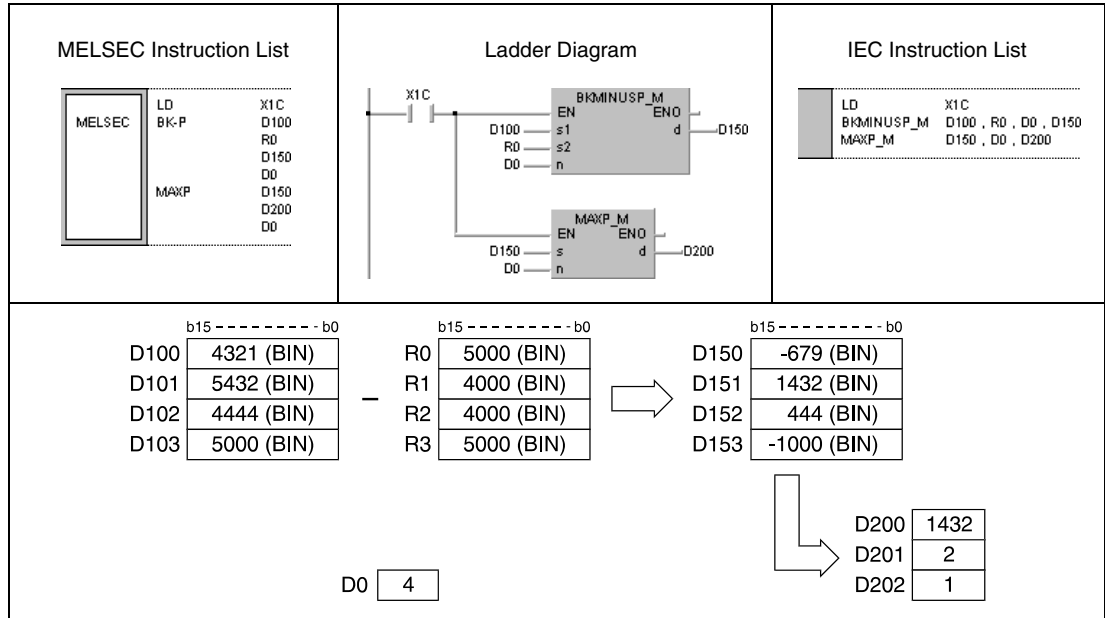
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The number of data blocks specified by n stored in the devices specified by s exceeds the relevant storage device range (error code 4101).

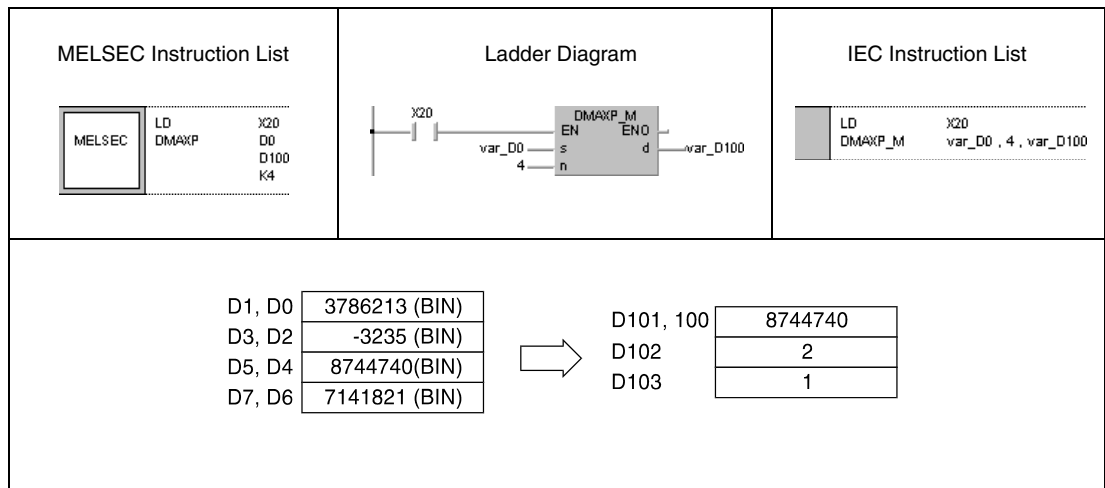
**Program Example 1** MAXP

With leading edge from X1C, the following program subtracts data in R0 through R3 from data in D100 through D103 and stores the result in D150 through D153. The number of 16-bit data blocks (4) is specified in D0. In the following step, as well with leading edge from X1C, the registers D150 through D153 are searched through for the maximum value. The value found is stored in D200, its position is stored in D201, and the number of identical maximum values is stored in D202.



**Program Example 2** DMAXP

With leading edge from X20, the following program searches for the maximum value of 32-bit data in D100 and D101. The position of the value is stored in D102, the number of identical maximum values is stored in D103.



**NOTE** *The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.5.11 MIN, MINP, DMIN, DMINP

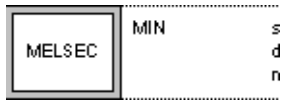
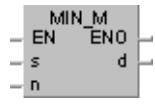
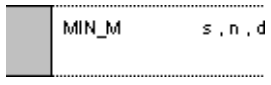
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

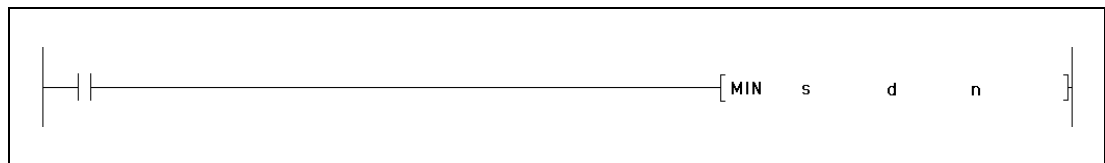
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	SM0	4	
d	—	●	●	—	—	—	—	—			
n	●	●	●	●	●	●	●	—			

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

GX Developer



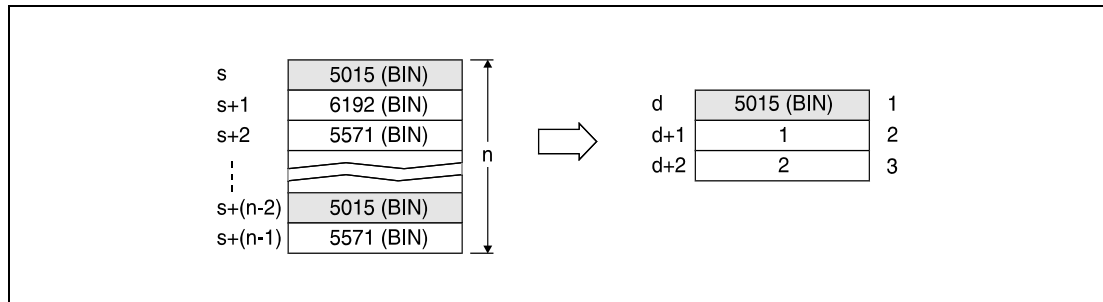
Variables

Set Data	Meaning	Data Type
s	First number of device storing data to be searched through for minimum values.	BIN 16-/32-bit
d	First number of device storing search result.	
n	Number of data blocks to be searched through.	BIN 16-bit

**Functions**    **Searching minimum values in 16-/32-bit data****MIN**    **Searching minimum values in 16-bit data**

The MIN instruction searches for minimum values in 16-bit data blocks. The number of data blocks to be searched through is specified by n. The smallest value found in s through s+(n-1) is stored in d.

The first position in s through s+(n-1) where the minimum value is found is counted beginning from s = 1 and stored in d+1. The number of existing identical minimum values is stored in d+2.



<sup>1</sup> Found minimum value

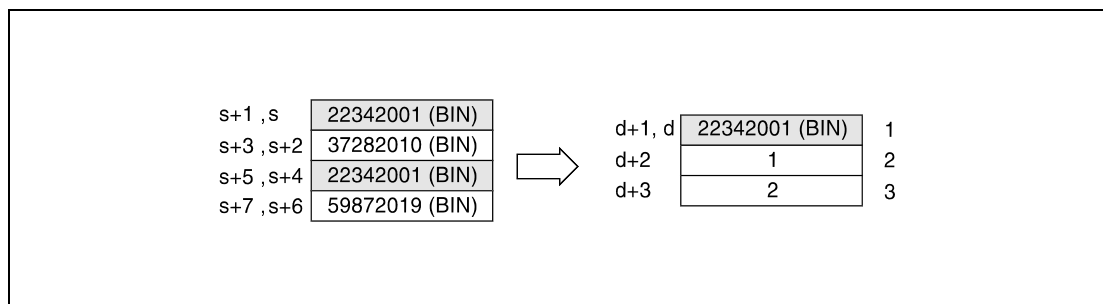
<sup>2</sup> First position the value has been found at

<sup>3</sup> Number of identical minimum values

**DMIN**    **Searching minimum values in 32-bit data**

The DMIN instruction searches for minimum values in 32-bit data blocks. The number of data blocks to be searched through is specified by n. The smallest value found in s through s+(n-1) is stored in d and d+1.

The first position in s through s+(n-1) where the minimum value is found is stored in d+2. The number of existing identical minimum values is stored in d+3.



<sup>1</sup> Found minimum value

<sup>2</sup> First position the value has been found at

<sup>3</sup> Number of identical minimum values

**Operation Errors**

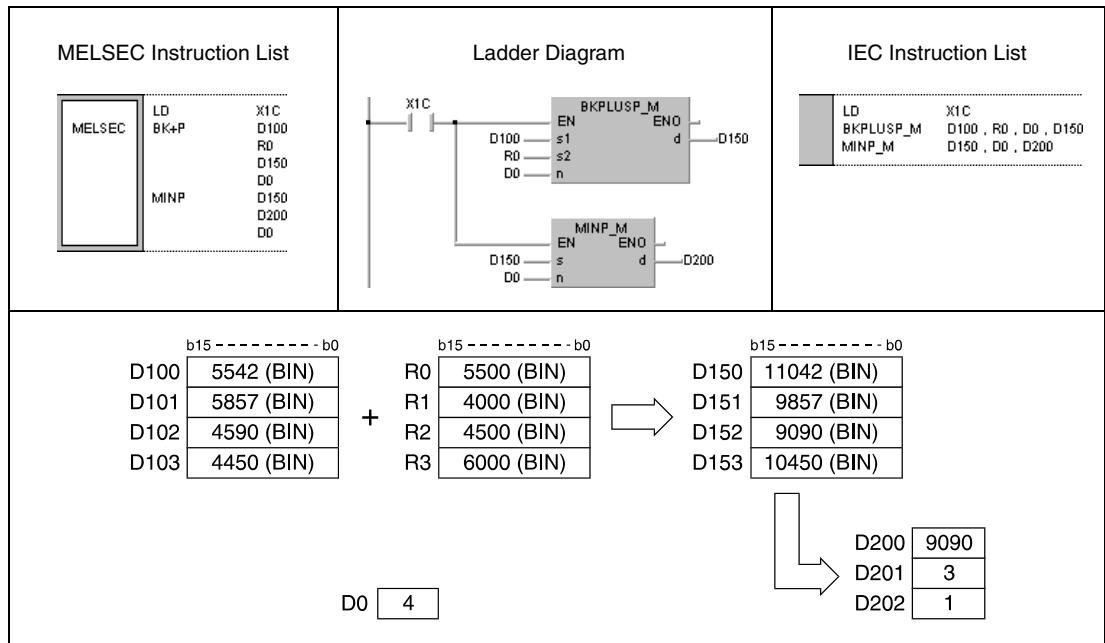
In the following cases an operation error occurs and the error flag is set:

- The number of data blocks specified by n stored in the devices specified by s exceeds the relevant storage device range (error code 4101).

**Program Example 1**

MINP

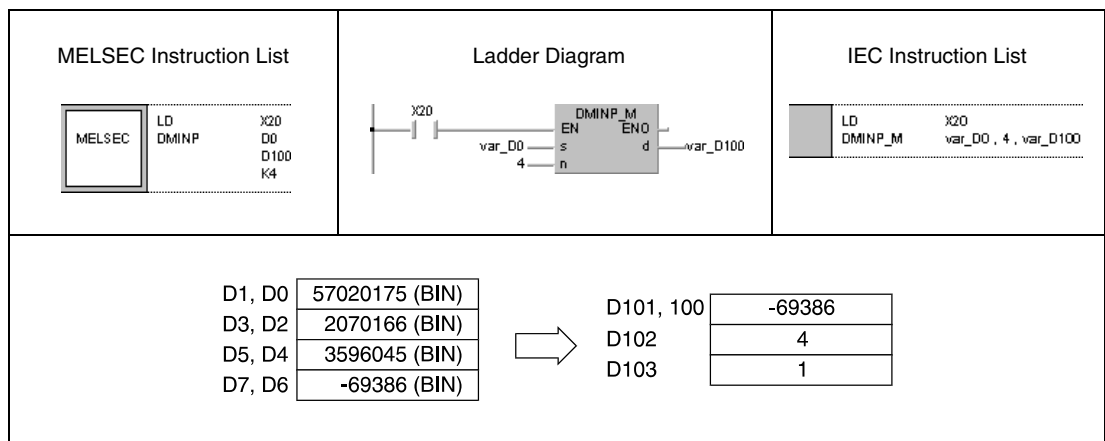
With leading edge from X1C, the following program adds data in D100 through D103 to data in R0 through R3 and stores the result in D150 through D153. The number of 16-bit data blocks (4) is specified in D0. In the following step, as well with leading edge from X1C, the registers D150 through D153 is searched through for the minimum value. The value found is stored in D200, its position is stored in D201, and the number of identical minimum values is stored in D202.



**Program Example 2**

DMINP

With leading edge from X20, the following program searches for the minimum value of 32-bit data in D0 through D7 and stores the value in D100 and D101. The position of the value is stored in D102, the number of identical minimum values is stored in D103.



**NOTE**

The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

**7.5.12 SORT, SORTP, DSORT, DSORTP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

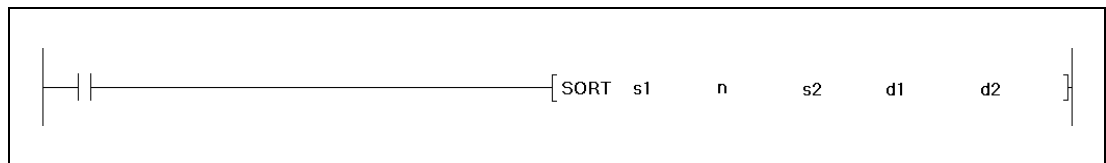
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	6
n	●	●	●	—	—	—	—	—	—		
s2	●	●	●	●	●	●	●	●	—		
d1	●	—	—	●	●	●	●	●	—		
d2	—	●	●	—	—	—	—	—	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">SORT</td> <td style="padding: 2px;">s1</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">s2</td> <td style="padding: 2px;">d1</td> <td style="padding: 2px;">d2</td> </tr> </table> </div>	MELSEC	SORT	s1	n	s2	d1	d2	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">SORT_M</td> <td style="padding: 2px;">s1 , n , s2 , d1 , d2</td> </tr> </table> </div>	SORT_M	s1 , n , s2 , d1 , d2
MELSEC	SORT	s1	n	s2	d1	d2					
SORT_M	s1 , n , s2 , d1 , d2										

**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s1	First number of device storing data to be sorted.	BIN 16-/32-bit
n	Number of data blocks to be sorted.	BIN 16-bit
s2	Number of data blocks to be compared each sort operation.	BIN 16-bit
d1	Number of bit to be set after finishing sort operation.	Bit
d2	For system use only.	BIN 16-bit

## Functions     **Sorting 16-/32-bit data**

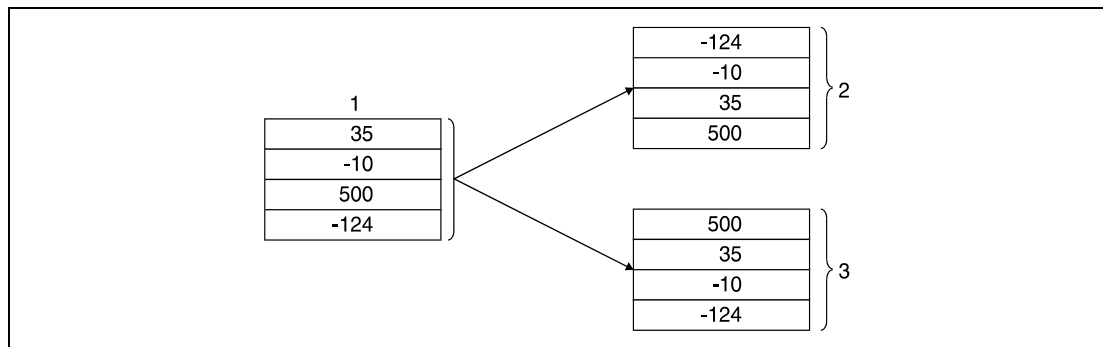
### **SORT     Sorting 16-bit data**

The SORT instruction sorts 16-bit data specified by s1 in ascending or descending order. The number of data to be sorted is specified by n.

The sorting order is set via the special relay SM703:

SM703 OFF: Ascending order

SM703 ON: Descending order



<sup>1</sup> Data to be sorted

<sup>2</sup> Data sorted in ascending order (SM703 = OFF)

<sup>3</sup> Data sorted in descending order (SM703 = ON)

For finishing the SORT instruction several scans are required. The number of required scans can be calculated by the division of the maximum number of scans by the number of 16-bit data specified in s2, to be compared each scan (decimal fractions are rounded up). Increasing the number of 16-bit data specified in s2 reduces the number of required scans for sorting but increases the processing time per scan.

The required number of sorting scans until finishing the sort operation is calculated via the following equation:

$$\text{Required number of sorting scans} = ((n) \times (n-1)) / (2 \times (s2))$$

For example, for n = 10 and s2 = 1 the result is 45 sort scans until finishing the sort operation.

For n = 10 and s2 = 2 the result is 22,5. Rounded up, 23 sort scans are required.

The bit specified in d1 is reset during the sort operation and will be set again when the sort operation is finished. This bit remains set and must be reset by appropriate programming.

The devices specified in d2 and (d2)+1 are used for internal system processing during the sort operation. So, these devices must not be changed by programming.

If the value in n is changed during the operation, the operation is processed with the currently set number of 16-bit data.

By resetting the execution condition, the operation will be terminated. Upon setting the execution condition again, the sort operation will be restarted.



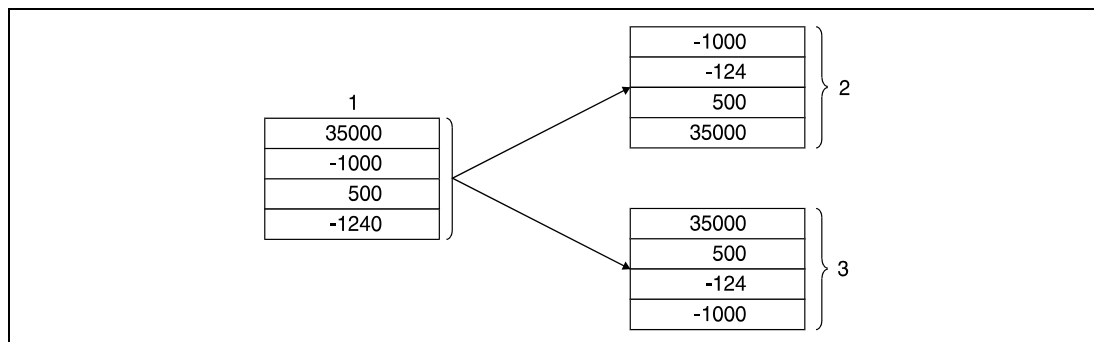
**DSORT Sorting 32-bit data**

The DSORT instruction sorts 32-bit data specified by s1 in ascending or descending order. The number of data to be sorted is specified by n.

The sorting order is set via the special relay SM703:

SM703 OFF: Ascending order

SM703 ON: Descending order



<sup>1</sup> Data to be sorted

<sup>2</sup> Data sorted in ascending order (SM703 = OFF)

<sup>3</sup> Data sorted in descending order (SM703 = ON)

For finishing the DSORT instruction several scans are required. The number of required scans can be calculated by the division of the maximum number of scans by the number of 32-bit data specified in s2, to be compared each scan (decimal fractions are rounded up). Increasing the number of 32-bit data specified in s2 reduces the number of required scans for sorting but increases the processing time per scan.

The required number of sorting scans until finishing the sort operation is calculated via the following equation:

$$\text{Required number of sorting scans} = ((n) \times (n-1)) / (2 \times (s2))$$

For example, for  $n = 10$  and  $s2 = 1$  the result is 45 sort scans until finishing the sort operation.

For  $n = 10$  and  $s2 = 2$  the result is 22,5. Rounded up, 23 sort scans are required.

The bit specified in d1 is reset during the sort operation and will be set again when the sort operation is finished. This bit remains set and must be reset by appropriate programming.

The devices specified in d2 and (d2)+1 are used for internal system processing during the sort operation. So, these devices must not be changed by programming.

If the value in n is changed during the operation, the operation is processed with the currently set number of 32-bit data.

By resetting the execution condition, the operation will be terminated. Upon setting the execution condition again, the sort operation will be restarted.

**Operation Errors**

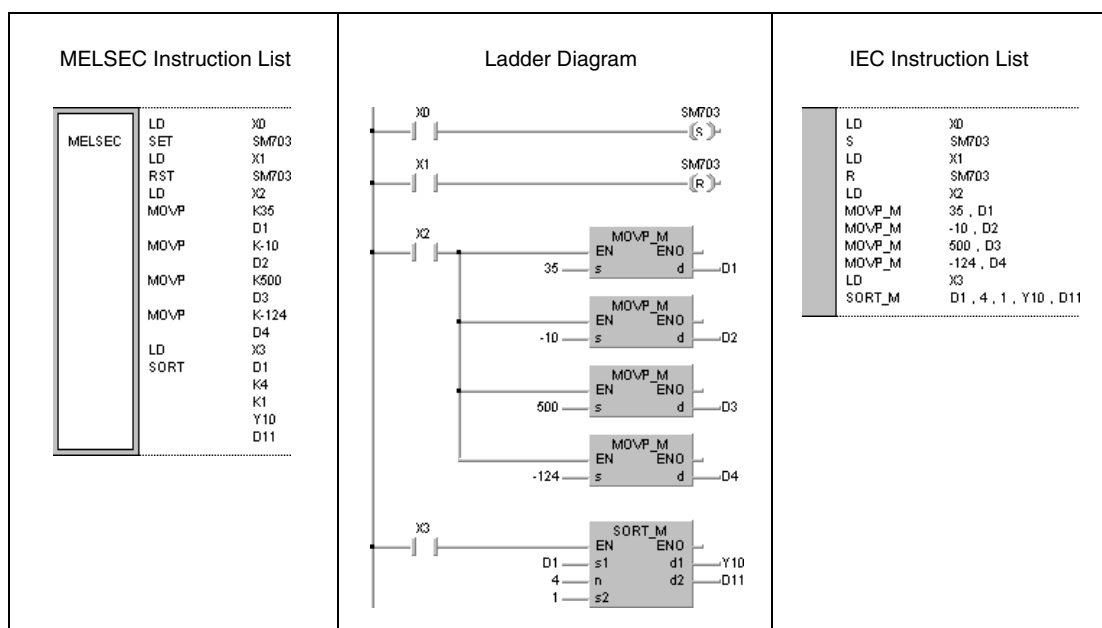
In the following cases an operation occurs and the error flag is set:

- The range specified by n or 2 x n in the device specified by s1 exceeds the relevant storage device range (error code 4101).
- The value specified in s2 is equal to or less than 0 (error code 4100).
- The value in d2 is greater than that in n (error code 4101).
- The value in (d2)+1 is greater than that in d2 (error code 4101).

**Program Example**

**SORT**

While X3 is set, the following program sorts 16-bit data in D1 through D4. In a first step with leading edge from X2, the values 35, -10, 500, and -124 are written to the registers D1 through D4. Then sorting starts. The sorting order is determined via X0 (set SM703) and X1 (reset SM703). After finishing the sort operation the output Y10 is set.



**7.5.13 WSUM, WSUMP**

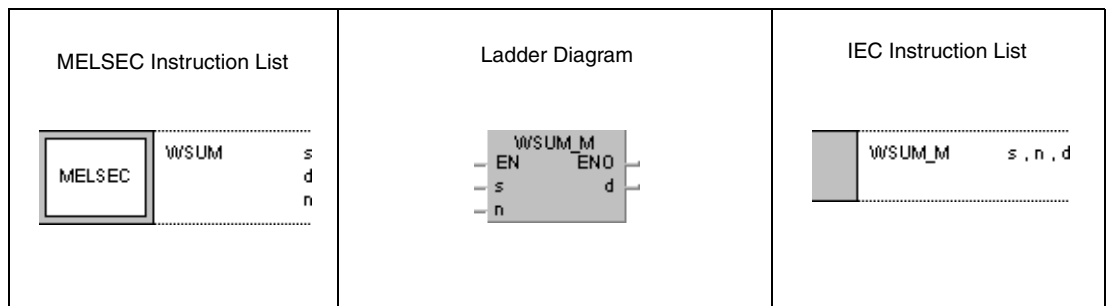
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

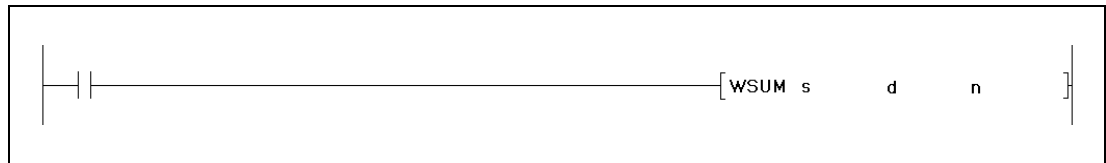
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	SM0	4	
d	●	●	●	●	●	●	●	—			
n	●	●	●	●	●	●	●	—			

**GX IEC Developer**



**GX Developer**



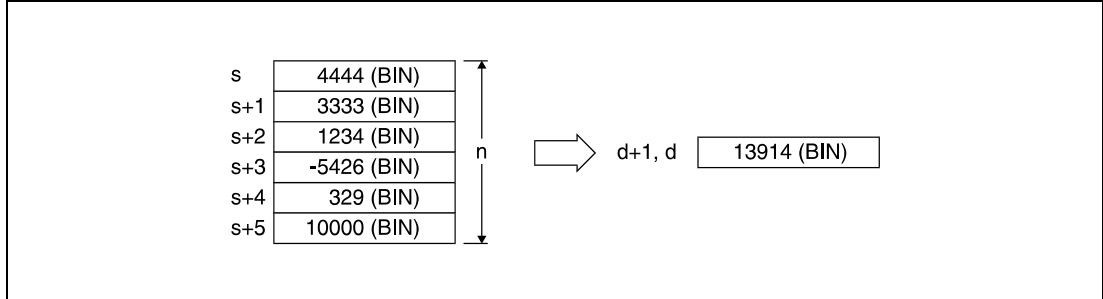
**Variables**

Set Data	Meaning	Data Type
s	First number of device storing data to be added.	BIN 16-bit
d	First number of device storing result.	BIN 32-bit
n	Number of data blocks to be added.	BIN 16-bit

**Functions**     **Calculating totals of 16-bit BIN data blocks**

**WSUM**    **Calculation of totals**

The WSUM instruction calculates the total of 16-bit data blocks in the device specified by s. The number of data blocks to be summed up is specified by n. The result is stored in the device specified by d.



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The range specified by n in the device specified by s exceeds the relevant storage device range (error code 4101).

**Program Example**

**WSUMP**

With leading edge from X1C, the following program adds BIN 16-bit data blocks in D10 through D14 and stores the result in D100 and D101.

<p>MELSEC Instruction List</p> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 5px;">MELSEC</td> <td style="padding: 5px;">LD</td> <td style="padding: 5px;">WSUMP</td> <td style="padding: 5px;">X1C</td> </tr> <tr> <td colspan="3"></td> <td style="padding: 5px;">D10</td> </tr> <tr> <td colspan="3"></td> <td style="padding: 5px;">D100</td> </tr> <tr> <td colspan="3"></td> <td style="padding: 5px;">K5</td> </tr> </table>	MELSEC	LD	WSUMP	X1C				D10				D100				K5	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 5px;">LD</td> <td style="padding: 5px;">WSUMP_M</td> <td style="padding: 5px;">X1C</td> </tr> <tr> <td colspan="2"></td> <td style="padding: 5px;">D10 , 5 , var_D100</td> </tr> </table>	LD	WSUMP_M	X1C			D10 , 5 , var_D100
MELSEC	LD	WSUMP	X1C																					
			D10																					
			D100																					
			K5																					
LD	WSUMP_M	X1C																						
		D10 , 5 , var_D100																						
<table style="margin: auto;"> <tr> <td style="padding: 5px;">D10</td> <td style="padding: 5px;">4500 (BIN)</td> <td rowspan="5" style="padding: 0 20px;">⇒</td> <td rowspan="5" style="padding: 0 20px;">D101, D100</td> <td style="padding: 5px;">14948 (BIN)</td> </tr> <tr> <td style="padding: 5px;">D11</td> <td style="padding: 5px;">2500 (BIN)</td> </tr> <tr> <td style="padding: 5px;">D12</td> <td style="padding: 5px;">-3276 (BIN)</td> </tr> <tr> <td style="padding: 5px;">D13</td> <td style="padding: 5px;">6780 (BIN)</td> </tr> <tr> <td style="padding: 5px;">D14</td> <td style="padding: 5px;">4444 (BIN)</td> </tr> </table>			D10	4500 (BIN)	⇒	D101, D100	14948 (BIN)	D11	2500 (BIN)	D12	-3276 (BIN)	D13	6780 (BIN)	D14	4444 (BIN)									
D10	4500 (BIN)	⇒	D101, D100	14948 (BIN)																				
D11	2500 (BIN)																							
D12	-3276 (BIN)																							
D13	6780 (BIN)																							
D14	4444 (BIN)																							

**7.5.14 DWSUM, DWSUMP**

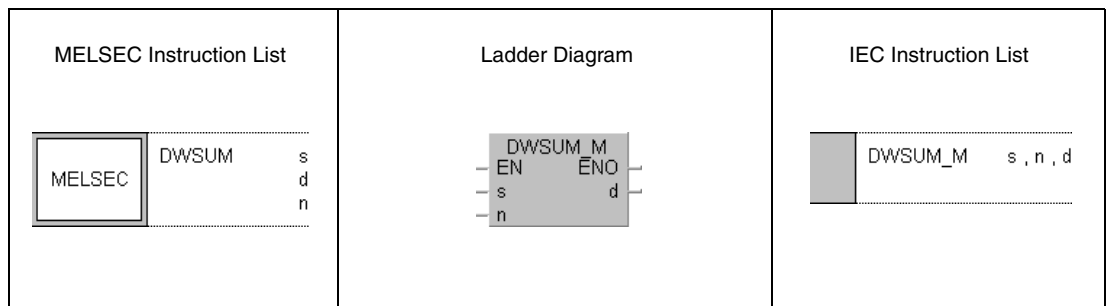
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

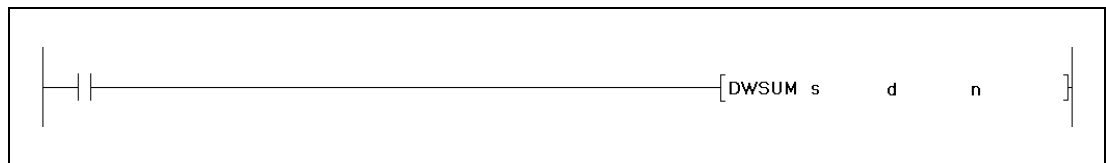
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	4
d	●	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

**GX IEC Developer**



**GX Developer**



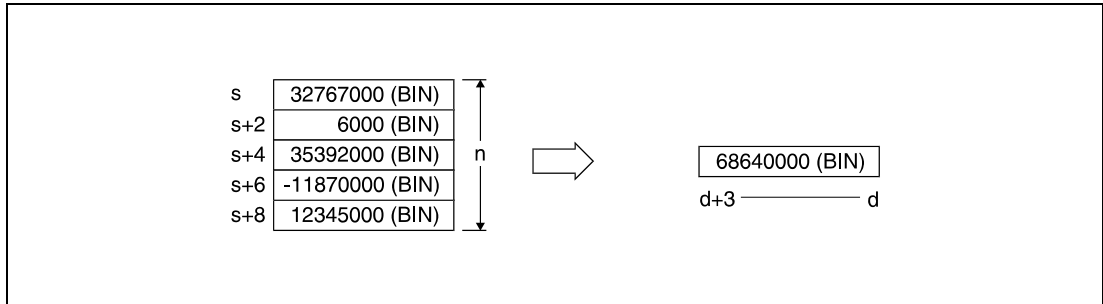
**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	First number of device storing data to be added.	BIN 32-bit	ANY32
d	First number of device storing result.	BIN-64-Bit	Array [1..4] of ANY16
n	Number of data blocks to be added.	BIN 16-bit	ANY16

**Functions**     **Calculating totals of 32-bit BIN data blocks**

**DWSUM Calculation of totals**

The DWSUM instruction calculates the total of 32-bit data blocks in the device specified by s. The number of data blocks to be summed up is specified by n. The result is stored in array[1] through array[4] in the device specified by d.



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The range specified by n in the device specified by s exceeds the relevant storage device range (error code 4101).

**Program Example**

**DWSUMP**

With leading edge from X20, the following program adds 32-bit BIN data blocks in D100 through D107 and stores the result in D10 through D13.

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">MELSEC</td><td style="border: none;">LD</td><td style="border: none;">X20</td></tr> <tr><td style="border: none;"></td><td style="border: none;">DWSUMP</td><td style="border: none;">D100 D10 K5</td></tr> </table>	MELSEC	LD	X20		DWSUMP	D100 D10 K5	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">LD</td><td style="border: none;">X20</td></tr> <tr><td style="border: none;"></td><td style="border: none;">DWSUMP_M</td><td style="border: none;">var_D100 , 4 , var_D10</td></tr> </table>	LD	X20		DWSUMP_M	var_D100 , 4 , var_D10	
MELSEC	LD	X20												
	DWSUMP	D100 D10 K5												
LD	X20													
	DWSUMP_M	var_D100 , 4 , var_D10												
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%; border-collapse: collapse;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>D101, D100</td><td>11245600 (BIN)</td></tr> <tr><td>D103, D102</td><td>27543200 (BIN)</td></tr> <tr><td>D105, D104</td><td>558800 (BIN)</td></tr> <tr><td>D107, D106</td><td>-15675000 (BIN)</td></tr> </table> </td> <td style="text-align: center; vertical-align: middle;">➔</td> <td style="width: 40%; border-collapse: collapse;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>23672600 (BIN)</td></tr> </table> <p style="text-align: center;">D13 ————— D10</p> </td> </tr> </table>			<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>D101, D100</td><td>11245600 (BIN)</td></tr> <tr><td>D103, D102</td><td>27543200 (BIN)</td></tr> <tr><td>D105, D104</td><td>558800 (BIN)</td></tr> <tr><td>D107, D106</td><td>-15675000 (BIN)</td></tr> </table>	D101, D100	11245600 (BIN)	D103, D102	27543200 (BIN)	D105, D104	558800 (BIN)	D107, D106	-15675000 (BIN)	➔	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>23672600 (BIN)</td></tr> </table> <p style="text-align: center;">D13 ————— D10</p>	23672600 (BIN)
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>D101, D100</td><td>11245600 (BIN)</td></tr> <tr><td>D103, D102</td><td>27543200 (BIN)</td></tr> <tr><td>D105, D104</td><td>558800 (BIN)</td></tr> <tr><td>D107, D106</td><td>-15675000 (BIN)</td></tr> </table>	D101, D100	11245600 (BIN)	D103, D102	27543200 (BIN)	D105, D104	558800 (BIN)	D107, D106	-15675000 (BIN)	➔	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>23672600 (BIN)</td></tr> </table> <p style="text-align: center;">D13 ————— D10</p>	23672600 (BIN)			
D101, D100	11245600 (BIN)													
D103, D102	27543200 (BIN)													
D105, D104	558800 (BIN)													
D107, D106	-15675000 (BIN)													
23672600 (BIN)														

**NOTE**

*This program will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 7.6 Structured program instructions

Structured program instructions call programs and parts of programs or switch over between them. In addition, instructions for index qualification and program repetitions (loops) are supplied.

The following table gives an overview of all instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Repetition instructions	FOR	FOR_M
	NEXT	NEXT_M
	BREAK	BREAK_MD
	BREAKP	BREAK_P_MD
Subroutine program calls	CALL	CALL_M
	CALLP	CALLP_M
	RET	RET_M
	FCALL	FCALL_MD
	FCALLP	FCALL_P_MD
Subroutine calls between program files (only possible with GX Developer)	ECALL	ECALL_M
	ECALLP	ECALLP_M
	EFCALL	EFCALL_M
	EFCALLP	EFCALLP_M
Main/subprogram switching	CHG	CHG_M
Microcomputer program call	SUB	SUB_M
	SUBP	SUBP_M
Index qualification of entire ladders	IX	IX_MD
	IXEND	IXEND_MD
Designation of qualification values in index qualification of entire ladders	IXDEV	IXDEV_M
	IXSET	IXSET_M

### 7.6.1 FOR, NEXT

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

**Devices  
MELSEC A**

	Usable Devices																			Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011
	Bit Devices							Word Devices (16-bit)							Constant		Pointer		Level					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P					
n							●	●	●	●	●	●	●	●	●	●	●				3/1	●		●

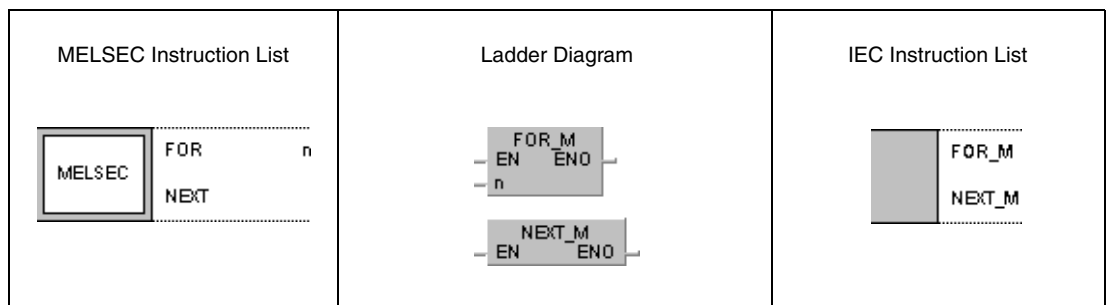
<sup>1</sup> The FOR instruction requires three steps, the NEXT instruction requires one step.  
Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

**Devices  
MELSEC Q**

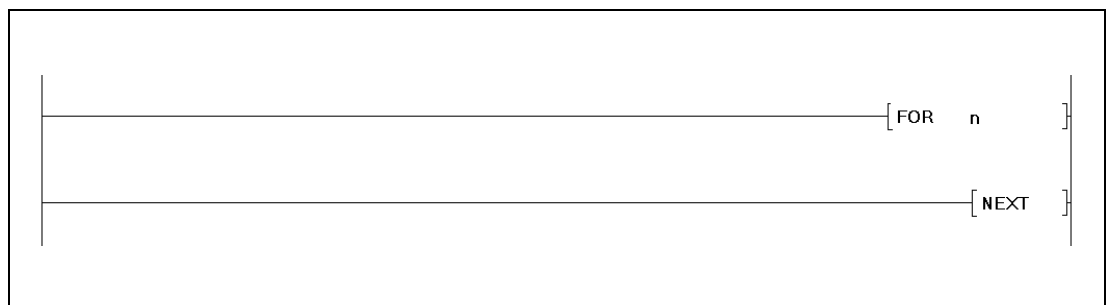
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
n	●	●	●	●	●	●	●	●	—	SM0	2/1 ●

<sup>1</sup> The FOR instruction requires two steps, the NEXT instruction requires one step.

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
n	Number of repetitions of the FOR/NEXT loops (from 1 to 32767).	BIN 16-bit



**Functions FOR/NEXT loop instruction****FOR/NEXT Loop instruction**

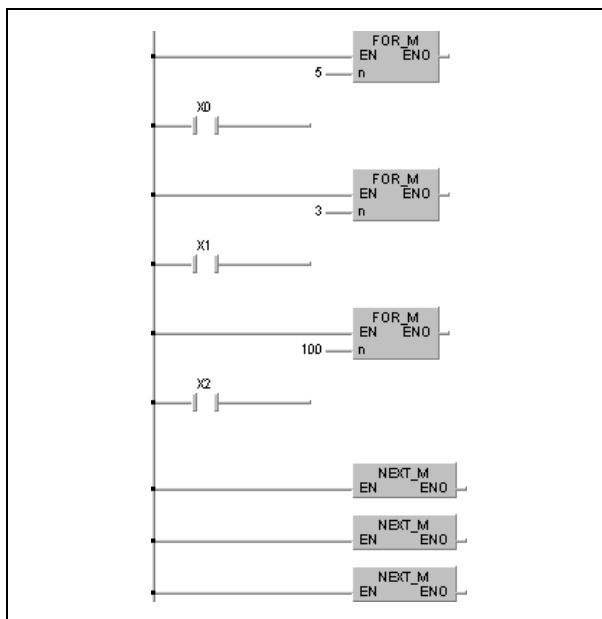
The FOR/NEXT loop repeats single program sequences without setting an input condition. The program sequence located between the FOR and the NEXT command is repeated for n times.

After executing the FOR/NEXT loop for n times, the next program step following the NEXT command is executed.

The variable n can be specified from 1 to 32767. If n is less than or equal to 0, it is processed as 1. Thus, the FOR/NEXT loop will be executed at least once.

If a program sequence between the FOR/NEXT loop is not intended to be executed, it can be skipped by a jump instruction (CJ or SCJ).

In total, up to 16 levels (A series = 5 levels) of FOR/NEXT loops can be nested up. The following diagram illustrates the principle of nesting:

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The END/FEND instruction is executed after a FOR instruction and before the NEXT instruction. (Q series and System Q = error code 4200).
- The NEXT instruction is executed before the FOR instruction (Q series and System Q = error code 4201).
- The number of FOR instructions does not match the number of NEXT instructions.
- A JMP instruction with a jump destination outside the FOR/NEXT loop is executed within a FOR/NEXT loop.
- A STOP instruction is programmed within a FOR/NEXT loop (Q series and System Q = error code 4200).
- The maximum number of nesting levels is exceeded (Q series and System Q = error code 4202).

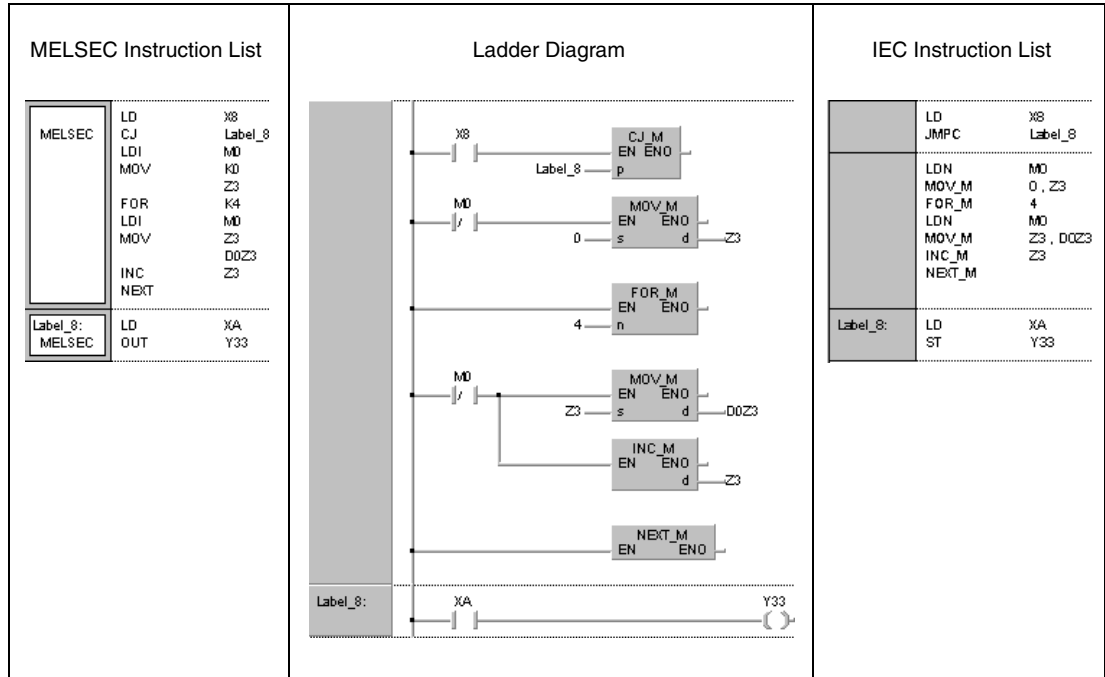
**NOTE**

*For Q series and System Q only:  
In order to terminate the execution of a FOR/NEXT loop before it is finished, a BREAK instruction must be programmed.*

*Apply the EGP/EGF instruction, to connect a switch condition to the FOR/NEXT instruction.*

**Program Example**

The following program processes the program sequence between FOR and NEXT for four times, if X8 is not set. The FOR/NEXT loop is skipped, if X8 is set.



**7.6.2 BREAK, BREAKP**

**CPU**

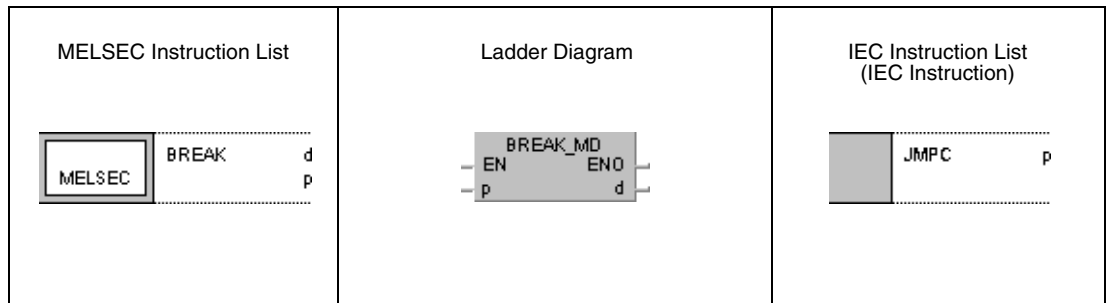
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

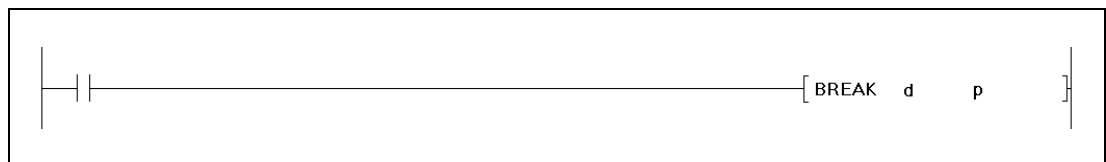
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other P
	Bit	Word		Bit	Word						
d	●	●	●	●	●	●	●	—	—	SM0	3
p	●	●	●	●	●	●	●	—	●		

**GX IEC  
Developer**



**GX  
Developer**

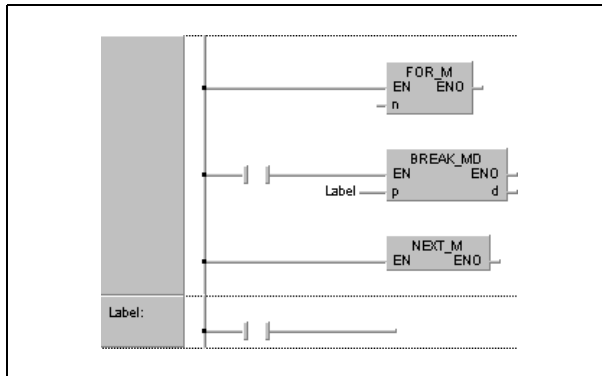


**Variables**

Set Data	Meaning	Data Type
d	Device storing the remaining number of FOR/NEXT loops.	BIN 16-bit
p	Destination address (Pointer/Label) to be jumped to after executing the BREAK instruction.	Pointer/label

**Functions**    **Terminating a FOR/NEXT loop****BREAK**    **Terminating the FOR/NEXT execution**

The BREAK instruction terminates a FOR/NEXT loop execution and jumps to the pointer/label specified by p.



The number of remaining FOR/NEXT loops to be executed is stored in the device specified by d.

The BREAK instruction can only be applied during the execution of a FOR/NEXT loop.

The BREAK instruction can only be applied to one nesting level. For several nesting levels the appropriate number of BREAK instructions must be executed.

**Operation Errors**

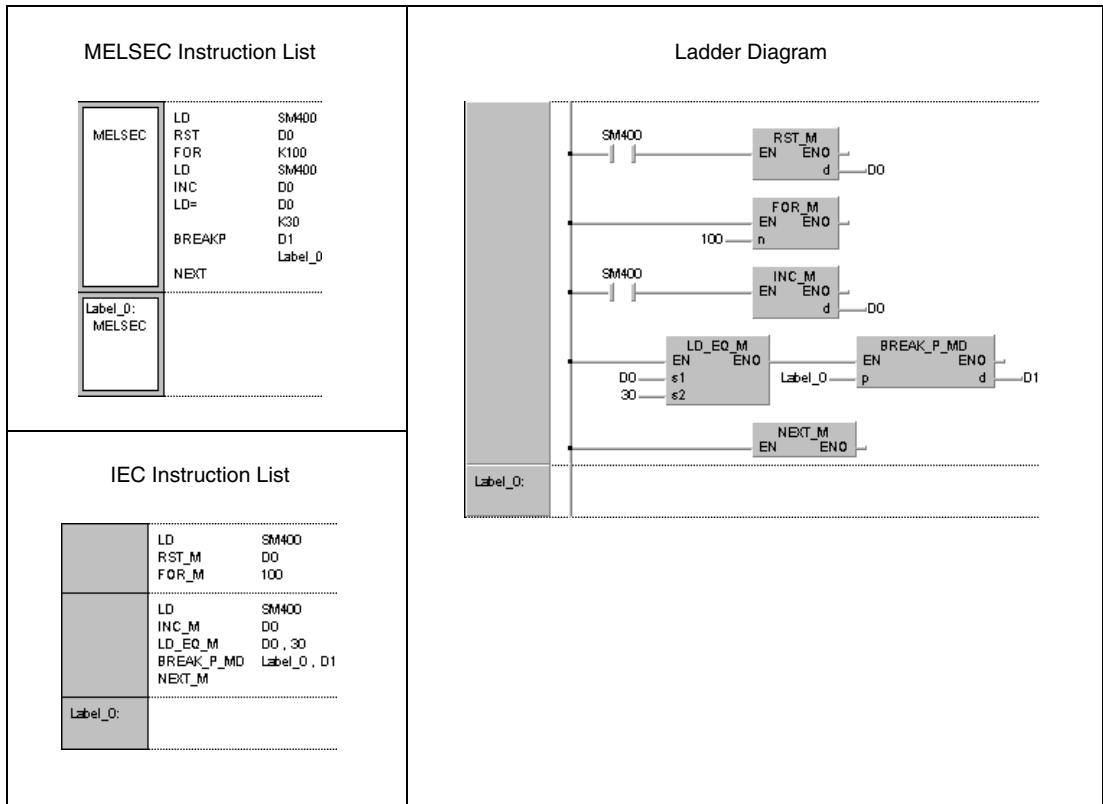
In the following cases an operation error occurs and the error flag is set:

- The BREAK instruction was executed without a FOR/NEXT loop (Q series and System Q = error code 4203).
- There is no subroutine program stored at the specified pointer/label (Q series and System Q = error code 4210).

**Program Example**

**BREAKP**

The following program terminates the execution in the 30th FOR/NEXT loop and jumps to the program part specified with label\_0. The number of remaining FOR/NEXT loops (70) is stored in D1.



7.6.3 CALL, CALLP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

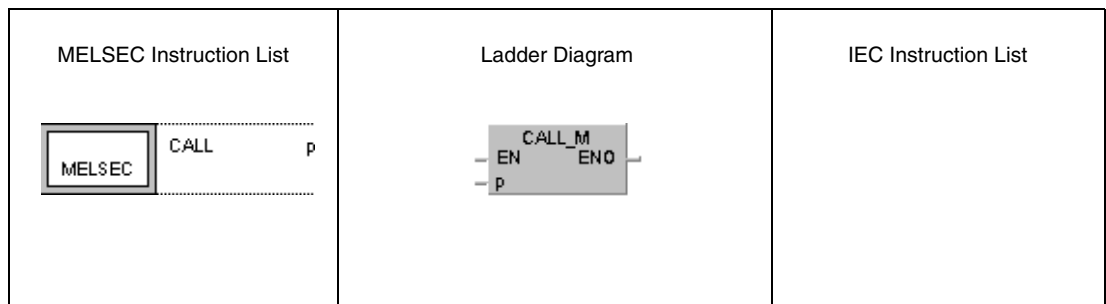
p	Usable Devices																	Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9010 M9011			
	Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level								
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K						H (16#)	P	I
																						●	3 <sup>1</sup>		●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

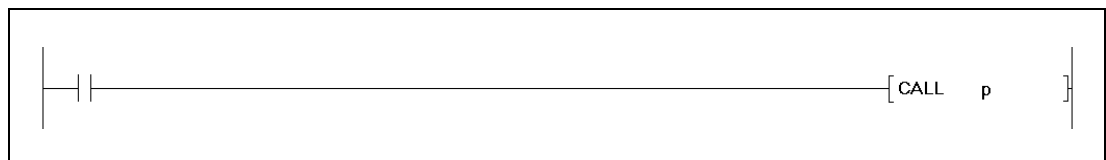
Devices  
MELSEC Q

p	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other		
	Bit	Word		Bit	Word						
	—	●	●	—	—	—	—	—	—	SM0	2

GX IEC  
Developer



GX  
Developer



Variables

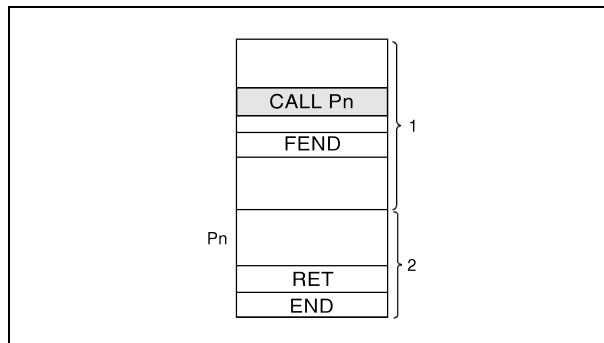
Set Data	Meaning	Data Type
pn	Address number (pointer/label) of subroutine program.	Pointer/label

NOTE

The CALL instruction should not be used with the IEC editor because the subroutine structure is generated by the GX IEC Developer.

**Functions**    **Calling a subroutine program****CALL**    **Subroutine program call**

The CALL instruction calls a subroutine program specified by a pointer Pxx in the GX Developer or by a label in the GX IEC Developer, respectively. The pointer (label) addresses of the A series range from P(label)0 to P(label)255. The pointer (label) addresses of the Q series and System Q range from P(label)0 to P(label)4095. Refer to the notes on programming pointer (label) addresses for the jump instructions (CJ, SCJ, JMP).



<sup>1</sup> Main routine program

<sup>2</sup> Subroutine program

The CALL instruction calls a subroutine program specified by pointer (label) addresses. In total, up to 5 subprogram nesting levels for the A series and 16 subprogram nesting levels for the Q series can be addressed.

Devices that were set during the execution of a subroutine program remain set, even if the routine is not executed any longer. In order to reset these devices the FCALL instruction has to be applied.

**Operation Errors**

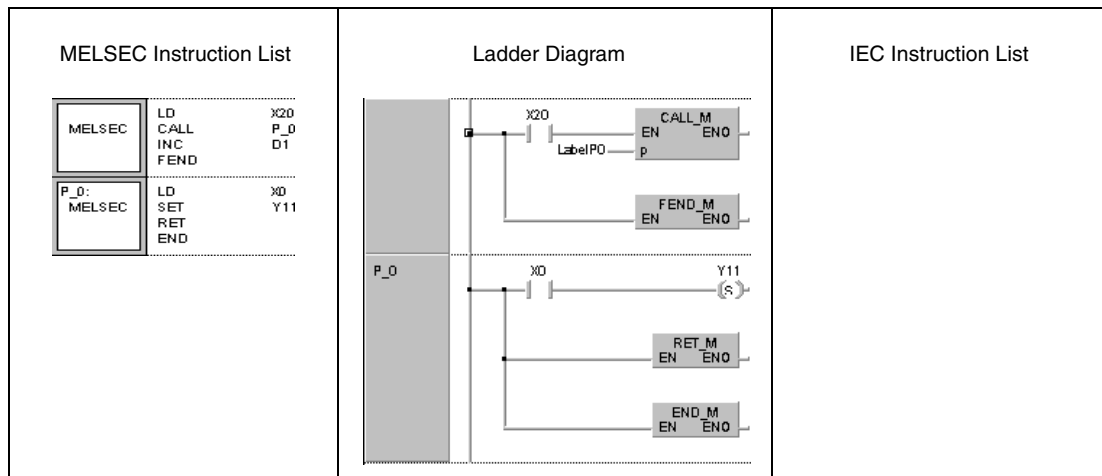
In the following cases an operation error occurs and the error flag is set:

- After execution of a CALL instruction an END, FEND, GOEND, or STOP instruction is executed, without a prior RET instruction (Q series and System Q = error code 4211).
- A RET instruction is executed before a CALL instruction (Q series and System Q = error code 4212).
- More than 5 nesting levels (A series) or 16 nesting levels (Q series) are executed (Q series and System Q = error code 4210).
- There is no subroutine program stored at the specified pointer/label (Q series and System Q = error code 4210).
- A CALL instruction specifies a pointer (label) address beyond of P(label)255 (A series).
- The sub routine is exited via a JMP instruction before executing a RET instruction (A series).

**Program Example**

**CALL**

While X20 is set, the following program executes the subroutine program at pointer/label P\_0.



**NOTE**

*In MELSEC-mode, the FEND, END, and RET instructions have to be programmed by the user. After the program organization unit has been processed no further one will be executed because it would follow the FEND instruction.*

*Alternatively to this programming, the IEC editor can be used. In that case the FEND instruction would be set by the compiler of the GX IEC Developer automatically.*



7.6.4 RET

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

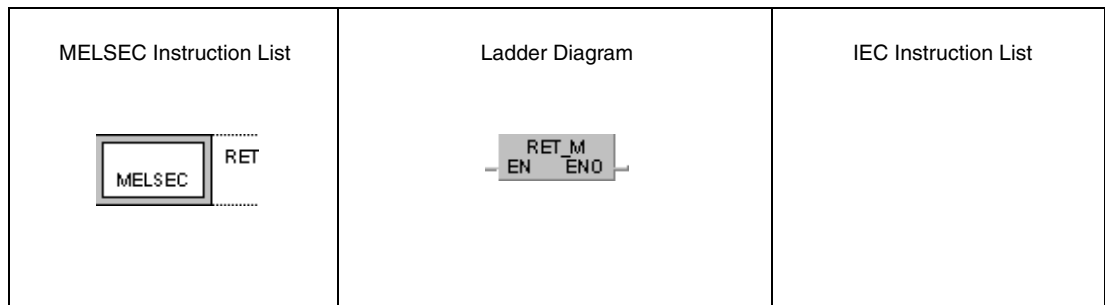
Usable Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag	
Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N	
																					1	1		●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

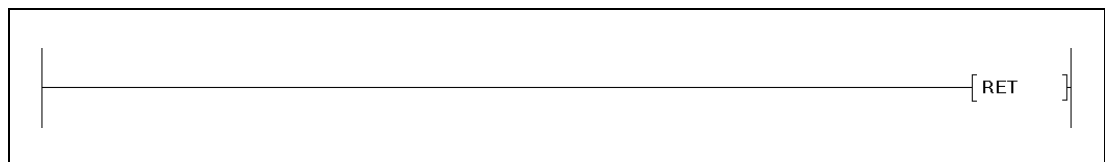
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	SM0	1

GX IEC  
Developer



GX  
Developer



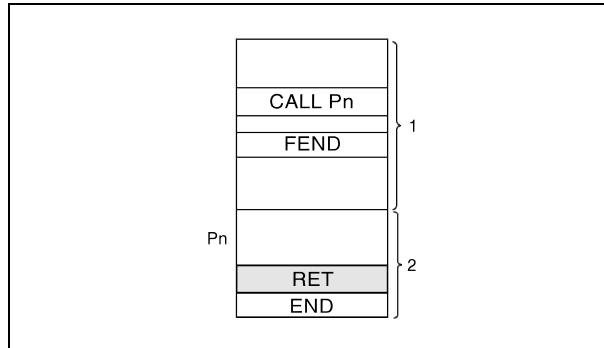
Variables

Set Data	Meaning	Data Type
—	—	—

**Functions**      **End of subroutine program**

**RET**      **Return to main program**

The RET instruction marks the end of a subroutine program. The program jumps back to the program step, that is specified after the CALL, FCALL, ECALL, or EFCALL instruction.



<sup>1</sup> Main routine program

<sup>2</sup> Subroutine program

**NOTE**

*Between a RET instruction in the subroutine program and the END instruction in the main routine program, a NOP instruction has to be programmed, because otherwise the CPU will not process the program properly (A series only).*

*In the MELSEC-mode the FEND, END, and RET instructions have to be programmed by the user. After the program organization unit has been processed no further one will be executed because it would follow the FEND instruction.*

*Alternatively to this programming, the IEC editor can be used. In that case the FEND instruction would be set by the compiler of the GX IEC Developer automatically.*

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- After execution of a CALL instruction an END, FEND, GOEND, or STOP instruction is executed, without a prior RET instruction (Q series and System Q = error code 4211).
- A RET instruction is executed before a CALL instruction (Q series and System Q = error code 4212).

**7.6.5 FCALL, FCALLP**

**CPU**

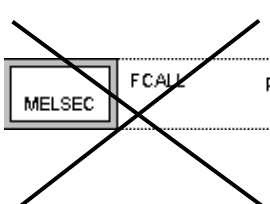
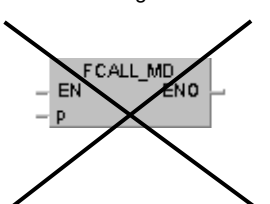
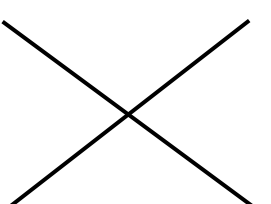
AnS	AnN	AnA (S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant			Other
	Bit	Word		Bit	Word						
p	—	●	●	—	—	—	—	—	—	SM0	2

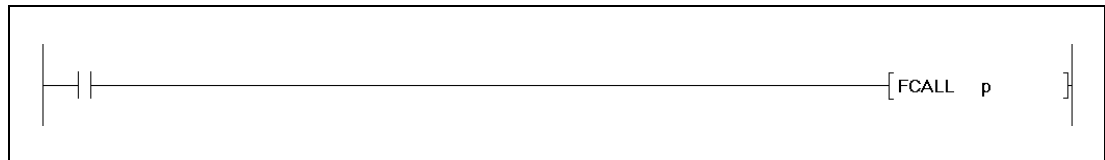
**GX IEC Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	---	--

**NOTE**

*These instructions are not available in GX IEC Developer.*

**GX Developer**



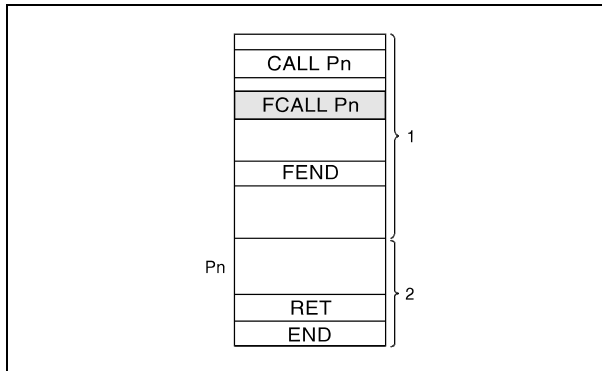
**Variables**

Set Data	Meaning	Data Type
pn	Address number (pointer/label) of subroutine program.	Pointer/label

**Functions     Resetting outputs in subroutine programs**

**FCALL    Resetting outputs (in conjunction with CALL instruction)**

On resetting the execution condition for the FCALL instruction, the contacts and coils in the subroutine program specified in p (pointer/label) are treated as if the execution condition of the according instruction was not set.



<sup>1</sup> Main routine program

<sup>2</sup> Subroutine program

The condition of coils and contacts after execution of the FCALL instruction or the respective condition of coils and contacts with the according execution condition not set is listed below:

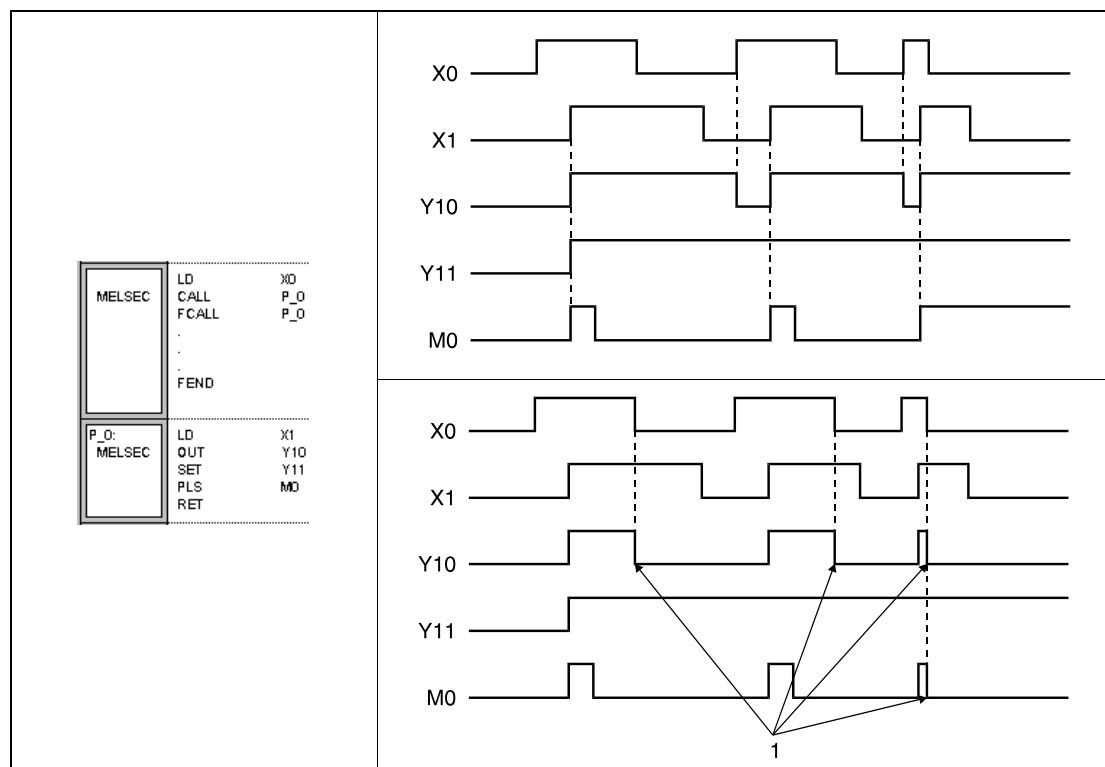
Instruction	Condition of contacts and coils
OUT instruction	All contacts and coils, designated by the OUT instruction are reset.
SET instruction	All contacts and coils, designated by these instructions remain their condition.
RST instruction	
SFT instruction	
Basic instructions	
Application instructions	
PLS instruction	All contacts and coils, designated by these instructions adopt a condition as if the execution conditions of the instructions were not set.
Instructions generating an output pulse	The setting values are reset to 0.
Setting values of low- and high-speed timers	
Setting values of retentive timers	
Setting values of counters	The setting values remain set.

The FCALL instruction is used in conjunction with a CALL instruction.

The following diagrams show a program, applying the CALL and FCALL instructions. The diagrams on the right show the signal condition of several contacts designated by several instructions. The diagram on the top right shows the contact conditions without applying an FCALL instruction. The diagram on the bottom right shows the contact conditions applying an FCALL instruction.

If only the CALL instruction is applied, the conditions of contacts and coils designated in a subroutine program are remained after resetting the execution condition of the CALL instruction (see diagram on top right).

If the FCALL instruction is applied, the conditions of contacts and coils designated in a subroutine program are reset after resetting the execution condition of the FCALL instruction (see diagram on bottom right). The same applies to coils and contacts designated by an OUT or PLS instruction, or by a pulse generating instruction.



<sup>1</sup> Forced OFF by FCALL instruction

The FCALL instruction calls a subroutine program specified via the pointer address (label). In total up to 16 nesting levels can be programmed.

### Operation Errors

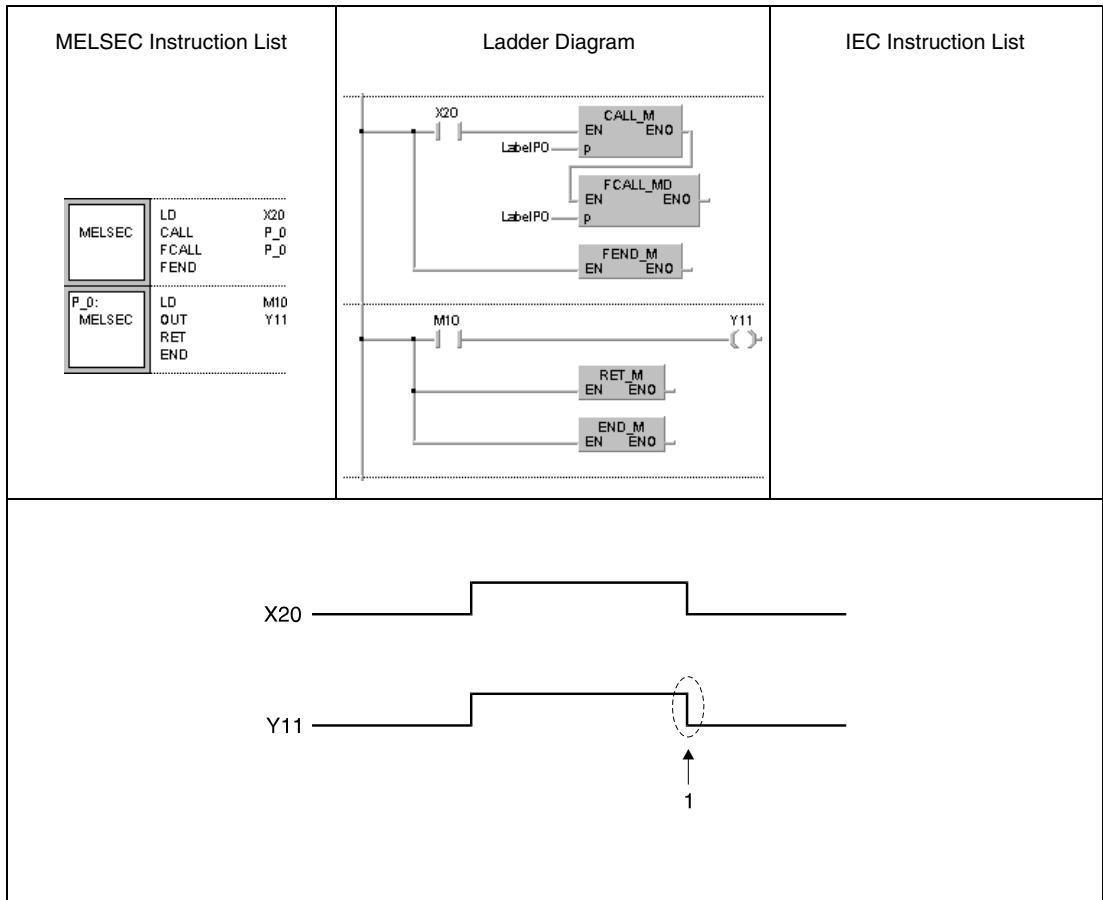
In the following cases an operation error occurs and the error flag is set:

- After execution of an FCALL instruction an END, FEND, GOEND, or STOP instruction is executed, without a prior RET instruction (Q series and System Q = error code 4211).
- A RET instruction is executed before an FCALL instruction (Q series and System Q = error code 4212).
- More than 16 nesting levels are executed (Q series and System Q = error code 4213).
- There is no subroutine program stored at the specified pointer/label (Q series and System Q = error code 4210).

**Program Example**

**FCALL**

While X20 is set, the following program executes the subroutine program at pointer address (label) P\_0. If X20 is reset, the FCALL instruction resets the output Y11 as well (1).



**7.6.6 ECALL, ECALLP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>


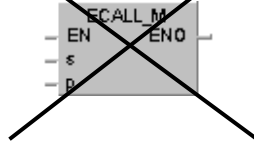
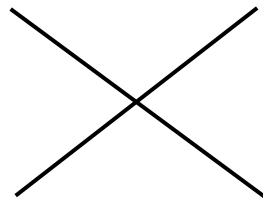
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant			Other
	Bit	Word		Bit	Word						
p	—	—	—	—	—	—	—	●	SMO	3	
s1	● <sup>1</sup>	●	●	●	●	●	●	—			
s2	● <sup>1</sup>	●	●	●	●	●	●	—			
s3	● <sup>1</sup>	●	●	●	●	●	●	—			
s4	● <sup>1</sup>	●	●	●	●	●	●	—			
s5	● <sup>1</sup>	●	●	●	●	●	●	—			

<sup>1</sup> Annunciators (F) cannot be used

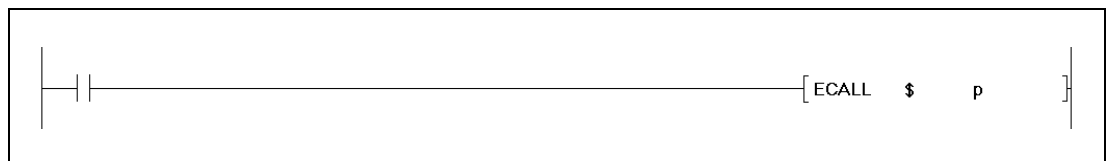
**GX IEC Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	--	---

**NOTE**

*These instructions are not available in GX IEC Developer.*

**GX Developer**



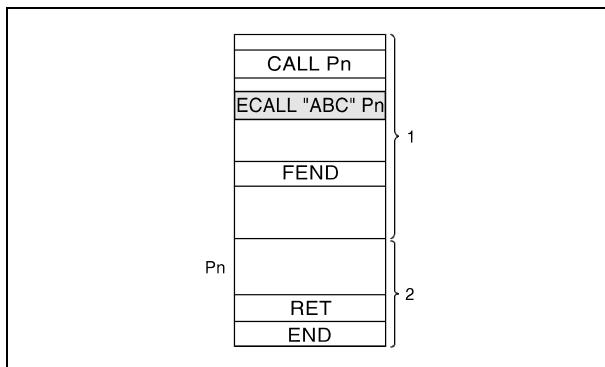
**Variables**

Set Data	Meaning	Data Type
File name	Name of program file containing the subroutine program.	Character string
pn	Address number (pointer/label) of subroutine program.	Pointer/label
s1 to s5	Device number that passes to subroutine	Bit BIN16-bit BIN 32-bit

**Functions**     **Calling a subroutine program in a program file**

**ECALL Subroutine program call**

The ECALL instruction calls a subroutine program specified by pointer address (label) in a program file specified by a file name. The pointer address (label) ranges from P(label)0 to P(label)4095. Refer to the notes on programming pointer (label) addresses for the jump instructions (CJ, SCJ, JMP).



<sup>1</sup> Main routine program (file name: MAIN)

<sup>2</sup> Subroutine program (file name: ABC)

Only files stored in internal memory (drive 0) can be specified by the file name.

When calling program files no file extension is required.

The ECALL instruction calls a subroutine program specified via the pointer address (label). In total up to 16 nesting levels can be programmed. However, this 16 levels is the total number of levels in the CALL, FCALL, ECALL, and EFCALL instructions.

Devices that were set during the execution of a subroutine program remain set, even if the routine is not executed any longer. In order to reset these devices the EFCALL instruction has to be applied.

When function devices (FX, FY, FD) are used by a sub-routine program, specify a device with s1 through s5 corresponding to the function device. Prior to execution of the sub-routine program, bit data is transmitted to FX, and word data is transmitted to FD. After the execution of the sub-routine program, the contents of FY and FD are transmitted to the corresponding device.

The amount of data which can be moved to a function register FD depends on the devices specified in s1 through s5: Up to 2 words of constants, Index registers or digit designated bit devices or up to 4 words of word devices can be stored. For example, if the device D0 is designated in s2, the registers D0, D1, D2 and D3 will be stored in FD1.

The number of function devices used by sub-routine programs must be identical to the number of devices handed over by the ECALL instruction in s1 through s5.

The function devices must be identical to the types of devices handed over by the ECALL instruction.

The devices specified in s1 through s5 must not overlap.



**Operation Errors**

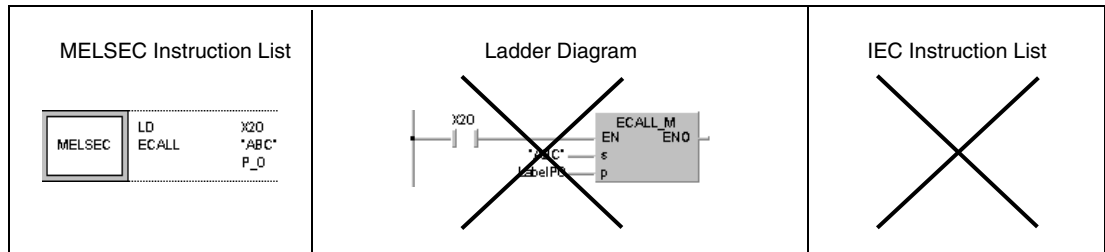
In the following cases an operation error occurs and the error flag is set:

- After execution of an ECALL instruction an END, FEND, GOEND, or STOP instruction is executed, without a prior RET instruction (error code 4211).
- A RET instruction is executed before an ECALL instruction (error code 4212).
- More than 16 nesting levels are executed (error code 4213).
- A function device (FX, FY, or FD) is specified in s1 to s5 (error code 4101)
- There is no subroutine program stored at the specified pointer/label (error code 4210).
- The specified program file does not exist (error code 4210).
- The specified program file cannot be executed (error code 2411).

**Program Example**

**ECALL**

While X20 is set, the following program executes the subroutine program at pointer/label P\_0 in the program file "ABC".



7.6.7 EFCALL, EFCALLP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant			Other
	Bit	Word		Bit	Word						
p	—	—	—	—	—	—	—	—	●	SM0	3
s1	● <sup>1</sup>	●	●	●	●	●	●	●	—		
s2	● <sup>1</sup>	●	●	●	●	●	●	●	—		
s3	● <sup>1</sup>	●	●	●	●	●	●	●	—		
s4	● <sup>1</sup>	●	●	●	●	●	●	●	—		
s5	● <sup>1</sup>	●	●	●	●	●	●	●	—		

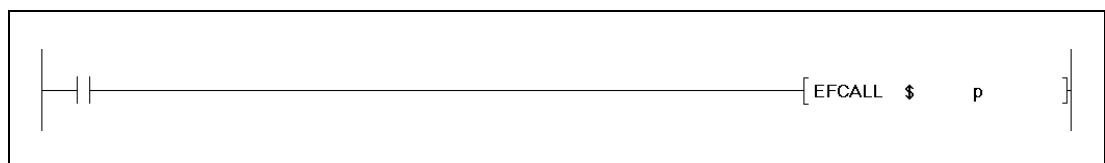
<sup>1</sup> Annunciators (F) cannot be used

GX IEC Developer

<p>MELSEC Instruction List</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p>
--------------------------------	-----------------------	-----------------------------

**NOTE** *These instructions are not available in GX IEC Developer.*

GX Developer



Variables

Set Data	Meaning	Data Type
file name	Name of program file containing the subroutine program.	Character string
pn	Address number (pointer/label) of subroutine program.	Pointer/label
s1 to s5	Device number that passes to subroutine	Bit BIN16-bit BIN 32-bit

**Functions     Resetting outputs in subroutine programs in program files**

**EFCALL Resetting outputs (in conjunction with ECALL)**

On resetting the execution condition for the EFCALL instruction, the contacts and coils in the subroutine program specified in p (pointer/label) are treated as if the execution condition of the according instruction was not set.

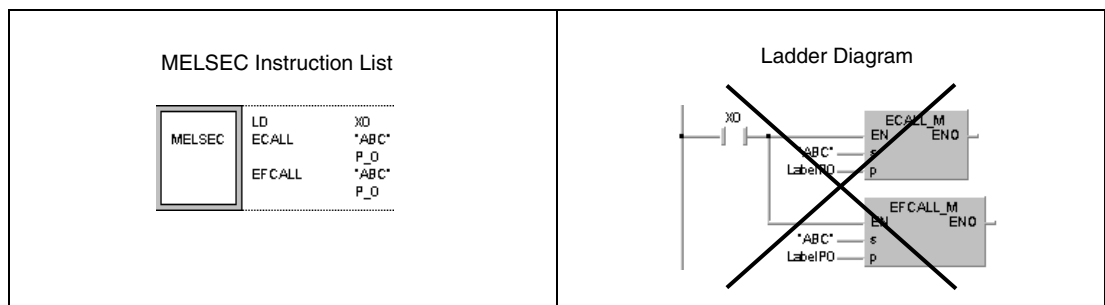
The EFCALL instruction executes subroutine programs, that are located within a different program file from that one calling them.

The condition of coils and contacts after execution of the EFCALL instruction or the respective condition of coils and contacts with the according execution condition not set is listed below:

Instruction	Condition of contacts and coils
OUT instruction	All contacts and coils, designated by the OUT instruction are reset.
SET instruction	All contacts and coils, designated by these instructions remain their condition.
RST instruction	
SFT instruction	
Basic instructions	
Application instructions	
PLS instruction	All contacts and coils, designated by these instructions adopt a condition as if the execution conditions of the instructions were not set.
Instructions generating an output pulse	
Setting values of low- and high-speed timers	The setting values are reset to 0.
Setting values of retentive timers	The setting values remain set.
Setting values of counters	

The EFCALL instruction is used in conjunction with a CALL instruction.

The following diagrams show a program applying the ECALL and EFCALL instructions.



The EFCALL instruction calls a subroutine program specified via the pointer address (label). In total up to 16 nesting levels can be programmed. However, this 16 levels is the total number of levels in the CALL, FCALL, ECALL, and EFCALL instructions.

Devices that were set during the execution of a subroutine program remain set, even if the routine is not executed any longer. In order to reset these devices the EFCALL instruction has to be applied.

When function devices (FX, FY, FD) are used by a sub-routine program, specify a device with s1 through s5 corresponding to the function device. Prior to execution of the sub-routine program, bit data is transmitted to FX, and word data is transmitted to FD. After the execution of the sub-routine, the contents of FY and FD are transmitted to the corresponding device.

The amount of data which can be moved to a function register FD depends on the devices specified in s1 through s5: Up to 2 words of constants, Index registers or digit designated bit devices or up to 4 words of word devices can be stored. For example, if the device D0 is designated in s2, the registers D0, D1, D2 and D3 will be stored in FD1.

The number of function devices used by sub-routine programs must be identical to the number of devices handed over by the ECALL instruction in s1 through s5.

The function devices must be identical to the types of devices handed over by the ECALL instruction.

The devices specified in s1 through s5 must not overlap.

**Operation Errors**

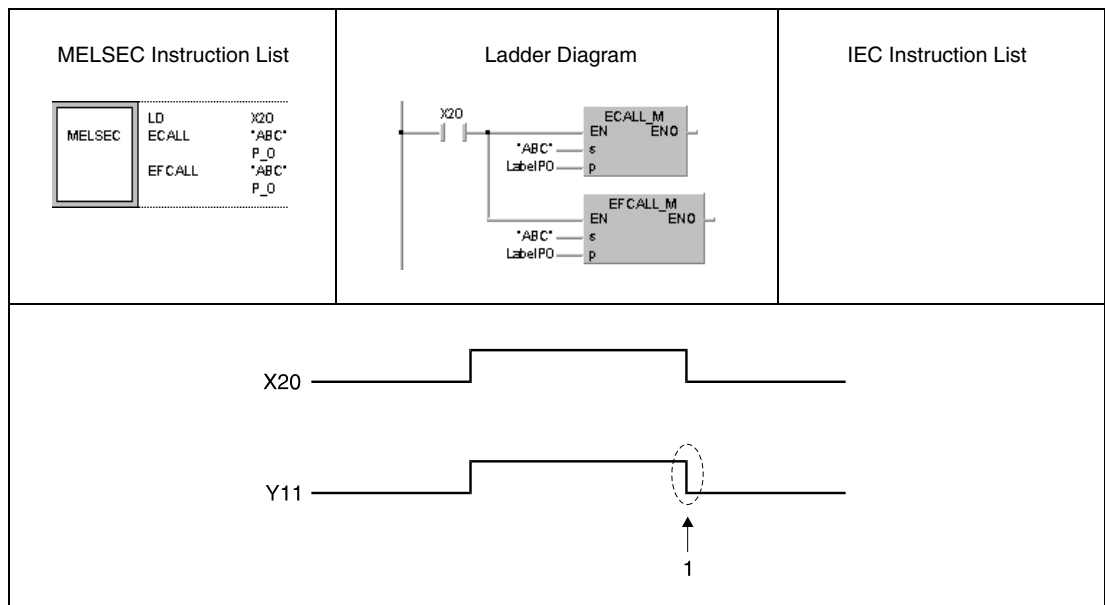
In the following cases an operation error occurs and the error flag is set:

- After execution of an EFCALL instruction an END, FEND, GOEND, or STOP instruction is executed, without a prior RET instruction (error code 4211).
- A RET instruction is executed before an EFCALL instruction (error code 4212).
- More than 16 nesting levels are executed (error code 4213).
- There is no subroutine program stored at the specified pointer/label (error code 4210).
- A function device (FX, FY, or FD) is specified in s1 to s5 (error code 4101)
- The specified program file does not exist (error code 4210).
- The specified program file cannot be executed (error code 2411).

**Program Example**

EFCALL

While X20 is set, the following program executes the subroutine program at pointer address (label) P\_0 in the program file "ABC". If X20 is reset, the EFCALL instruction resets the output Y11 as well (1).



7.6.8 CHG

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
	● <sup>1</sup>	● <sup>2</sup>	● <sup>3</sup>		

<sup>1</sup> A3N CPUs only

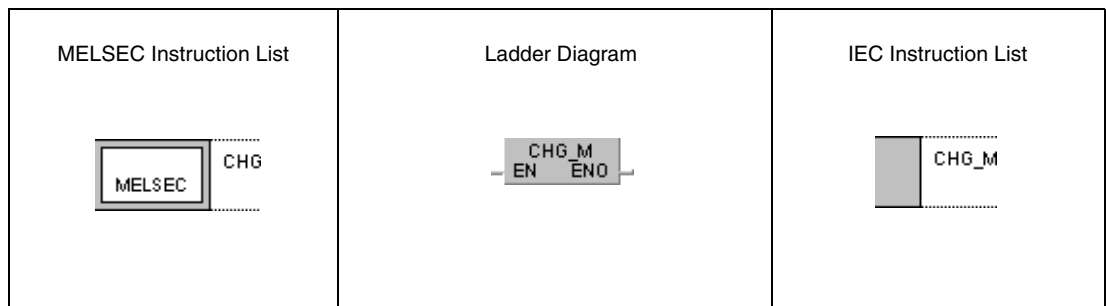
<sup>2</sup> A3A CPUs only

<sup>3</sup> A3U CPUs only

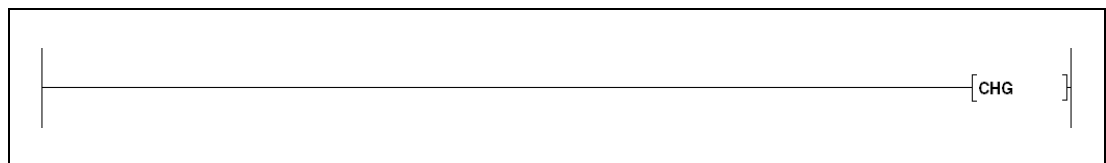
Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag		
Bit Devices							Word Devices (16-bit)						Constant		Pointer							Level	
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I	N	M9012	M9010 M9011	
																						1	

GX IEC  
Developer



GX  
Developer



Variables

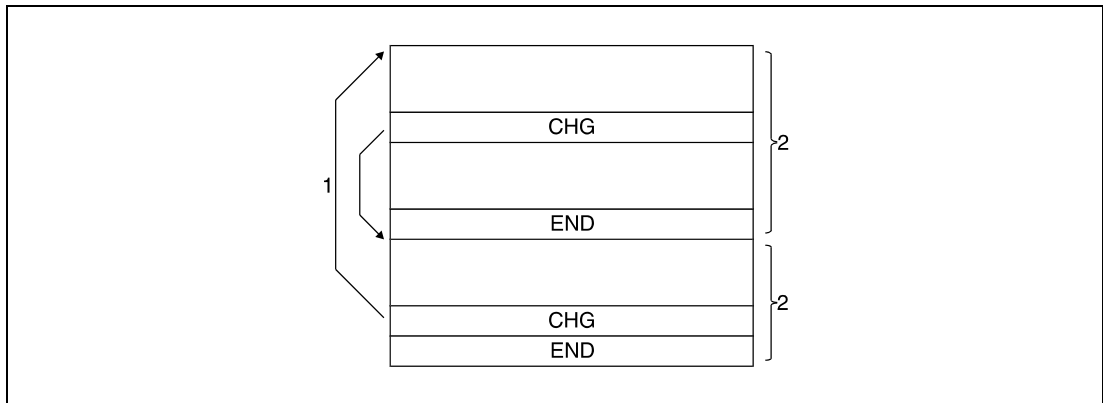
Set Data	Meaning	Data Type
—	—	—

**Functions Switching between MAIN and SUB program**

**CHG Switch instruction**

With the input condition set, the CHG instruction enables switching between MAIN and SUB programs. Switching is performed after processing timers, counters, and self diagnostics.

Refer to chapter 7.6.9 of this manual for functions and application of the SUB program parts.



<sup>1</sup> Timer, counter processing, self diagnostics

<sup>2</sup> Sequence program

**Switching between MAIN and SUB program part**

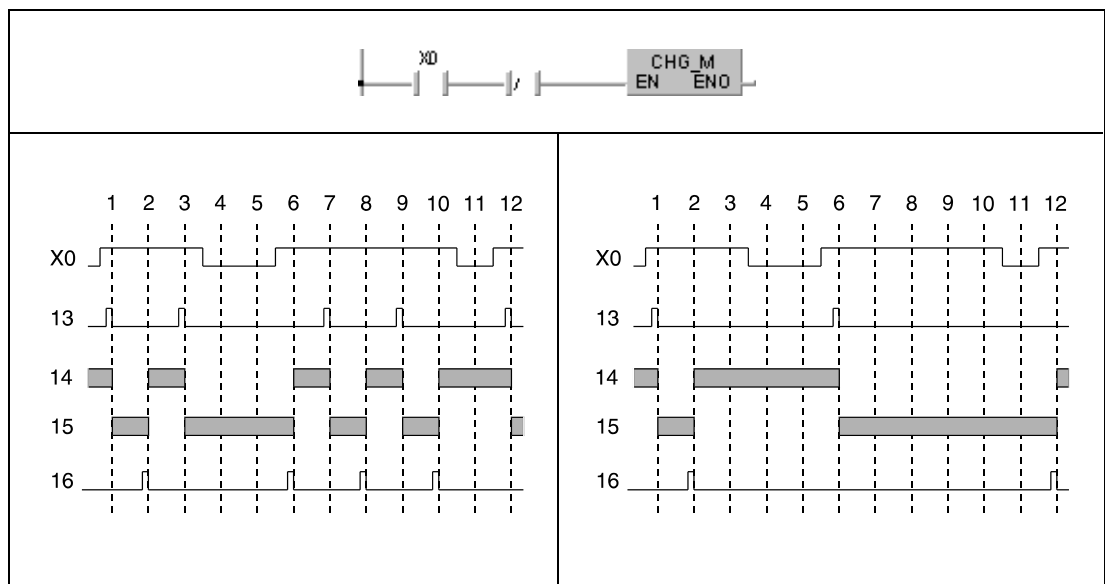
**CHG Using an A3□CPU**

With an A3□CPU the CHG instruction is only executed with leading edge from the input condition. The operation result of the input condition depends on the status of the internal relay M9050. The function of the CHG instruction therefore changes depending on the status of M9050.

An A3 NCPU does not support the internal relay M9050. It processes data as if M9050 was set.

The following upper diagram shows a programmed CHG instruction. This program part is located prior to an END or FEND instruction within a MAIN or SUB program.

The bottom diagrams show the corresponding signal conditions. The signal conditions on the bottom left correspond to the internal relay M9050 not set. The signal conditions on the bottom right correspond to M9050 set. The following table shows processing depending on the operation status of X0.



The execution of the CHG instruction in the MAIN sequence is indicated 13, the MAIN sequence is indicated 14, the subsequence is indicated 15, and the execution of the CHG instruction is indicated 16.

Status of X0	Status of M9050	
	OFF	ON
0	No switching between MAIN and SUB sequence programs (4, 5, 11).	No switching between MAIN and SUB sequence programs (4, 5, 11).
1	The CHG instruction is executed every scan and switches between MAIN and SUB sequence programs (2, 3, 7, 8, 9, 10).	The MAIN sequence program is only switched to the SUB sequence program, then back to the MAIN sequence program on the first leading edge from X0 (2).
0 → 1	Switching between MAIN and SUB sequence programs (1, 6, 12).	Switching between MAIN and SUB sequence programs (1, 6, 12).

After execution of the CHG instruction END processing is performed for the current program. Processing starts from step 0 of the other program. The GX IEC Developer automatically switches over at the end of the MAIN or SUB sequence.

**CHG instruction in conjunction with a PLS instruction**

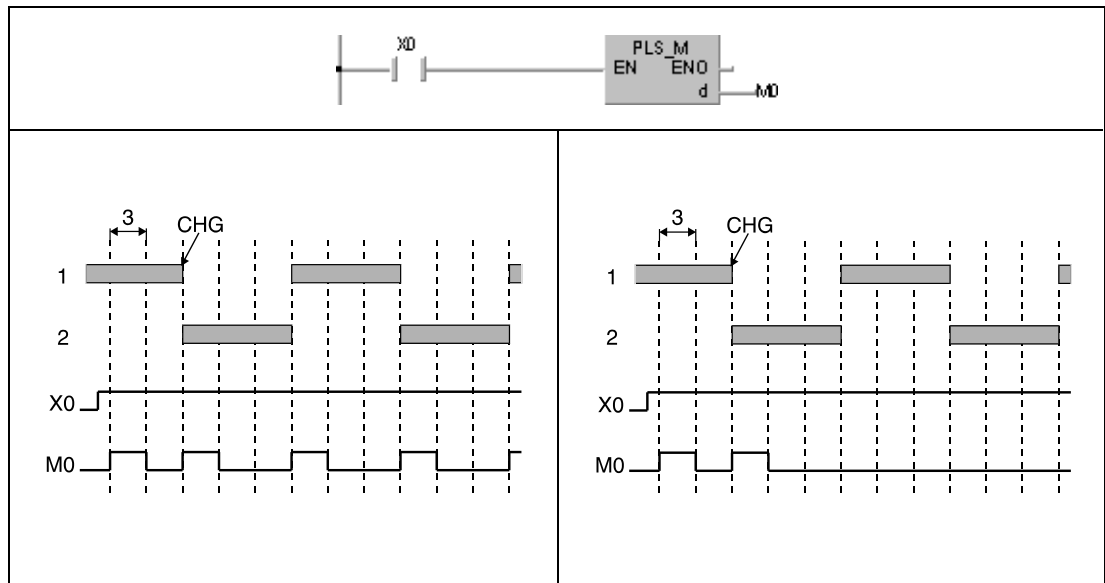
**CHG Using an A3□CPU**

With an A3□CPU the functions of the PLS instruction depends on the status of the internal relay M9050.

An A3 NCPU does not support the internal relay M9050. It processes data as if M9050 was set.

The following upper diagram shows a programmed PLS instruction. This program part is located at the beginning (step 0) of the MAIN or SUB sequence.

The bottom diagrams show the corresponding signal conditions. The signal conditions on the bottom left correspond to the internal relay M9050 not set. The signal conditions on the bottom right correspond to M9050 set. The following table shows processing depending on the operation status of X0.



Processing of the MAIN sequence is indicated 1, processing of the SUB sequence is indicated 2, and one program scan is indicated 3.

Status of X0	Status of M9050	
	OFF	ON
0	M0 is not set.	M0 is not set.
1	M0 is only set during the first scan after switching by the CHG instruction.	M0 is only set during the first scan of the SUB sequence program selected by the CHG instruction executed after X0 is switched ON.
0 → 1	M0 is only set during one scan.	M0 is only set during one scan.



**CHG instruction in conjunction with a pulsed instruction (xP)**

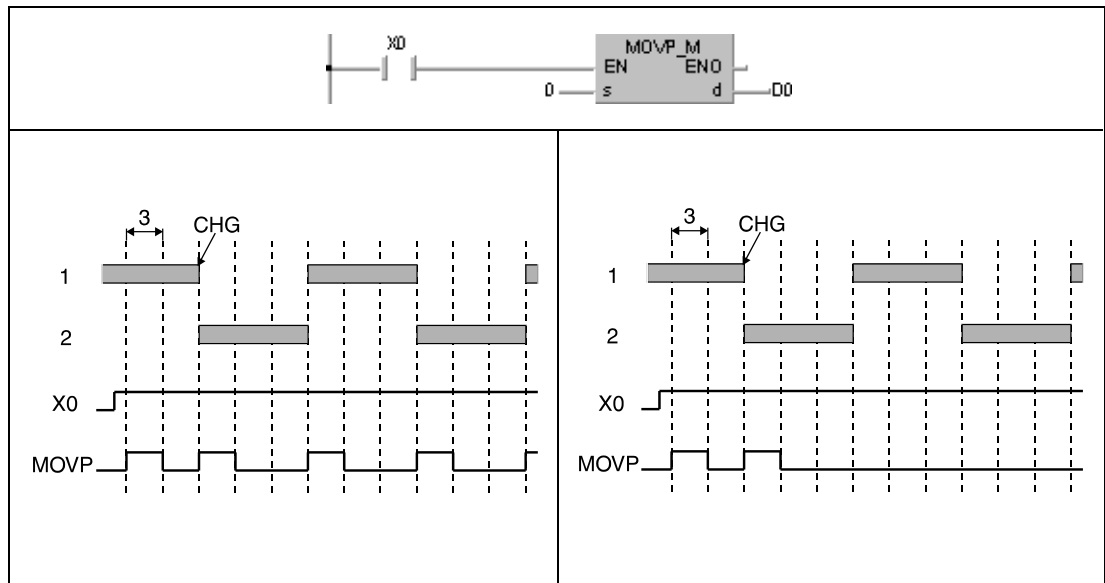
**CHG Using an A3□CPU**

With an A3□CPU the function of a pulsed instruction (xP) depends on the status of the internal relay M9050.

An A3 NCPU does not support the internal relay M9050. It processes data as if M9050 was set.

The following upper diagram shows a programmed pulsed instruction. This program part is located at the beginning (step 0) of the MAIN or SUB sequence.

The bottom diagrams show the corresponding signal conditions. The signal conditions on the bottom left correspond to the internal relay M9050 not set. The signal conditions on the bottom right correspond to M9050 set. The following table shows processing depending on the operation status of X0.



Processing of the MAIN sequence is indicated 1, processing of the SUB sequence is indicated 2, and one program scan is indicated 3.

Staus of X0	Status of M9050	
	OFF	ON
0	The MOV P instruction is not executed.	The MOV P instruction is not executed.
1	The MOV P instruction is only executed during the first scan after switching by the CHG instruction.	The MOV P is only executed during the first scan of the SUB sequence program selected by the CHG instruction executed after X0 is switched ON.
0 → 1	The MOV P instruction is executed once.	The MOV P instruction is executed once.

**CHG instruction and counting of counters**

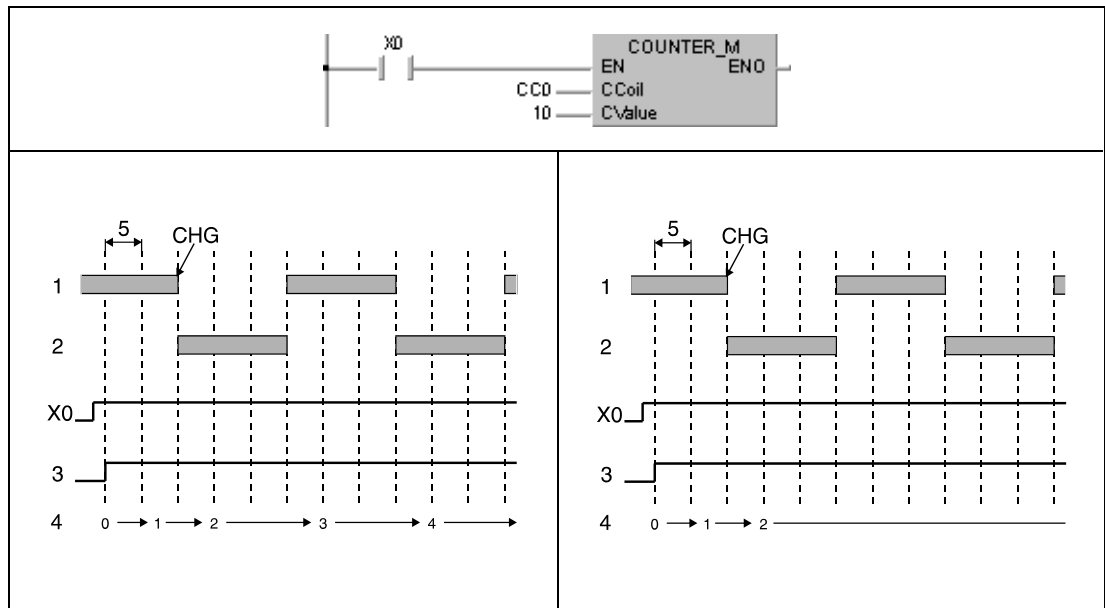
**CHG Using an A3□CPU**

With an A3□CPU the function of counters depends on the status of the internal relay M9050, provided that all other input conditions remain set.

An A3 NCPU does not support the internal relay M9050. It processes data as if M9050 was set.

The following upper diagram shows a programmed counter instruction. This program part is located at the beginning (step 0) of the MAIN or SUB sequence.

The bottom diagrams show the corresponding signal conditions. The signal conditions on the bottom left correspond to the internal relay M9050 not set. The signal conditions on the bottom right correspond to M9050 set. The following table shows processing depending on the operation status of X0.



Processing of the MAIN sequence is indicated 1, processing of the SUB sequence is indicated 2, the contact of C0 is indicated 3, the current value of C0 is indicated 4, and one program scan is indicated 5.

Status of X0	Status of M9050	
	OFF	ON
0	The current value of the counter is not changed.	The current value of the counter is not changed.
1	The current value of the counter is incremented by 1, after END (FEND, CHG) is executed during the first scan of the program selected by the CHG instruction.	The current value of the counter is incremented by 1, after END (FEND, CHG) is executed during the first scan of the program selected by the CHG instruction executed after X0 is switched ON.
0 → 1	The current value of the counter is incremented by 1, after END (FEND, CHG) is executed.	The current value of the counter is incremented by 1, after END (FEND, CHG) is executed.

**CHG instruction and timing of timers**

All CPUs capable of processing the CHG instruction supply two different storage areas for timer setting values. One for the MAIN sequence and one for the SUB sequence. Hence, timers are only processed due to the currently processed storage area (MAIN/SUB).

The setting values of timers currently not in use are reset to 0 in the according storage area. A setting value of 0 corresponds to an infinite value, so the timer will never expire.

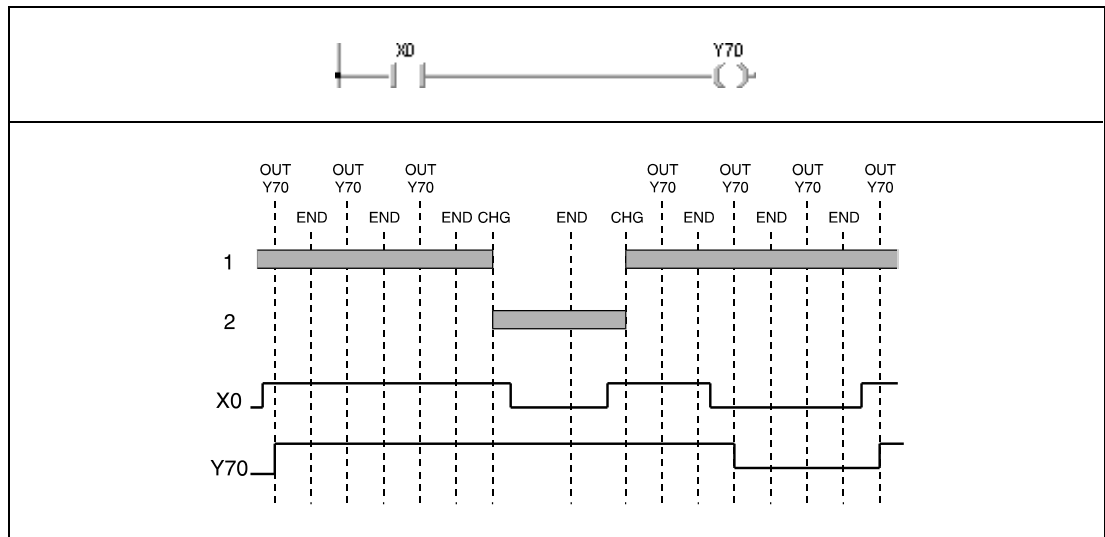
If after starting a timer the storage area is switched from MAIN/SUB via the CHG instruction, the timer is not processed in the program part being switched over. This is because the timer was programmed in the suspended program and its timer setting is regarded as 0 in the current program being switched to. After switching back to the suspended program the timer processing is continued. The timer expires if the current value is greater than the setting value or less than 0. When the timer has expired, the timer contact is switched ON.

**CHG instruction and processing of OUT instructions**

All CPUs capable of processing the CHG instruction switch the output contacts depending on the currently processed program part.

The output contacts retain their status after switching from the current to a different program part (MAIN/SUB area). Their status even remains unchanged, if the input conditions change.

The following upper diagram shows a programmed OUT instruction. This program part is located in the MAIN storage area. The output Y70 is not used in the SUB storage area.



The bottom diagram shows the signal conditions. Processing of the MAIN area is indicated 1, processing of the SUB area is indicated 2.

While processing the MAIN area, the output Y70 is switched ON/OFF depending on the input condition of X0. While processing the SUB area, the status of Y70 even remains unchanged if the input condition changes.

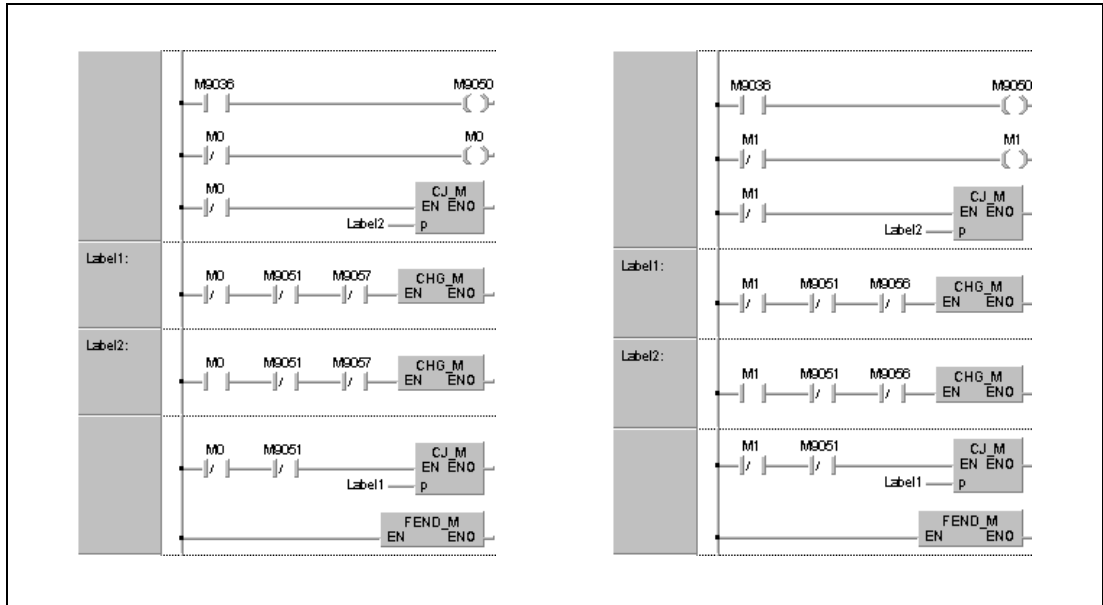
**Program Example 1**

**CHG (A3□CPU)**

For accurate operation of the CHG instruction the operation result of one program scan must be compared to the previous scan. For this reason, the internal relay M9050 must be set prior to the CHG instruction in order to load the operation result of the previous scan from the buffer memory into the main memory.

Since the CHG instruction is only executed by an A3□CPU with set input condition, programs must be written according to the following structure. The program on the left is located in the MAIN storage area, the program on the right is located in the SUB storage area.

The internal relay M9036 is always set.



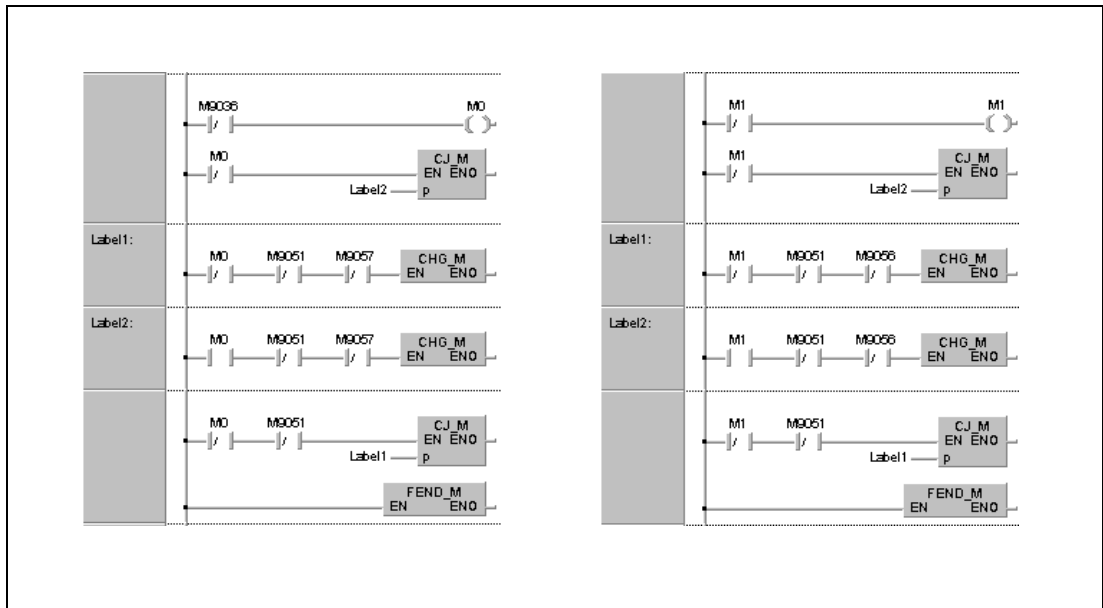
**NOTE**

When modifying a SUB program during MAIN program run or vice versa, the internal relays M9051, M9056, and M9057 must be used to disable the CHG instruction so that the CHG instruction cannot switch the currently running program to the program currently being corrected.

Thus, during an online change in the SUB area the MAIN area is not processed. With GX IEC Developer and GX Developer accurate programming is achieved automatically.

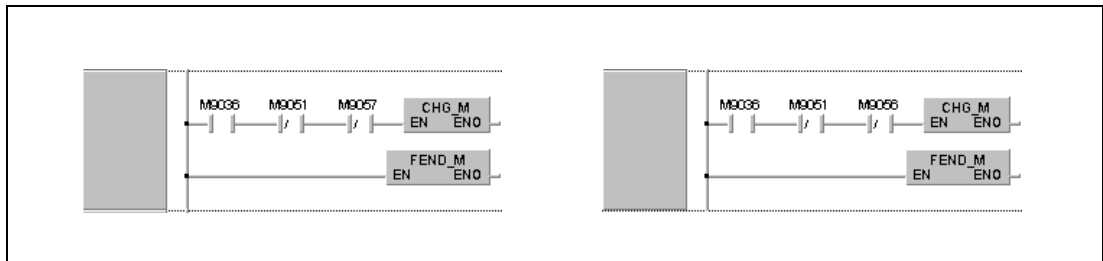
**Program Example 2** CHG (A3N CPU)

Since the CHG instruction is only executed by an A3N CPU with set input condition, programs must be written according to the following structure. The program on the left is located in the MAIN storage area, the program on the right is located in the SUB storage area.



**Program Example 3** CHG (A3H CPU)

Programs must be written according to the following structure. The program on the left is located in the MAIN storage area, the program on the right is located in the SUB storage area.



7.6.9 SUB, SUBP

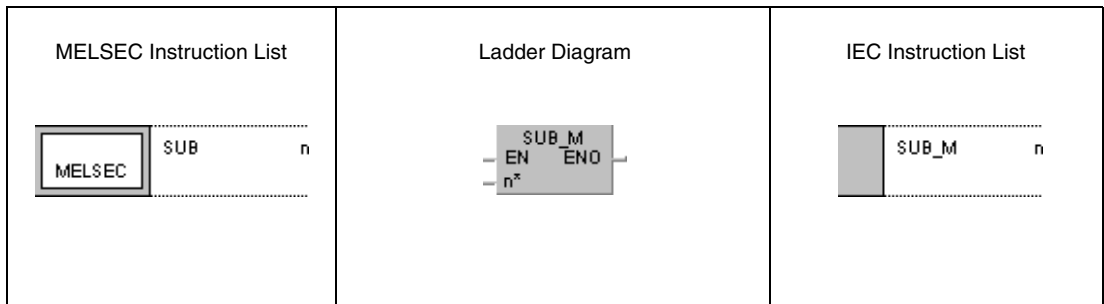
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●				

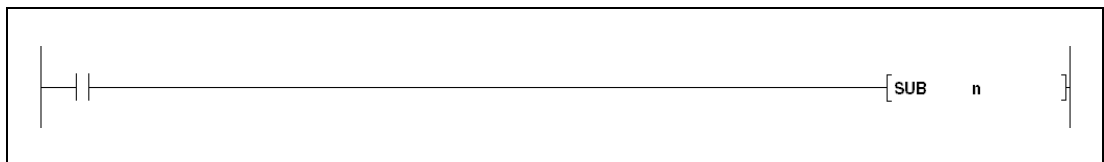
Devices  
MELSEC A

Usable Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag
Bit Devices							Word Devices (16-bit)							Constant		Pointer		Level					
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
n							●	●	●	●	●	●	●	●	●	●	●						●

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
n	Address of microcomputer program to be called.	Address

**Functions    Microcomputer program call**

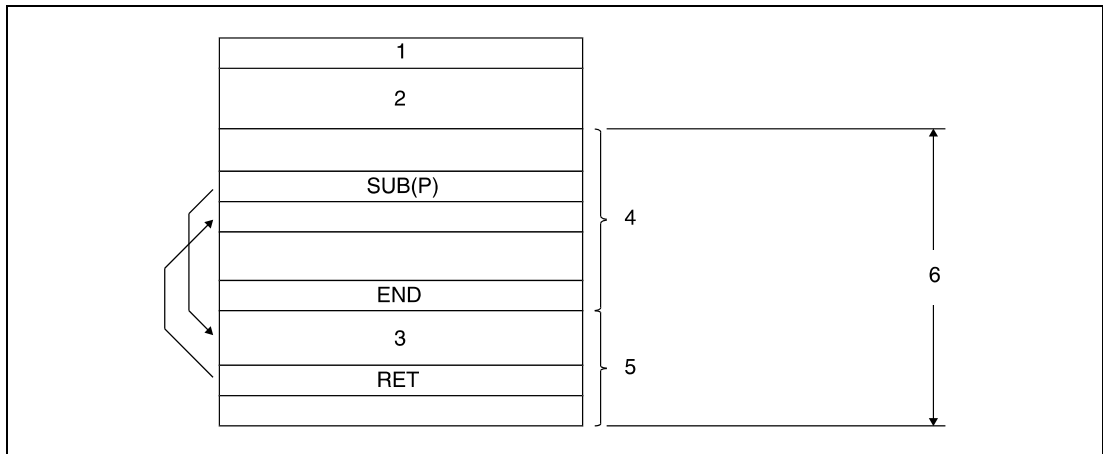
**SUB    Calling a microcomputer program**

The SUB/SUBP instruction calls a microcomputer program created by a user.

If the input condition is set, the SUB instruction calls the microcomputer program located at the address "n".

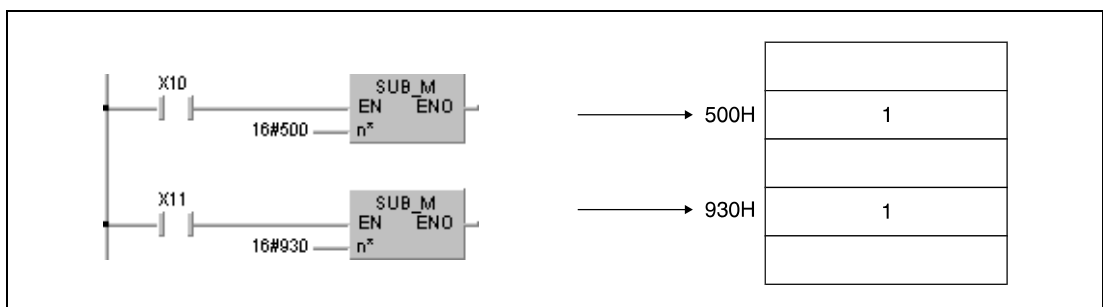
After execution of the microcomputer program the sequence program is processed from the program step on following the SUB/SUBP instruction.

The SUB/SUBP instruction may be programmed in the sequence program of the MAIN and SUB areas.



- 1 Parameter
- 2 Setting value of timer and counter
- 3 Microcomputer program
- 4 Sequence program area
- 5 Microcomputer program area
- 6 MAIN or SUB storage area

Within one microcomputer program area several programs may be created.



- 1 Microcomputer program

**NOTE** *Among the dedicated instructions for AnA, AnAS, and AnU CPUs the SUB instruction determines a 16-bit constant in the instruction block.*

Refer to chapter 10 of this manual for further details on microcomputer programs.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The maximum capacity of the microcomputer program is exceeded.
- The address specified by n exceeds the relevant device range.

**NOTE** *The processing time of a microcomputer program called by one SUB(P) instruction must not exceed 5ms. If it exceeded 5ms, it would conflict with the sequence program and the PLC would not run accurately.*

*If a microcomputer program is to be executed that needs more than 5ms processing time, it has to be split into several blocks that are called consecutively. This method can shorten the processing time of a microcomputer program called by one SUB instruction.*



7.6.10 IX, IXEND

CPU

AnS	AnN	AnA, AnAS	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

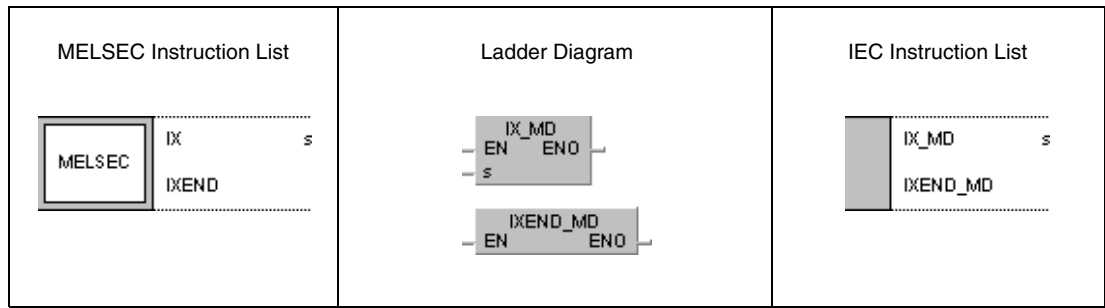
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

Devices  
MELSEC Q

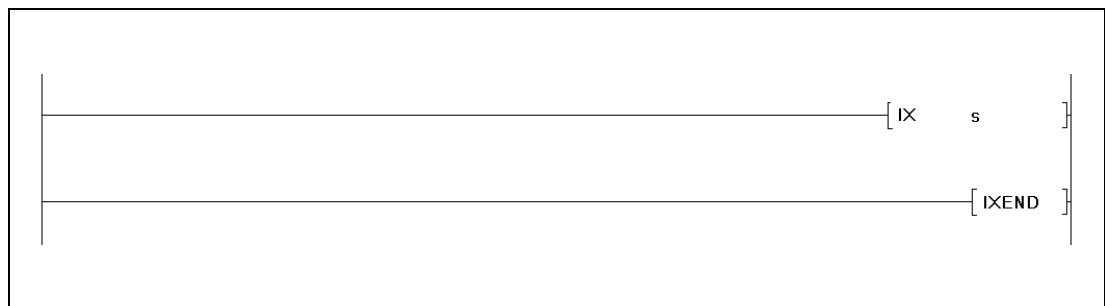
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	2/1 ● <sup>1</sup>

<sup>1</sup> The IX instruction requires two steps; the IXEND instruction requires one step.

GX IEC Developer



GX Developer



Variables

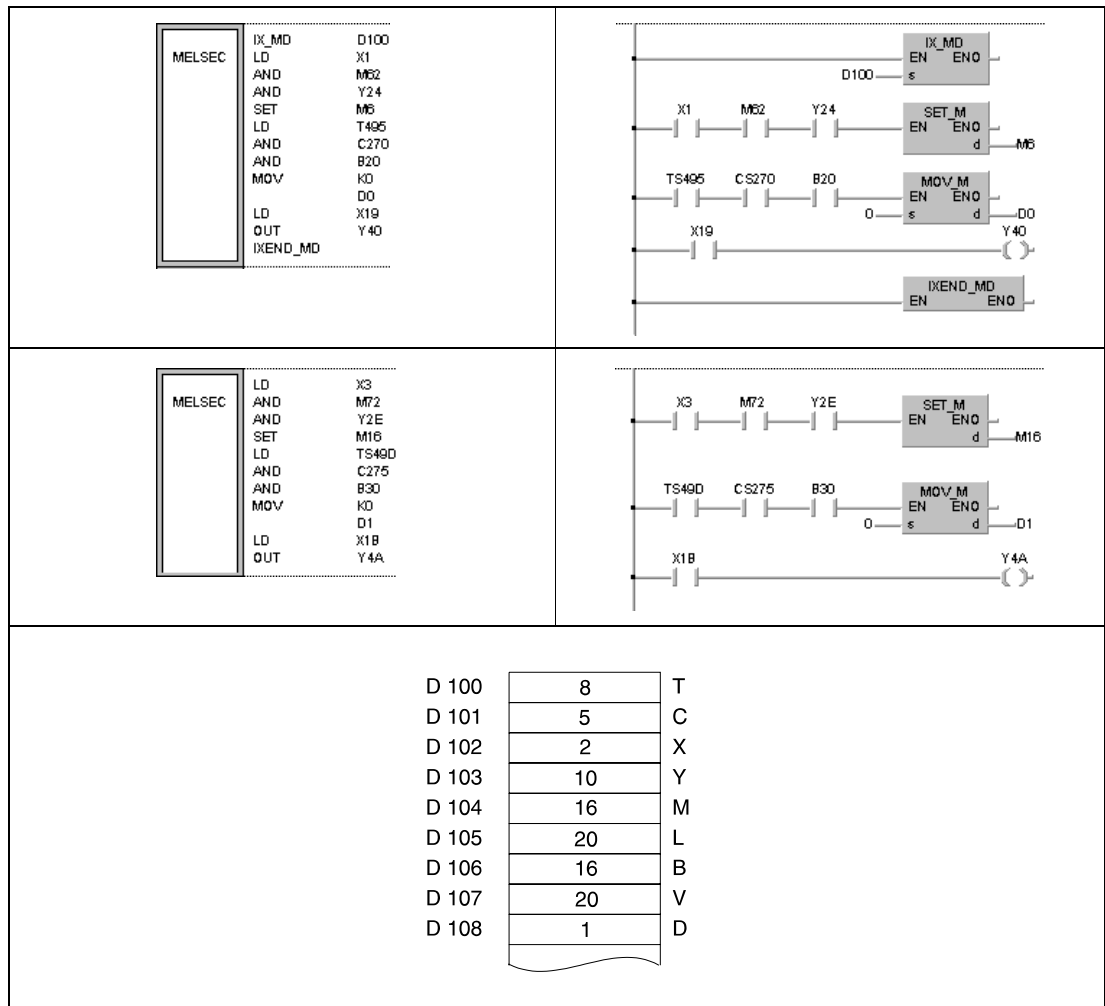
Set Data	Meaning	Data Type
s	First number of device storing data for index qualification.	BIN 16-bit

**Functions** Index qualification of entire program parts

**IX, IXEND** Index qualification instruction

The instructions IX and IXEND are supported only in MELSEC mode in the GX IEC Developer. The IX and IXEND instructions perform index qualification on those devices in the program part located between the IX and IXEND instructions.

On index qualification, decimal values from an index table (s) are added to the device numbers. This new address in hexadecimal format becomes the valid address for further processing. Each device specified in s is assigned a specific type of device, on which the addition is applied. The following diagrams illustrate index qualification:



The value in D100 (8) is added to the timer address TS495. The new address is TS49D.

The value in D101 (5) is added to the counter address CS270. The new address is CS275.

The value in D102 (2) is added to the addresses of the inputs X1 and X19. The new addresses are X3 and X1B.

The value in D103 (10) is added to the addresses of the outputs Y24 and Y40. The new addresses are Y2E and Y4A.

The value in D104 (16) is added to the addresses of the internal relays M6 and M62. The new addresses are M16 and M72.

The value in D106 (16) is added to the address of the link relay B20. The new address is B30.

The value in D108 (1) is added to the register address D0. The new address is D1.

PLS, PLF, and pulsed instructions that are executed once only on set input condition, cannot be addressed by index qualification via the IX/IXEND instruction

In cases where the new address, resulted from the addition exceeds the relevant address range, the instruction cannot be processed accurately.

If the IX and IXEND instructions are executed during a change between program sequences in the online mode (modifying in RUN mode) the instruction cannot be processed neither.

The values added to the addresses of word devices of which each bit can be accessed are stored as binary data. The initial addresses of the devices these values are specified for are stored in s.

In a program, between the IX and the IXEND instruction no index qualification can be performed.

When a program is expanded, the indexed addresses of devices in a program part located between the IX and the IXEND instruction are transformed to addresses using index registers (Zn). The assignment of indexed addresses to the corresponding index registers is shown below:

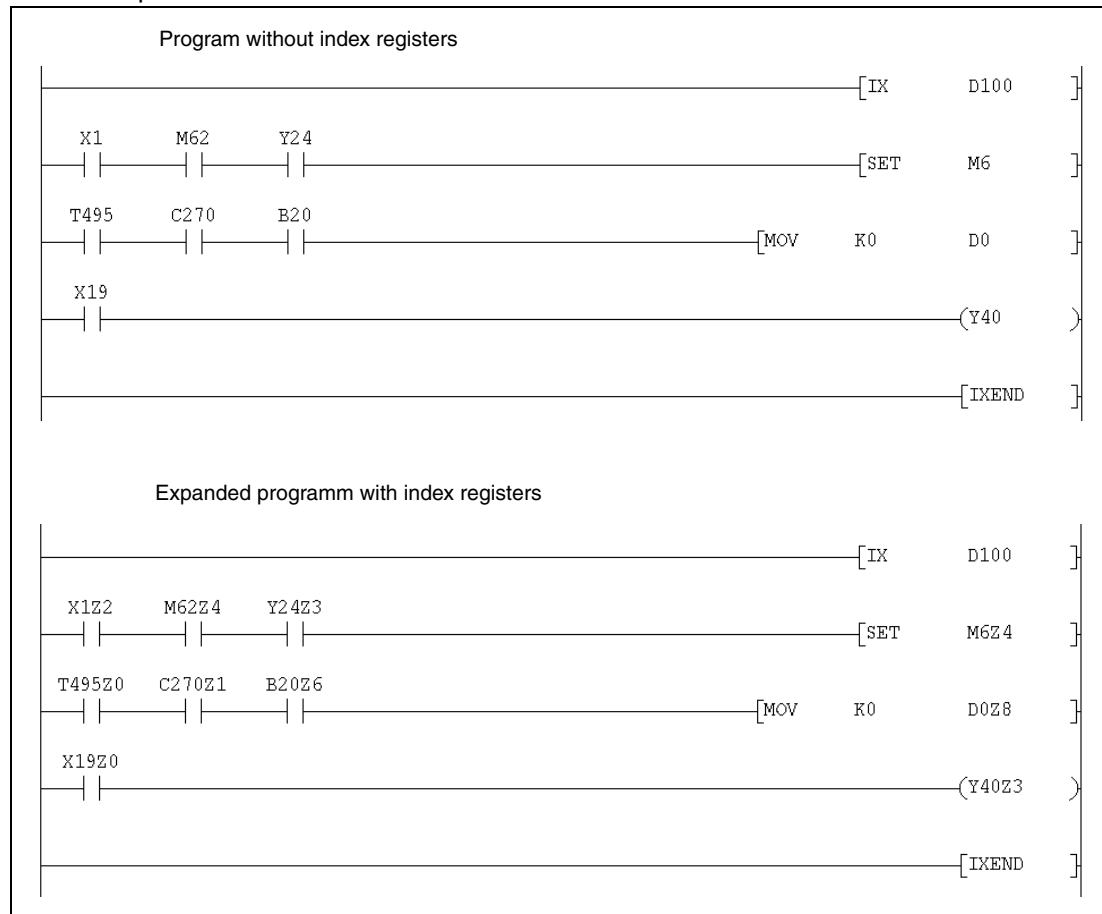
s	Device	Index Register	s	Device	Index Register
s	Qualification value of timer (T)	Z0	s+8	Qualification value of data register (D)	Z8
s+1	Qualification value of counter (C)	Z1	s+9	Qualification value of link register (W)	Z9
s+2	Qualification value of input (X)	Z2	s+10	Qualification value of file register (R)	Z10
s+3	Qualification value of output (Y)	Z3	s+11	Qualification value of buffer register I/O (U)	Z11
s+4	Qualification value of internal relay (M)	Z4	s+12	Qualification value of buffer register (G)	Z12
s+5	Qualification value of latch relay (L)	Z5	s+13	Qualification value of network numbers of link devices with direct access (J)	Z13
s+6	Qualification value of link relay (B)	Z6	s+14	Qualification value of file register (ZR)	Z14
s+7	Qualification value of edge relay (V)	Z7	s+15	Qualification value of pointer (label)	Z15

The Index Registers Z10 to Z15 are not available for the Q00JCPU, Q00CPU, and Q01CPU.

Depending on the programming software used the user has to add the index registers in the sequence program between the IX and the IXEND instructions manually.

**Example**

GX Developer



The index registers used between the IX and the IXEND instructions (Z0 to Z15) do not affect the index registers used by other instructions elsewhere in the program.

**NOTE**

*For index qualification program parts, peripheral devices must be started up in general purpose mode and program expansion must be performed (Q-series only).*

*If peripheral devices are started up by a Q2A, Q2A-S1, Q3A or Q4A CPU and index qualification program parts are created by the IX and IXEND instruction accurate processing is not possible.*

*When using the IX and IXEND instructions in both a normal sequence program and an interrupt sequence program, establish an interlock to avoid simultaneous execution. Disable interrupts between the IX and the IXEND instructions.*

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The IX and IXEND instructions are not programmed in conjunction (error code 4231).
- After execution of the IX instruction an END, FEND, GOEND or STOP instruction is executed before the IXEND instruction is executed (error code 4231).

**Program Example**

**IX, IXEND**

The following program processes the program loop between IX and IXEND for 10 times. With each loop the device numbers programmed within the loop are increased by 1. The table below shows the registers containing the values of the corresponding devices to be added. In addition the changes in the device numbers for the 1st, 2nd, and 10th loop are shown.

```

MELSEC Instruction List

MELSEC  LD      SM400
        FMOV   K0
        D100
        K7
        FOR   K10
        IX    D100
        LD    B0
        OR    Y30
        ANI  X30
        OUT  Y30
        SET  M0
        +    D0
        D10
        LD    T3
        AND  C3
        MOV  K1
        D40
        IXEND
        LD    SM400
        BK+  D100
        K1
        D100
        K7
        NEXT
    
```

D	Device	Device number change / loop			
		1.	2.	3.	10.
D100	Qualification value of timer (T)	T3	T4	T5	TC
D101	Qualification value of counter (C)	C4	C5	C6	CD
D102	Qualification value of input (X)	X10	X11	X12	X19
D103	Qualification value of output (Y)	Y30	Y31	Y32	Y39
D104	Qualification value of internal relay (M)	M0	M1	M2	M9
D106	Qualification value of link relay (B)	B0	B1	B2	B9
D108	Qualification values of data registers (D)	D0	D1	D2	D9
		D10	D11	D12	D19
		D40	D41	D42	D49

7.6.11 IXDEV, IXSET

CPU

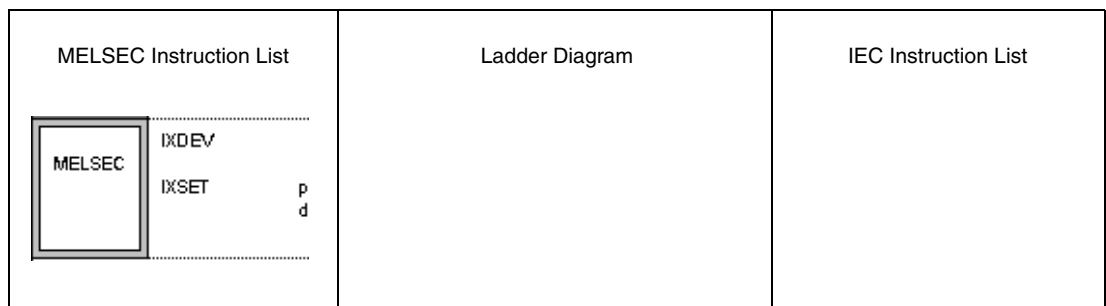
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

Devices  
MELSEC Q

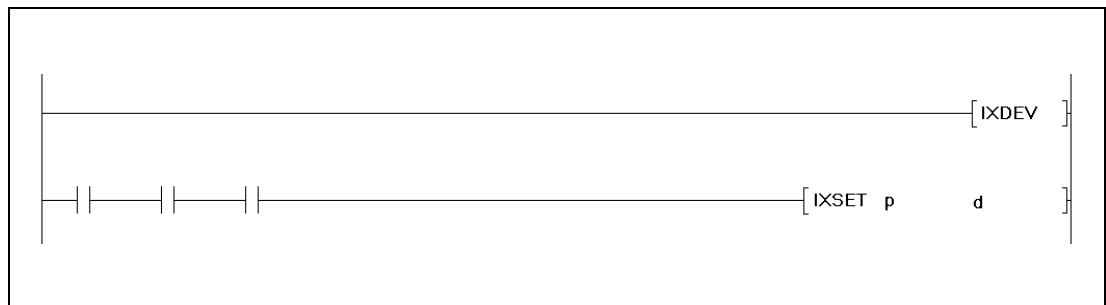
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other P
	Bit	Word		Bit	Word						
p	—	—	—	—	—	—	—	—	●	SM0 1/3 ● <sup>1</sup>	
d	—	●	●	—	—	—	—	—	—		

<sup>1</sup> The IXDEV instruction requires one step; the IXSET instruction requires three steps.

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
p	First number of device (pointer/label only) storing data for index qualification.	Pointer/label
d	First number of device storing indexed addresses of devices.	BIN 16-bit

**Functions**     **Storing indexed device numbers in an index qualification table****IXDEV/IXSET Instruction for writing to an index table**

The instructions IXDEV and IXSET are supported in the GX Developer or in MELSEC mode in the GX IEC Developer only.

The IXDEV and IXSET instructions read the addresses of the devices in the offset designation area and write these offset numbers to an index table in the device designated by d.

Refer to the instructions IX and IXEND for the assignment of device types to their corresponding registers.

If a device type is not assigned in the offset designation the value 0 is stored in the index table.

The single bits of word devices are processed as dummy contact, i.e. only the address of a single bit can be read and written to the index table. In order to address the dummy the corresponding bit is specified. Bit 0 (b0) in data register D0 is addressed D0.0. For bit designation in a 16-bit data word the hexadecimal values 0 through F are used.

Reading in the offset values applies as follows:

- Reading in the devices: T□, C□, X□, Y□, M□, L□, V□, B□  
The offset value indicated □ is read in and written to the corresponding registers.
- Reading the devices: D□.XX, W□.XX, R□.XX<sup>1</sup>, U□\G□.XX<sup>1</sup>, ZR□.XX<sup>1</sup>  
The offset value indicated □ is read in and written to the corresponding registers.  
The value indicated XX serves as variable for the bit designation.  
<sup>1</sup> Not possible for Q00JCPU, Q00CPU, and Q01CPU
- Reading in the devices: J□/B□<sup>1</sup>, J□/W□<sup>1</sup>, J□/X□<sup>1</sup>, J□/Y□<sup>1</sup>  
The offset value indicated □ is read in and written to the corresponding registers.  
If no offset value is to be written for the device following J□/, this value is to be set to 0.  
<sup>1</sup> Not possible for Q00JCPU, Q00CPU, and Q01CPU
- On programming the IXSET instruction the offset value of the device P□ is designated directly via address (pointer/label).

If in the offset designation area two identical device types are specified, the offset value of the latter device is valid.

The IXDEV and IXSET instructions have to be programmed in conjunction.

The offset value of the device ZR□.XX may range from 0 to 32767. The offset value is the remainder of the quotient of the device number divided by 32767, and is written to the corresponding register.

For the dummy contacts in the offset designation area only LD and AND instructions are valid. All other instructions are ignored.

**Operation Errors**

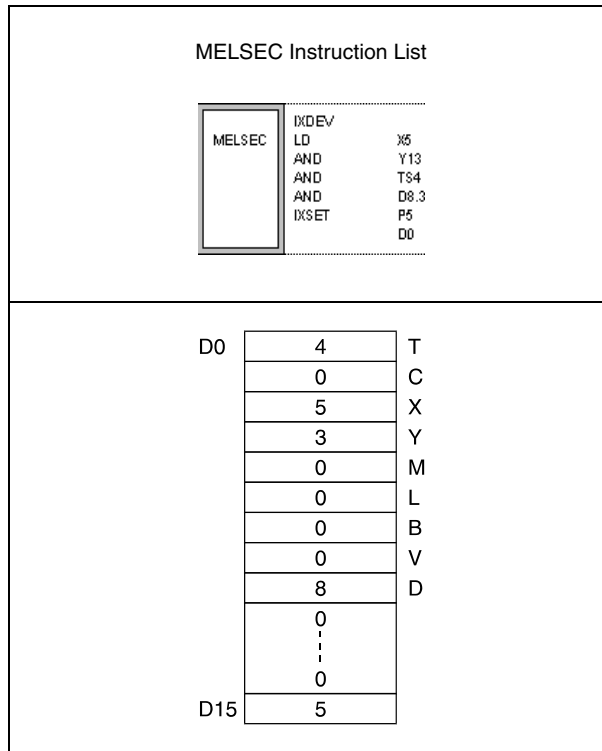
In the following cases an operation error occurs and the error flag is set:

- The IXDEV and IXSET instructions are not programmed in conjunction (error code 4231).

**Program Example**

**IXDEV, IXSET**

The following program writes the addresses (offset values) of the dummy contacts in the offset designation area to the corresponding register. The offset value of the pointer/label is specified by the IXSET instruction. Refer to the instructions IX and IXEND for the assignment of device types to their corresponding registers.





## 7.7 Data table operation instructions

The operation instructions for data tables write and read data to and from a data table. Current data are written to the table and read out in a different order for further processing. In addition, these instructions enable deleting and inserting specific data blocks.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Write data to a data table	FIFW	FIFW_M
	FIFWP	FIFWP_M
Read data entered first from data table	FIFR	FIFR_M
	FIFRP	FIFRP_M
Read data entered last from data table	FPOP	FPOP_M
	FPOPP	FPOPP_M
Delete specified data blocks from data table	FDEL	FDEL_M
	FDELP	FDELP_M
Insert specified data blocks in data table	FINS	FINS_M
	FINSP	FINSP_M

### 7.7.1 FIFW, FIFWP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Usable Devices																		Digit designation K1 ↓ K4	Number of steps 7	Index	Carry Flag M9012	Error Flag M9010 M9011				
	Bit Devices								Word Devices (16-bit)								Constant							Pointer		Level	
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I	N						
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				K1 ↓ K4	7				
d								●	●	●	●																●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	SM0	3
d	—	●	●	—	—	—	—	—	—		

GX IEC  
Developer

<p>MELSEC Instruction List</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p>
--------------------------------	-----------------------	-----------------------------

GX  
Developer

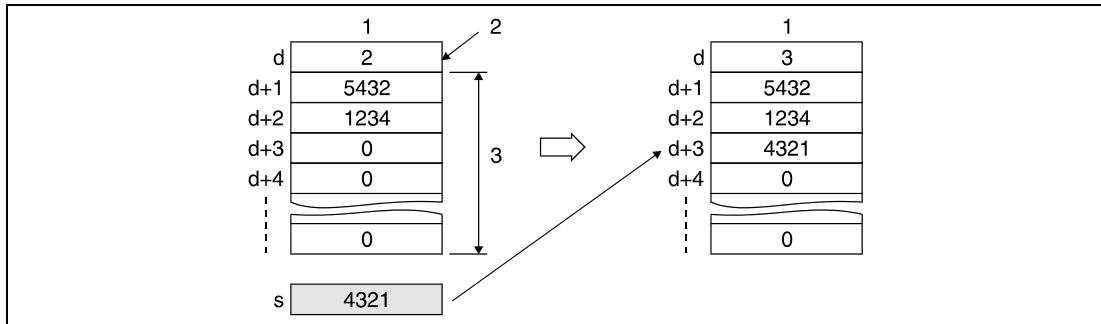


Variables

Set Data	Meaning	Data Type
s	Data to be written to the data table or devices storing such data.	BIN 16-bit
d	First number of data table.	

**Functions**     **Writing data to a data table****FIFW**     **Instruction for data entry**

The FIFW instruction writes data in a sequence specified by *s* to a data table. This table is specified by the address range in *d* and conducts data in the sequence of their entry. In the first address of the data range in *d* the total number of data records contained in the table is stored. Therefore, the value at this address is the position pointer for data to be recorded in the table. On each execution of the FIFW instruction this value is increased by 1. Thus, following data are recorded from the address *d*+1.



<sup>1</sup> Data table

<sup>2</sup> Position pointer

<sup>3</sup> Data table range

Prior to the first FIFW instruction the contents of the device specified in *d* have to be cleared.

The number of data records to be recorded and the address range of the data table have to be controlled on programming by the user.

For management of several data records in different data tables an application program should be used.

**Operation Errors**

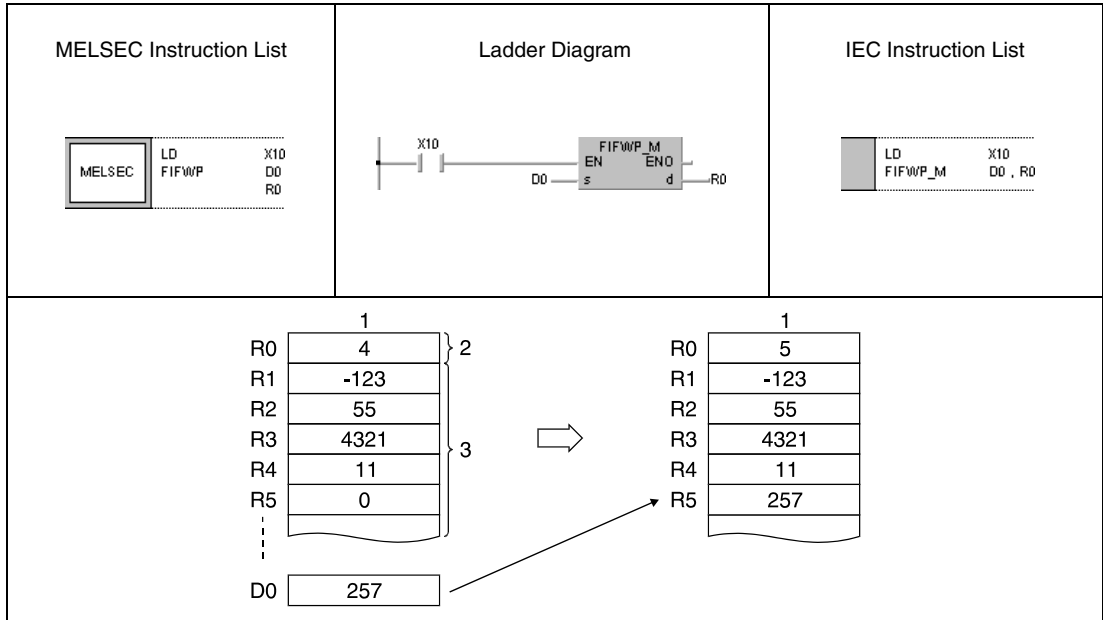
In the following case an operation error occurs and the error flag is set:

- The data table range of the FIFO table exceeds the relevant storage device range when executing the FIFW instruction (Q series and System Q = error code 4101)

**Program Example 1**

FIFWP

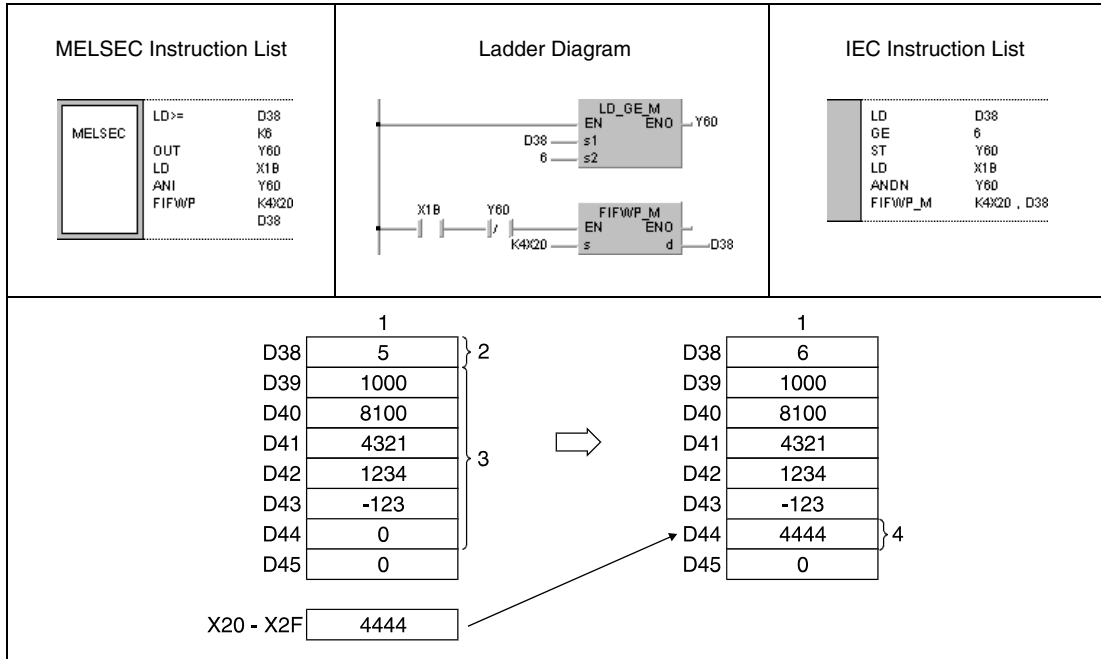
The following program specifies the storage range of the data table via the data registers R0 through R5. The initial address of the storage range (R0) contains the position pointer, indicating the number of stored data records. With leading edge from X10, data in D0 are stored at the next available storage position of the data table (R5).



- <sup>1</sup> Data table
- <sup>2</sup> Position pointer
- <sup>3</sup> Data table range

**Program Example 2** FIFWP

The following program specifies the storage range of the data table via the data registers D38 through D44. The initial address of the storage range (D38) contains the position pointer, indicating the number of stored data records. With leading edge from X1B, data at the inputs X20 through X2F are stored at the next available storage position of the data table (D44). The data table specified here stores at maximum 6 data records. Therefore, Y60 is programmed as a limiter of the FIFW instruction. The output is set, if the contents of D38 are greater than or equal to 6.



- 1 Data table
- 2 Position pointer
- 3 Data table range
- 4 Highest available storage address

7.7.2 FIFR, FIFRP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

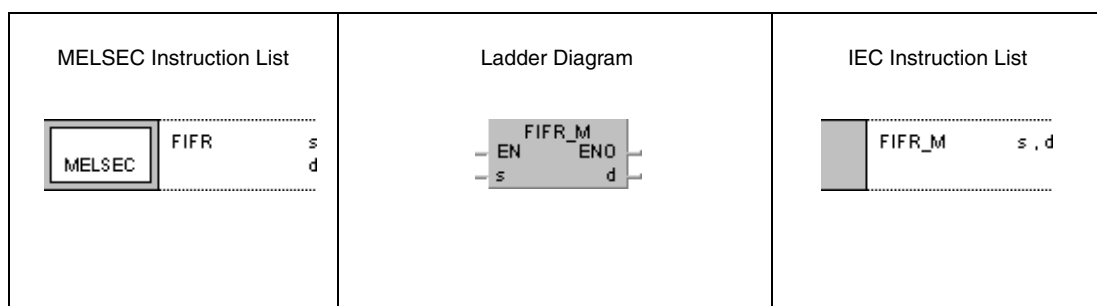
	Usable Devices																				Digit designation K1 ↓ K4	Number of steps 7 <sup>1</sup>	Index ●	Carry Flag M9012	Error Flag M9010 M9011			
	Bit Devices							Word Devices (16-bit)							Constant		Pointer		Level									
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I				N				
s	●	●	●	●	●	●	●	●	●	●	●	●	●	●														
d							●	●	●	●	●																●	

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

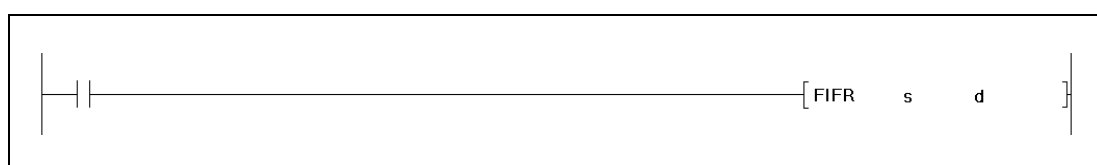
Devices  
MELSEC Q

	Usable Devices								Error Flag SM0	Number of steps 3	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	—	
d	—	●	●	—	—	—	—	—	—	—	

GX IEC  
Developer



GX  
Developer



Variables

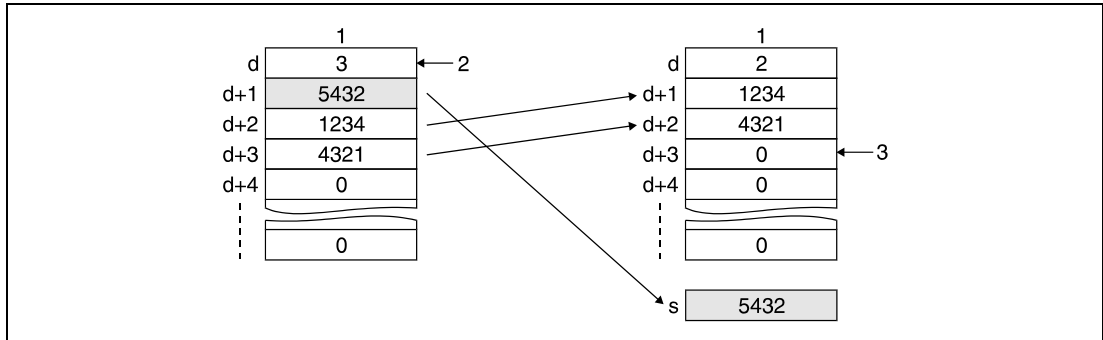
Set Data	Meaning	Data Type
s	First number of device storing read out data.	BIN 16-bit
d	First number of data table.	

**Functions**      **Reading data entered first from a data table**

**FIFR      Instruction for reading data entered first**

The FIFR instruction reads data from a data table and stores them in a specified storage range. Reading the data begins with the first address d+1 after the position pointer. The data is transferred to the storage range specified by s.

The data in the data table are moved successively to the beginning of the table in order of their entry. All preceding data are cleared. After reading out, the value of the position pointer (first address in d) is decreased by 1.



- <sup>1</sup> Data table
- <sup>2</sup> Position pointer
- <sup>3</sup> This register is reset to 0

**NOTE**      *Make sure this instruction is not executed, while d (position pointer) contains the value 0.*

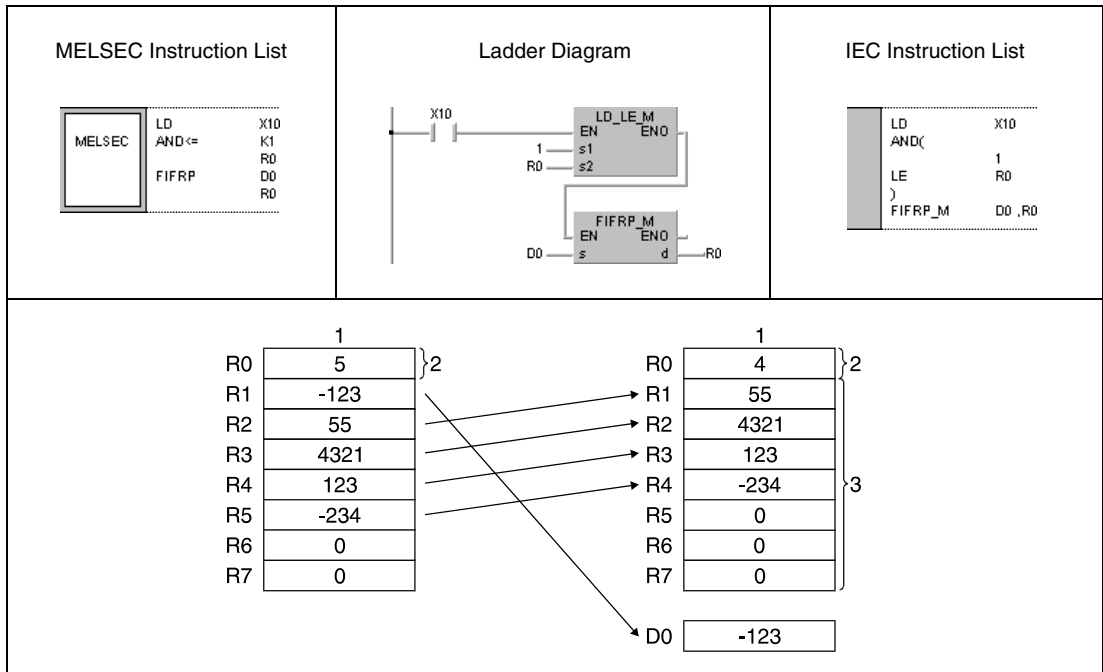
**Operation Errors**      In the following cases an operation error occurs and the error flag is set:

- An FIFR instruction is executed while the position pointer contains the value 0 (Q series and System Q = error code 4100)
- The device table range exceeds the corresponding device range when executing the FIFR instruction (Q series and System Q = error code 4101)

**Program Example 1**

FIFRP

With leading edge from X10, the following program reads the data value in R1 (first entered value) of the data table from R0 through R7 and stores the value in the register D0. At the beginning the value of the position pointer is 5 and after the execution 4. The preceding comparison operation avoids the execution of the FIFR instruction, if the position pointer (R0) contains the value 0.

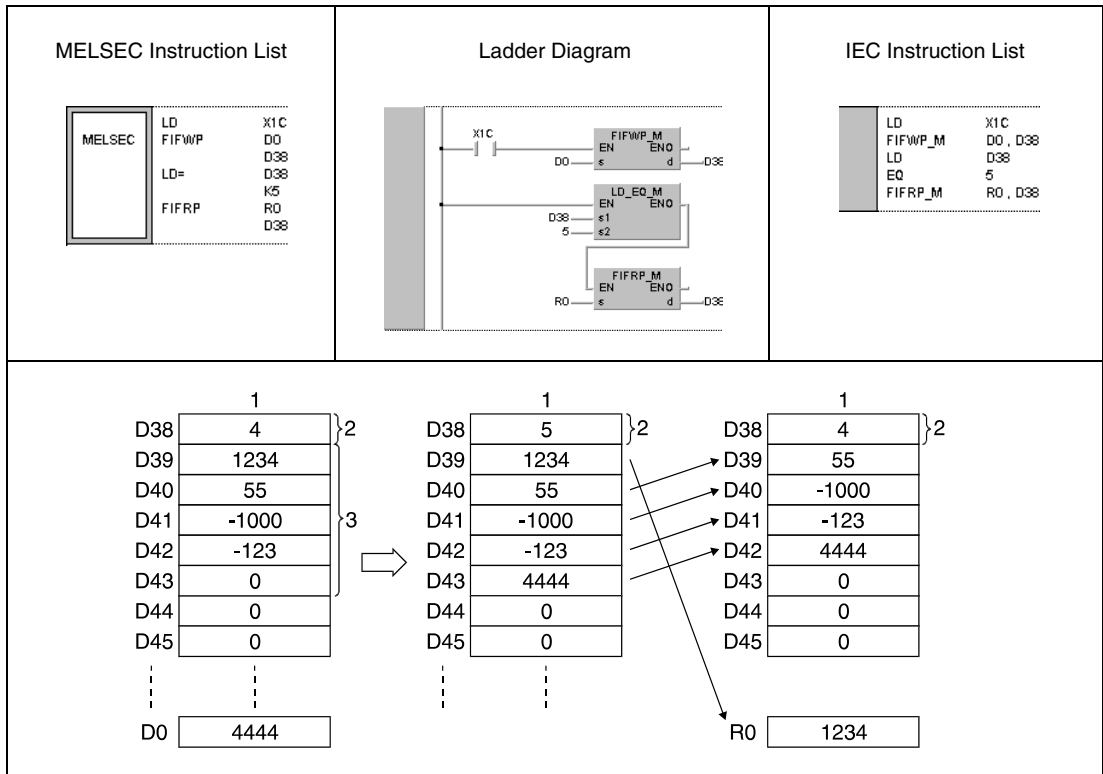


- <sup>1</sup> Data table
- <sup>2</sup> Position pointer
- <sup>3</sup> Data table range



**Program Example 2** FIFRP

With leading edge from X1C, the following program writes a value from D0 to the data table from D38 through D43. If the value of the position pointer is 5, the first value of the FIFO table is read and passed on to R0. This process is repeated with every leading edge from X1C.



- <sup>1</sup> Data table
- <sup>2</sup> Position pointer
- <sup>3</sup> Data table range

7.7.3 FPOP, FPOPP

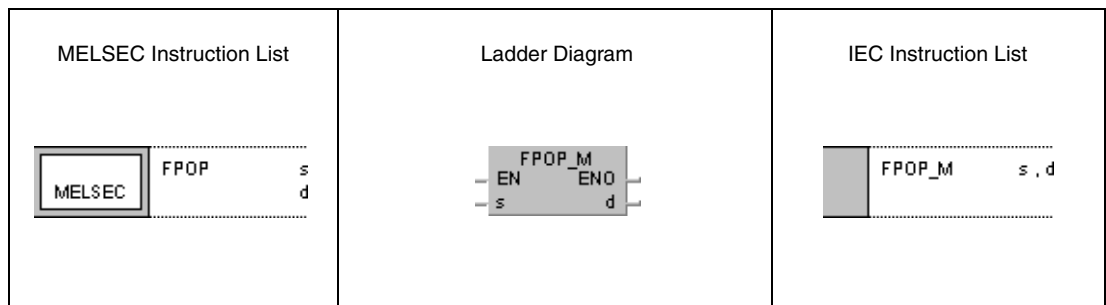
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

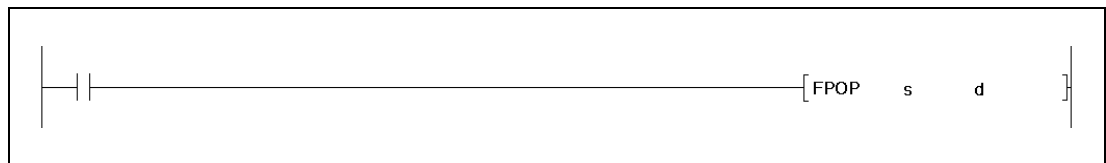
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	SM0	3	
d	—	●	●	—	—	—	—	—			

GX IEC  
Developer



GX  
Developer



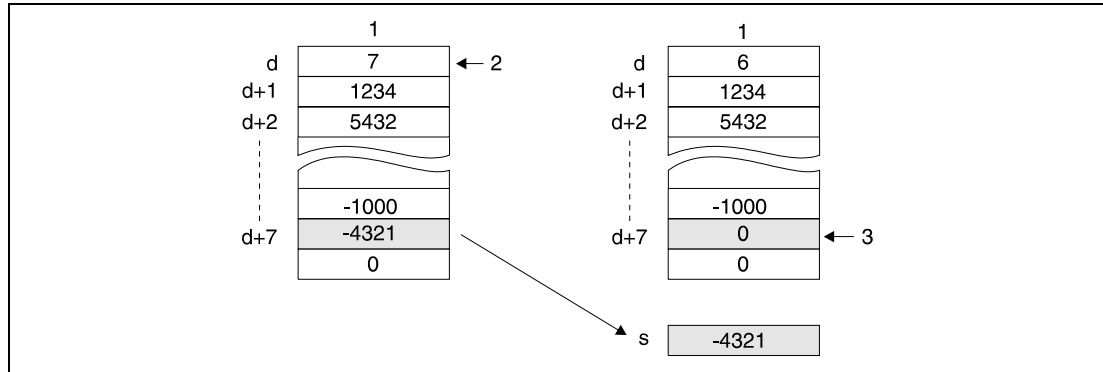
Variables

Set Data	Meaning	Data Type
s	First number of device storing read data.	BIN 16-bit
d	First number of data table.	

**Functions**    **Reading data entered last from a data table****FPOP**    **Instruction for reading data entered last**

The FPOP instruction reads data from a data table and stores them in a specified storage range. Reading the data begins with the last address  $d+n$  in the data table. The data is transferred to the storage range specified by  $s$ .

The read address in the data table is reset to 0. After reading out, the value of the position pointer (first address in  $d$ ) is decreased by 1.



<sup>1</sup> Data table

<sup>2</sup> Position pointer

<sup>3</sup> This register is reset to 0

**NOTE**    *Make sure this instruction is not executed, while  $d$  (position pointer) contains the value 0.*

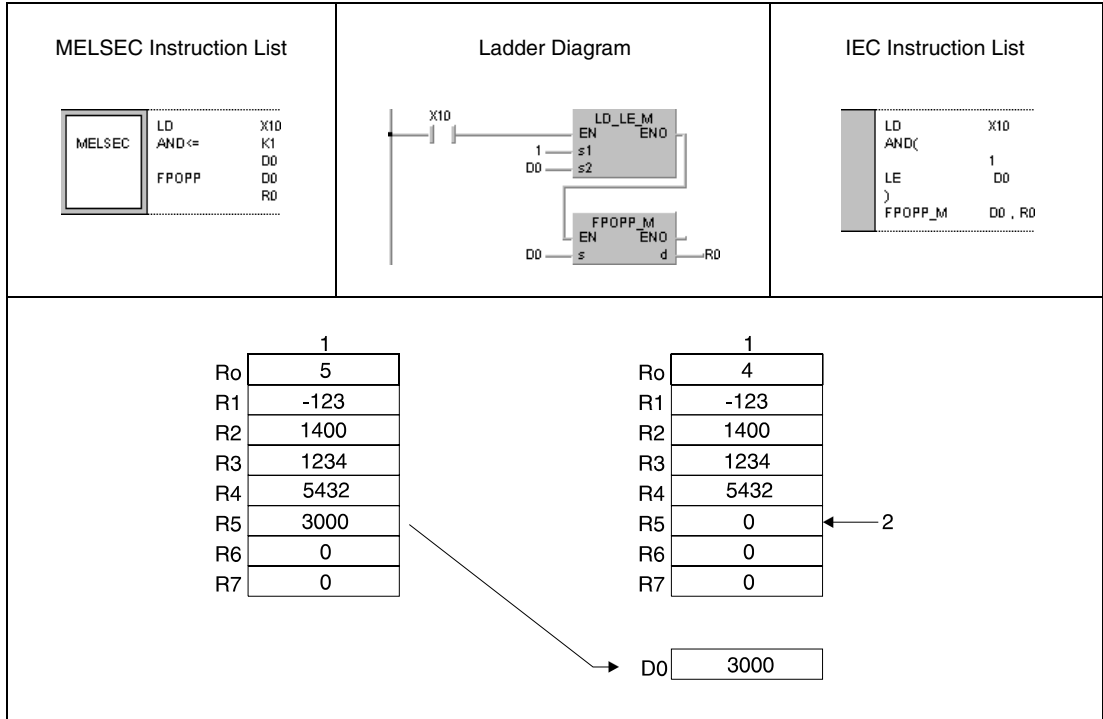
**Operation Errors**    In the following cases an operation error occurs and the error flag is set:

- An FPOP instruction is executed while the position pointer contains the value 0 (error code 4100)
- The data table range exceeds the corresponding device range when executing the FPOP instruction (error code 4101).

**Program Example 1**

**FPOPP**

With leading edge from X10, the following program reads the data value in R5 (value entered last) of the data table from R0 through R7 and stores the value in the register D0. At the beginning the value of the position pointer is 5 and after the execution 4. The preceding comparison operation avoids the execution of the FPOPP instruction, if the position pointer (R0) contains the value 0.

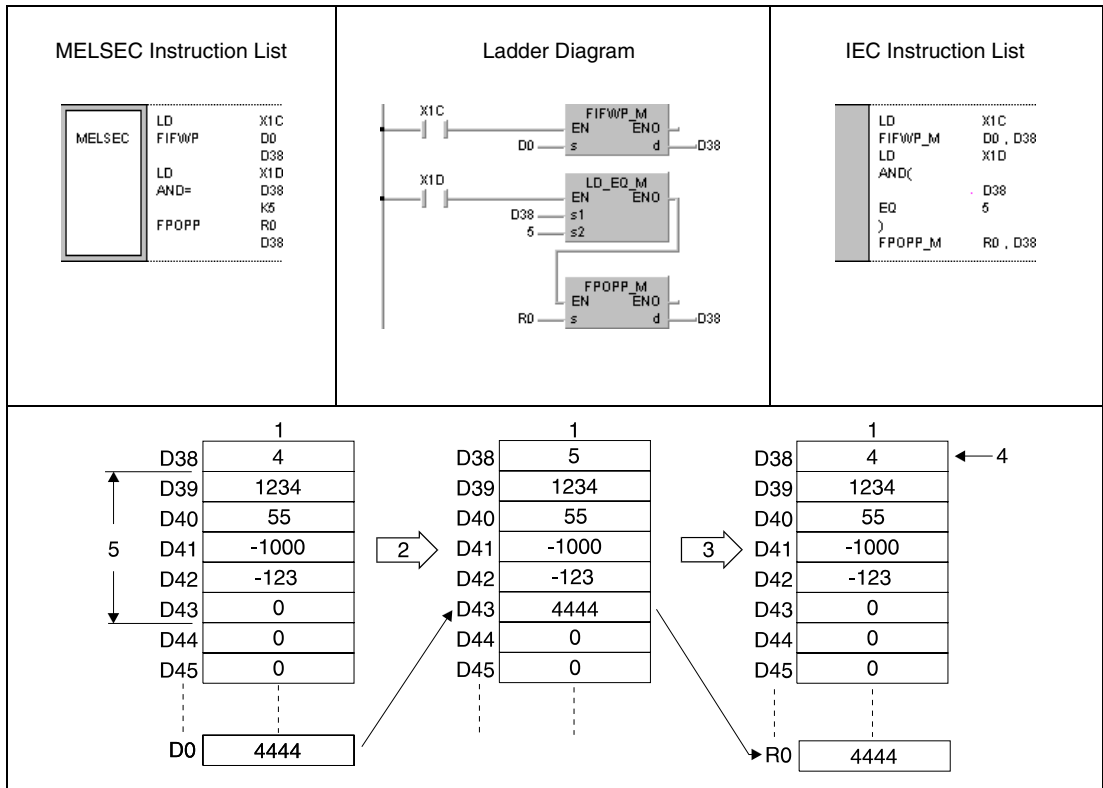


<sup>1</sup> Data table

<sup>2</sup> This register is reset to 0

**Program Example 2** FPOPP

With leading edge from X1C, the following program writes a value from D0 to the data table from D38 through D43. If the value of the position pointer is 5, with leading edge from X1D the value in register D43 is read and passed on to R0.



- <sup>1</sup> Data table
- <sup>2</sup> Leading edge from X1C
- <sup>3</sup> Leading edge from X1D
- <sup>4</sup> Position pointer
- <sup>5</sup> Current address range of data table

7.7.4 FDEL, FDELP, FINS, FINSP

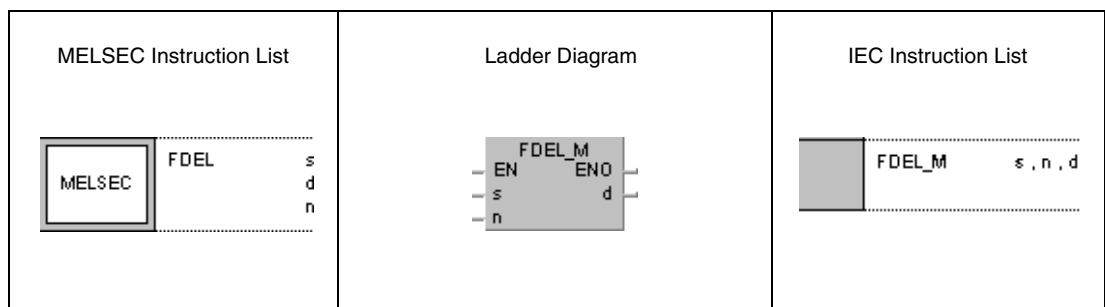
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

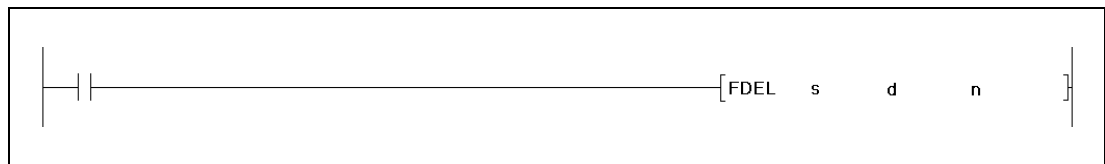
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC Developer



GX Developer



Variables

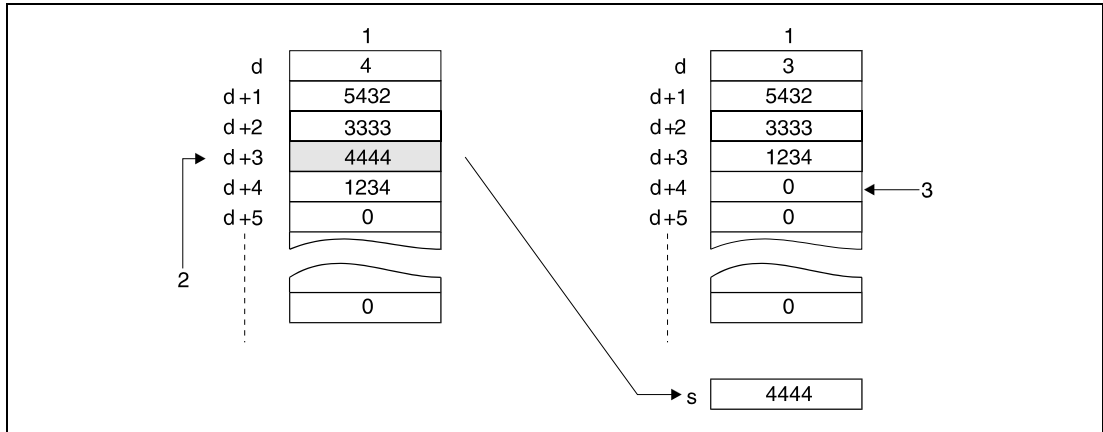
Set Data	Meaning	Data Type
s	Data to be inserted into the data table at a specified address or device storing such data. First number of device storing data to be deleted from a data table at a specified address.	BIN 16-bit
d	First number of data table.	
n	Number of address where data is to be inserted or deleted.	

**Functions Deleting and inserting specified data blocks in a data table**

**FDEL Deleting specified data blocks**

The FDEL instruction deletes the nth data block after the position pointer from a data table specified by d and stores this value in a device specified in s.

The data in the data table are shifted together after deletion of one data block. After reading, the value of the position pointer (first address in d) is decreased by 1.

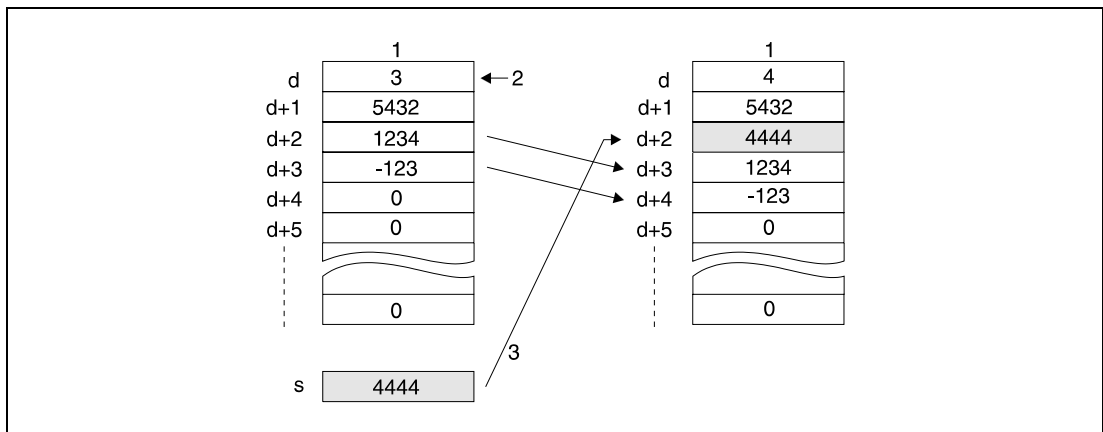


- <sup>1</sup> Data table
- <sup>2</sup> For n=3 the data block d+3 is deleted.
- <sup>3</sup> This register is reset to 0

**FINS/FINSP Inserting specified data blocks**

The FINS instruction inserts a 16-bit data block specified by s at the nth position after the position pointer into the data table specified by d.

The data blocks following the inserting position are shifted on by one address. After inserting, the value of the position pointer (first address in d) is increased by 1.



- <sup>1</sup> Data table
- <sup>2</sup> Position pointer
- <sup>3</sup> For n=2 the data block is inserted at d+2

## Operation Errors

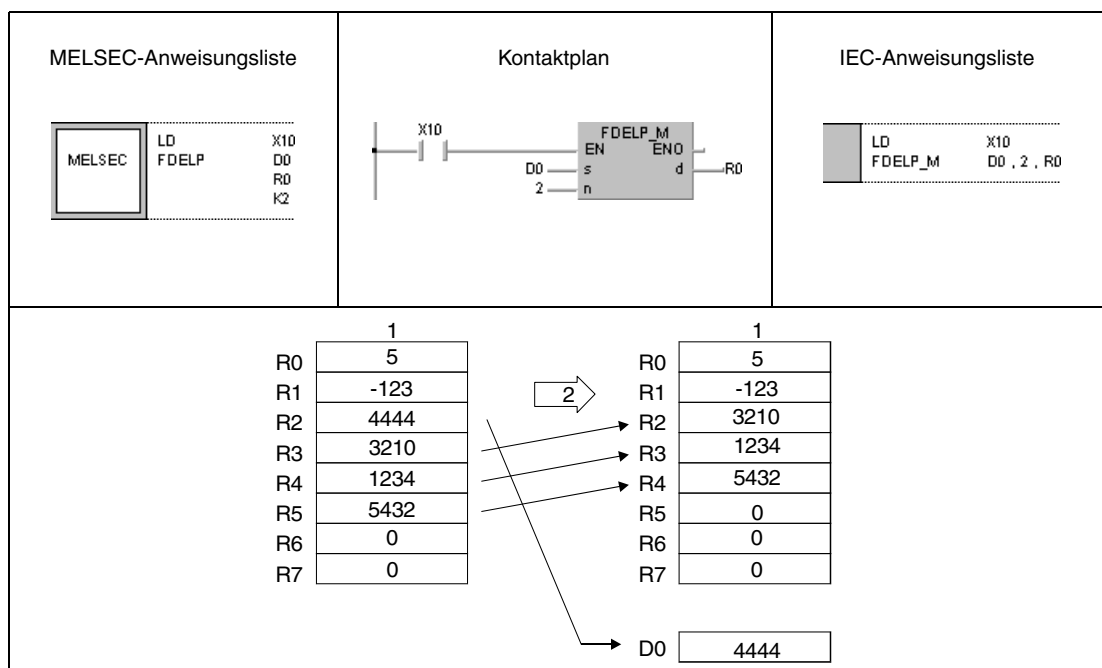
In the following cases an operation error occurs and the error flag is set:

- The inserting position in d specified by n via the FINS instruction exceeds the address range of existing data blocks plus 1 (error code 4101).
- The value of n exceeds the device range of the table d (error code 4101).
- The FDEL or FINS instruction was executed when n = 0 (error code 4100).
- The FDEL was executed when the value of d was 0 (error code 4100)
- The data table range exceeds the corresponding device range when the FDEL or FINS instruction is executed (error code 4100).

## Program Example 1

### FDELP

When X10 goes ON, the data from the 2nd position (R2) of the data table ranging from R0 to R7 will be deleted and the data stored in D0.



<sup>1</sup> Data table

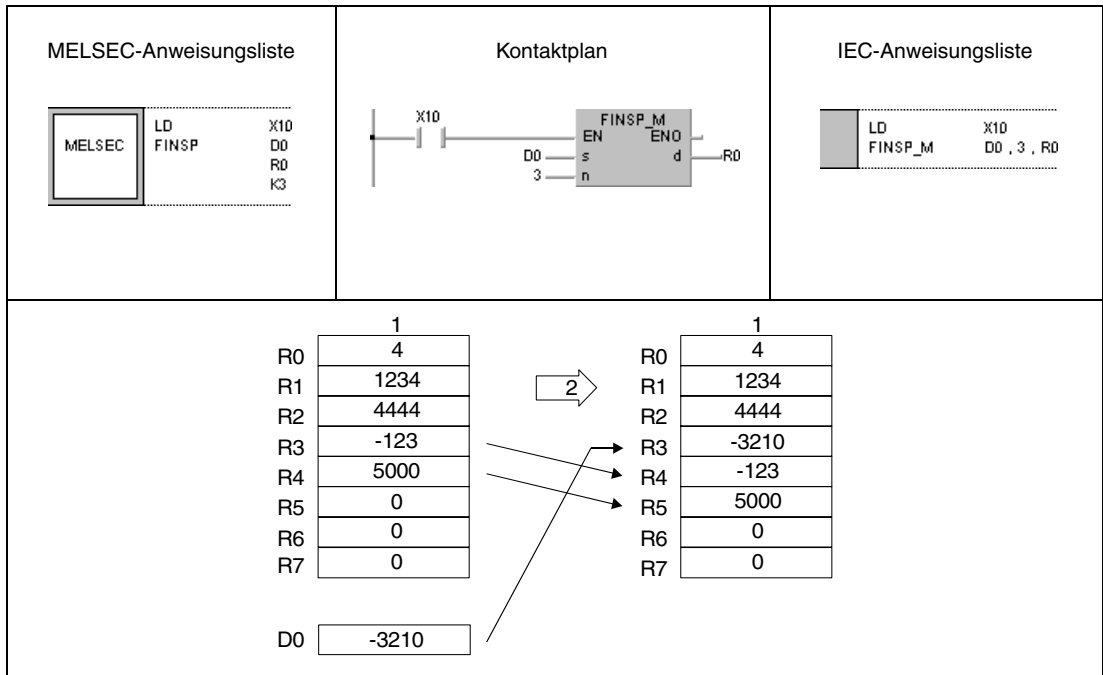
<sup>2</sup> Leading edge of X10



**Program** FINSP

**Example 2**

The following program inserts the data at D0 at the 3rd position of the data table ranging from R0 to R7 when X10 goes ON.





## 7.8 Buffer Memory Access Instructions

The following instructions access the buffer memory of special function modules. These instructions enable the CPU to exchange data with the according modules. The following table gives an overview of the instructions:

Function	MELSEC instruction in MELSEC Editor	MELSEC instruction in IEC Editor
Reading data from a special function module	FROM	FROM_M
	FROMP	FROMP_M
	DFRO	DFRO_M
	DFROP	DFROP_M
Writing data to a special function module	TO	TO_M
	TOP	TOP_M
	DTO	DTO_M
	DTOP	DTOP_M

7.8.1 FROM, DFRO

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag	
Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level						
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I
n1																●	●					
n2																						
d	●	●	●	●	●	●	●	●	●	●	●					●	●					●
n3																●	●					

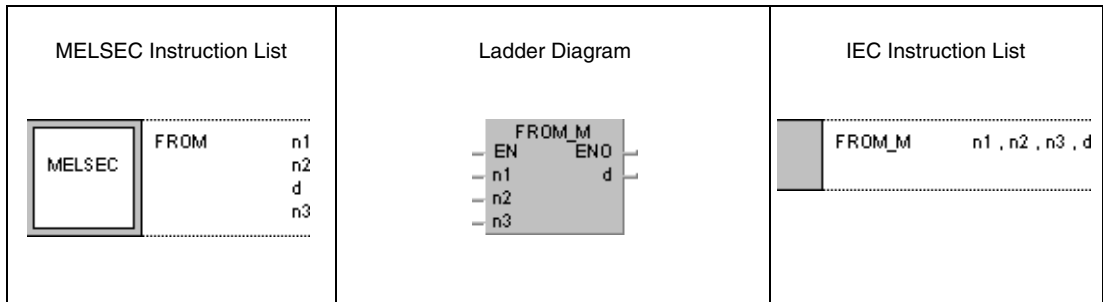
<sup>1</sup> The digit designation can be specified K1 to K4 via a FROMP instruction and K1 to K8 via a DFROP instruction.

<sup>2</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

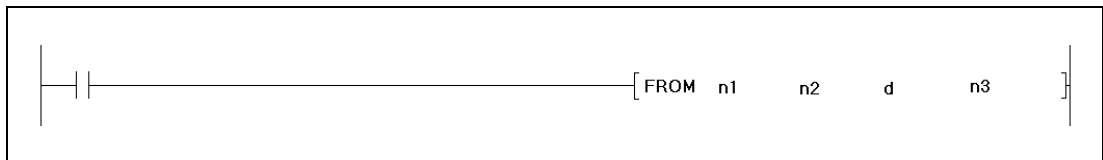
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word				U			
n1	●	●	●	●	●	●	●	●	●	SM0	5
n2	●	●	●	●	●	●	●	—	—		
d	●	●	●	—	—	—	—	—	—		
n3	●	●	●	●	●	●	●	●	—		

GX IEC  
Developer



GX  
Developer



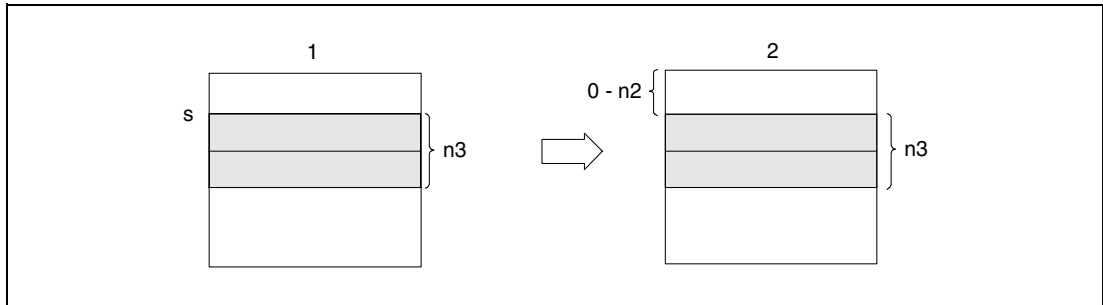
Variables

Set Data	Meaning	Data Type
n1	Head address of special function module on base unit.	BIN 16-bit
n2	First number of memory address area for data to be read.	BIN 16-/ 32-bit
d	First number of memory address area of the CPU to be written to.	BIN 16-bit
n3	Number of data words to be read.	

**Functions Reading 1-word and 2-word data from a special function module**

**FROM Reading 1-word data (16-bit)**

The FROM instruction reads 1-word data from the buffer memory of a special function module and stores it in a specified memory address area of the CPU. The first address of data to be read is specified by n2, the number of data words is specified by n3, and the head address of the special function module, resulting from the position of the module on the base unit is specified by n1. The memory address area of the CPU storing the data is specified by d.

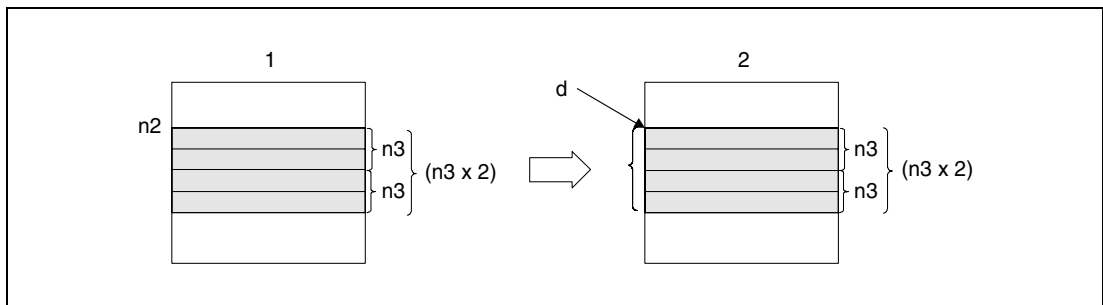


<sup>1</sup> Buffer memory of special function module  
<sup>2</sup> Memory of the CPU

**NOTE** *The FROM instruction can also be used to read data from shared memory of another station in a multi CPU system. Refer to chapter 9.6.2 for more details.*

**DFRO Reading 2-word data (32-bit)**

The DFRO instruction reads 2-word data from the buffer memory of a special function module. The first address of data to be read is specified by n2, the number of data words (2-multiple) is specified by n3, and the head address of the special function module is specified by n1. The memory address area of the CPU storing the data is specified by d.



<sup>1</sup> Buffer memory of special function module  
<sup>2</sup> Memory of the CPU

**NOTE** *A OnA or a System Q CPU can also access the buffer memory of special function modules directly. In this case the devices are specified as U□\G□ (U(Headaddress of the special function module)/G(Buffer memory adress)).*

**Operation Errors**

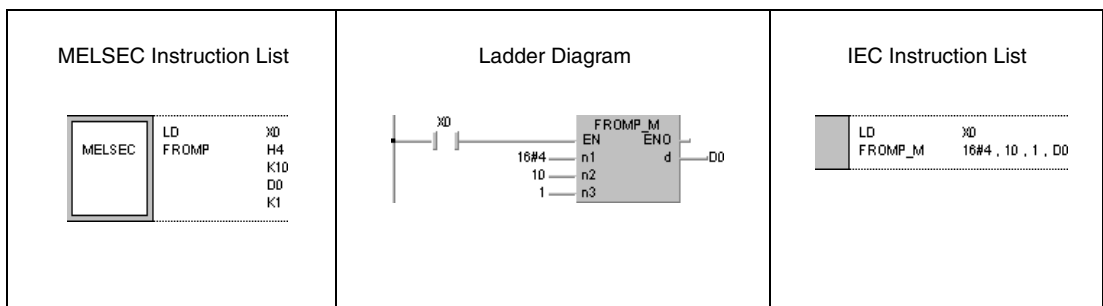
In the following cases an operation error occurs and the error flag is set:

- No signals have been exchanged with the special function module prior to the execution of the instruction (error code 1412).
- An error has occurred in the special function module prior to the execution of the instruction (error code 1402).
- The I/O number specified by n1 is not a special function module (Q series and System Q = error code 2110)
- The number of data words specified in n3 exceeds the storage range of the device specified by d (Q series and System Q = error code 4101).
- The address specified by n2 exceeds the buffer memory range (Q series and System Q = error code 4100)
- The address specified by n2 is inaccurate (AJ71QC24) (Q series and System Q = error code 4100).
- A special function module cannot be accessed.

**Program Example 1**

**FROMP**

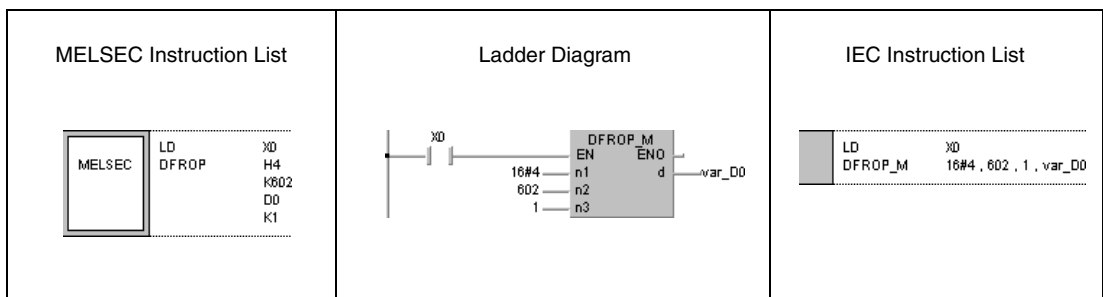
With leading edge from X0, the following program reads the digital values of channel CH1 from address 10 of the buffer memory of an A68AD module. The memory address area of the module is 040 through 05F. The read data is stored in D0.



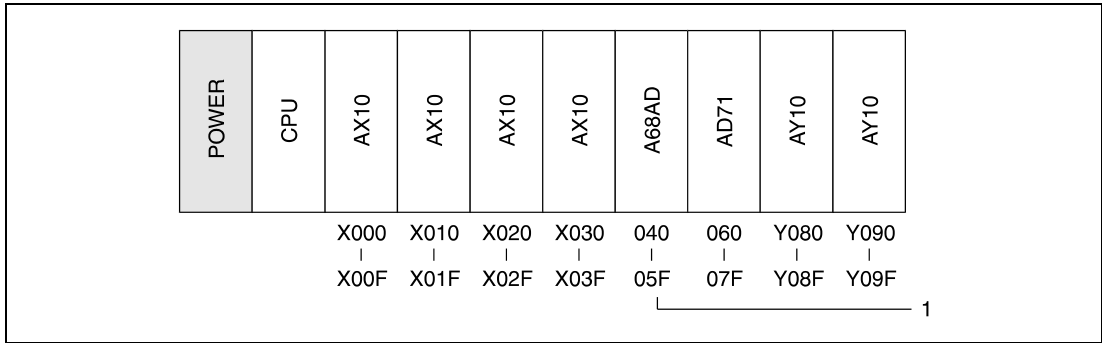
**Program Example 2**

**DFROP**

With leading edge from X0, the following program reads the x-axis data at the addresses 602 and 603 in the buffer memory of an AD71 module. The memory address area of the module is 040 through 05F. The read data is stored in D0 and D1.



**NOTE**      *The head address in n1 has to be specified as follows:*  
*n1 = 10 → head address = 1*  
*n1 = 20 → head address = 2*



<sup>1</sup> Head address of special register: n1 = K4 or H4

*The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.8.2 TO, DTO, DTO, DTO

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag		Error Flag					
	Bit Devices						Word Devices (16-bit)						Constant		Pointer					Level		M9012	M9010 M9011				
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)			P	I	N	
n1																	●	●					K1 ↓ K4  ● <sup>2</sup>  K1 ↓ K8	9/11  ● <sup>3</sup>	●		
n2																●	●										
s	●	●	●	●	●	●	●	●	●	●	●					● <sup>1</sup>	● <sup>1</sup>										
n3																	●	●									

<sup>1</sup> The designation range of constant s is: H0 through FFFF, K-32768 through 32767.

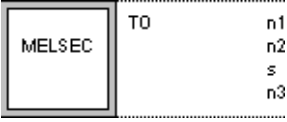
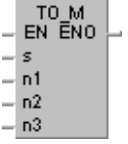

<sup>2</sup> The digit designation can be specified K1 to K4 via a TO(P) instruction and K1 to K8 via a DTO(P) instruction.

<sup>3</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

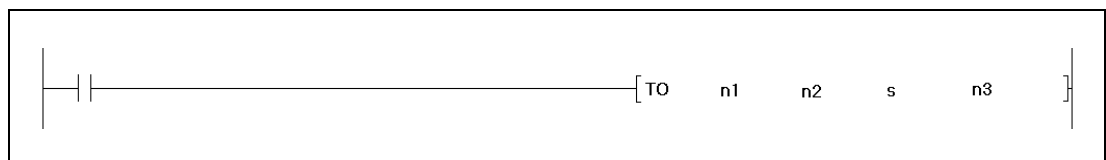
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other U
	Bit	Word		Bit	Word						
n1	●	●	●	●	●	●	●	●	●	SM0	5
n2	●	●	●	●	●	●	●	●			
s	●	●	●	—	—	—	—	●			
n3	●	●	●	●	●	●	●	●			

GX IEC  
Developer

MELSEC Instruction List  	Ladder Diagram  	IEC Instruction List  
--	---	---

GX  
Developer



Variables

Set Data	Meaning	Data Type
n1	Head address of special function module on base unit.	BIN 16-bit
n2	First number of memory address area to be written to.	



**Variables**

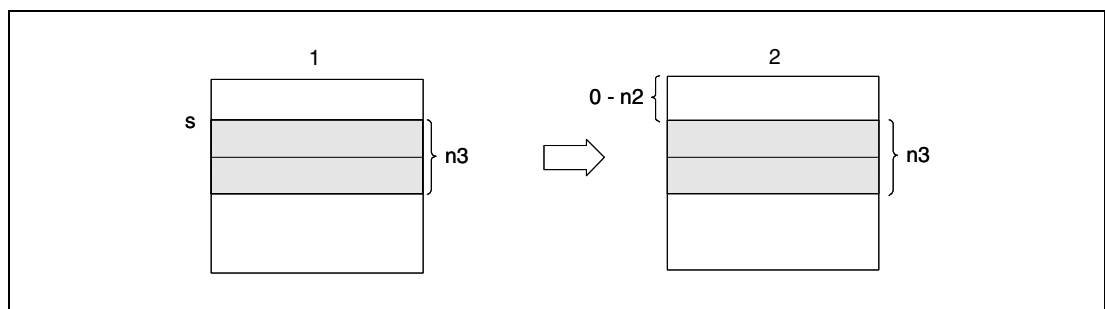
Set Data	Meaning	Data Type
s	Data to be written or first number of memory address area of the CPU storing data to be written.	BIN 16-/32-bit
n3	Number of data words to be written.	BIN 16-bit

**Functions**

**Writing 1-word and 2-word data to the buffer memory of a special function module**

**TO Writing 1-word data (16-bit)**

The TO instruction writes 1-word data from the memory of the CPU to the buffer memory of a special function module. The first address of the memory area data is to be written to is specified by n2, the number of data words is specified by n3, and the address of the special function module, resulting from the position of the module on the base unit is specified by n1. The first address of the memory address area the data is to be read from is specified by s.

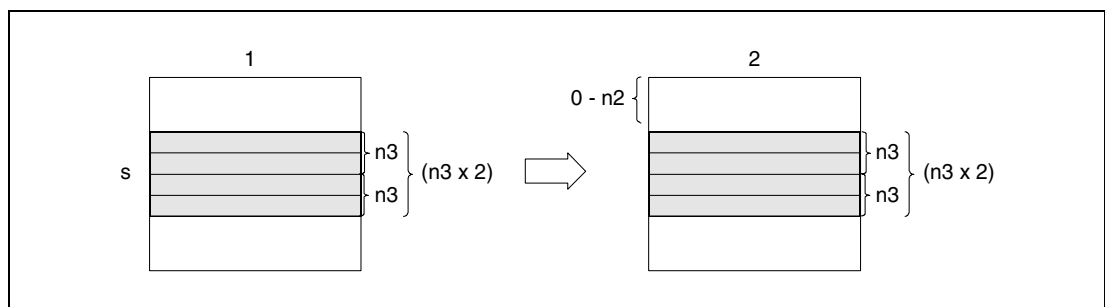


<sup>1</sup> Memory of the CPU

<sup>2</sup> Buffer memory of special function module

**DTO Writing 2-word data (32-bit)**

The DTO instruction writes 2-word data from the memory of the CPU to the buffer memory of a special function module. The first address of the memory area data is to be written to is specified by n2, the number of data words (2-multiple) is specified by n3, and the address of the special function module is specified by n1. The first address of the memory address area the data is to be read from is specified by s.



<sup>1</sup> Memory of the CPU

<sup>2</sup> Buffer memory of special function module

**Operation Errors**

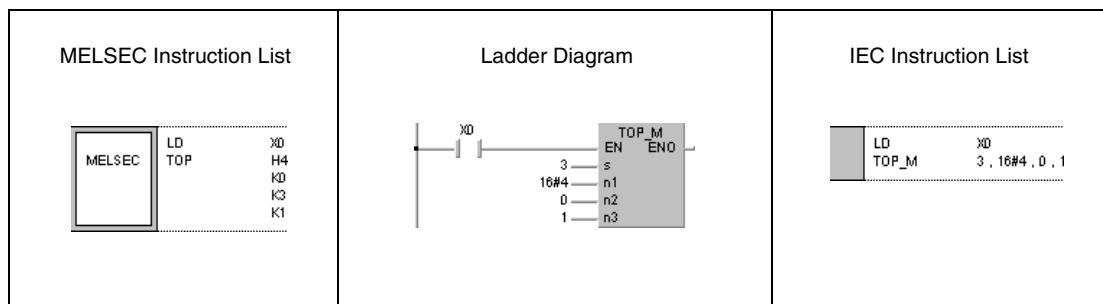
In the following cases an operation error occurs and the error flag is set:

- No signals have been exchanged with the special function module prior to the execution of the instruction (error code 1412).
- An error has occurred in the special function module prior to the execution of the instruction (error code 1402).
- The I/O number specified by n1 is not a special function module (Q series and System Q = error code 2110)
- The number of data words specified by n3 exceeds the storage range of the device specified by d (Q series and System Q = error code 4101).
- The address specified by n2 exceeds the buffer memory range (Q series and System Q = error code 4100)
- The address specified by n2 is inaccurate (AJ71QC24) (Q series and System Q = error code 4100).
- A special function module cannot be accessed.

**Program Example 1**

**TOP**

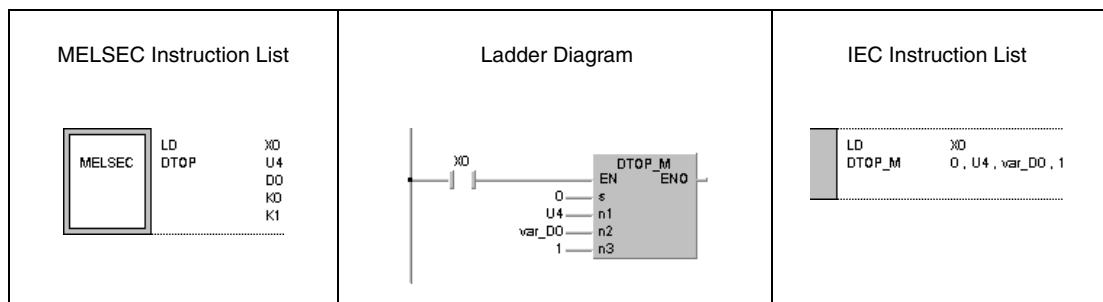
With leading edge from X0, the following program sets the channels CH1 and CH2 on an A68AD module to execute A/D conversion. The special function module is at address 040 through 05F. The value 3 is written to the buffer memory at address 0.



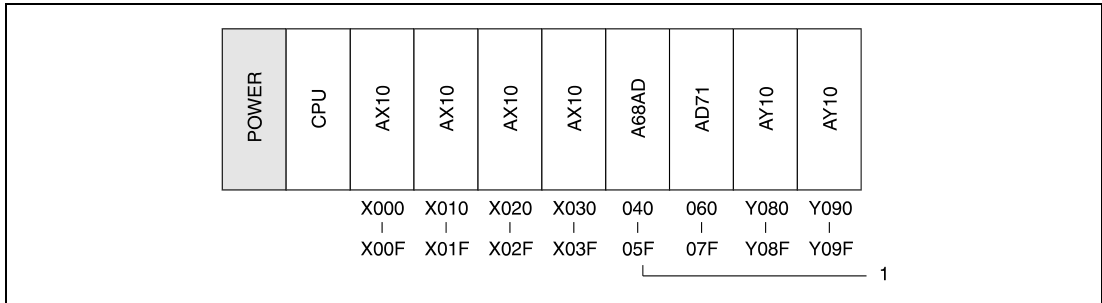
**Program Example 2**

**DTOP**

With leading edge from X0, the following program resets the x-data values at the buffer memory addresses 41 and 42 of a AD71 module to 0. The special function module is at address 040 through 05F.



**NOTE**      *The head address in n1 has to be specified as follows:*  
*n1 = 10 → head address = 1*  
*n1 = 20 → head address = 2*



<sup>1</sup> Head address of special register: n1 = K4 or H4

*The program example 2 will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 7.9 Display Instructions

The MELSEC Q and A series as well as the System Q supply several instructions that output ASCII characters at the outputs of an output module or on a LED display on the front panel of suitable CPU modules. In total, 7 different display instructions are supplied.

Function	MELSEC instruction in MELSEC Editor	MELSEC instruction in IEC Editor
ASCII character output	PR	PR_M
	PRC	PRC_M
Display of ASCII character and comments	LED	LED_M
	LEDC	LEDC_M
	LEDA	LEDA_M
	LEDB	LEDB_M
Clear display	LEDR	LEDR_M

**NOTE**

*Using an A3A, the LEDA and LEDB instructions cannot be processed directly as display instructions. Here, the instructions serve as start command for the Dedicated Application Instructions.*

*In order to use the functions of the LEDA and LEDB instructions with an A3A CPU, the sequence of the character string data has to be altered via the Dedicated Application Instructions of the AnA or AnAS CPUs. For details, refer to the separate programming manual for the AnA and AnAS series for details (Dedicated Instructions).*

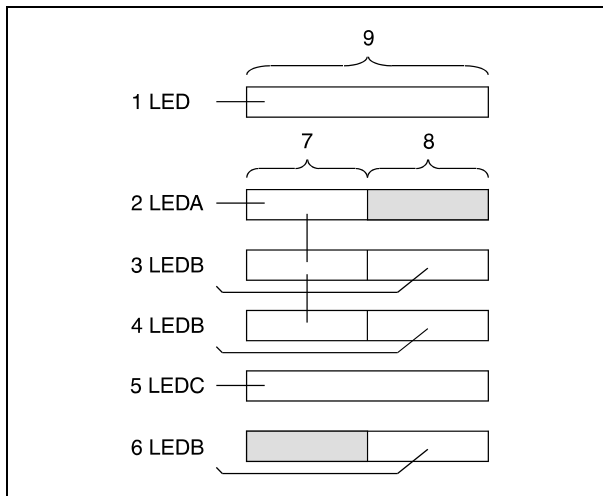
The LED display complies to the following priority:

1. Display of self diagnostics error
2. Display of CHK instruction
3. Display of number of annunciator F
4. Display of ASCII character via LED (A, B, C) instruction
5. BATTERY ERROR

Using an A3A CPU the priority can be freely changed. Refer to the manuals of the AnA series for further details.

If one of the first three displays is indicated, the execution of a display instruction does not change the current reading. If "BATTERY ERROR" is displayed, the reading on the display is changed when executing a LED (A, B, C) instruction.

The diagram below illustrates the LED display after execution of a LED (A, B, C) instruction.



On execution of a LED instruction (1) up to 16 characters (9) are displayed. After execution of a LEDA instruction (2) the first 8 characters (7) are displayed; the latter 8 characters (8) remain blank. If a LEDB instruction (3) follows, data is displayed on the latter 8 characters. If the LEDB instruction (4) is executed again, the data displayed in the latter 8 characters is overwritten; the data in the first 8 characters remain unchanged. After execution of a LEDC instruction (5) a preset comment (15 characters) is displayed. The execution of a LEDB instruction (6) overwrites the original data; the first 8 characters remain blank.

The following items can be displayed on the LED display on the front panel of a suitable CPU:

Numeral: 0 to 9

Alphabet: A to Z (capitals)

Special symbol: < > = \* / ' + -

7.9.1 PR

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices  
MELSEC A

	Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices							Word Devices (16-bit)							Constant	Pointer	Level					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P
s							●	●	●	●	●											
d	● <sup>1</sup>																					● <sup>7,1</sup>

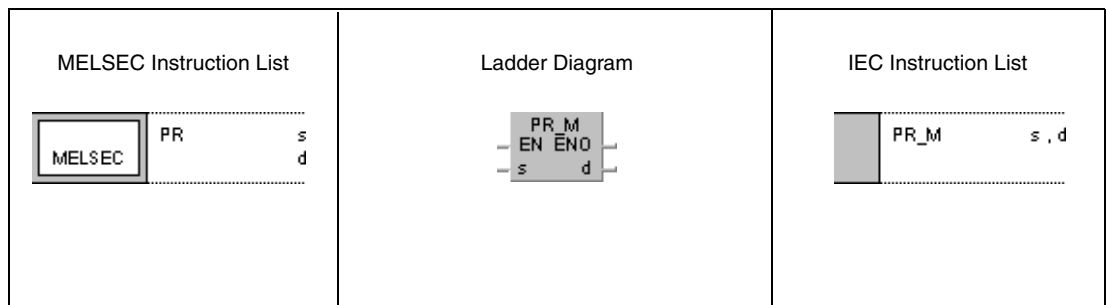
<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

Devices  
MELSEC Q

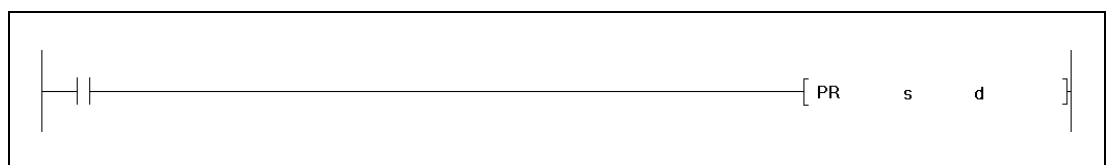
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	●	●	—	—	3
d	● <sup>1</sup>	—	—	—	—	—	●	—	—		

<sup>1</sup> Y only

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
s	First number of device storing ASCII code.	Character string
d	Head address of output module for ASCII code output.	Bit

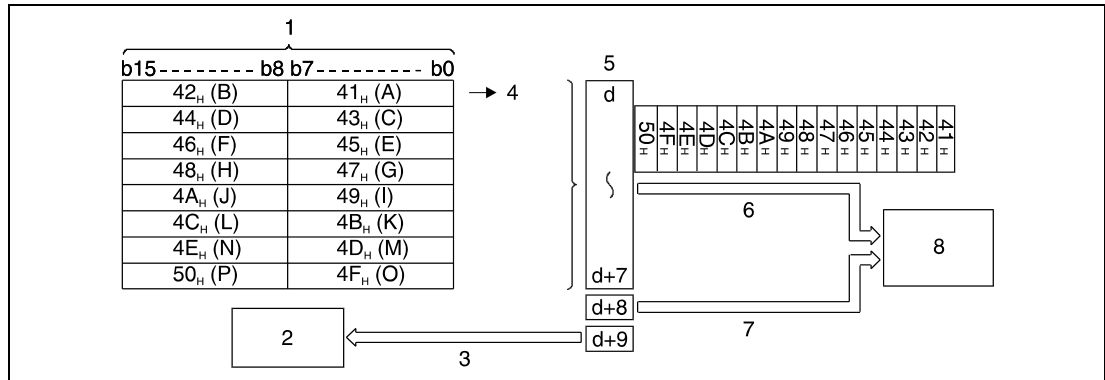
**Functions**     **Output to a peripheral device**

**PR**     **Output of an ASCII character string**

The PR instruction supplies two functions. Its function depends on the status of special relay M9049 (A series) or special relay SM701 (Q series and System Q) respectively:

M9049/SM701 set (1) (function 1):

Output of an ASCII character string of 16 characters to an output module. The character string, divided into twice 8 characters, is read from the address area s and output to the outputs specified by d.



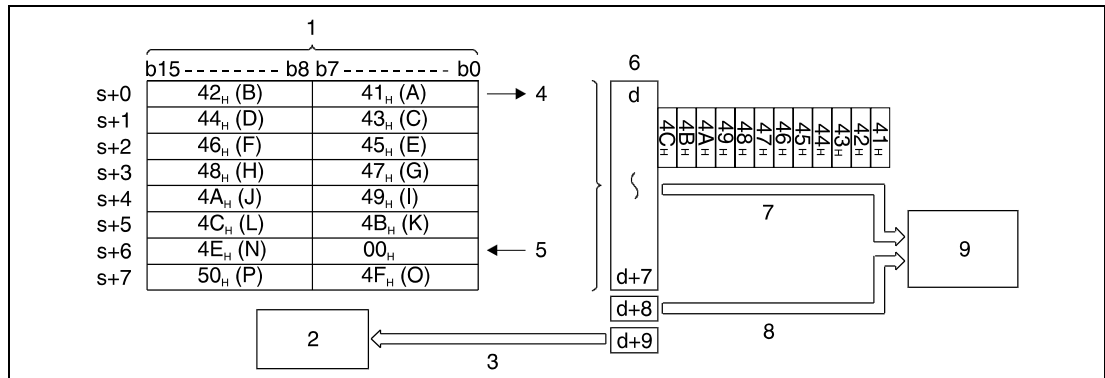
- 1 Device storing ASCII code
- 2 Sequence program
- 3 Flag indicating that PR instruction is in progress (used as interlock)
- 4 Start of output
- 5 Outputs Y
- 6 Output of ASCII code
- 7 Output of strobe signal
- 8 Printer or display device

The PR instruction can only access ASCII data already stored. If the stored data changes, the current data is output. For conversion from alphanumeric data into ASCII code an ASC instruction has to be applied.

During the output of 16 characters of ASCII code, the PR instruction execution flag d+9 is set ON. Thus, the output Y at address d+9 is set as long as the PR instruction is executed.

M9049/SM701 not set (0) (function 2):

Output of ASCII character string data up to the character code "00H" in hexadecimal format from the address area s to the outputs specified by d.



- 1 Device storing ASCII code
- 2 Sequence program
- 3 Flag indicating that PR instruction is in progress (used as interlock)
- 4 Start of output
- 5 End of character string (end of transmission)
- 6 Outputs Y
- 7 Output of ASCII code
- 8 Output of strobe signal
- 9 Printer or display device

If the content of the devices storing ASCII code is overwritten during the output, the current data is output.

The end of ASCII character string is indicated by the character code "00H".

If the hexadecimal code "00H" does not exist in the specified device, the execution is terminated and an error indicator is set.

During the output of ASCII code, the PR instruction execution flag d+9 is set ON.

**NOTE** An A series CPU can only execute function 1.

For the execution of a PRC instruction an output module with 10 successive binary outputs is needed. The address area begins at the output number specified by d.

Output signals from the output module are transmitted at the rate of 30 ms per character. Thus, processing n characters takes n x 30 ms. The output transmission is controlled via 10 ms interrupts, so the sequence program is processed continuously.

The 10 output addresses of the output module are processed independently from an I/O refresh after the END instruction in the program sequence.

In addition to the ASCII code a strobe signal (ON = 10 ms, OFF = 20 ms) is output at address Y= d+8.

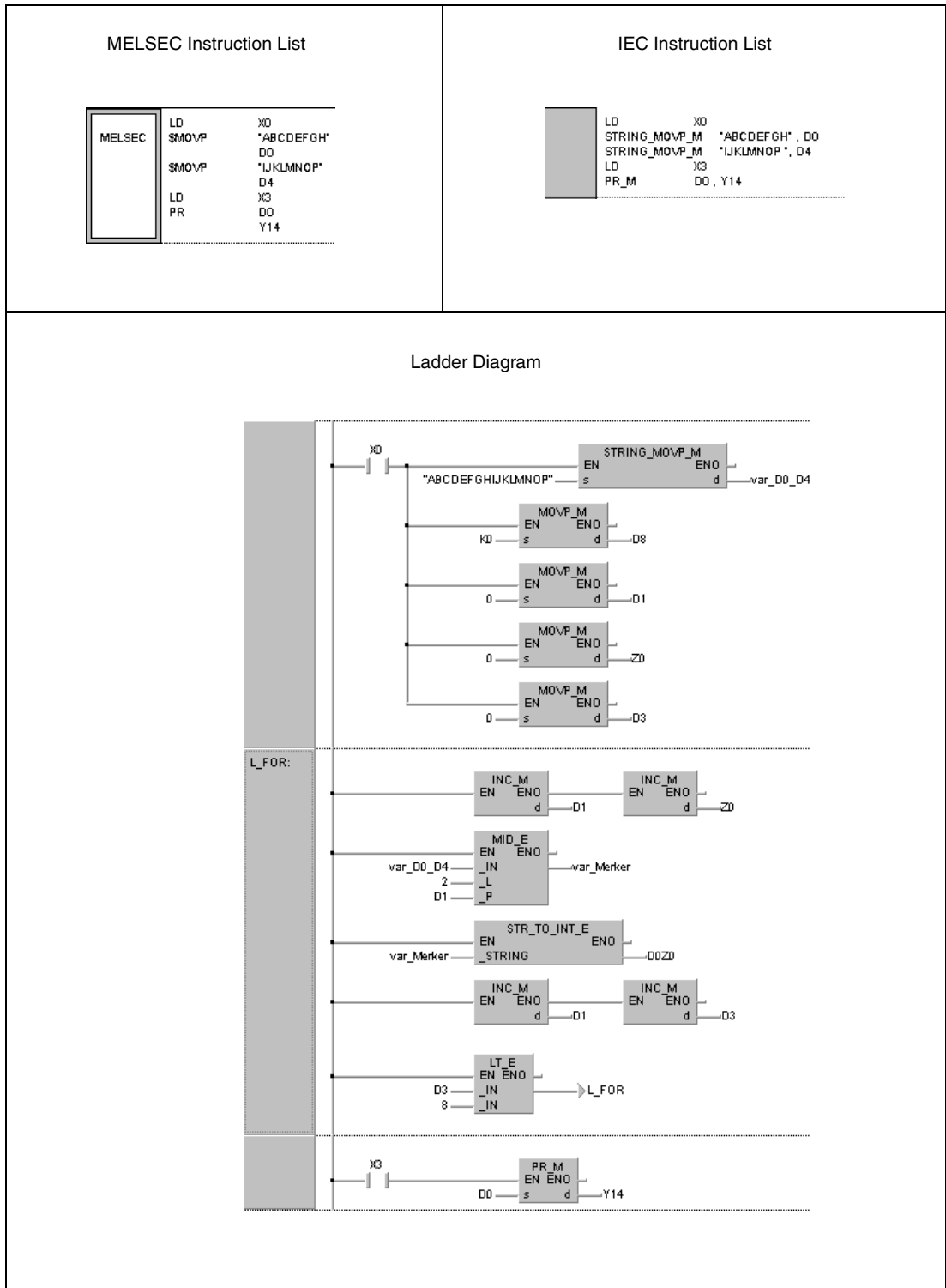
The PR and PRC instructions can be executed multiple times. Yet, an interlock should be established via the PR instruction execution flag (output device Y= d+9) so the PR and PRC instructions are not executed simultaneously.



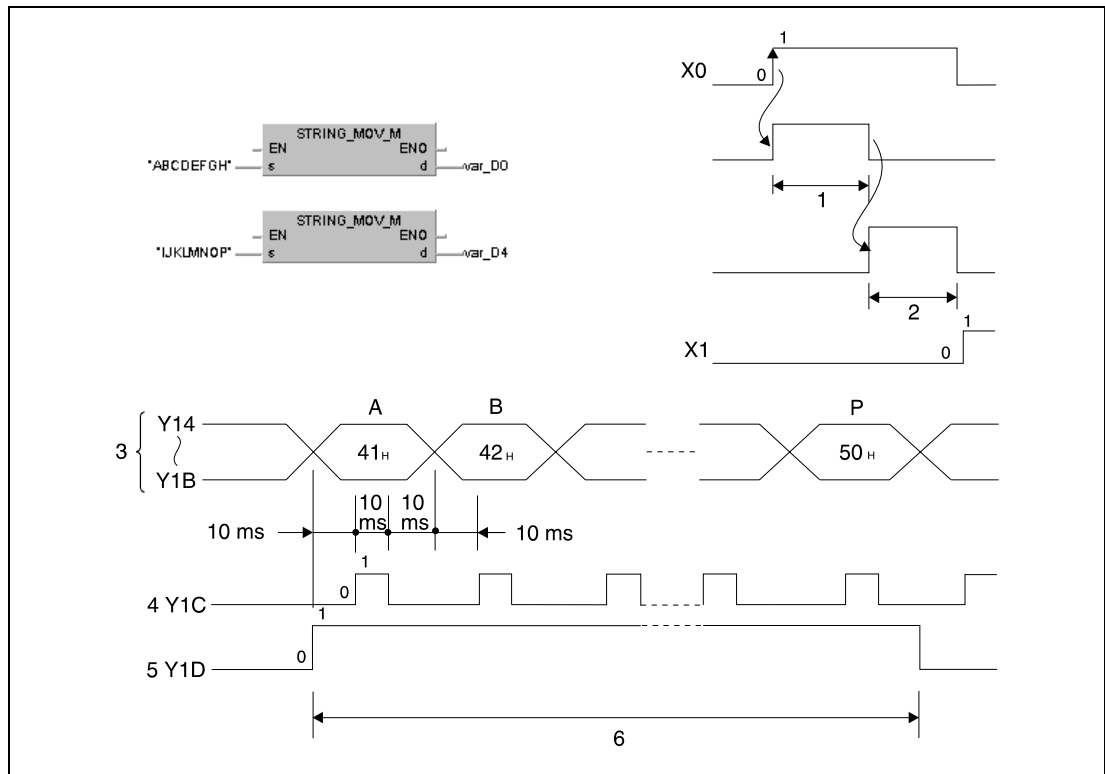
**Program Example**

PR

With leading edge from X0, the following program converts the character string "ABCDEFGH-IJKLMNOP" into ASCII code and stores it in data registers D0 through D7. After setting X3 ON, the ASCII code in D0 through D7 is output to the outputs Y14 through Y1D.



The following timing charts illustrate the processing of the program:



- <sup>1</sup> Storage of character string "ABCDEFGH" in D0 through D3
- <sup>2</sup> Storage of character string "IJKLMNOP" in D4 through D7
- <sup>3</sup> ASCII code
- <sup>4</sup> Strobe signal
- <sup>5</sup> PR instruction execution flag
- <sup>6</sup> Processing the PR instruction (period = 480 ms)

**NOTE**

*If no A series CPU is used and SM701 is not set, the value "00H" has to be written to register D8. Without this character code an operation error would occur in the program example above.*

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 7.9.2 PRC

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC A**

		Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
		Bit Devices						Word Devices (16-bit)						Constant		Pointer							Level	
s	d	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
●	●																							

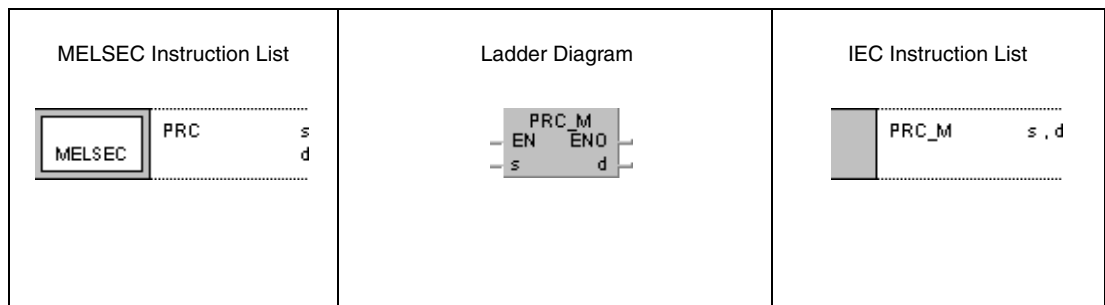
<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

**Devices  
MELSEC Q**

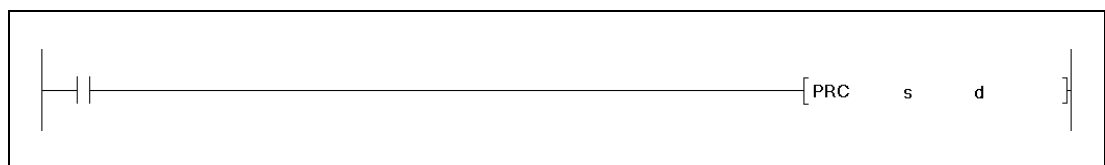
		Usable Devices								Error Flag	Number of steps	
		Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other P, I, J, U
s	d	Bit	Word		Bit	Word						
●	●	●	●	●	●	●	—	—	●	—	3	
● <sup>1</sup>	—	—	—	—	—	—	—	—	—	—		

<sup>1</sup> Y only

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
s	First number of device storing comment to be output.	BIN 16-bit
d	Head address of output module for comment output.	Bit

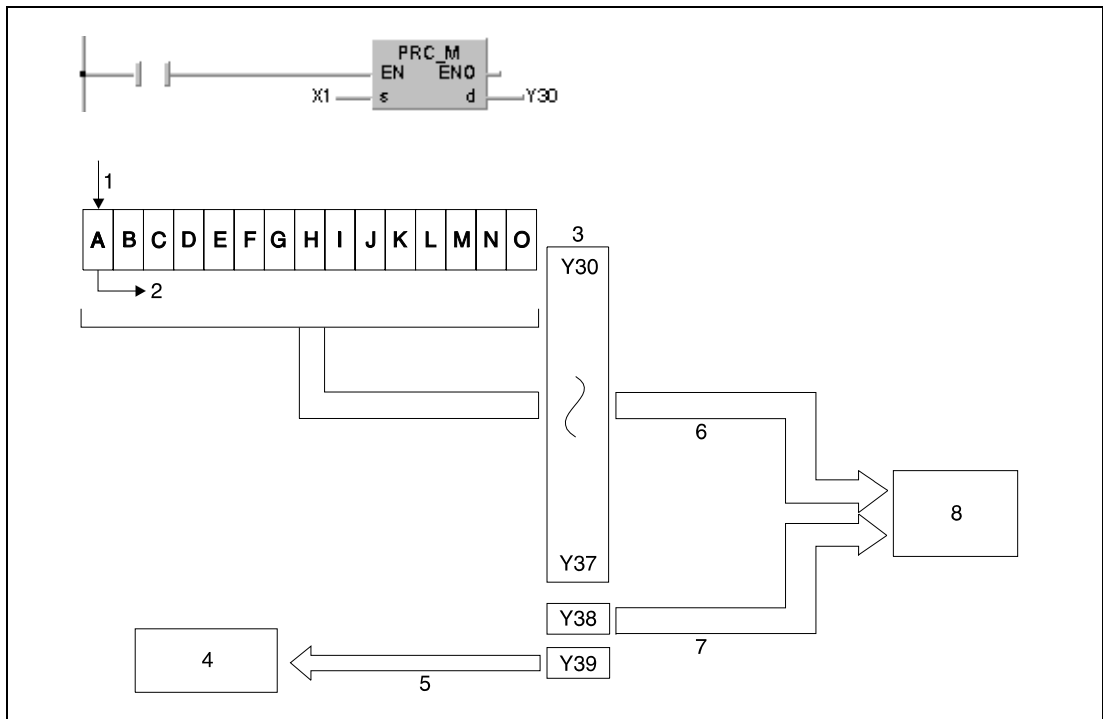
**Functions Output to a peripheral device**

**PRC Output of a comment**

The PRC instruction outputs a comment of a device (in ASCII code) to an output module.

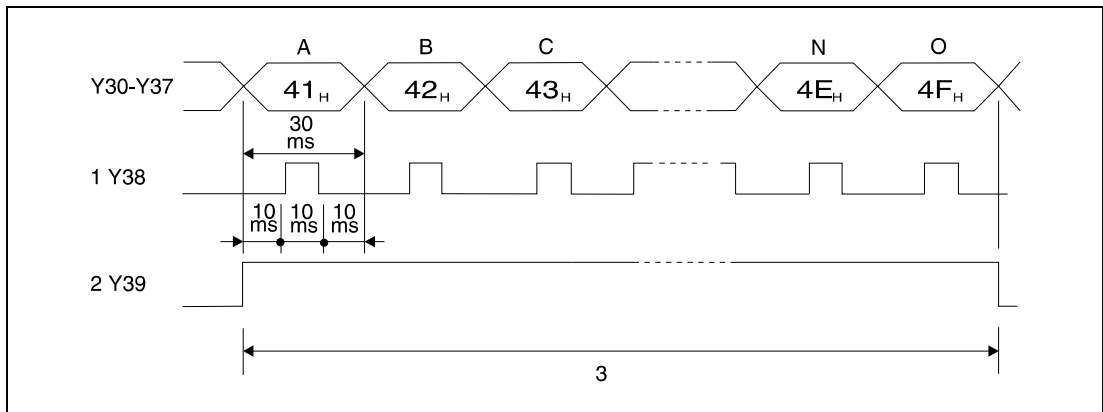
The MELSEC A series reads the character string divided into twice 8 characters from the address area s and outputs it to the outputs specified by d.

With the MELSEC Q series and the System Q the output of either 16 or 32 characters can be chosen. The choice is specified via special relay SM701. If SM701 is set (1), 16 characters are output; if SM701 is not set (0), 32 characters are output.



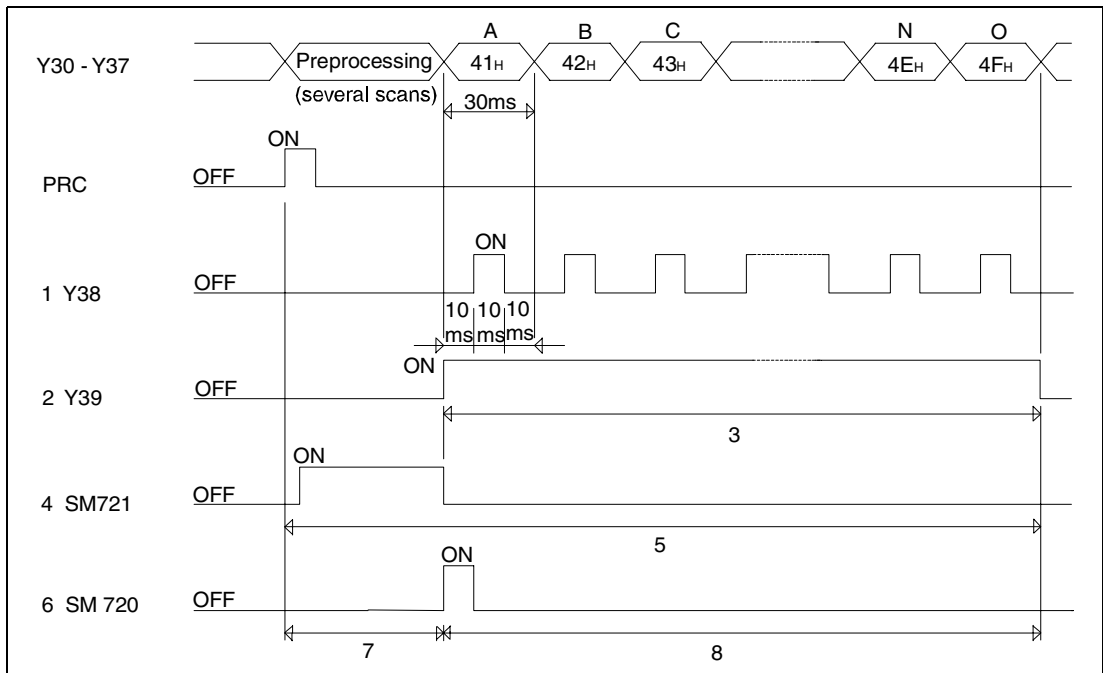
- 1 Comment (ASCII code) from X1 onwards
- 2 Start of output
- 3 Outputs Y
- 4 Sequence program
- 5 PR instruction execution flag (used as interlock)
- 6 Output of ASCII code
- 7 Output of strobe signal
- 8 Printer or display device

The following timing charts illustrate the processing of the PRC instruction in a QnA CPU:



- 1 Strobe signal
- 2 PRC instruction execution flag
- 3 Processing the PR instruction (period = 480 ms)

The processing of the PRC instruction in a multi processor CPU of the System Q is shown in the following timing chart:



- 1 Strobe signal
- 2 PRC instruction execution flag
- 3 Processing time (16 x 30 ms = 480 ms) for the PRC instruction
- 4 File access in process flag
- 5 The PRC instruction cannot be executed again
- 6 File access completion flag
- 7 No other instruction can be executed
- 8 Instructions other than PRC, S.FREAD, S.FWRITE, PLOAD, PUNLOAD and PSWAPP can be executed

There are 10 binary outputs of a digital output module assigned. The address area begins at the output address Y specified by d.

Output signals from the output module are transmitted at the rate of 30 ms per character. Thus, processing n characters takes n x 30 ms. The output transmission is controlled via 10 ms interrupts, so the sequence program is processed continuously.

In addition to the ASCII code a strobe signal (ON = 10 ms, OFF = 20 ms) is output at address Y= d+8.

During the output of 16 characters of ASCII code, the PRC instruction execution flag d+9 is set ON. Thus, the output Y at address d+9 is set as long as the PRC instruction is executed. The PR and PRC instructions can be executed multiple times. Yet, an interlock should be established via the PRC instruction execution flag (output device Y= d+9) so the PR and PRC instructions are not executed simultaneously.

If the address area s does not contain data, the instruction is not executed.

The PRC instruction can only access comments already stored in the PLC. For conversion from alphanumeric data into ASCII code an ASC instruction has to be applied.

After the execution of the PRC instruction is finished, SM720 turns ON for one scan. SM721 turns ON during the execution of the PRC instruction. The PRC instruction cannot be executed when SM721 is already ON. If an attempt is made, the processing will not be performed.

**NOTE**

*The PRC instruction can only access comments stored in a memory card. The PRC instruction can not access comments stored in the internal memory.*

*The comment file accessed by the PRC instruction is set at the "PC File Setting" in the Parameter mode.*

*The output of a comment file with the PRC instruction is not possible if no comment file has been set.*

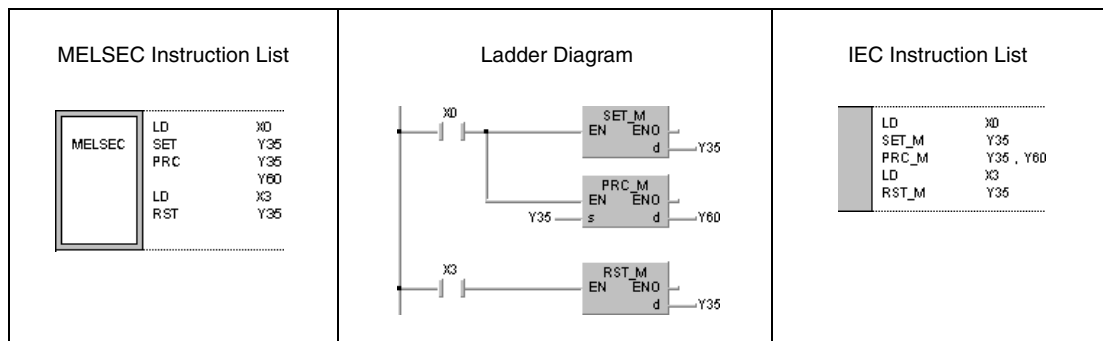
*Do not execute the PRC instruction during an interrupt program. Otherwise, malfunction may result.*

*The comment devices for the PRC instruction are stored on an IC memory card. The internal memory of the CPU cannot store comments (Q series and System Q only).*

**Program Example**

PRC

If X0 is set ON, the following program sets output Y35 ON and outputs the comment at Y35 in ASCII code simultaneously at the outputs Y60 through Y69. After setting X3 ON, Y35 is reset OFF.



### 7.9.3 LED

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
	● <sup>1</sup>	● <sup>2</sup>	●	● <sup>3</sup>	

<sup>1</sup> A3N CPU only.

<sup>2</sup> A3A CPU only.

<sup>3</sup> Except Q2A (S1) CPU

**Devices  
MELSEC A**

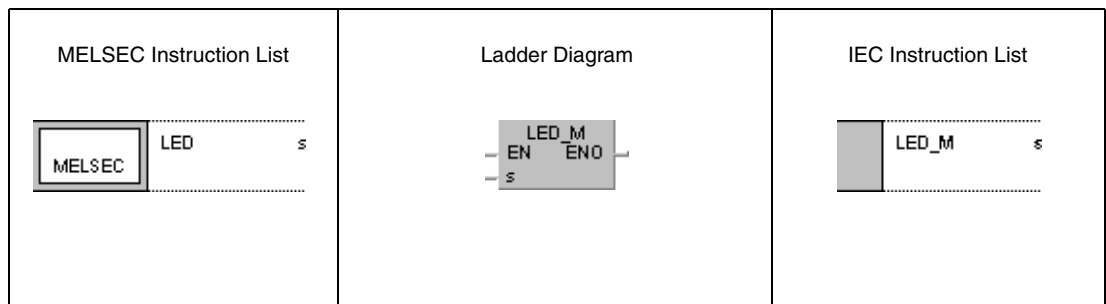
	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag					
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level											
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P	I	N
s							●	●	●	●	●												3	● <sup>1</sup>		●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

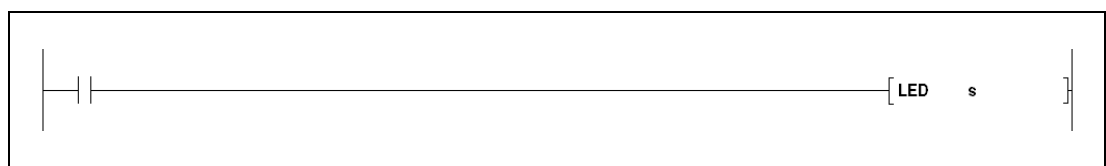
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	—	2

**GX IEC  
Developer**



**GX  
Developer**



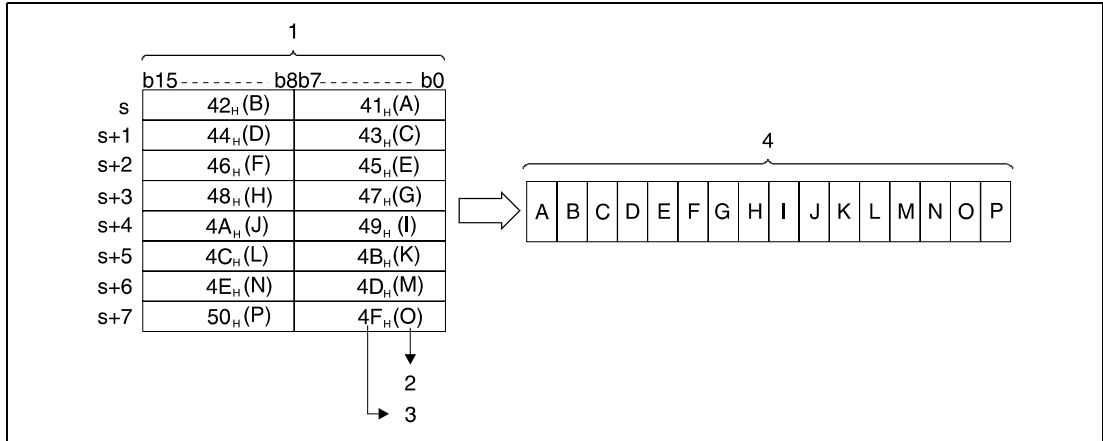
**Variables**

Set Data	Meaning	Data Type
s	First number of device storing ASCII data to be displayed.	Character string

**Functions**     **Output to a LED display**

**LED**     **Display ASCII data in the LED display on the CPU**

The LED instruction reads ASCII data (16 characters) from a specified address area and displays it on a suitable CPU display. The first number of device storing ASCII code in 8 addresses is specified by s (see illustration below).



- <sup>1</sup> Data to be displayed
- <sup>2</sup> ASCII character
- <sup>3</sup> ASCII code (hexadecimal)
- <sup>4</sup> LED display on the CPU front panel

If no ASCII data is stored in the specified address area, the display of timers, counters, and data and link registers remains blank. For file registers R the display is arbitrary; it remains blank if the according file registers are already cleared.

The following items can be displayed on the LED display on the front panel of a suitable CPU:

- Numeral: 0 to 9
- Alphabet: A to Z (capitals)
- Special symbol: < > = \* / ' + -

The LED instruction can only access ASCII data already stored. For conversion from alphanumeric data into ASCII code a \$MOV or ASC instruction has to be applied.

**NOTE**

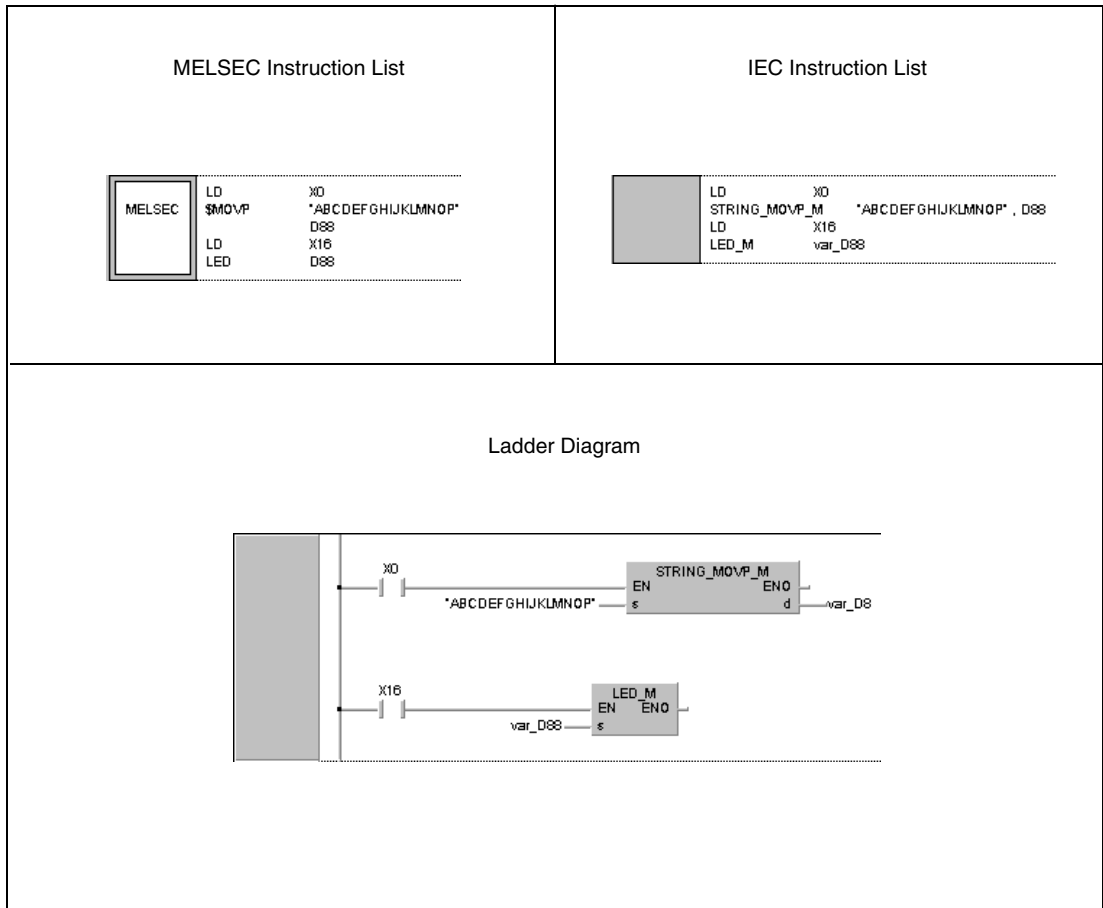
*The LED instruction can be used only in combination with a A3N, A3A, Q3A, Q4A or Q4AR CPU. If the instruction is executed on a CPU without LED-Display, no processing will be performed.*



**Program Example**

LED

The following program converts a character string into ASCII code, stores it in the registers specified, and outputs the contents of the registers on the LED display. In a first step after setting X0 ON, the following program converts the character string into "ABCDEFGHJKLMNQP" into ASCII code and stores it in the data registers D88 through D95. After setting X16 ON, ASCII data stored in D88 through D95 are displayed on the display on the CPU.



7.9.4 LEDC

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
	● <sup>1</sup>	● <sup>2</sup>	●	● <sup>3</sup>	

<sup>1</sup> A3N CPU only.

<sup>2</sup> A3A CPU only.

<sup>3</sup> Except Q2A (S1) CPU

Devices  
MELSEC A

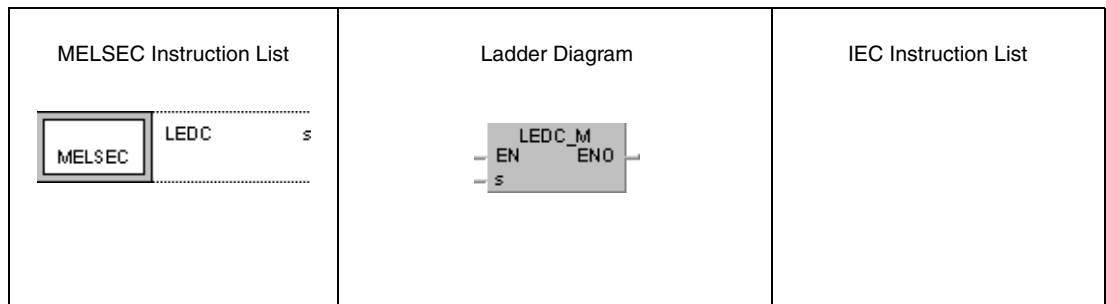
Usable Devices															Digit designation	Number of steps	Index	Carry Flag	Error Flag						
Bit Devices					Word Devices (16-bit)					Constant		Pointer		Level											
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z						V	K	H (16#)	P	I	N
●	●	●	●	●	●	●	●	●	●	●	●							●	●			3 ● <sup>1</sup>	●		●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

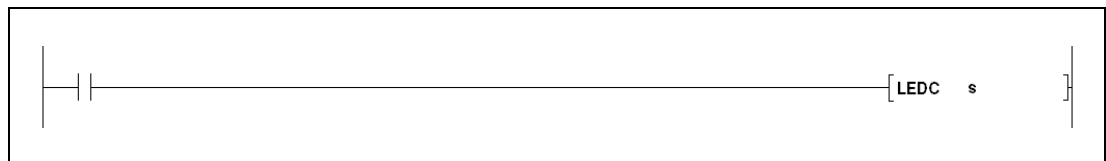
Devices  
MELSEC Q

s	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
●	●	●	●	●	●	—	—	●	—	2	

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s	First number of device storing comment to be displayed.	Device name

**Functions**      **Output to a LED display****LEDC**      **Display stored comment data in the LED display on the CPU**

The LEDC instruction reads comment data (16 characters) from a specified address area and displays it on a suitable CPU display. If more than 16 characters are to be displayed only the first 16 characters are displayed. The first number of device storing comment data is specified by s.

If no comment data is stored in the specified device, the display on the CPU front panel remains blank. If the data exceeds the comment range the LEDC instruction is not processed and the reading on the display remains unchanged.

If a comment contains characters that cannot be displayed, the display will be inaccurate. The following items can be displayed on the LED display on the front panel of a suitable CPU:

Numeral:0 to 9

Alphabet:A to Z (capitals)

Special symbol:< > = \* / ^ + -

A Q2ACPU(S1) cannot process a LED instruction. The instruction would be processed without any result.

**NOTE**

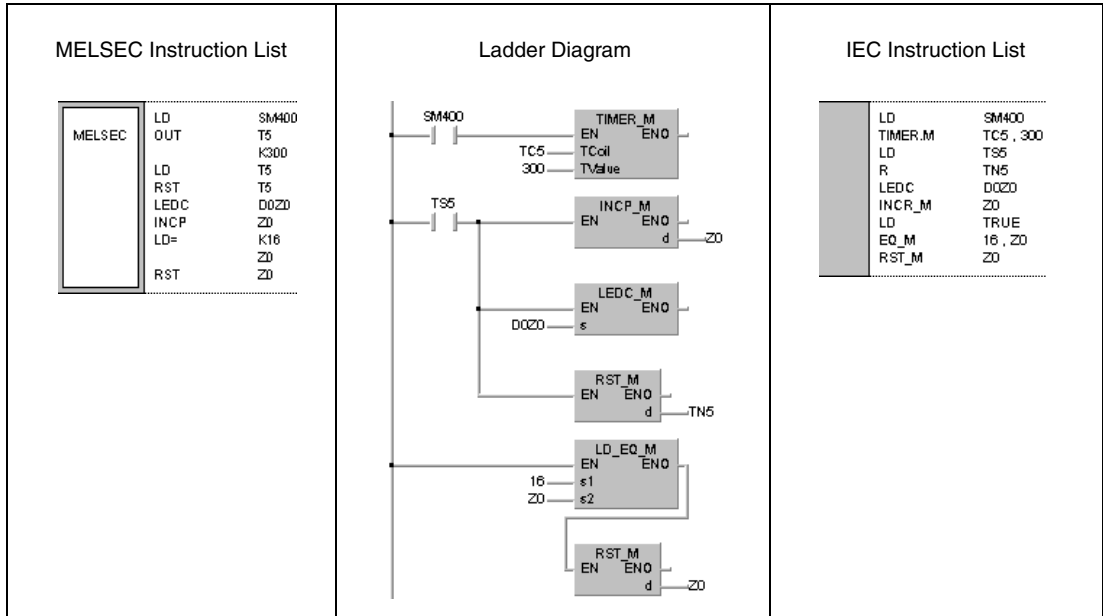
*In the Dedicated Instructions of the AnA CPUs the LEDC instruction sets devices.*

*Refer to the separate programming manual for the AnA CPUs for details on programming a LEDC instruction using an A3A CPU (Dedicated Instructions) (A series only).*

**Program Example**

LEDC

The following program displays comments in D0 through D15 in intervals of 30 seconds. Timer T5 sets the input condition for the LEDC instruction every 30 seconds. Once the timer switches ON, the comment in data register D(0+Z) is displayed and the value in Z is incremented by 1. If Z becomes 16, the value in Z is reset to 0.



**7.9.5 LEDA, LEDB**

**CPU**

AnS	AnN	An(S)	AnU	QnA(S), Q4AR	System Q
	● <sup>1</sup>				

<sup>1</sup> A3N CPU only.

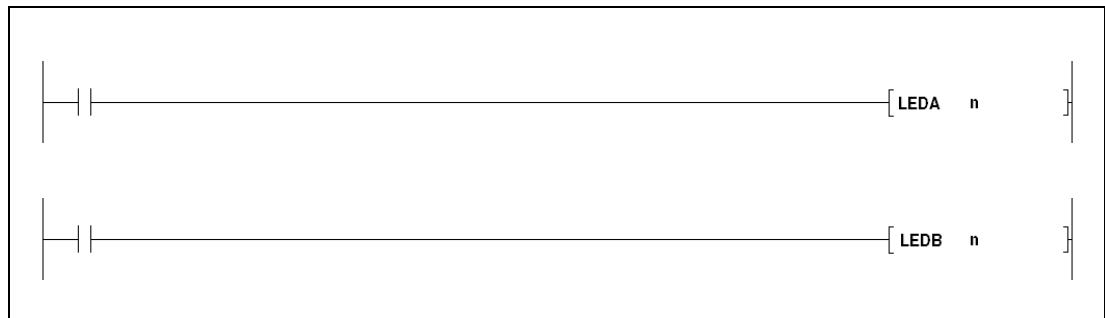
**Devices  
MELSEC A**

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag				
Bit Devices							Word Devices (16-bit)						Constant		Pointer							Level			
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I	N	M9012	M9010 M9011
n																							13		

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LEDA</td> <td style="text-align: right;">n</td> </tr> <tr> <td></td> <td>LEDB</td> <td style="text-align: right;">n</td> </tr> </table>	MELSEC	LEDA	n		LEDB	n	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td>LEDA_M</td> <td style="text-align: right;">n</td> </tr> <tr> <td>LEDB_M</td> <td style="text-align: right;">n</td> </tr> </table>	LEDA_M	n	LEDB_M	n
MELSEC	LEDA	n										
	LEDB	n										
LEDA_M	n											
LEDB_M	n											

**GX  
Developer**



**Variables**

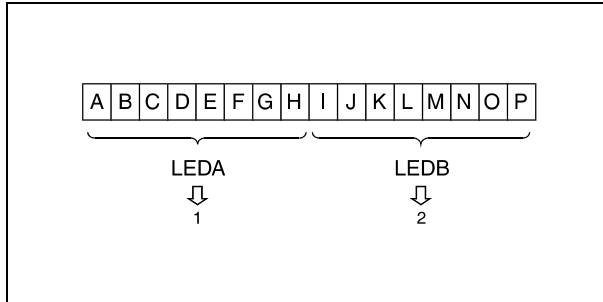
Set Data	Meaning	Data Type
n	ASCII data	Character string

**Functions**      **Output to a LED display**

**LEDA, LEDB    Display ASCII character string in the LED display on the CPU**

These instructions display an ASCII character string in the LED display of a suitable CPU. The ASCII character string consists of 8 characters for each instruction and is specified by the LEDA or LEDB instruction.

In total, up to 16 characters can be displayed with both instructions together. The LEDA instruction specifies the first 8 characters (left half), and the LEDB instruction specifies the latter 8 characters (right half) of the LED display.



- <sup>1</sup> Specification of first 8 characters
- <sup>2</sup> Specification of latter 8 characters

The following items can be displayed on the LED display on the front panel of a suitable CPU:

- Numeral: 0 to 9
- Alphabet: A to Z (capitals)
- Special symbol: < > = \* / ' + -

**NOTE**      *Using an AnA or AnU CPU the LEDA / LEDB instructions indicates the begin of the Dedicated Instructions. Refer to the separate programming manual for the AnA or AnU CPUs (Dedicated Instructions) for details on programming the LEDA / LEDB instructions.*

**Program Example**

LEDA, LEDB

If XC is set ON, the following program outputs the character string "ABCDEFGH IJKLMNPO" in the LED display of the CPU.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List									
<table border="1" style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px 5px;">MELSEC</td> <td style="padding: 2px 5px;">LD</td> <td style="padding: 2px 5px;">XC</td> </tr> <tr> <td style="padding: 2px 5px;"></td> <td style="padding: 2px 5px;">LEDA</td> <td style="padding: 2px 5px;">"ABCDEFGH"</td> </tr> <tr> <td style="padding: 2px 5px;"></td> <td style="padding: 2px 5px;">LEDB</td> <td style="padding: 2px 5px;">"IJKLMNPO"</td> </tr> </table>	MELSEC	LD	XC		LEDA	"ABCDEFGH"		LEDB	"IJKLMNPO"		
MELSEC	LD	XC									
	LEDA	"ABCDEFGH"									
	LEDB	"IJKLMNPO"									

**NOTE**      *The latter half of a character string displayed via the LED instruction is cleared, if the first 8 characters are overwritten via a LEDA instruction.*

*Vice versa, the first half of a character string is cleared, if the latter 8 characters are overwritten via a LEDB instruction.*

### 7.9.6 LEDR

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

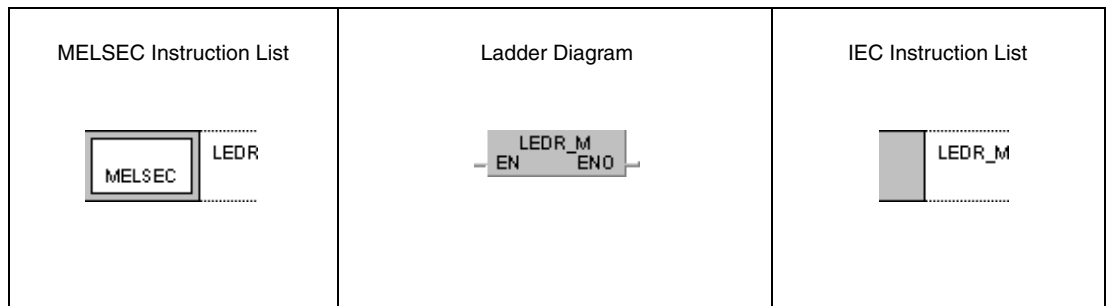
**Devices  
MELSEC A**

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag	
Bit Devices					Word Devices (16-bit)						Constant		Pointer		Level							
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I
																		1				

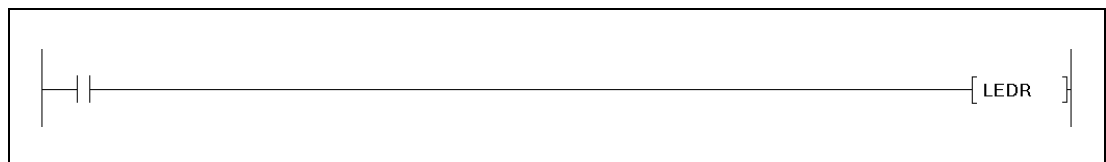
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions    Resetting annunciators and error displays****LEDR    Reset instruction**

The LEDR instruction resets annunciators that were set automatically when an operation error occurred. The LEDR instruction has the same effect as the actuation of the INDICATOR RESET button on CPU modules with a LED display (A series only).

**Operation of the LEDR instruction with an annunciator set during self-diagnosis (Q series and System Q only):**

If during self-diagnosis an error occurs that does not affect the accurate operation of the CPU, the execution of a LEDR instruction clears the "ERROR" LED and the error display on the CPU.

In addition, SM1 and SD0 at the user program have to be reset, because they are not reset automatically by the LEDR instruction. Further steps required to reset the annunciator are not executed neither.

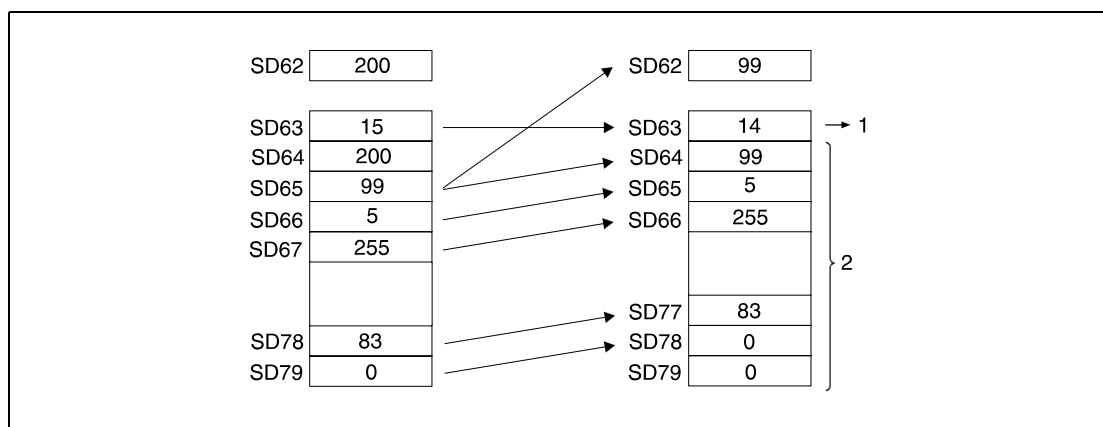
**Operation of the LEDR instruction on occurrence of a battery error (Q series only):**

If the LEDR instruction is executed after a battery replacement, the "BAT. ARM" LED on the front panel of the CPU and the error display on the CPU are cleared. At the same time, SM51 is reset automatically.

**Operation of the LEDR instruction with an annunciator F set on a CPU without LED display:**

After execution of the LEDR instruction the following operations are executed:

- The ERROR LED on the front panel of the CPU flickers and then turns off.
- The annunciator F stored in D9009 (A series) or SD62 (Q series/System Q) respectively is reset.
- The registers D9009 and D9125 (A series) or SD62 and SD64 (Q series/System Q) respectively are reset and the annunciators stored in D9126 through D9131 (A series) or SD65 through SD79 (Q series/System Q) respectively are shifted for further processing.
- The new number of annunciator F shifted to D9125 (A series) or SD62 (Q series/System Q) respectively is written to D9009 (A series) or SD62 (Q series/System Q) respectively.
- The accumulator of the annunciator in D9124 (A series) or SD63 (Q series/System Q) respectively is decremented by 1. If D9124 (A series) or SD63 (Q series/System Q) respectively is already at 0, this value remains unchanged.

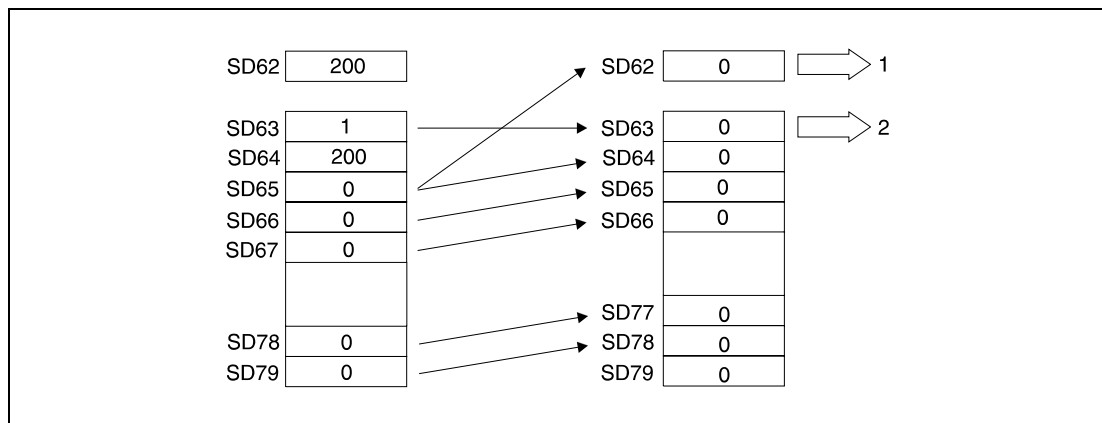




**Operation of the LEDR instruction with an annunciator F set on a CPU with LED display:**

After execution of the LEDR instruction, the following operations are executed:

- The annunciator displayed on the LED display of the CPU is cleared.
- The annunciator F stored in D9009 (A series) or SD62 (Q series/System Q) is cleared.
- The data registers D9009 and D9125 (A series) or SD62 and SD64 (Q series) respectively are reset, and the annunciators stored in D9126 through D9131 (A series) or SD65 through SD79 (Q series/System Q) respectively are shifted for further processing.
- The new number of annunciator F shifted to D9125 (A series) or SD62 (Q series/System Q) respectively is written to D9009 (A series) or SD62 (Q series/System Q) respectively.
- The accumulator of the annunciator in D9124 (A series) or SD63 (Q series/System Q) respectively is decremented by 1. If D9124 (A series) or SD63 (Q series/System Q) respectively is already at 0, this value remains unchanged.
- The current number of annunciator stored in D9009 (A series) or SD62 (Q series/System Q) respectively is displayed. If D9124 (A series) or SD63 (Q series/System Q) respectively is already at 0, there is nothing displayed.



<sup>1</sup> Since SD63 is at value 0, no annunciator is displayed on the LED display.

<sup>2</sup> Number of stored annunciators

**NOTE**

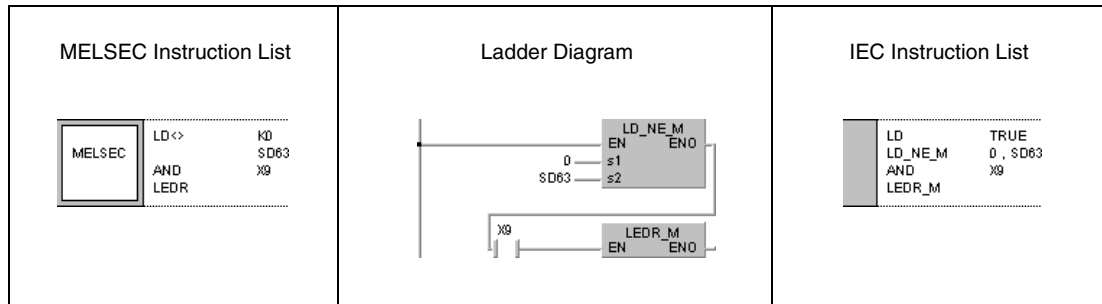
*A series only:*

*Using an AnA or AnU CPU the LEDR instruction indicates the completion of the Dedicated Instructions. Refer to the separate programming manual for the AnA CPU (Dedicated Instructions) for details on programming the LEDR instructions using an A3A CPU .*

**Program Example**

LEDR

If X9 is set and the value in register SD63 is not equal to 0, the following program executes a LEDR instruction.



**NOTE**

The defaults for the error item numbers set in special register SD207 to SD209 and the order of priority is shown in the table below:

Order of priority	Error item number (Hexadecimal)	Description	Remark
1	1	AC DOWN	Power supply cut
2	2	UNIT VERIFY ERR. FUSE BREAK OFF P. UNIT ERROR	I/O module verify error Blown fuse Special function module verify error
3	3	OPERATIN ERROR LINK PARA ERROR SFCP OPE. ERROR SFCP EXE. ERROR	Operation error Link parameter error SFC instruction operation error SFC program execution error
4	4	ICM.OPE ERROR FILE OPE ERROR EXTEND INST. ERROR	Memory card operation error File assess error Extend instruction error
5	5	PRG.TIME OVER	Constant scan setting time over error Low speed execution monitoring time over error
6	6	CHK instruction	Fehler wurde mit der CHK-Anweisung festgestellt
7	7	Annunciator	
8	8	LED instruction	
9	9	BATTERY ERR.	
10	A	Clock data	

## 7.10 Failure diagnosis and debugging

The instructions for failure diagnosis and debugging support failure checks, setting and resetting the status latch, sampling trace, and program trace. The following table gives an overview of these instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Failure check	CHKST	CHKST_M
	CHK	CHK_M
	CHKCIR	CHKCIR_M
	CHKEND	CHKEND_MD
Set / reset status latch	SLT	SLT_M
	SLTR	SLTR_M
Set / reset sampling trace	STRA	STRA_M
	STRAR	STRAR_M
Execute / set / reset program trace	PTRA	PTRA_M
	PTRAR	PTRAR_M
	PTRAEXE	PTRAEXE_M
	PTRAEXEP	PTRAEXEP_M

**NOTE** *Please check, whether these functions are available and supported by your version of the GX IEC Developer.*

7.10.1 CHKST, CHK (Q series and System Q only)

CPU

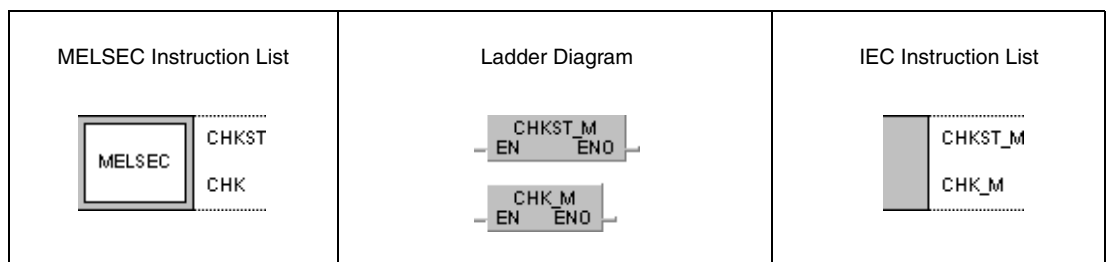
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

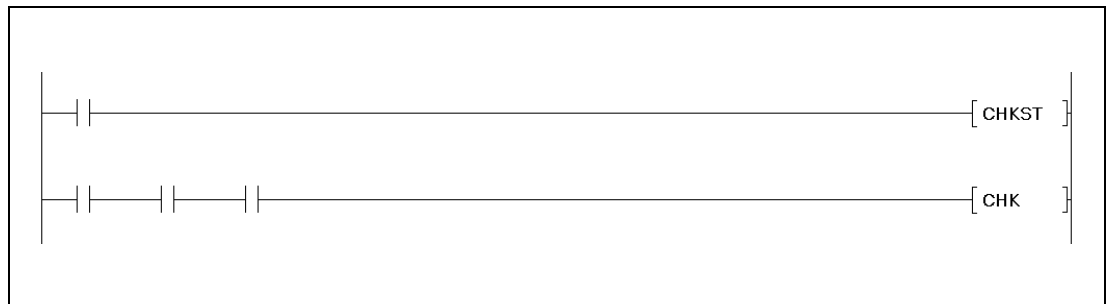
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word				DY			
—	—	—	—	—	—	—	—	—	SM0	1	

GX IEC Developer



GX Developer



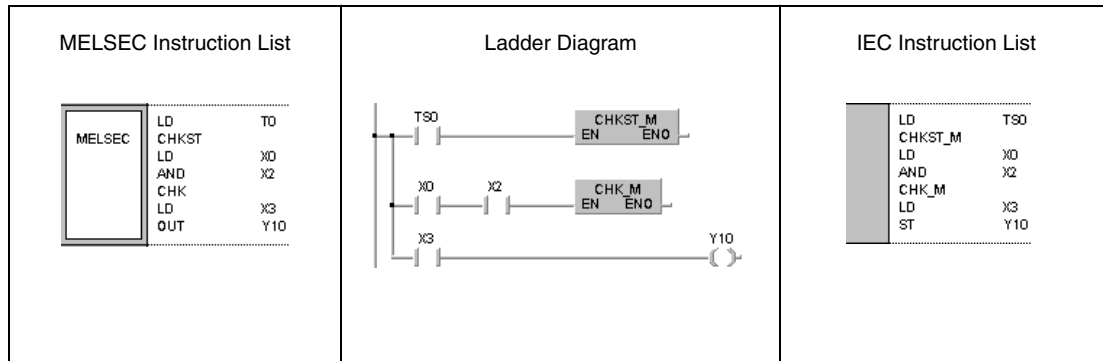
Variables

Set Data	Meaning	Data Type
—	—	—

**Functions Failure check for bidirectional operations (Q series and System Q only)**

**CHKST Start instruction for the CHK instruction**

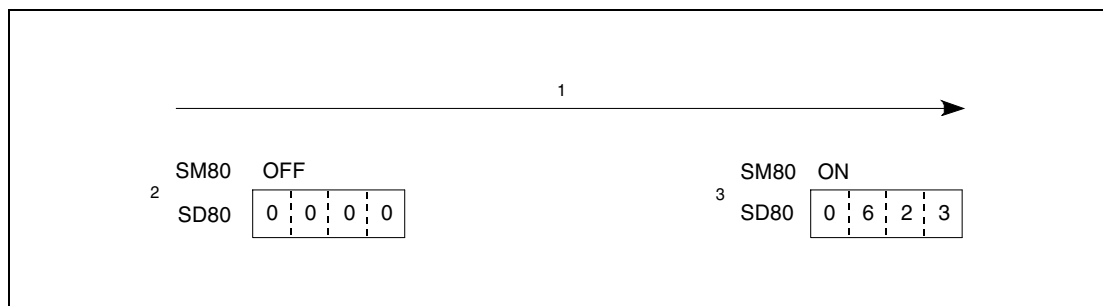
The CHKST instruction starts the execution of the CHK instruction. If the execution condition for the CHKST instruction is not set (0), the program step following the CHK instruction will be executed. With the execution condition for the CHKST instruction set (1), the CHK instruction is executed. In the ladder diagram below these instructions are programmed.



**CHK Failure check instruction**

The CHK instruction with some CPU types (and depending on the control mode) supports failure check operations for contact circuits with limit switches that monitor bidirectional movement. Once an error occurs within such a circuit, the special relay SM80 is set and the corresponding error code is stored in special register SD80.

The Q series and the CPUs of the System Q stores the error code as BCD 4-digit data value in special register SD80. The upper 3-digits store the contact number of the corresponding contact (here contact 62) and the lower digit stores the number of the failure check circuit (coil number 1 - 6; here coil number 3).



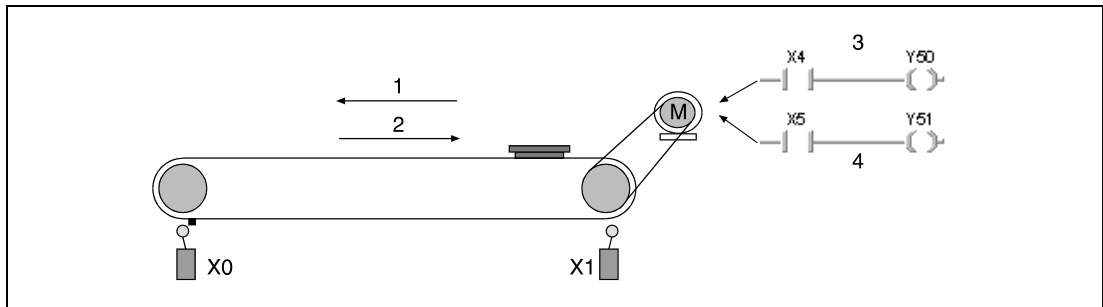
<sup>1</sup> Contact 62; coil number 3 (during failure check)

<sup>2</sup> Before failure check

<sup>3</sup> After failure check

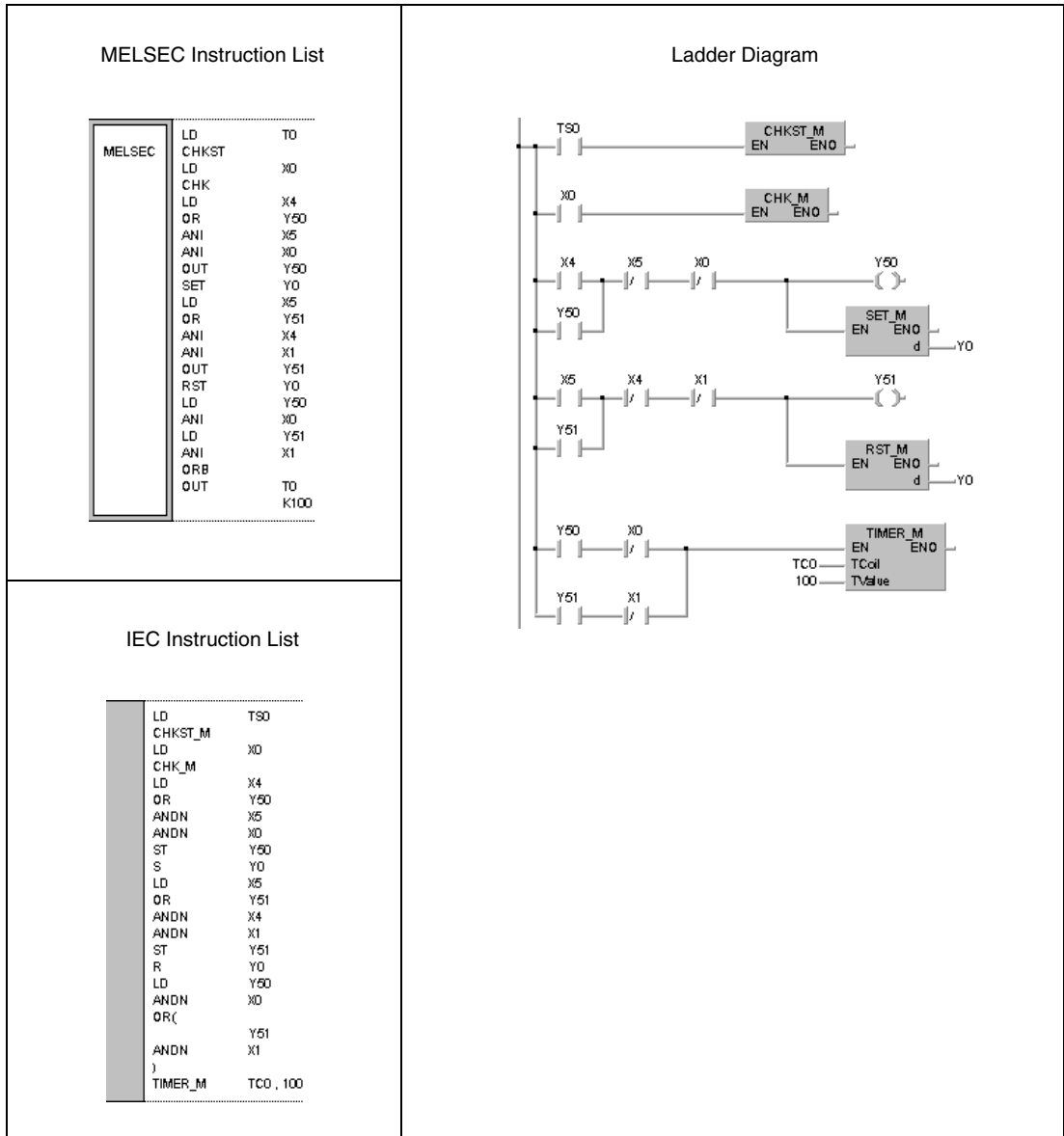
The input contacts programmed prior to the CHK instruction do not serve as execution condition for the CHK instruction but as specification of the check conditions.

In the following, the failure check programming via the CHK instruction is illustrated with a concrete example. The following illustration shows a conveyor belt that moves from the left to the right travel limit. The corresponding travel limits are detected via limit switches (X0 and X1). The start contact for advance movement is X4 and for retract movement is X5.



- <sup>1</sup> Advance movement
- <sup>2</sup> Retract movement
- <sup>3</sup> Advance command
- <sup>4</sup> Retract command

The diagrams below show a sample program for the operation and failure check of the conveyor belt shown above using a Q series CPU. During error free operation the program jumps to the program step following the CHK instruction. With leading edge from X4, the conveyor belt is advanced, and Y0 is set for failure check. With leading edge from X5, the conveyor belt is retracted, and Y0 is reset. The timer T0 watches the duty cycle time. If the duty cycle time is exceeded the CHKST instruction is set via the contact TS0. In the next program step the CHK instruction is executed, and the error code is stored in the special register SD80.



The operations of the CHK instruction can be illustrated through the following ladder diagrams, of which the functions are similar to the execution of the CHK instruction. The contact numbers of the limit switches for advance movement  $X□$  and retract movement  $X□+1$  have to be designated successively. The number of the advance limit switch  $X□$  must be less than the number of the retract limit switch  $X□+1$ . The contact number of the advance limit switch is assigned to an output  $Y□$  with the same address. According to the program example, this output is set during advance movement and reset during retract movement.

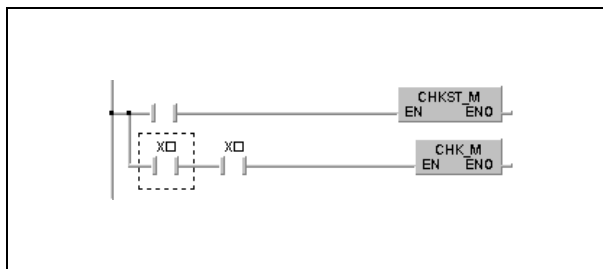
For better comprehensibility of the program example above, the contacts  $X0$  ( $X□$ ),  $X1$  ( $X□+1$ ) and  $Y0$  ( $Y□$ ) are applied directly for specification of the coil number. Depending on the program they can be replaced by any other number.

**NOTE**

*The outputs  $Y□$  are treated as internal relays and cannot be output to external devices.*

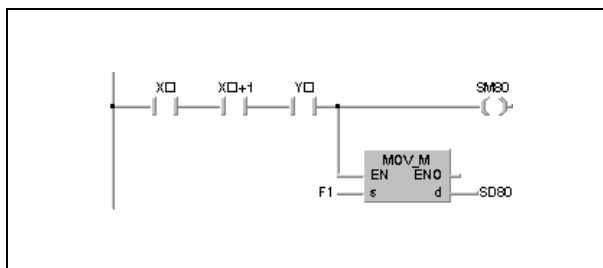
The following diagrams concerning the CHK instructions and the 6 generated failure check circuits (error conditions) are arranged in pairs.

In the following, the CHK instructions are illustrated. The contact indicated  $X□$  serves as variable for maximum 150 contacts (150 conveyor belts or similar applications).



Failure check circuit 1 (coil number 1):

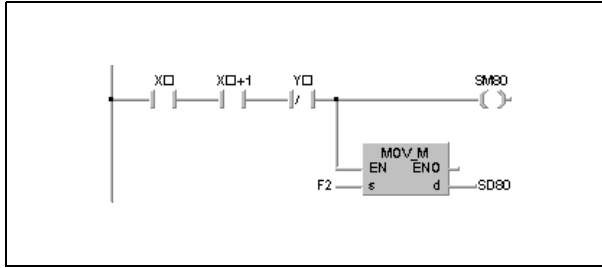
Both limit switches respond to the advance movement of the conveyor belt.





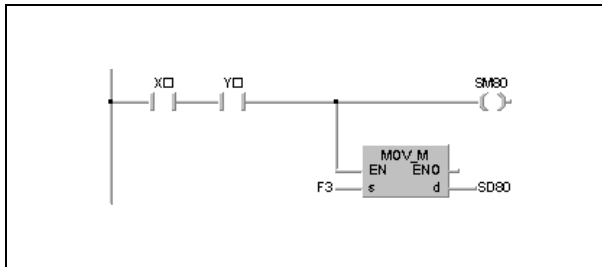
Failure check circuit 2 (coil number 2):

Both limit switches respond to the retract movement of the conveyor belt.



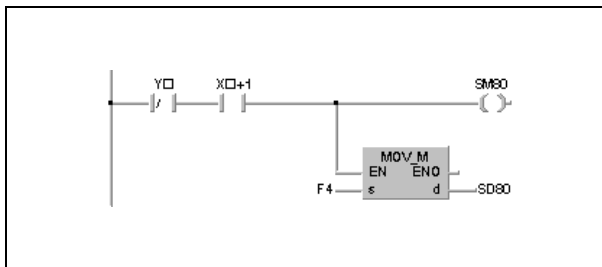
Failure check circuit 3 (coil number 3):

Advance command for set advance limit switch.



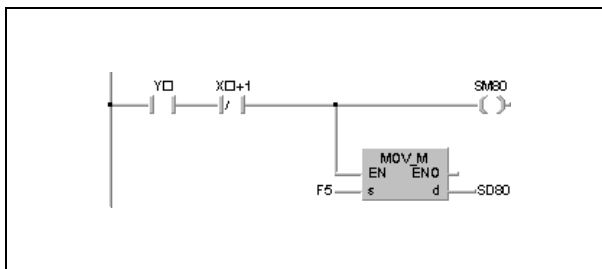
Failure check circuit 4 (coil number 4):

Retract command for set retract limit switch.

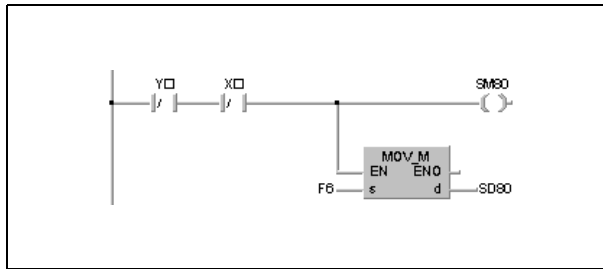


Failure check circuit 5 (coil number 5):

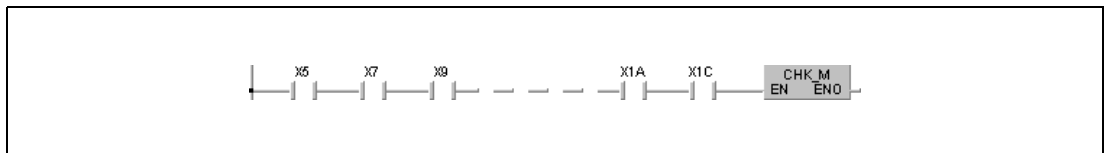
Advance command for reset retract limit switch.



Failure check circuit 6 (coil number 6):  
Retract command for reset advance limit switch.



The CHK instruction can designate a maximum of 150 contact numbers for advance limit switches. For the designation of contact numbers any contact number of the retract limit switch is skipped.

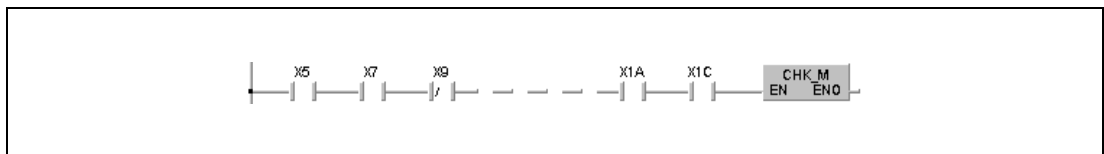


The relay SM80 and the special register SD80 have to be reset after execution of the CHK instruction because they retain their condition after being set. If they are not reset prior to another CHK instruction, the instruction cannot be executed.

The CHKST instruction has to be programmed prior to the CHK instruction.

The CHK instruction can be programmed in any program step of the sequence program. The CHK instruction can be executed twice at most within one program organization unit (POU).

The coil numbers have to be programmed via a LD or AND instruction prior to the CHK instruction. Other input instructions are not supported. If an LDI or ANI instruction is programmed, the failure check of the CHK instruction cannot be executed. The contact numbers designated for the failure check however can be designated via the LDI and ANI instructions. In the diagram below the switch with the number X9 is ignored because it is an NC contact (normally closed).



Using a Q series or a System Q CPU, the failure detection method depends on the status of the special relay SM710 as follows.

SM710 is reset (0):

The failure check is performed in coil number (failure check circuit) sequence from contact 1 (limit switch) to contact n (limit switch).

The first contact is checked from coil number 1 through coil number 6. Then the next contact is checked from coil number 1 through coil number 6. The operation is completed after the nth contact is checked from coil number 1 through coil number 6.

SM710 is set (1):

The failure check is performed in contact number (limit switch) sequence from coil 1 (failure check circuit) through coil 6 (failure check circuit).

The first coil is checked from contact number 1 through contact number n. Then the next coil is checked from contact number 1 through contact number n. The operation is completed after the 6th coil is checked from contact number 1 through contact number n.

If more than one failure is detected, the number of the first failure detected is stored. Further detected failures are ignored.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- Two failure check input contacts within one failure check circuit are connected in parallel (error code 4235).
- More than 150 input devices are specified (error code 4235).
- A CHKST instruction is not followed by a CHK instruction (error code 4235).
- A CHK instruction is executed without a prior CHKST instruction (error code 4235).

7.10.2 CHK (A series only)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
● <sup>1</sup>	● <sup>1</sup>	●	●		

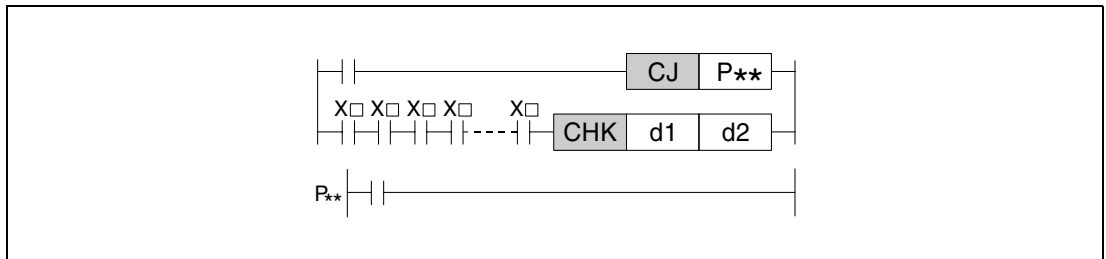
<sup>1</sup> In direct mode only

Devices MELSEC A

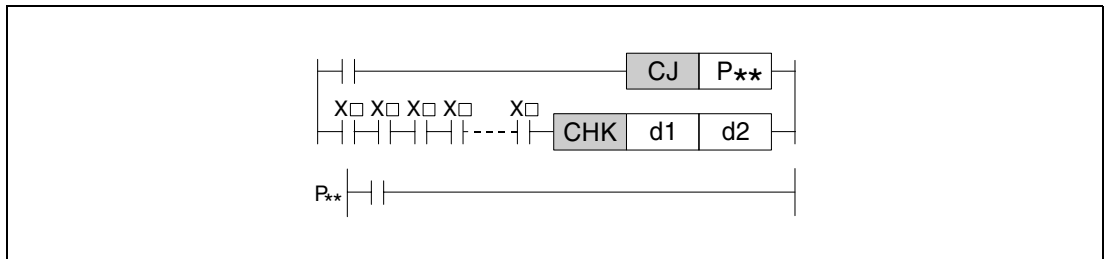
	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
d1	●	●	●	●	●	●																	
d2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						5 <sup>1</sup>	

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
d1	Device to be set during failure check.	Bit
d2	Device storing error code.	BIN 16-bit

**Functions**    **Failure check for bidirectional operations (A series only)****CHK**    **Failure check instruction**

The function of the CHK instruction depends on the selected I/O control mode. Using A1S and AnN CPUs in refresh mode, the check instruction generates a flip-flop.

In direct I/O control mode (except for AnA, AnAS, AnU, and A2C CPUs) the check instruction checks for failures in bidirectional operations.

Due to the pointer 254, the CHK instruction can only be programmed in a instruction list.

The CHK instruction in combination with some CPU types (and depending on the control mode) supports a failure check in a contact circuit with limit switches for detection of failures in bidirectional movement operations. Once an error occurs within such a circuit the device in d1 is set and the corresponding error code is stored in d2.

The input contacts programmed prior to the CHK instruction do not serve as execution conditions for the CHK instruction but as specification of the check conditions.

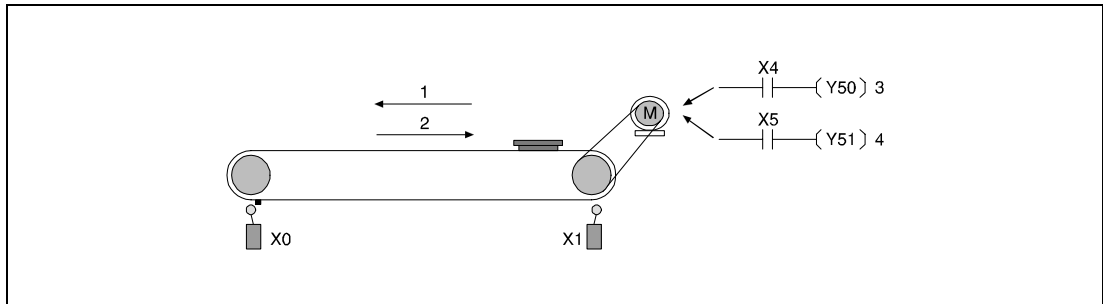
The purpose of the CHK instruction is to detect the occurrence and cause of failures in the program execution, e.g., if the processing time of a duty cycle is exceeded. If no errors occur during program execution, the execution of the program part containing the CHK instruction should be skipped via the CJ, SCJ, or JMP instruction.

The CHK instruction is executed with every program scan and is independent from the status of the input devices programmed prior to the CHK instruction as specification of the check conditions.

The following program sets Y60 and executes the check instruction, if the processing time of one duty cycle is exceeded. Once the failure is detected by the CHK instruction, M0 is set, and the program jumps to the jump destination P31 (not shown below). The jump destination P31 (not shown below) for example could store a program part for error processing. If the processing time is not exceeded, the program part for the failure check is skipped and step 18 at jump destination P30 is executed. Due to pointer 254, this program can only be programmed in a instruction list.

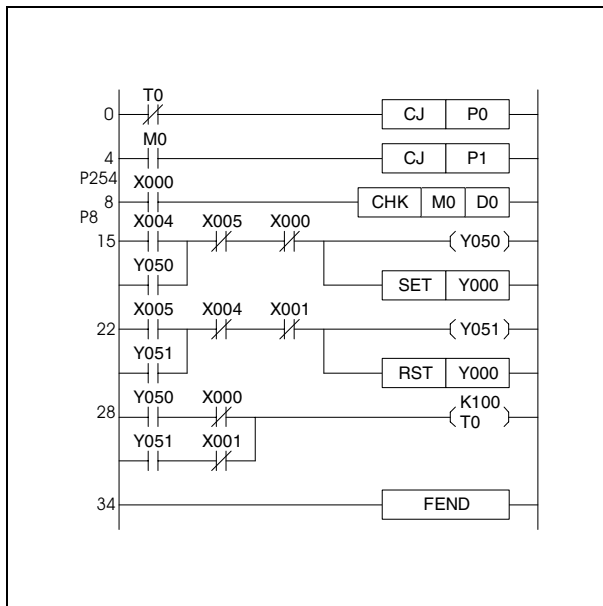
100	LDI	Y060	
101	CJ	P30	
104	LD	M0	
105	CJ	P30	
108	P254		
109	LD	X010	
110	AND	X015	
111	AND	X008	
112	AND	X01A	
113	CHK	M0	D0
118	P30		
118	LD	M10	
119	OUT	Y040	
121	END		

In the following, the failure check programming via the CHK instruction is illustrated with a concrete example. The following illustration shows a conveyor belt that moves from the left to the right travel limit. The corresponding travel limits are specified via limit switches (X0 and X1). The start contact for advance movement is X4 and for retract movement is X5.

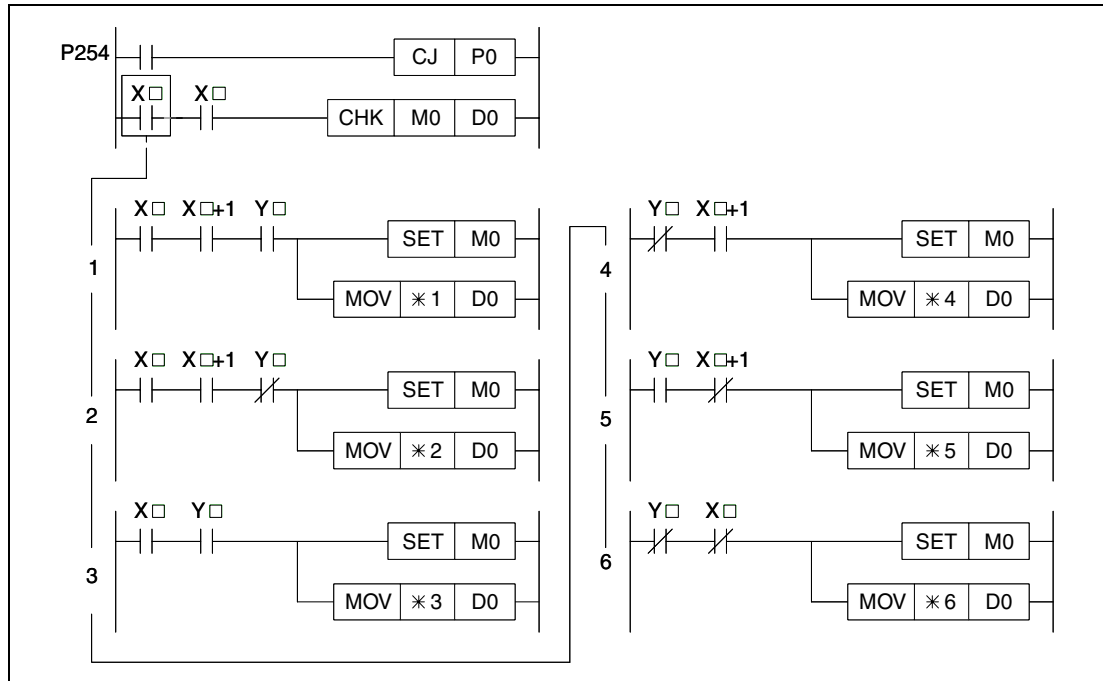


- 1 Advance movement
- 2 Retract movement
- 3 Advance command
- 4 Retract command

The diagram below shows a sample program for the operation and failure check of the conveyor belt shown above. Due to the pointer 254, this program can only be programmed in a instruction list or the ladder diagram of the GX Developer. During error free operation the program jumps to the jump destination P0. If X4 is set, the conveyor belt is advanced and Y0 is set for failure check. If X5 is set, the conveyor belt is retracted and Y0 is reset. The timer T0 watches the duty cycle time. If the cycle time is exceeded, M0 is set via the CHK instruction and the error code is stored in D0. The program execution is proceeded for further failure check at the jump destination P1 (step 35).



The operations of the CHK instruction can be illustrated through the following ladder diagram, of which the functions are similar to the execution of the CHK instruction. For better comprehensibility of the program example above, the contacts X0, X1, and Y0 are applied directly for the specification of the check conditions. Depending on the program they can be replaced by any other contact numbers.



The following fault conditions may result:

Condition 1: Both limit switches are actuated while the conveyor belt is advanced.

Condition 2: Both limit switches are actuated while the conveyor belt is retracted.

Condition 3: Advance command for set advance limit switch.

Condition 4: Retract command for set retract limit switch.

Condition 5: Advance command for reset retract limit switch.

Condition 6: Retract command for reset advance limit switch.

The error code number stored in D0 corresponds to the fault condition number above.

The CHK instruction performs failure check following the circuit pattern illustrated above. The circuit pattern cannot be changed.

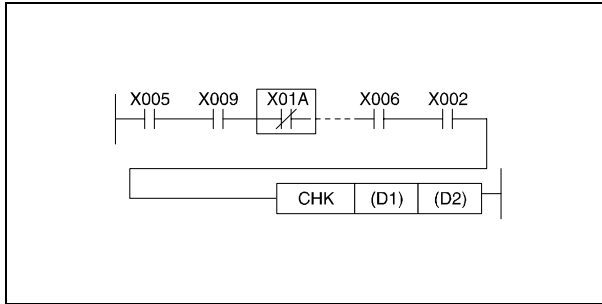
The devices in d1 and d2 must be reset after execution of the CHK instruction, since they retain their conditions after being set via the CHK instruction. If these devices remain set, the CHK instruction cannot be executed again.

The pointer P254 must always be specified as jump destination in the head of the CHK instruction. This pointer indicates the begin of a failure check.

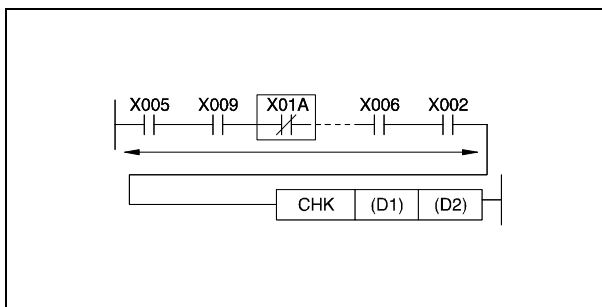
The CHK instruction can be written to any desired step in the sequence program. However, it can only be programmed once within one program.

The CHK instruction cannot be written in the RUN operation mode of the CPU.

The check conditions have to be set via the LD or AND instruction prior to the CHK instruction. Other contact commands cannot set the check condition. If the ANI instruction is applied to set the check condition, the failure check will not be processed.



The failure check is performed in the order of input contact numbers that are specified as check variables. If more than one error is detected, only the error code with the higher priority is stored.





The error code stored in d2 depends on the stored fault condition and on the contact number:

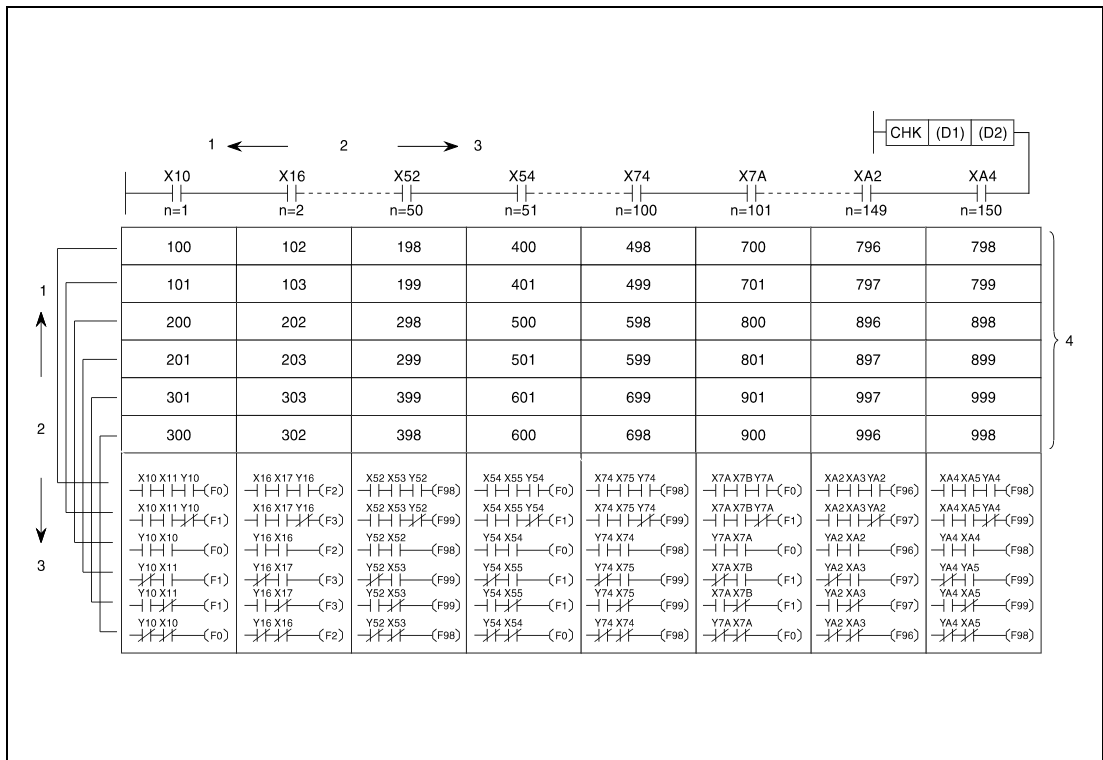
Fault condition number	Input contact number 1 to 50	Input contact number 51 to 100	Input contact number 101 to 150
1 (error code 1)	$100 + (2 \times (\text{contact No.} - 1))$	$400 + (2 \times (\text{contact No.} - 1))$	$700 + (2 \times (\text{contact No.} - 1))$
2 (error code 2)	$101 + (2 \times (\text{contact No.} - 1))$	$401 + (2 \times (\text{contact No.} - 1))$	$701 + (2 \times (\text{contact No.} - 1))$
3 (error code 3)	$200 + (2 \times (\text{contact No.} - 1))$	$500 + (2 \times (\text{contact No.} - 1))$	$800 + (2 \times (\text{contact No.} - 1))$
4 (error code 4)	$201 + (2 \times (\text{contact No.} - 1))$	$501 + (2 \times (\text{contact No.} - 1))$	$801 + (2 \times (\text{contact No.} - 1))$
5 (error code 5)	$300 + (2 \times (\text{contact No.} - 1))$	$600 + (2 \times (\text{contact No.} - 1))$	$900 + (2 \times (\text{contact No.} - 1))$
6 (error code 6)	$301 + (2 \times (\text{contact No.} - 1))$	$601 + (2 \times (\text{contact No.} - 1))$	$901 + (2 \times (\text{contact No.} - 1))$

- <sup>1</sup> Contact number 1
- <sup>2</sup> Contact number 50
- <sup>3</sup> Contact number 51
- <sup>4</sup> Contact number 100
- <sup>5</sup> Contact number 101
- <sup>6</sup> Contact number 150

The error code numbers displayed after the execution of the CHK instruction indicate the kind of error occurred. Prepare a troubleshooting table corresponding to the system for quick remedies.

Error code No.	Cause	Corrective action
301	Conveyor 1: Retract run occurred when the advance limit switch was not actuated	- Check limit switch X1 - Check conveyor
302	Conveyor 1: ...	...
...	...	...

Overview of error code numbers

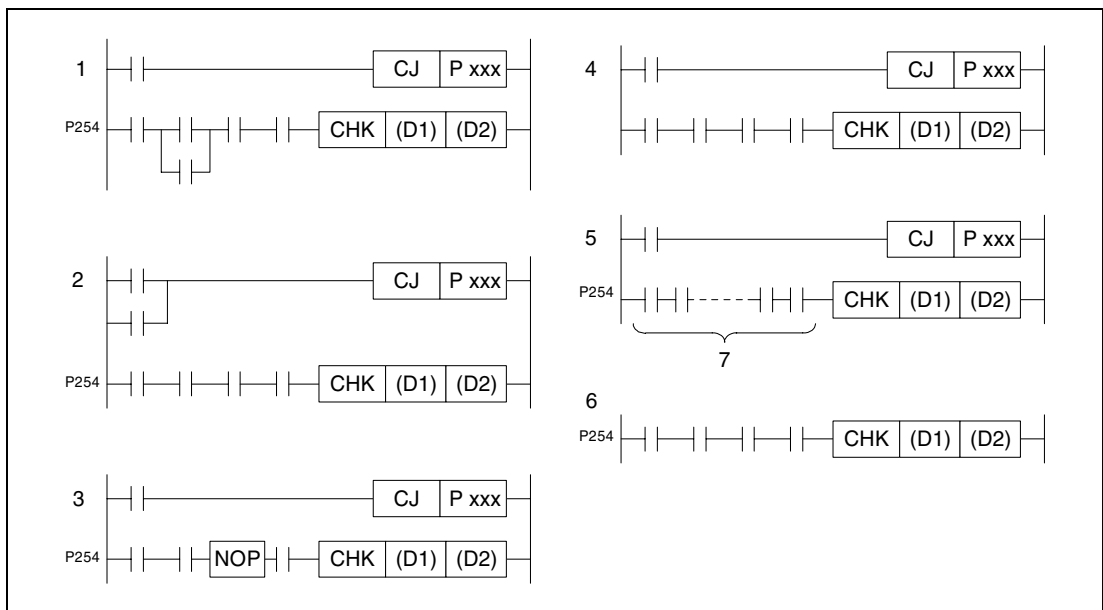


- 1 high priority
- 2 priority
- 3 low priority
- 4 error code number

**Operation Errors**

In the following cases an operation error occurs and the error flag is set (the numbers in brackets refer to the following diagrams):

- Two input contacts are connected in parallel in the check conditions (1) or in the head of the CJ instruction (2).
- A NOP instruction is programmed within the check conditions of the CHK instruction (3).
- The jump destination P254 does not exist in the program (4).
- The check conditions of the CHK instruction contain more than 150 input devices (5).
- There is no jump instruction (CJ) prior to the CHK instruction (CJ)(6).



<sup>7</sup> More than 150 input contacts

7.10.3 CHKCIR, CHKEND

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

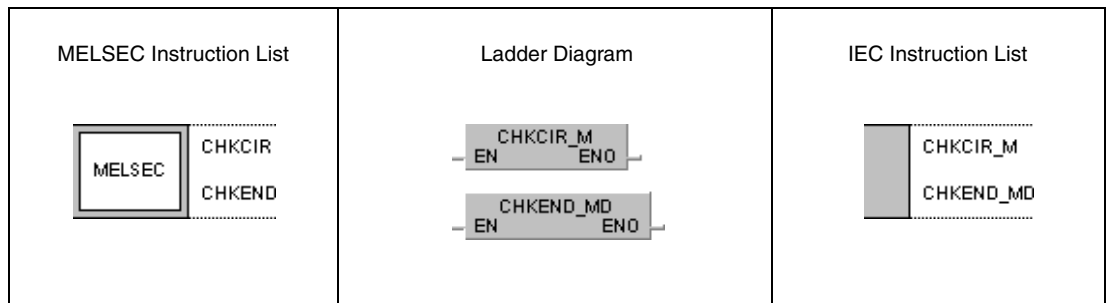
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions (CHKEND only).

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

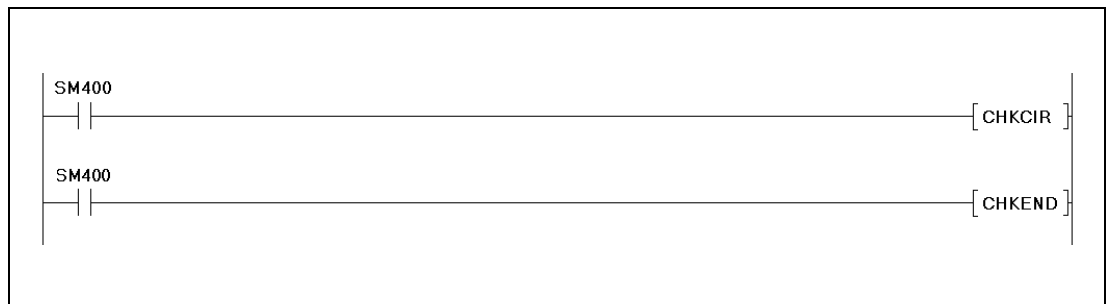
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	SM0	1

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
—	—	—

**Functions**      **Generating check circuits for the CHK instruction**

**CHKCIR, CHKEND**      **Start and end instructions for a program part with generated check circuits.**

The CHKCIR and CHKEND instructions alter check circuits for the CHK instruction. Any required check format can be generated. The actual failure check is performed via the CHKST and CHK instructions.

The failure check is executed via the error check circuits programmed between the CHK and the CHKEND instruction.

**NOTE**

*If the check circuit format for the CHK instruction was altered via the CHKCIR and CHKEND instructions, connected peripheral devices have to be started up in "General Mode", and a program expansion has to be performed.*

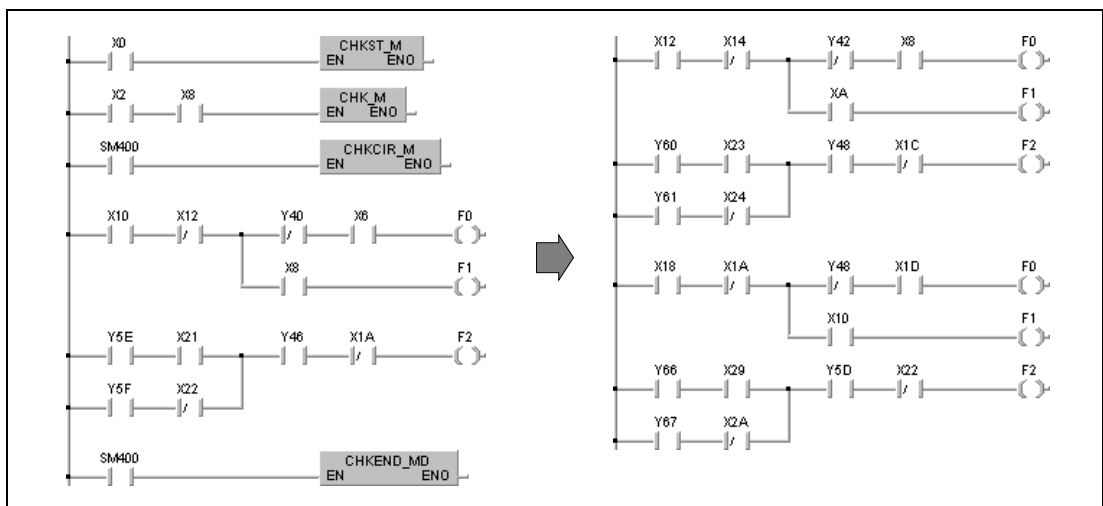
*In cases where a peripheral device is started up by a Q2A, Q2AS, Q3A or Q4A CPU and an attempt was made to generate altered error check circuits for the CHK instruction via CHKCIR and CHKEND instructions, accurate processing cannot be ensured.*

From the error check circuits between the CHKCIR and CHKEND instructions altered error check circuits are generated through index qualification. The error check circuits programmed between these instructions can be assigned 9 annunciators (F1 - F9). Index qualification is performed through the addition of contact numbers designated prior to the CHK instruction and contact numbers of the error check circuits. For example, the contact X10 in the error check circuits shown below will be assigned X12 and X18 in the index qualified check circuits due to the contacts X2 and X8, programmed prior to the CHK instruction.

The error check algorithm depends on the status of the special relay SM710 as follows:

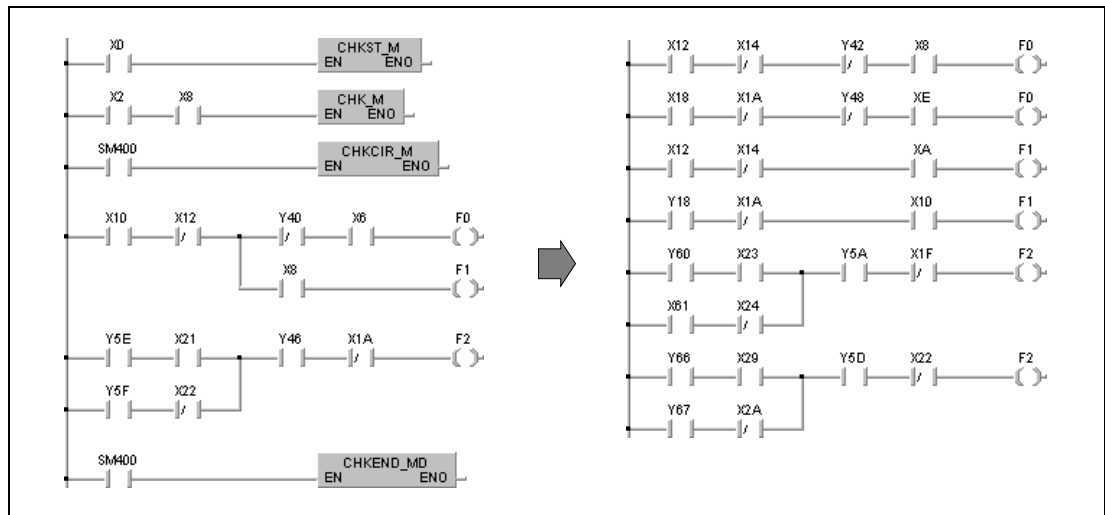
SM710 is reset (0):

First in this case, each contact number in the error check circuit programmed between the CHKCIR and CHKEND instruction is index qualified with the first contact number designated prior to the CHK instruction. Then, each programmed check circuit is index qualified again with the second contact number designated prior to the CHK instruction. This operation is completed as for any programmed check circuit with assigned annunciator (F) a total of new check circuits equivalent to the number of input contacts of the CHK instruction exists.



SM710 is set (1):

First in this case, the first programmed error check circuit with assigned annunciator is index qualified with all contact numbers programmed prior to the CHK instruction. Then, the following check circuit is index qualified with all contact numbers programmed prior to the CHK instruction. This operation is completed as for any programmed check circuit with assigned annunciator (F) a total of new check circuits equivalent to the number of input contacts of the CHK instruction exists.



During error check of the index qualified error check circuits, the outputs (F) that can only be set via the OUT F instruction are checked for their status. If an output (F) is set, the special relay SM80 is set. The error code consisting of contact number and error check circuit (F1 - F9) is stored in special register SD80 in BCD data format.

The error check circuits between the CHKCIR and CHKEND instruction can be programmed with the following instructions:

Contacts:

LD, LDI, AND, ANI, OR, ORI, ANB, ORB, MPS, MPP, MRD, comparison operation instructions.

Coils:

OUT F

The inputs X and outputs Y have to be programmed as devices for the contacts.

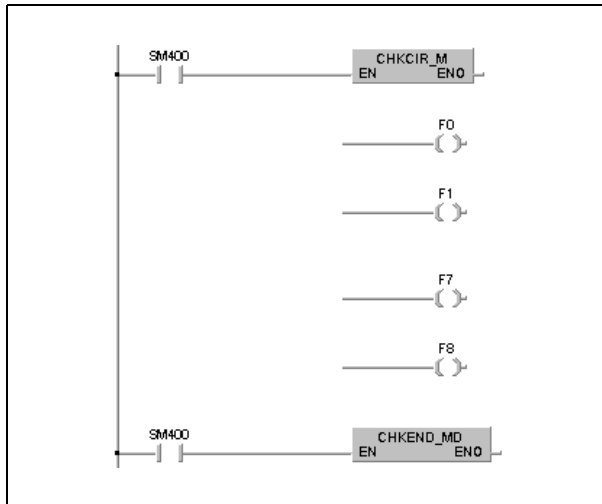
Only annunciators (F) can be programmed as outputs of error check circuits. The error check circuits can be specified any random designation from F0 on, since these outputs are processed as dummy contacts. For this reason, no errors occur with annunciators (F) overlapping.

The status of annunciators (F) can even be checked accurately, if one annunciator (F) is programmed twice beyond the CHK instruction, because both of these annunciator functions are processed separately.

Since the status (0/1) of annunciators (F) applied by the CHK instruction is not updated, the annunciators even remain reset, if they are monitored by a peripheral device.

The error check circuits programmed between the CHKCIR and CHKEND instructions can be created with maximum 256 program steps (contact branches) and 9 outputs (annunciators F1-F9) addressed by OUT F instructions.

The error check circuits between the CHKCIR and CHKEND instructions are designated from top error check circuit 1 (F0) to bottom error check circuit 9 (F8).



The CHKCIR and CHKEND instructions can be programmed at any program step of the sequence program. In total, these instructions may only exist twice in all program files to be executed and once within one program file.

The CHKCIR and CHKEND instructions cannot be applied in low-speed programs, otherwise an operation error occurs and the CPU terminates processing.

### Operation Errors

In the following cases an operation error occurs and the error flag is set:

- The CHKCIR and CHKEND instructions appear more than twice in all program files (error code 4235).
- The CHKCIR and CHKEND instructions appear more than once within one program file (error code 4235).
- The CHKEND instruction is not executed after the CHKCIR instruction (error code 4230).
- The CHKEND instruction is executed without a preceding CHKCIR instruction (error code 4230).
- The CHKCIR and CHKEND instructions are programmed in a low-speed program (error code 4235).
- More than 9 annunciators (F) (error check circuits) are addressed (error code 4235).
- The created error check circuits contain more than 256 program steps (contact branches) (error code 4235).
- The error check circuits contain invalid devices (error code 4235).
- The error check circuits contain devices already index qualified (error code 4235).

**NOTE**

The following errors occurring during program expansion at a peripheral device prevent the program expansion from execution:

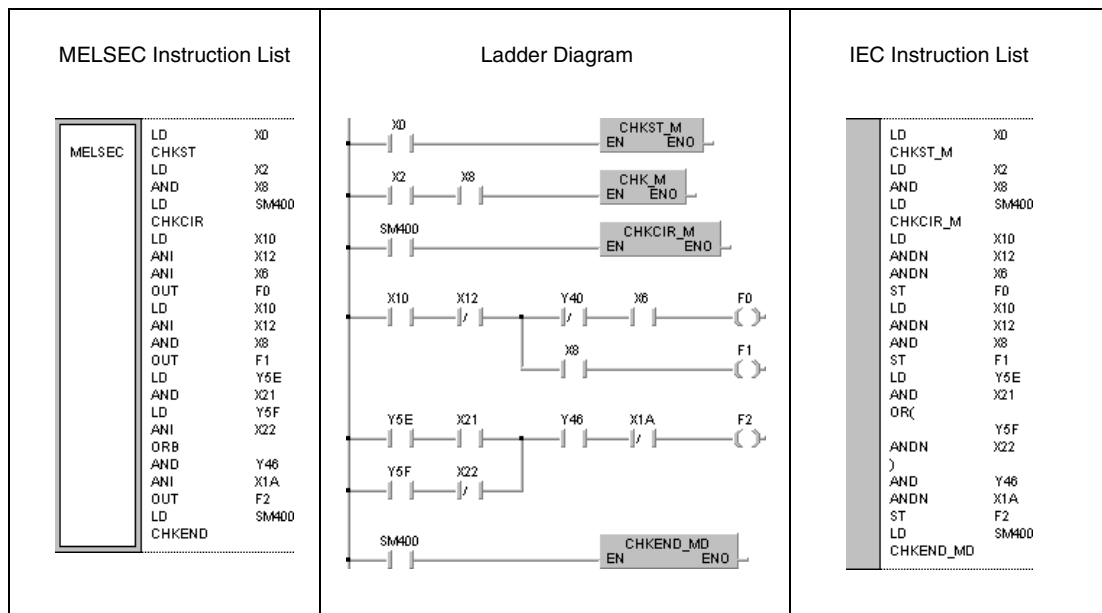
- The error check circuits contain invalid devices.
- The error check circuits contain devices already index qualified.

Correct the error check circuits accordingly, if any of the errors above occur.

**Program Example**

CHKCIR, CHKEND

The following program creates index qualified error check circuits. The operations of this program are illustrated under the topic "functions". In addition, the MELSEC and IEC instruction lists are shown below.





7.10.4 SLT, SLTR

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	● <sup>1</sup>	●	●	●	

<sup>1</sup> Except A1N CPU.

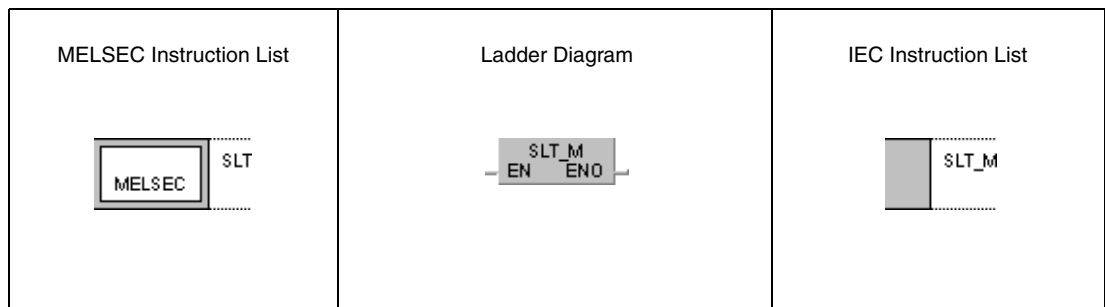
Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag	
Bit Devices					Word Devices (16-bit)						Constant		Pointer		Level							
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I
																		1				

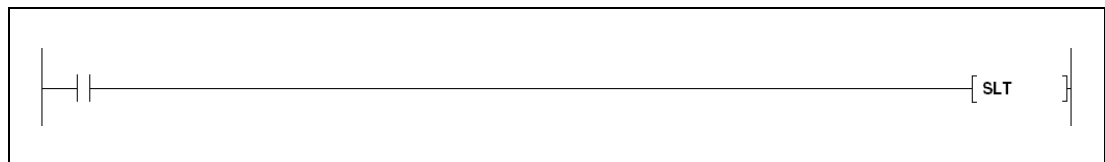
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
—	—	—

**Functions     Setting and resetting status latch**

**SLT     Set status latch**

Although the program execution is monitored by the GX IEC Developer, not any status of devices can be transmitted and displayed. For this purpose the CPU supplies a special status memory area (status latch). The status latch memory is set via parameter settings and stores the data of one program scan (refer to the manuals of the GX Developer for further details).

The SLT instruction executes the temporary storage of specified device data. The data are stored in the status latch memory and can be checked and displayed.

The SLT instruction can only be executed once within one program scan. For another execution of the SLT instruction, it has to be reset (re-enabled) via the SLTR instruction.

**SLTR     Reset status latch**

The SLTR instruction clears the data temporarily stored in the status latch area, and resets (re-enables) the SLT instruction.

The SLT instruction can only be executed once within one program scan. For another execution of the SLT instruction, it has to be reset (re-enabled) via the SLTR instruction.

**NOTE**

***Please check, whether these functions are available and supported by your version of the GX IEC Developer.***

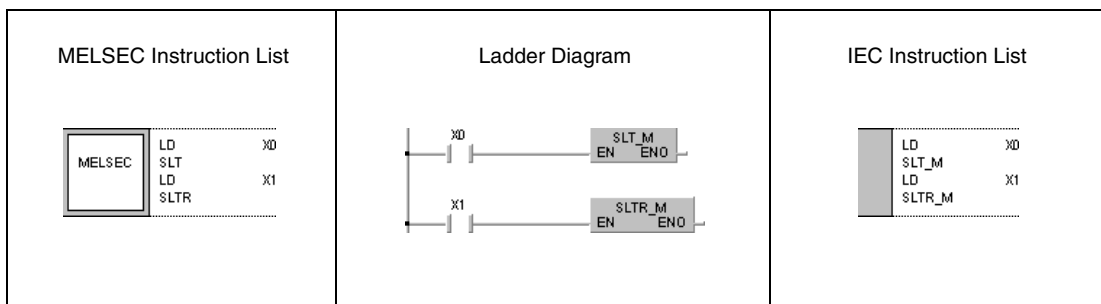
*Refer to the user’s manuals for the CPUs and the GX Developer for further details on status latch operations.*

*The execution of the SLT instruction increases the program scan time depending on the CPU type. The setting value of the watch dog timer has to be set according to the increased program scan time. Refer to the user’s manual of the according CPU for the amount of time increased.*

**Program Example**

**SLT/SLTR**

While X0 is set, the following program executes the SLT instruction. While X1 is set, the SLTR instruction resets the SLT instruction.



### 7.10.5 STRA, STRAR

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	

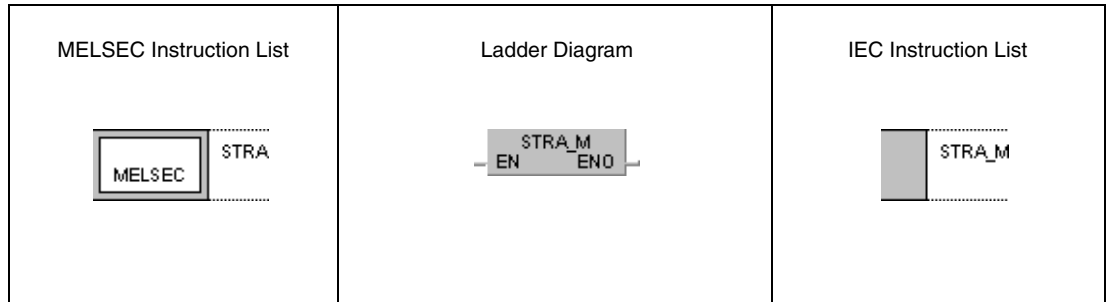
**Devices  
MELSEC A**

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag				
Bit Devices							Word Devices (16-bit)						Constant	Pointer	Level										
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I	N	M9012	M9010 M9011
																							1		

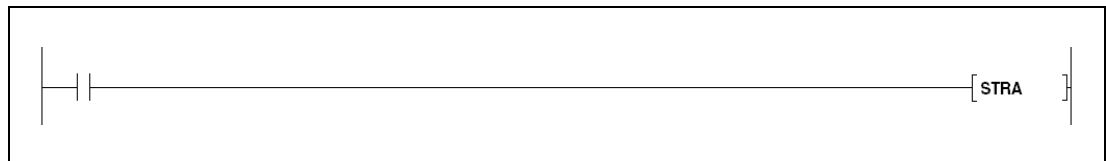
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions**      **Setting and resetting sampling trace**

**STRA**    **Set sampling trace**

The sampling trace monitors the data and status of specified devices for a specified period of time and stores the cumulative data of the traced devices in a separate storage area. The selection of devices and the trace period are specified via parameters.

**STRAR**    **Reset sampling trace**

The STRAR instruction clears the data from the sampling trace program file, and resets the STRA instruction and the special relay M9043 (A series) or the special relay SM801 - SM805 (Q series and System Q) respectively.

The STRA instruction can only be executed once again after the execution of the STRAR instruction.

**NOTE**

**Please check, whether these functions are available and supported by your version of the GX IEC Developer.**

Refer to the user's manuals for the CPUs and the GX Developer for further details on sampling trace operations.

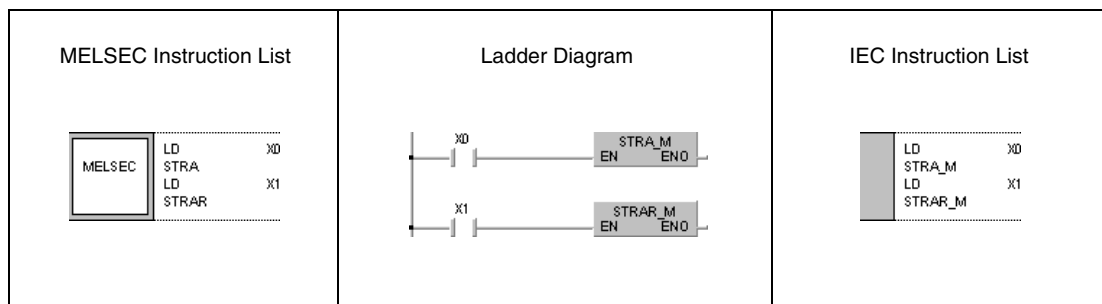
The execution of the SLT instruction increases the program scan time depending on the CPU type. The setting value of the watch dog timer has to be set according to the increased program scan time. Refer to the user's manual of the according CPU for the amount of time increased.

While accessing a ROM, the STRA or STRAR instruction cannot be executed (A series only).

**Program Example**

**STRA/STRAR**

While X0 is set, the following program executes an STRA instruction. While X1 is set, the STRAR instruction resets the STRA instruction.



**7.10.6 PTRA, PTRAR, PTRAEXE, PTRAEEXP**

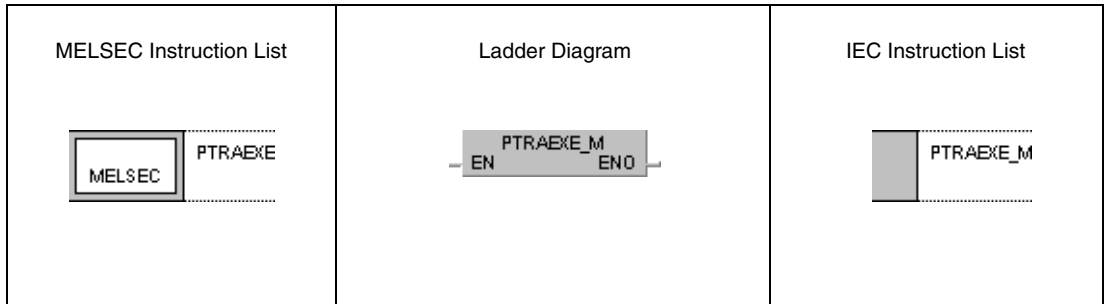
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

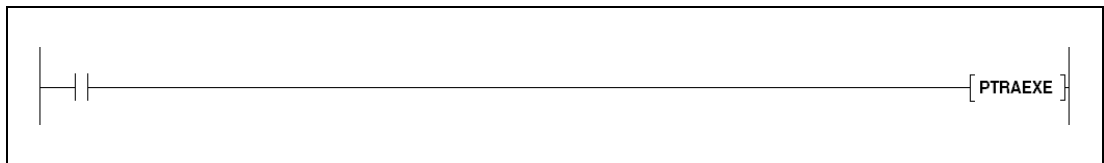
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
—	—	—

**Functions**      **Setting, resetting, and executing program trace****PTRA**              **Set program trace**

The program trace monitors the data and status of devices specified by programs for a specified period of time and stores the cumulative data of the traced programs in a separate storage area.

The PTRA instruction enables tracing programs for a specified number of trace scans and storing the data temporarily in a separate storage area of the CPU for the program trace operation. The PTRAEXE instruction starts the program trace execution. The special relays SM810 - SM812 require to be set (1) for data storage.

On execution of the PTRA instruction the special relay SM813 is set. After execution of the specified number of trace scans the data is stored for further processing and the program trace is terminated.

If the special relay SM811 is reset during program trace, the trace operation is terminated.

After the execution of the PTRA instruction is completed, the special relay SM815 is set.

Before the PTRA instruction can be executed once again, the PTRAR instruction has to be executed.

The results of the program trace operation can be monitored by a peripheral device.

**PTRAR**              **Reset program trace**

The PTRAR instruction clears the data from the program trace program file, and resets the PTRA instruction and the special relays SM811– SM815.

The PTRA instruction can only be executed once again after the execution of the PTRAR instruction.

**PTRAEXE**          **Execute program trace**

The PTRAEXE instruction starts the program scan execution.

If the special relay SM811 is reset during program trace, the trace operation is terminated.

If the execution condition for the PTRAEXE instruction is not set, program trace will not be executed.

**NOTE**

***Please check, whether these functions are available and supported by your version of the GX Developer.***

*Refer to the user's manuals for the CPUs and the GX Developer for further details on program trace operations.*

## 7.11 Character string processing instructions

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Conversion of 16-/32-bit binary data into decimal values in ASCII code	BINDA	BINDA_MD
		BINDA_K_MD
		BINDA_S_MD
	BINDAP	BINDA_P_MD
		BINDA_K_P_MD
		BINDA_P_S_MD
	DBINDA	DBINDA_MD
		DBINDA_K_P_MD
		DBINDA_P_S_MD
	DBINDAP	DBINDA_P_MD
		DBINDA_K_P_MD
		DBINDA_P_S_MD
Conversion of BIN 16-/32-bit binary data into ASCII code	BINHA	BINHA_MD
		BINHA_K_MD
		BINHA_S_MD
	BINHAP	BINHA_P_MD
		BINHA_K_P_MD
		BINHA_P_S_MD
	DBINHA	DBINHA_MD
		DBINHA_K_MD
		DBINHA_S_MD
	DBINHAP	DBINHA_P_MD
		DBINHA_K_P_MD
		DBINHA_P_S_MD
Conversion of 4-/8-digit BCD data into ASCII code	BCDDA	BCDDA_MD
		BCDDA_K_MD
		BCDDA_S_MD
	BCDDAP	BCDDA_P_MD
		BCDDA_K_P_MD
		BCDDA_P_S_MD
	DBCDDA	DBCDDA_MD
		DBCDDA_K_MD
		DBCDDA_S_MD
	DBCDDAP	DBCDDA_P_MD
		DBCDDA_K_P_MD
		DBCDDA_P_S_MD

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Conversion of decimal ASCII data into BIN 16-/32-bit binary data	DABIN	DABIN_MD
		DABIN_S_MD
	DABINP	DABIN_P_MD
		DABIN_P_S_MD
	DDABIN	DDABIN_MD
		DDABIN_S_MD
	DDABINP	DDABIN_P_MD
		DDABIN_P_S_MD
Conversion of hexadecimal ASCII data into BIN 16-/32-bit binary data	HABIN	HABIN_MD
		HABIN_S_MD
	HABINP	HABIN_P_MD
		HABIN_P_S_MD
	DHABIN	DHABIN_MD
		DHABIN_S_MD
	DHABINP	DHABIN_P_MD
		DHABIN_P_S_MD
Conversion of decimal ASCII data into 4-/8-digit BCD data	DABCD	DABCD_MD
		DABCD_S_MD
	DABCDP	DABCD_P_MD
		DABCD_P_S_MD
	DDABCD	DDABCD_MD
		DDABCD_S_MD
	DDABCDP	DDABCD_P_MD
		DDABCD_P_S_MD
Read-out of comment data	COMRD	COMRD_MD
		COMRD_S_MD
	COMRDP	COMRD_P_MD
		COMRD_P_S_MD
Detection of character string length	LEN	LEN_E
		LEN_MD
		LEN_S_MD
	LENP	LEN_P_S_MD
Conversion of BIN 16-/32-bit binary data into character string data	STR	STR_MD
		STR_K_MD
		STR_S_MD
	STRP	STR_P_MD
		STR_K_P_MD
		STR_P_S_MD
	DSTR	DSTR_MD
		DSTR_K_MD
		DSTR_S_MD
	DSTRP	DSTR_P_MD
DSTR_K_P_MD		
DSTR_P_S_MD		



Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Conversion of character string data into BIN 16-/32-bit binary data	VAL	VAL_MD
		VAL_S_MD
	VALP	VAL_P_MD
		VAL_P_S_MD
	DVAL	DVAL_MD
		DVAL_S_MD
DVALP	DVAL_P_MD	
	DVAL_P_S_MD	
Conversion of floating point data into character string data	ESTR	ESTR_M
	ESTRP	ESTRP_M
Conversion of character string data into decimal floating point data	EVAL	EVAL_M
	EVALP	EVALP_M
Conversion of alphanumerical character strings into ASCII code	ASC	ASC_MD
		ASC_K_MD
		ASC_S_MD
	ASCP	ASC_P_MD
		ASC_P_S_MD
		ASC_K_P_MD
Conversion of hexadecimal ASCII values into binary values	HEX	HEX_S_MD
		HEX_MD
		HEX_K_MD
	HEXP	HEX_P_S_MD
		HEX_P_MD
		HEX_K_P_MD
Extraction of character string data (right part of character string)	RIGHT	RIGHT_M
		RIGHT
		RIGHT_E
	RIGHTP	RIGHTP_M
Extraction of character string data (left part of character string)	LEFT	LEFT_M
		LEFT
		LEFT_E
	LEFTP	LEFTP_M
Random extraction of parts from character strings	MIDR	MIDR_M
	MIDRP	MIDRP_M
Selecting and moving parts of character strings into a character string	MIDW	MIDW_M
	MIDWP	MIDWP_M
Search for character strings	INSTR	INSTR_M
	INSTRP	INSTRP_M
Floating point data conversion with BCD representation	EMOD	EMOD_M
	EMODP	EMODP_M
BCD data conversion with decimal floating point format	EREXP	EREXP_M
	EREXPP	EREXPP_M

**7.11.1 BINDA, BINDAP, DBINDA, DBINDAP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

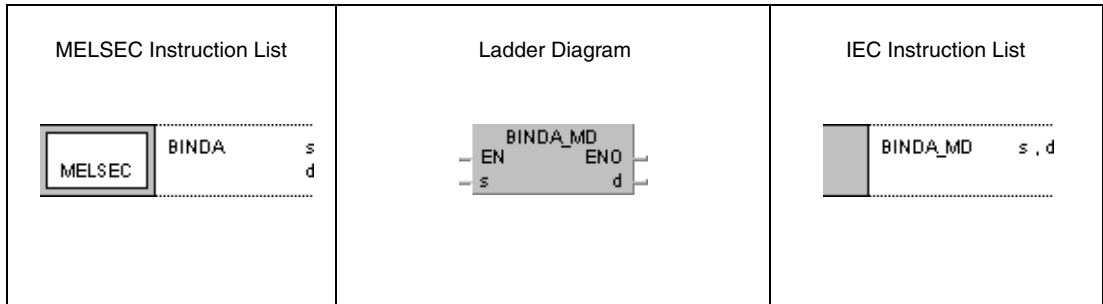
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	3	
d	—	●	●	—	—	—	—	—	—		

**GX IEC Developer**



**GX Developer**



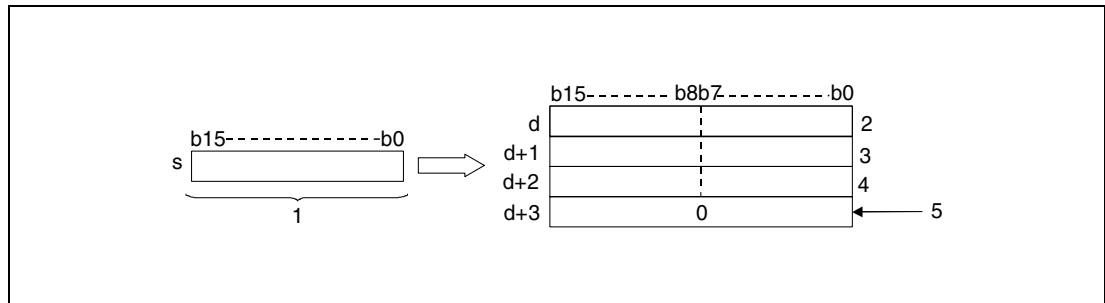
**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	Binary data to be converted into ASCII format.	BIN 16-/32-bit	ANY16/32
d	First number of device storing the conversion result.	Character string	Array [1..4]/ [1..6] of ANY16

**Functions Conversion of 16-/32-bit binary data into decimal values in ASCII code**

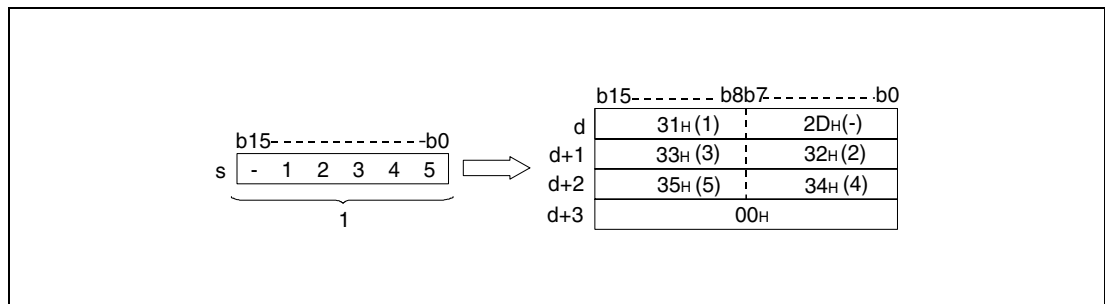
**BINDA Conversion of 16-bit binary data**

The BINDA instruction converts a 16-bit binary value specified by s into a decimal value in ASCII code and stores it in the device specified in d (Array\_d[1]) through d+3 (Array\_d[4]).



- <sup>1</sup> 16-bit binary data
- <sup>2</sup> Digit of tenths in ASCII code/ sign character
- <sup>3</sup> Digit of hundreds in ASCII code/ digit of thousands in ASCII code
- <sup>4</sup> Digit of ones in ASCII code/ digit of tens in ASCII code
- <sup>5</sup> With the relay SM701 not set

The value specified by s is stored as decimal value in ASCII code beginning from d (Array\_d[1]) through d+3 (Array\_d[4]).



- <sup>1</sup> Binary value

The 16-bit binary value may range from -32768 to 32767.

The results of the conversion operations are stored in d as follows:

- If the 16-bit binary value is positive, the sign character is stored as "20H".
- If the 16-bit binary value is negative, the sign character is stored as "2DH".

The stored sign character "20H" replaces the preceding zeroes.

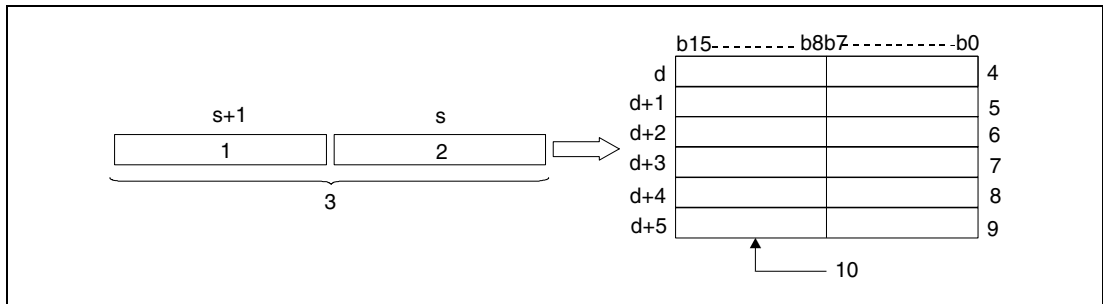
For the value 00325 the zeroes of the digits of tenths and thousands are replaced by "20H" so that only the actually required digits are stored.

The storage of data in the device specified by d+3 (Array\_d[4]) depends on the status of the relay SM701.

- If the relay is not set, a zero "00H" is stored in the area d+3 (Array\_d[4]).
- If the relay is set, the value in d+3 (Array\_d[4]) remains unchanged.

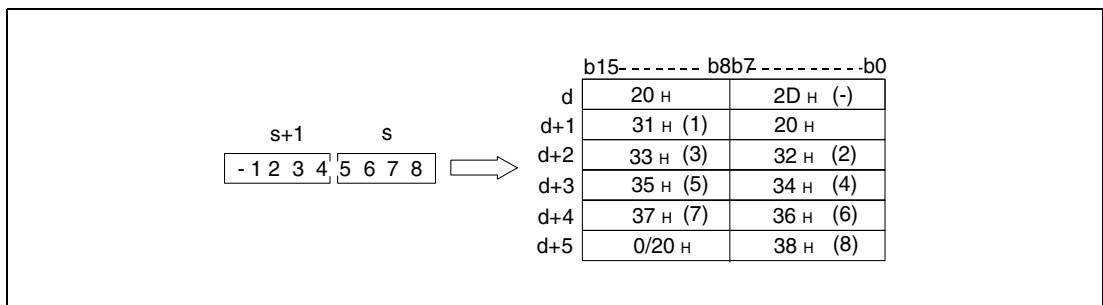
**DBINDA Conversion of 32-bit binary data**

The DBINDA instruction converts 32-bit binary data specified by s and s+1 into a decimal value in ASCII code and stores it in the device specified in d (Array\_d[1]) through d+5 (Array\_d[6]).



- <sup>1</sup> Upper 16 bits
- <sup>2</sup> Lower 16 bits
- <sup>3</sup> 32-bit binary data
- <sup>4</sup> Sign character/ digit of billions in ASCII code
- <sup>5</sup> Digit of ten millions/ digit of one hundred millions in ASCII code
- <sup>6</sup> Digit of one hundred thousands/ digit of millions in ASCII code
- <sup>7</sup> Digit of thousands/ digit of ten thousands in ASCII code
- <sup>8</sup> Digit of tens/ digit of hundreds in ASCII code
- <sup>9</sup> 0 or 20H/ digit of ones in ASCII code
- <sup>10</sup> With the relay SM701 not set (0)/ with the relay SM701 set (20H)

The value specified by s and s+1 is stored beginning from d (Array\_d[1]) through d+5 (Array\_d[6]) as decimal value in ASCII code.



The 32-bit binary value specified by s may range from -2147483648 to 2147483647.  
 The results of the conversion operation are stored in d (Array\_d[1]) through d+5 (Array\_d[6]) as follows:  
 If the binary value is positive, the sign character is stored as "20H".  
 If the binary value is negative, the sign character is stored as "2DH".  
 The stored sign character "20H" replaces the preceding zeroes.  
 For the value 0012034560 the zeroes of the digits of billions and hundred millions are replaced by "20H" so that only the actually required digits are stored.

The storage of data in the upper 8 bits of the device specified by d+5 (Array\_d[6]) depends on the status of the relay SM701.

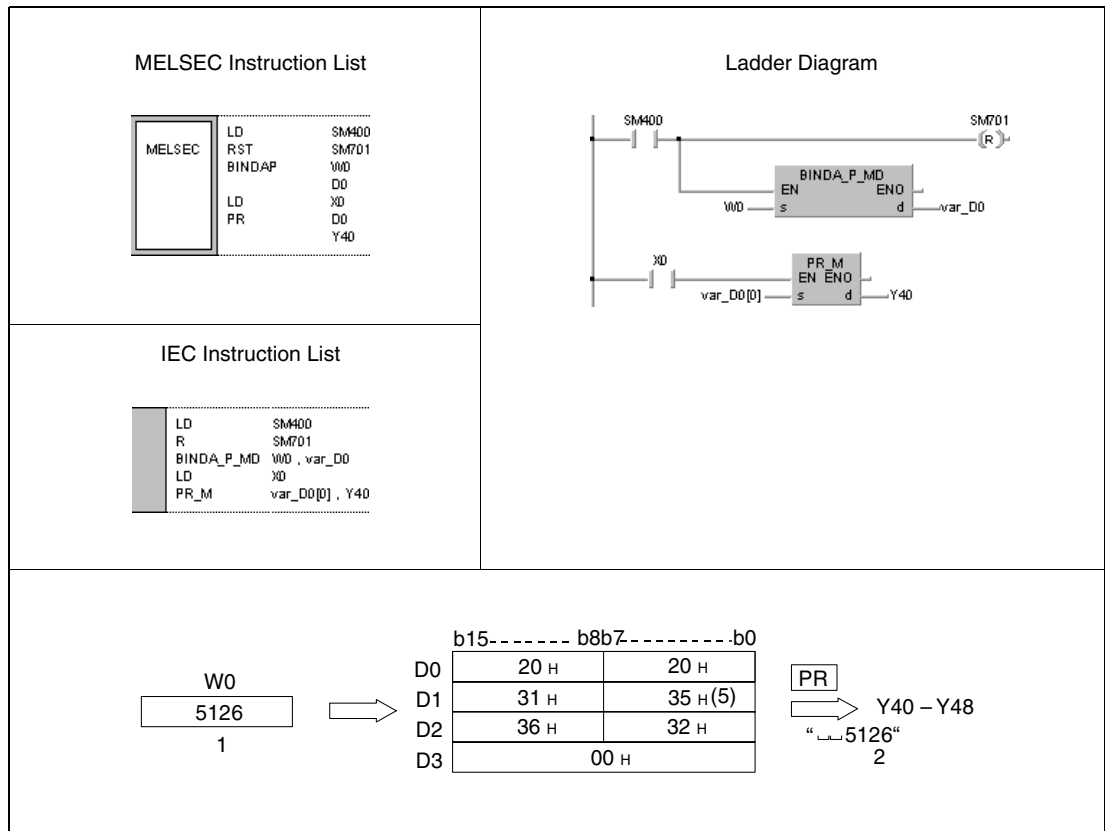
If this relay is not set, a zero "00H" is stored in the area d+5 (Array\_d[6]).

If this relay is set, a space character (20H) is stored in the area d+5 (Array\_d[6]).

**Program Example 1**

**BINDAP**

With leading edge from SM400, the following program outputs the value of the 16-bit binary data in W0 as decimal value in ASCII code via the BINDAP instruction. The PR instruction outputs the characters at Y40 through Y48.



<sup>1</sup> Binary value

<sup>2</sup> Output

**Program Example 2** DBINDAP

With leading edge from SM400, the following program outputs the value of the 32-bit binary data in W10 and W11 as decimal value in ASCII code via the DBINDAP instruction. The PR instruction outputs the characters at Y40 through Y48.

<p style="text-align: center;"><b>MELSEC Instruction List</b></p> <pre style="font-family: monospace; border: 1px solid black; padding: 5px;"> MELSEC LD      SM400         RST     SM701         DBINDAP W10         D0         LD      X0         PR      D0                 Y40                 </pre>	<p style="text-align: center;"><b>Ladder Diagram</b></p>																																					
<p style="text-align: center;"><b>IEC Instruction List</b></p> <pre style="font-family: monospace; border: 1px solid black; padding: 5px;"> LD      SM400 R       SM701 DBINDA_P_MD var_W10 , var_D0 LD      X0 PR_M    var_D0[0] , Y40                 </pre>																																						
<div style="display: flex; align-items: center; justify-content: space-around;"> <div style="text-align: center;"> <p>W11      W10</p> <table border="1" style="border-collapse: collapse;"> <tr> <td style="padding: 2px 5px;">-</td> <td style="padding: 2px 5px;">3</td> <td style="padding: 2px 5px;">8</td> <td style="padding: 2px 5px;">4</td> <td style="padding: 2px 5px;">2</td> <td style="padding: 2px 5px;">5</td> <td style="padding: 2px 5px;">6</td> <td style="padding: 2px 5px;">3</td> </tr> <tr> <td colspan="8" style="text-align: center; padding: 2px 5px;">1</td> </tr> </table> </div> <div style="font-size: 2em;">⇒</div> <table border="1" style="border-collapse: collapse; text-align: center;"> <thead> <tr> <th></th> <th style="border: none;">b15----- b8b7----- b0</th> <th style="border: none;"></th> </tr> </thead> <tbody> <tr> <td>D0</td> <td style="border: none;">20 H</td> <td style="border: none;">2D H (-)</td> </tr> <tr> <td>D1</td> <td style="border: none;">20 H</td> <td style="border: none;">20 H</td> </tr> <tr> <td>D2</td> <td style="border: none;">38 H (8)</td> <td style="border: none;">33 H (3)</td> </tr> <tr> <td>D3</td> <td style="border: none;">32 H (2)</td> <td style="border: none;">34 H (4)</td> </tr> <tr> <td>D4</td> <td style="border: none;">36 H (6)</td> <td style="border: none;">35 H (5)</td> </tr> <tr> <td>D5</td> <td style="border: none;">00 H</td> <td style="border: none;">33 H (3)</td> </tr> </tbody> </table> <div style="margin-left: 20px;"> <p>PR ⇒ Y40 – Y48</p> <p>“3842563” 2</p> </div> </div>		-	3	8	4	2	5	6	3	1									b15----- b8b7----- b0		D0	20 H	2D H (-)	D1	20 H	20 H	D2	38 H (8)	33 H (3)	D3	32 H (2)	34 H (4)	D4	36 H (6)	35 H (5)	D5	00 H	33 H (3)
-	3	8	4	2	5	6	3																															
1																																						
	b15----- b8b7----- b0																																					
D0	20 H	2D H (-)																																				
D1	20 H	20 H																																				
D2	38 H (8)	33 H (3)																																				
D3	32 H (2)	34 H (4)																																				
D4	36 H (6)	35 H (5)																																				
D5	00 H	33 H (3)																																				

- <sup>1</sup> Output
- <sup>2</sup> Binary value

**NOTE** These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

**7.11.2 BINHA, BINHAP, DBINHA, DBINHAP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

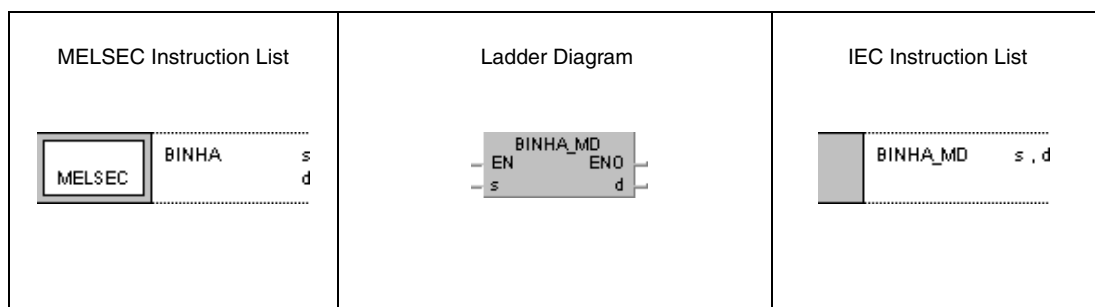
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

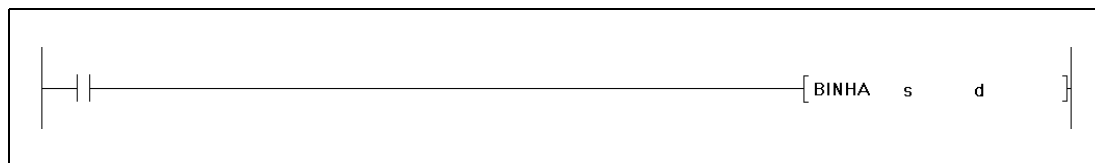
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	3	
d	—	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**



**GX  
Developer**



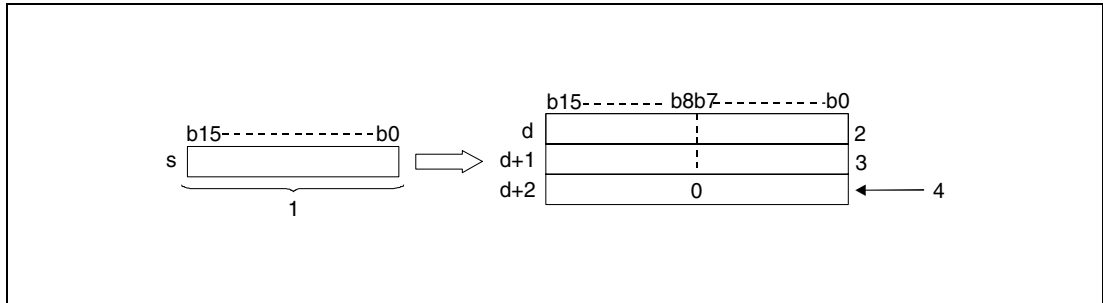
**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	Binary data to be converted into ASCII format.	BIN 16-/32-bit	ANY16/32
d	First number of device storing the conversion result.	Character string	Array [1..3]/ [1..5] of ANY16

**Functions Conversion of 16-/32-bit binary data into hexadecimal values in ASCII code**

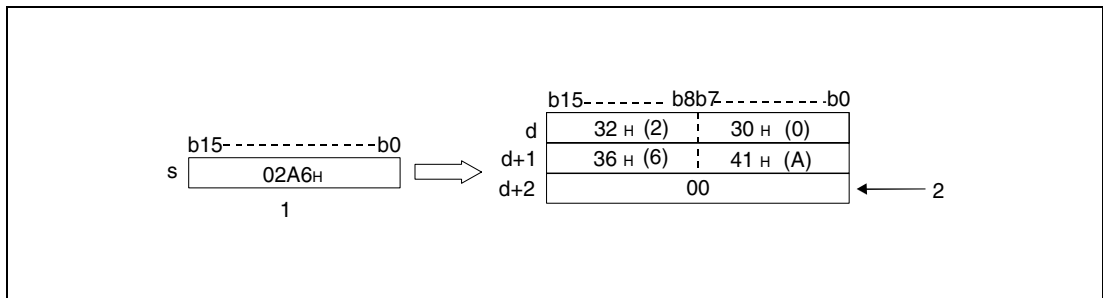
**BINHA Conversion of 16-bit binary data**

The BINHA instruction converts 16-bit binary data specified by s into a hexadecimal value in ASCII code and stores it in the devices specified by d (Array\_d[1]) through d+2 (Array\_d[3]).



- <sup>1</sup> 16-bit binary data
- <sup>2</sup> ASCII code of the 3rd digit/ ASCII code of the 4th digit
- <sup>3</sup> ASCII code of the 1st digit/ ASCII code of the 2nd digit
- <sup>4</sup> With the relay SM701 not set

The value specified by s is stored in ASCII code in d (Array\_d[1]) through d+2 (Array\_d[3]).



- <sup>1</sup> 16 bit binary data
- <sup>2</sup> With the relay SM701 not set

The 16-bit binary data specified by s may range from 0H to FFFFH.

The conversion result is stored as 4-digit hexadecimal value in d (Array\_d[1]) through d+2 (Array\_d[3]).

If one of the digits is 0, this digit is processed as value 0 (zeroes are not suppressed).

The storage of the data in the device specified by d+2 (Array\_d[3]) depends on the status of the relay SM701 as follows:

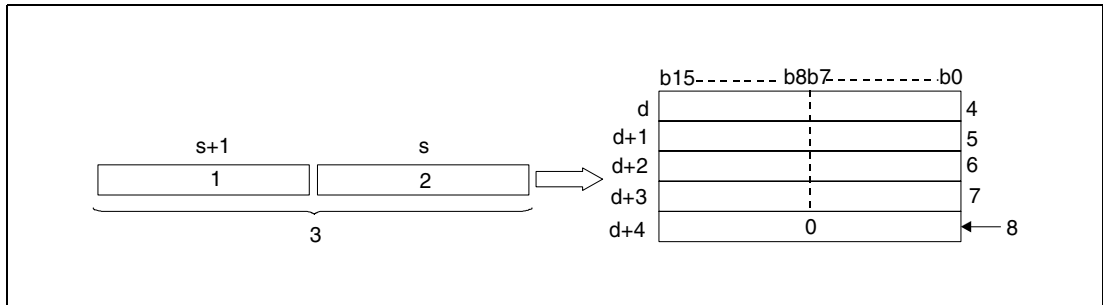
If this relay is not set, a zero "00H" is stored in the area d+2 (Array\_d[3]).

If this relay is set, the value in d+2 (Array\_d[3]) remains unchanged.



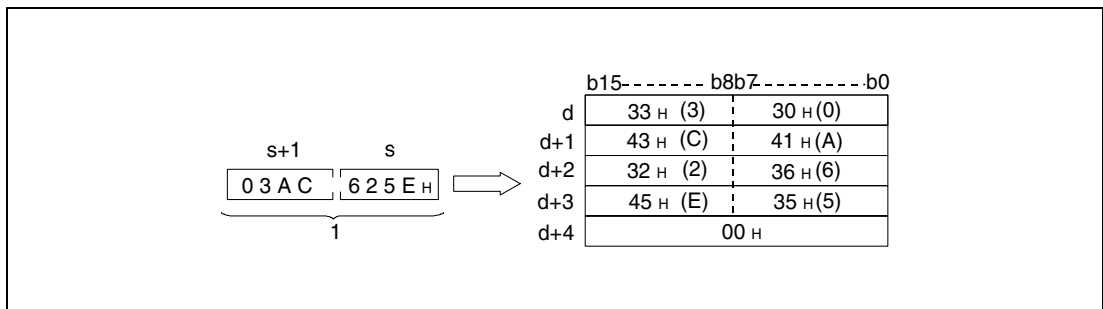
**DBINHA Conversion of 32-bit binary data**

The DBINHA instruction converts 32-bit binary data specified by s and s+1 into a hexadecimal value in ASCII code and stores it in the devices specified by d (Array\_d[1]) through d+4 (Array\_d[5]).



- <sup>1</sup> Upper 8 bits
- <sup>2</sup> Lower 8 bits
- <sup>3</sup> 32-bit binary data
- <sup>4</sup> ASCII code of the 7th digit/ ASCII code fo the 8th digit
- <sup>5</sup> ASCII code of the 5th digit/ ASCII code of the 6th digit
- <sup>6</sup> ASCII code of the 3th digit/ ASCII code of the 4th digit
- <sup>7</sup> ASCII code of the 1st digit/ ASCII code of the 2nd digit
- <sup>8</sup> With the relay SM701 not set

The value "03AC625EH" specified in s and s+1 is stored in d as follows:



- <sup>1</sup> BIN 32-bit data

The 32-bit binary value specified by s and s+1 may range from 0H to FFFFFFFFH.

The conversion result is stored as 8-digit hexadecimal value in d (Array\_d[1]) through d+4 (Array\_d[5]).

If one of the digits is 0, this digit is processed as value 0 (zeroes are not suppressed).

The storage of the data in the device specified by d+4 (Array\_d[5]) depends on the status of the relay SM701 as follows:

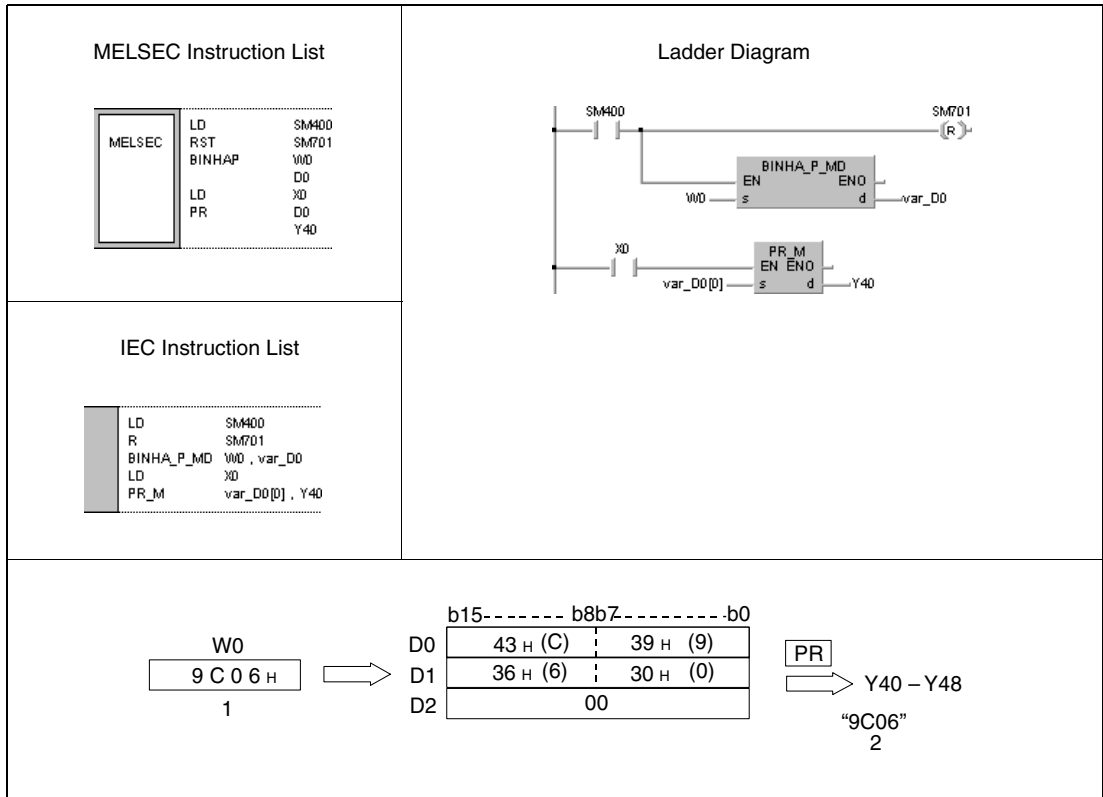
If this relay is not set, a zero "00H" is stored in the area d+4 (Array\_d[5]).

If this relay is set, the value in d+4 (Array\_d[5]) remains unchanged.

**Program Example 1**

**BINHAP**

With leading edge from SM400, the following program outputs the value of the 16-bit binary data in W0 as decimal value in ASCII code via the BINHAP instruction. The PR instruction outputs the characters at Y40 through Y48.

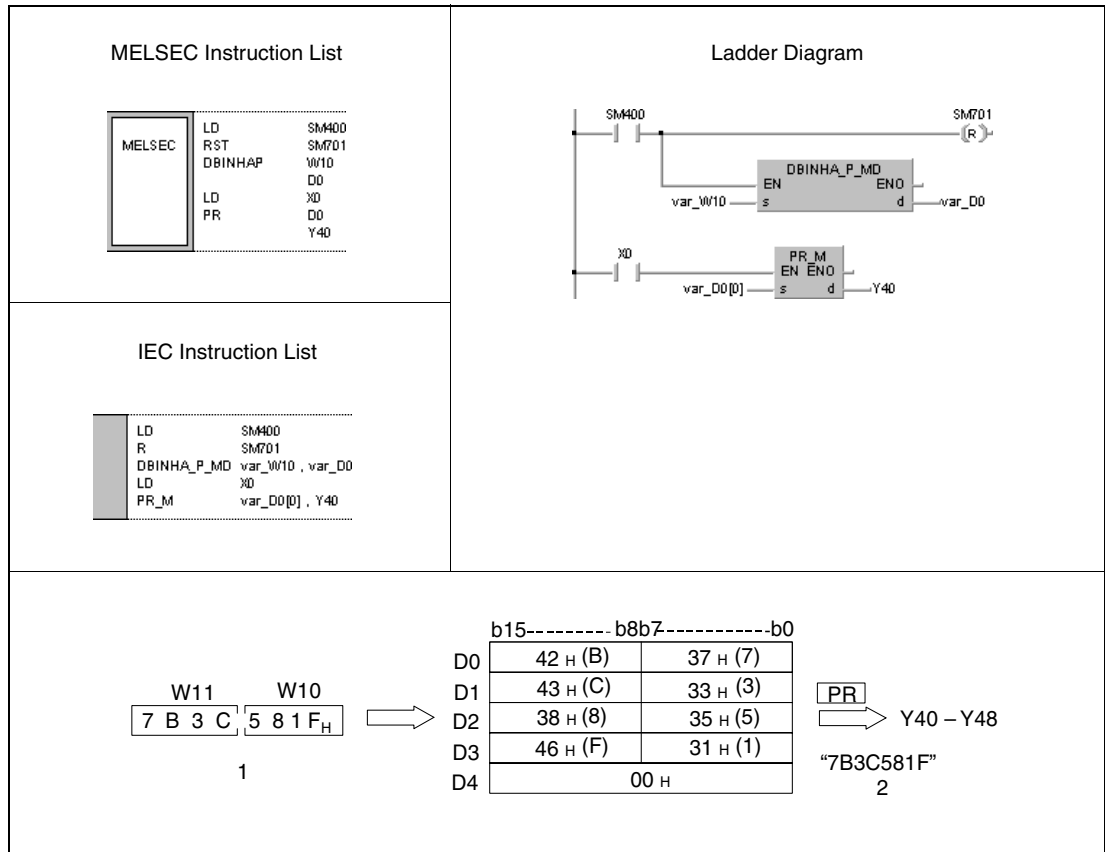


<sup>1</sup> Output

<sup>2</sup> Binary data

**Program Example 2** DBINHAP

With leading edge from SM400, the following program outputs the value of the 32-bit binary data in W10 and W11 via the DBINHAP instruction as decimal value in ASCII code. The PR instruction outputs the characters at Y40 through Y48.



<sup>1</sup> Output  
<sup>2</sup> Binary value

**NOTE** *These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.3 BCDDA, BCDDAP, DBCDDA, DBCDDAP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

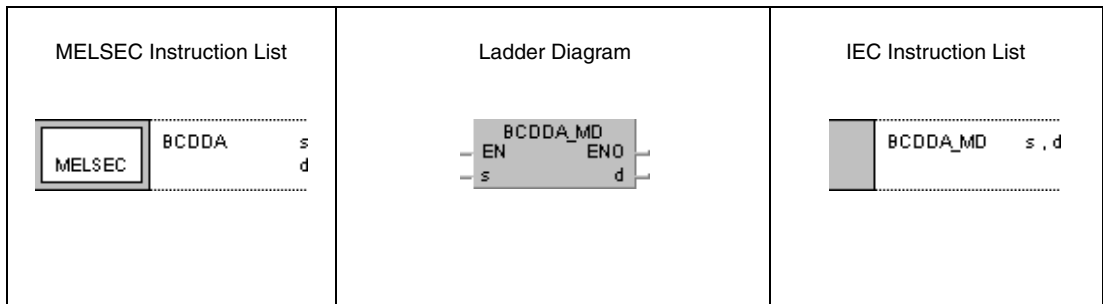
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

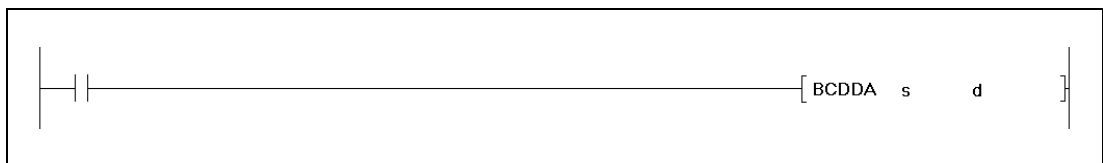
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	—	SM0	3
d	—	●	●	—	—	—	—	—	—		

GX IEC Developer



GX Developer



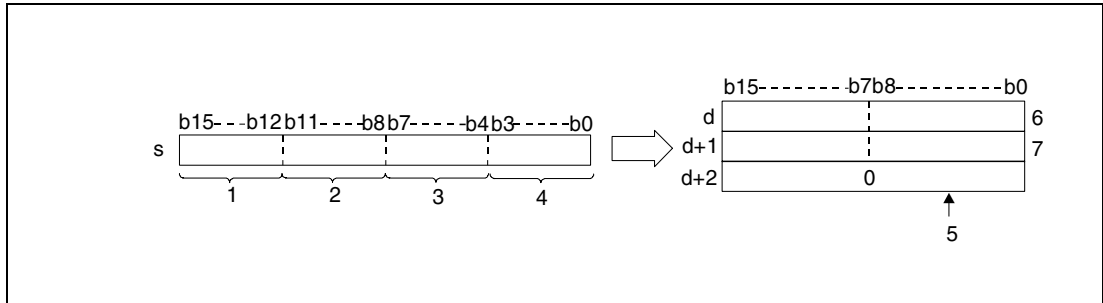
Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	BCD data to be converted into ASCII format.	Word	ANY16/32
d	First number of device storing the conversion result.	Character string	Array [1..3]/ [1..5] of ANY16

**Functions Conversion of 4-/ 8-digit BCD data into ASCII code**

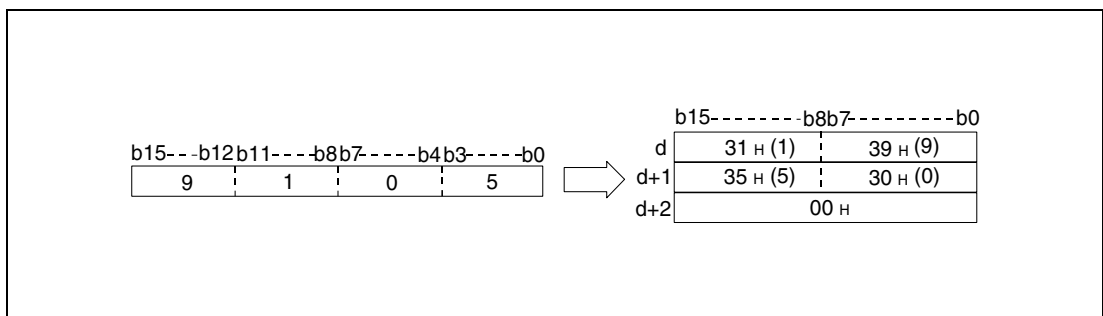
**BCDDA Conversion of 4-digit BCD data**

The BCDDA instruction converts 4-digit BCD data specified by s into the ASCII format and stores it in the devices specified by d (Array\_d[1]) through d+2 (Array\_d[3]).



- <sup>1</sup> Digit of thousands
- <sup>2</sup> Digit of hundreds
- <sup>3</sup> Digit of tens
- <sup>4</sup> Digit of ones
- <sup>5</sup> With the relay SM701 not set
- <sup>6</sup> ASCII code of the 3rd digit/ ASCII code of the 4th digit
- <sup>7</sup> ASCII code of the 1st digit/ ASCII code of the 2nd digit

The value 9105 specified in d is stored as follows:



The BCD value specified in s may range from 0 to 9999.

The conversion result is stored in d (Array\_d[1]) through d+2 (Array\_d[3]).

If one of the digits is 0, this digit is processed as "30H" (zeroes are not suppressed).

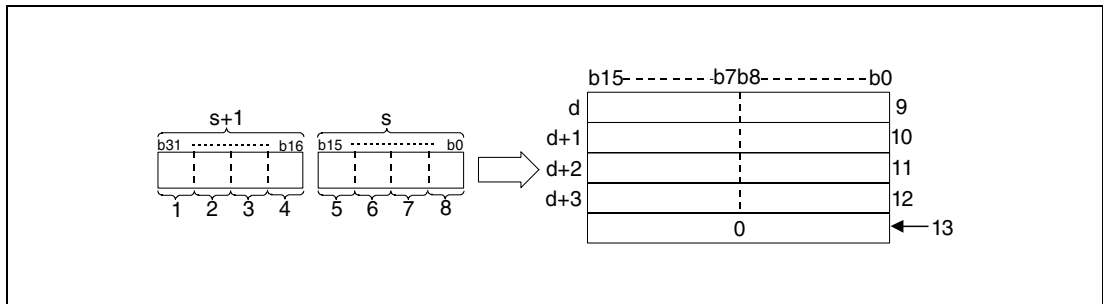
The storage of the data in the device specified by d+2 (Array\_d[3]) depends on the status of the relay SM701 as follows:

If this relay is not set, a zero "00H" is stored in the area d+2 (Array\_d[3]).

If this relay is set, the value in d+2 (Array\_d[3]) remains unchanged.

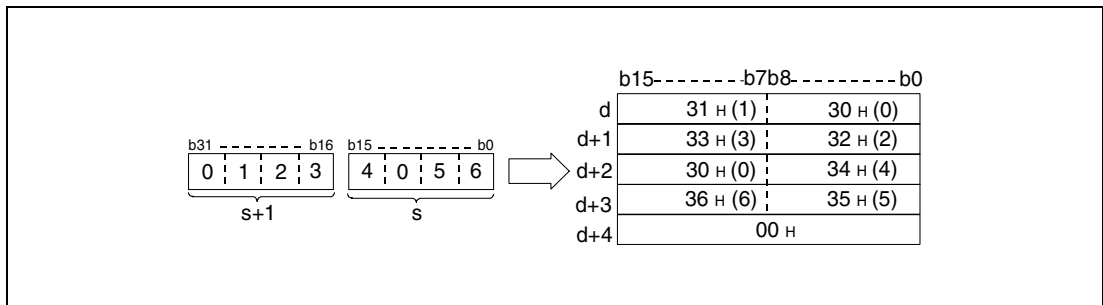
**DBCDDA Conversion of 8-digit BCD data**

The DBCDDA instruction converts 8-digit BCD data specified by s and s+1 into the ASCII format and stores it in the devices specified by d (Array\_d[1]) through d+4 (Array\_d[5]).



- <sup>1</sup> Digit of ten millions
- <sup>2</sup> Digit of millions
- <sup>3</sup> Digits of hundred thousands
- <sup>4</sup> Digit of ten thousands
- <sup>5</sup> Digit of thousands
- <sup>6</sup> Digit of hundreds
- <sup>7</sup> Digit of tens
- <sup>8</sup> Digit of ones
- <sup>9</sup> ASCII code of the 7th digit/ ASCII code of the 8th digit
- <sup>10</sup> ASCII code of the 5th digit/ ASCII code of the 6th digit
- <sup>11</sup> ASCII code of the 3rd digit/ ASCII code of the 4th digit
- <sup>12</sup> ASCII code of the 1st digit/ ASCII code of the 2nd digit
- <sup>13</sup> With the relay SM701 not set

The value 01234056 specified in s and s+1 is stored in d as follows:



The BCD value specified by s and s+1 may range from 0 to 99999999.  
 The conversion result is stored in d (Array\_d[1]) through d+4 (Array\_d[5]).  
 If one of the digits is 0, this digit is processed as "30H" (zeroes are not suppressed).  
 The storage of the data in the device specified by d+4 (Array\_d[5]) depends on the status of the relay SM701.  
 If this relay is not set, a zero "00H" is stored in the area d+4 (Array\_d[5]).  
 If this relay is set, the value in d+4 (Array\_d[5]) remains unchanged.

**Operation Errors**

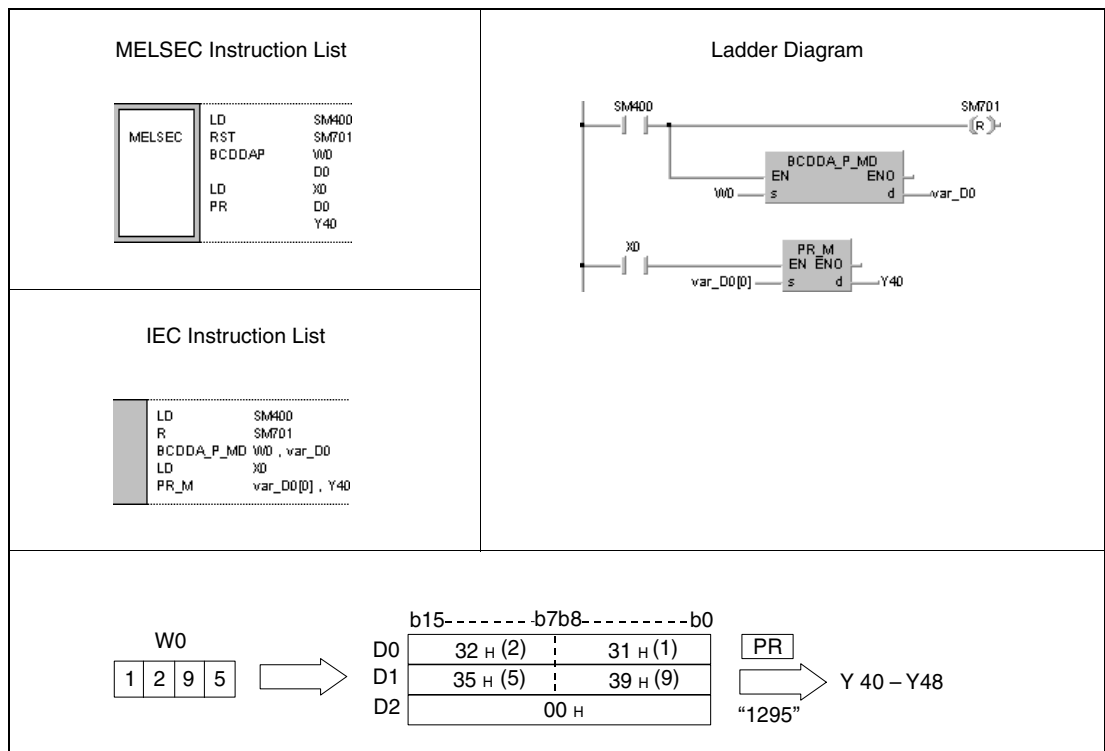
In the following cases an operation error occurs and the error flag is set:

- The BCD data in s exceed the range of 0 to 9999 during the execution of the BCDDA instruction (error code: 4100).
- The BCD data in s exceed the range of 0 to 99999999 during the execution of the DBCDDA instruction (error code: 4100).

**Program Example 1**

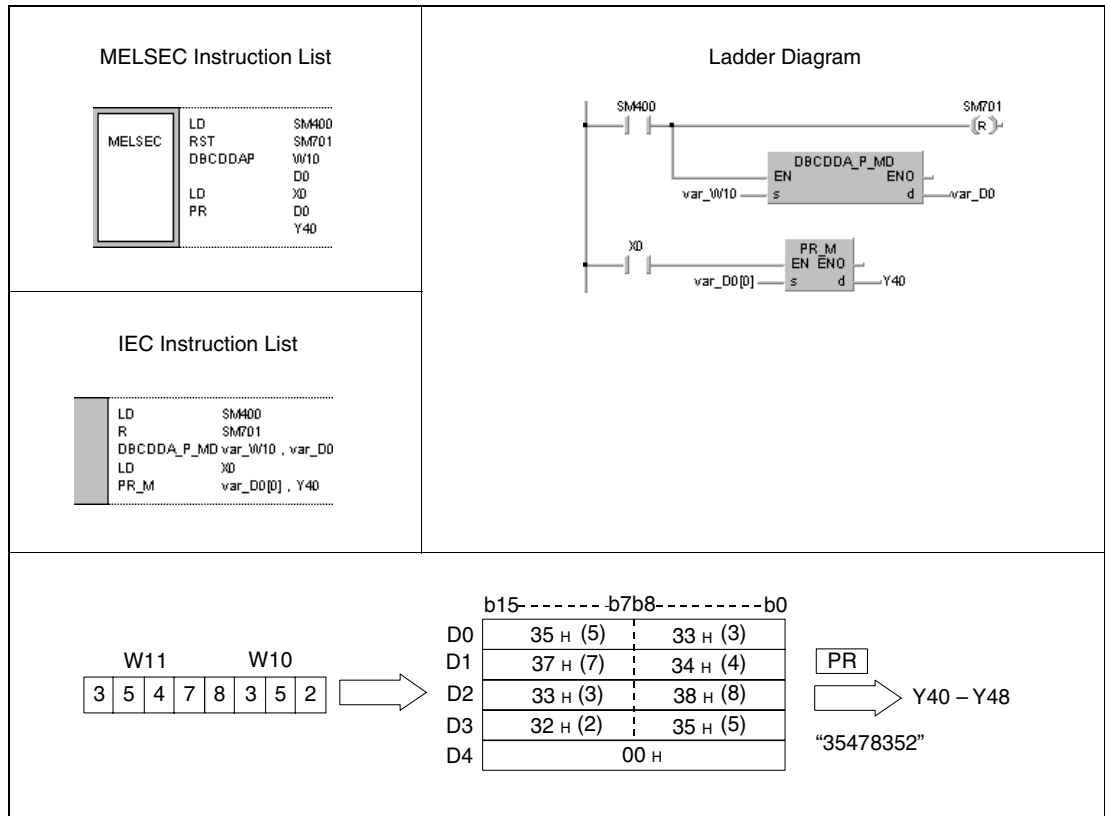
**BCDDAP**

With leading edge from SM400, the following program outputs the value of the 4-digit BCD data in W0 as decimal value in ASCII code via the BCDDAP instruction. The PR instruction outputs the characters at Y40 through Y48.



**Program Example 2** DBCDDAP

With leading edge from SM400, the following program outputs the value of the 8-digit BCD data in W10 and W11 as decimal value in ASCII code via the PR instruction. The PR instruction outputs the characters at Y40 through Y48.



**NOTE**

*These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



**7.11.4 DABIN, DABINP, DDABIN, DDABINP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

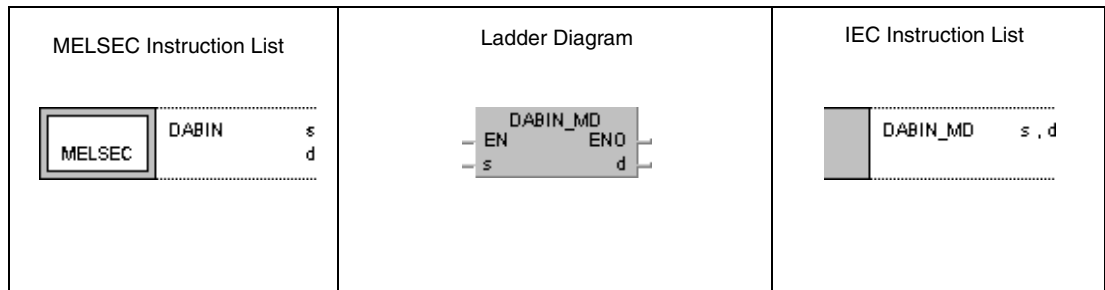
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

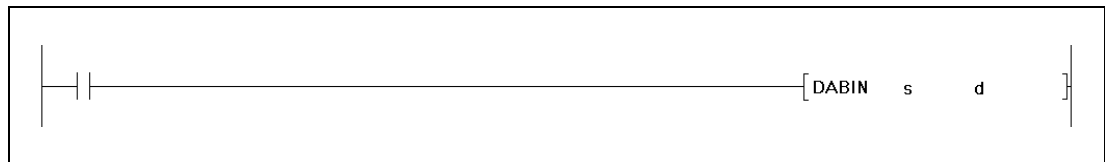
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

**GX IEC Developer**



**GX Developer**



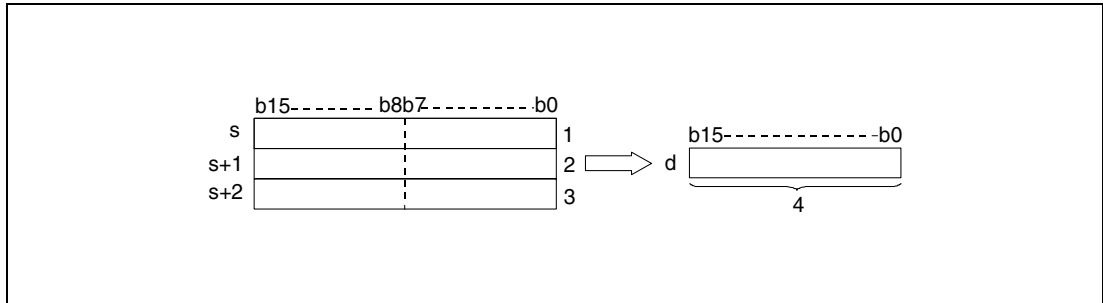
**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	Storage area storing the ASCII data to be converted.	Character string	Array [1..3]/ [1..6] of ANY16
d	Storage area storing the conversion result.	BIN 16-/32-bit	ANY16/32

**Functions Conversion of decimal ASCII data into BIN 16-/32-bit binary data**

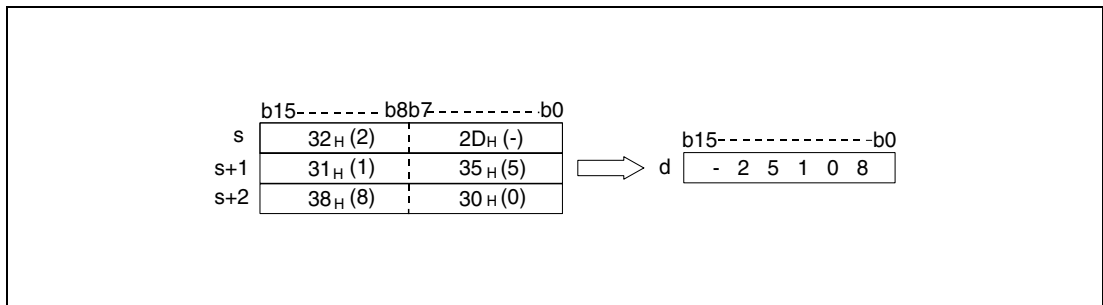
**DABIN Conversion of BIN 16-bit binary data**

The DABIN instruction converts the decimal ASCII data specified in the area s (Array\_s[1]) through s+2 (Array\_s[3]) into the BIN 16-bit format and stores it in the devices specified by d.



- <sup>1</sup> ASCII code of the digit of ten thousands/ sign character
- <sup>2</sup> ASCII code of the digit of hundreds/ ASCII code of the digit of thousands
- <sup>3</sup> ASCII code of the digit of ones/ ASCII code of the digit of tens
- <sup>4</sup> BIN 16-bit binary data

The value specified in the area s (Array\_s[1]) through s+2 (Array\_s[3]) is stored in d as -25018H as follows:



The ASCII value specified by s (Array\_s[1]) through s+2 (Array\_s[3]) may range from -32768 to 32767.

The sign character is stored as "20H" if the binary value is positive.

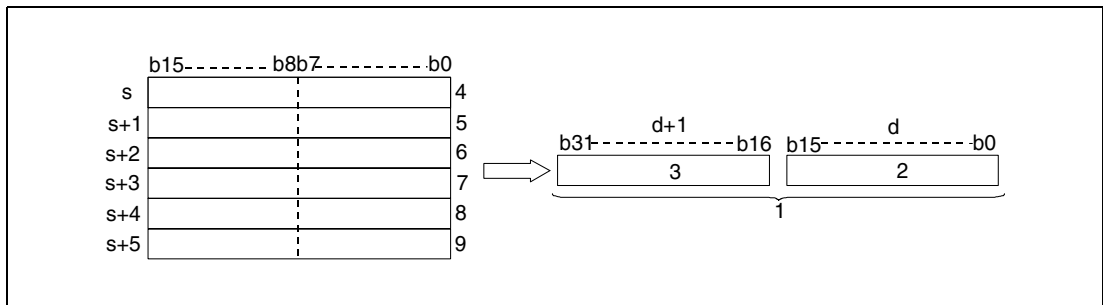
For a negative result the value "2DH" is stored.

Each stored digit of the ASCII code may range from "30H" to "39H".

If a digit contains the value "20H" or "00H", this value will be overwritten automatically with the value "30H".

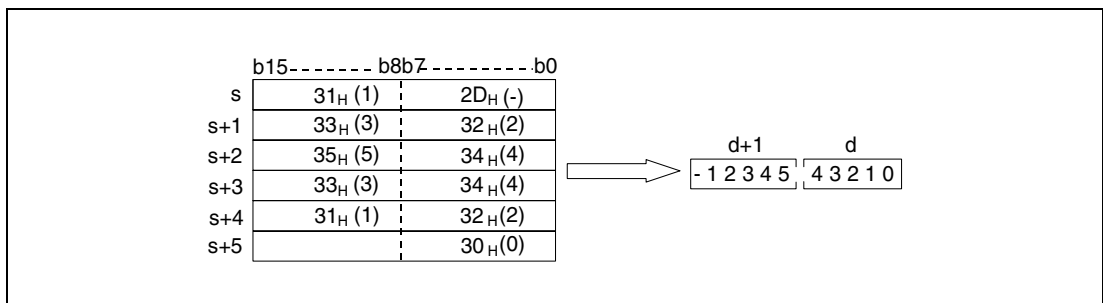
**DDABIN Conversion into BIN 32-bit data**

The DDABIN instruction converts the decimal ASCII data specified in the area s (Array\_s[1]) through s+5 (Array\_s[6]) into the BIN 32-bit format and stores it in the devices specified by d and d+1.



- <sup>1</sup> BIN 32-bit binary data
- <sup>2</sup> Lower 16-bit
- <sup>3</sup> Upper 16-bit
- <sup>4</sup> ASCII code of the digit of billions/ sign character
- <sup>5</sup> ASCII code of the digit of ten millions/ ASCII code of the digit of hundred millions
- <sup>6</sup> ASCII code of the digit of hundred thousands/ ASCII code of the digit of millions
- <sup>7</sup> ASCII code of the digit of thousands/ ASCII code of the digit of ten thousands
- <sup>8</sup> ASCII code of the digit of tens/ ASCII code of the digit of hundreds
- <sup>9</sup> Is ignored/ ASCII code of the digit of tens

The value specified in the area s (Array\_s[1]) through s+5 (Array\_s[6]) is stored in d as -1234543210<sub>H</sub> as follows:



The ASCII value specified in s (Array\_s[1]) through s+5 (Array\_s[6]) may range from -2147483648 to 2147483647.

The sign character is stored as "20<sub>H</sub>" if the binary value is positive.

For a negative result the value "2D<sub>H</sub>" is stored.

Each stored digit of the ASCII code may range from "30<sub>H</sub>" to "39<sub>H</sub>".

If a digit contains the value "20<sub>H</sub>" or "00<sub>H</sub>", this value will be overwritten automatically with the value "30<sub>H</sub>".

**Operation Errors**

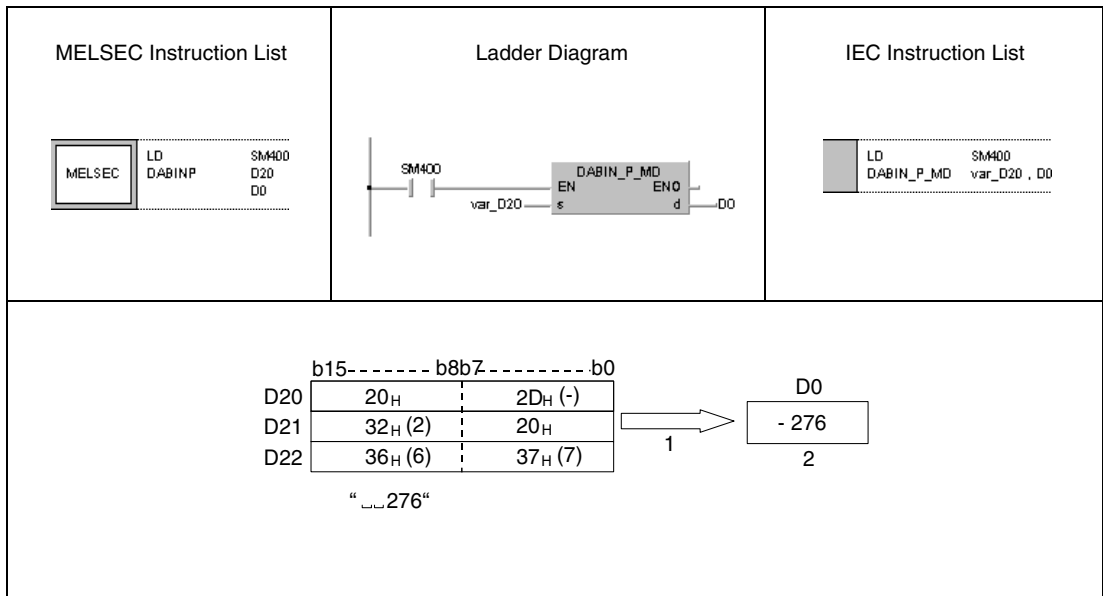
In the following cases an operation error occurs and the error flag is set:

- The sign character stored in the lower 16 bits of the device s (Array\_s[1]) contains a value different from "30H" to "39H, "20H" or "00H" (error code 4100).
- The ASCII code stored in the area s (Array\_s[1]) through s+5 (Array\_s[6]) contains values different from "30H" to "39H, "20H" to "00H" (error code 4100).
- The ASCII code stored in the area s (Array\_s[1]) through s+5 (Array\_s[6]) exceeds the following range of values:  
 For the DABIN instruction                      -32768 to 32767  
 For the DDABIN instruction                 -2147483648 to 2147483647 (error code 4100).

**Program Example 1**

DABINP

With leading edge from SM400, the following program converts the five-digit decimal ASCII value in D20 (var\_D20 Array [0]) through D22 (var\_D20 Array [2]) into a binary value and stores it in D0.



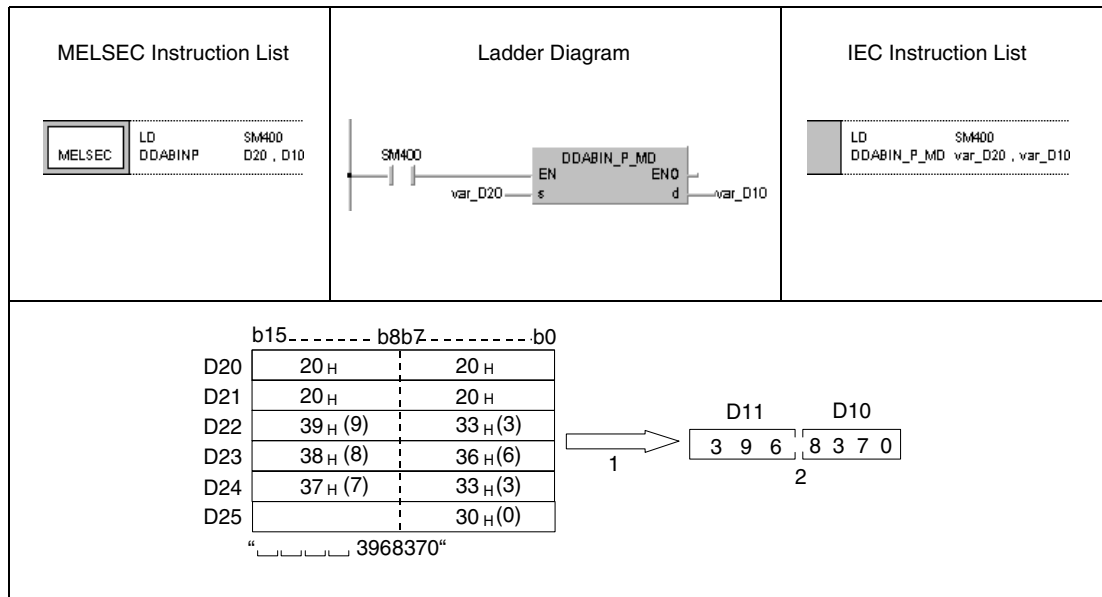
<sup>1</sup> Is read as -00276

<sup>2</sup> Binary value

**Program** DDABINP

**Example 2**

With leading edge from SM400, the following program converts the ten-digit decimal ASCII value in D20 (var\_D20 Array [0]) through D25 (var\_D20 Array [5]) into a binary value and stores it in D10 and D11.



<sup>1</sup> Is read as +0003968370

<sup>2</sup> Binary value

**NOTE**

*These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.5 HABIN, HABINP, DHABIN, DHABINP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

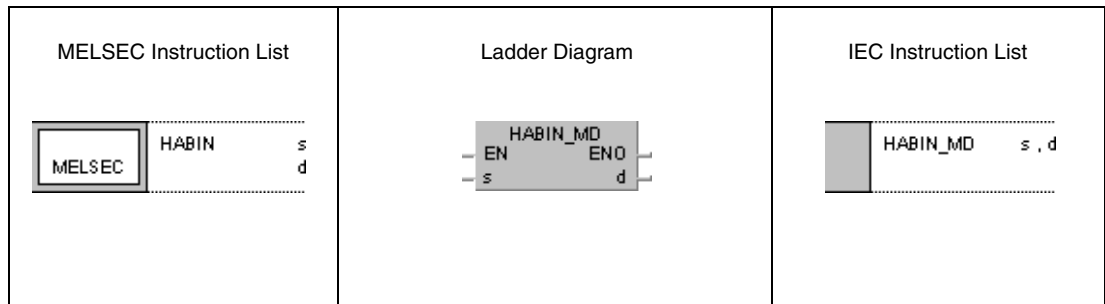
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

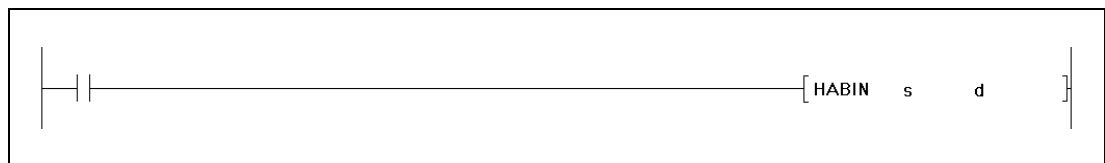
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



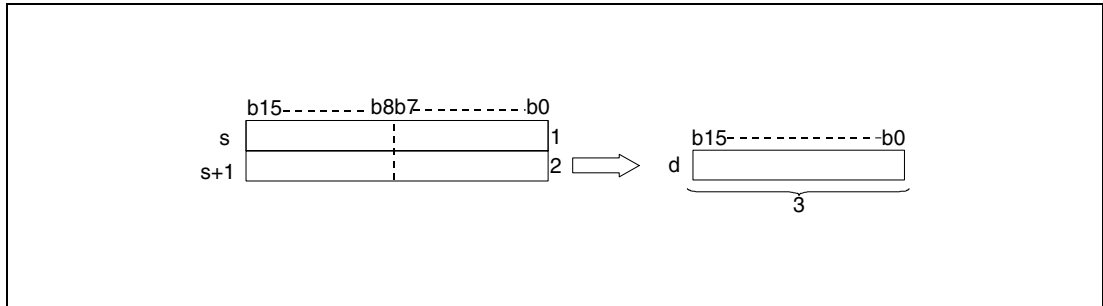
Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	Storage area storing the ASCII data to be converted.	Character string	ANY32/Array [1..4] of ANY16
d	Storage area storing the conversion result.	BIN 16-/32-bit	ANY16/32

**Functions Conversion of hexadecimal ASCII data into BIN 16-/32-bit binary data**

**HABIN Conversion into BIN 16-bit data**

The HABIN instruction converts the hexadecimal ASCII data in the device specified by s and s+1 into the BIN 16-bit binary format and stores it in the devices specified by d.

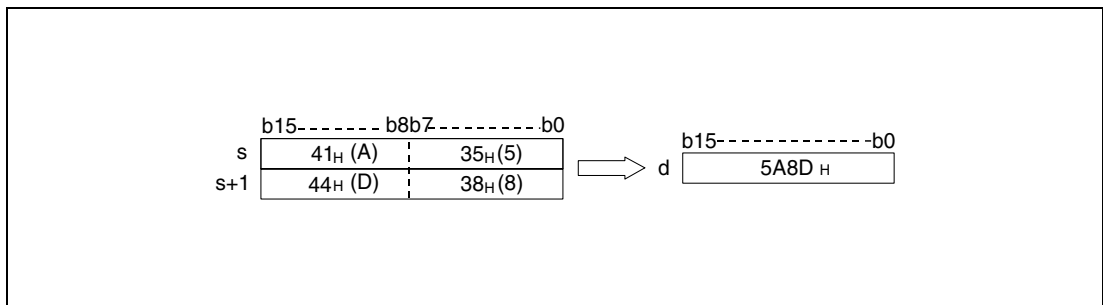


<sup>1</sup> ASCII code for the 3rd digit/ ASCII code for the 4th digit

<sup>2</sup> ASCII code for the 1st digit/ ASCII code for the 2nd digit

<sup>3</sup> BIN 16-bit binary data

The value "5A8D<sub>H</sub>" specified in s through s+1 is stored after being processed as follows:

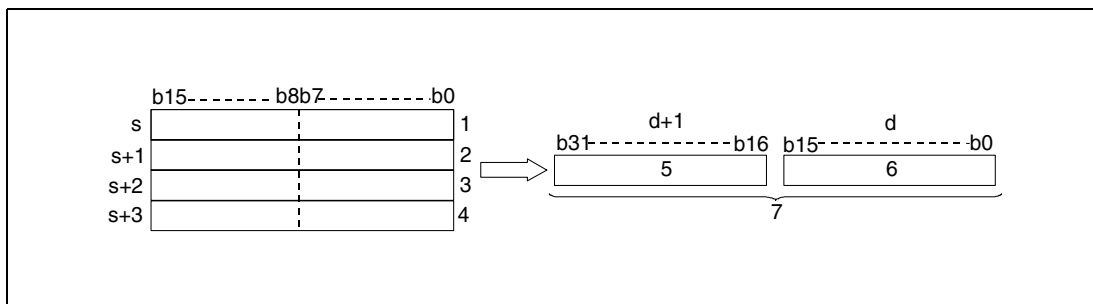


The ASCII value specified in s through s+1 may range from 0000<sub>H</sub> to FFFF<sub>H</sub>.

Each stored digit of the ASCII code may range from "30<sub>H</sub>" to "39<sub>H</sub>" and "41<sub>H</sub>" und "46<sub>H</sub>".

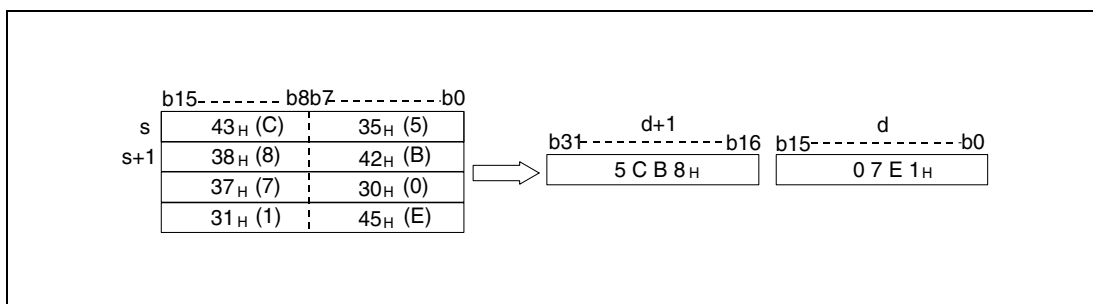
**DHABIN Conversion into BIN 32-bit data**

The DHABIN instruction converts the hexadecimal ASCII data specified in the area s (Array\_s[1]) through s+3 (Array\_s[4]) into the BIN 32-bit format and stores it in the devices specified by d and d+1.



- <sup>1</sup> ASCII code of the 7th digit/ ASCII code of the 8th digit
- <sup>2</sup> ASCII code of the 5th digit/ ASCII code of the 6th digit
- <sup>3</sup> ASCII code of the 3rd digit/ ASCII code of the 4th digit
- <sup>4</sup> ASCII code of the 1st digit/ ASCII code of the 2nd digit
- <sup>5</sup> Upper 16 bits
- <sup>6</sup> Lower 16 bits
- <sup>7</sup> BIN 32-bit binary data

The value "5CB807E1" specified in s (Array\_s[1]) through s+3 (Array\_s[4]) is stored after being processed in d and d+1 as follows:



The ASCII value specified in s (Array\_s[1]) through s+3 (Array\_s[4]) may range from 00000000H and FFFFFFFFH.

Each stored digit of the ASCII code may range from "30H" to "39H" and "41H" and "46H".

**Operation Errors**

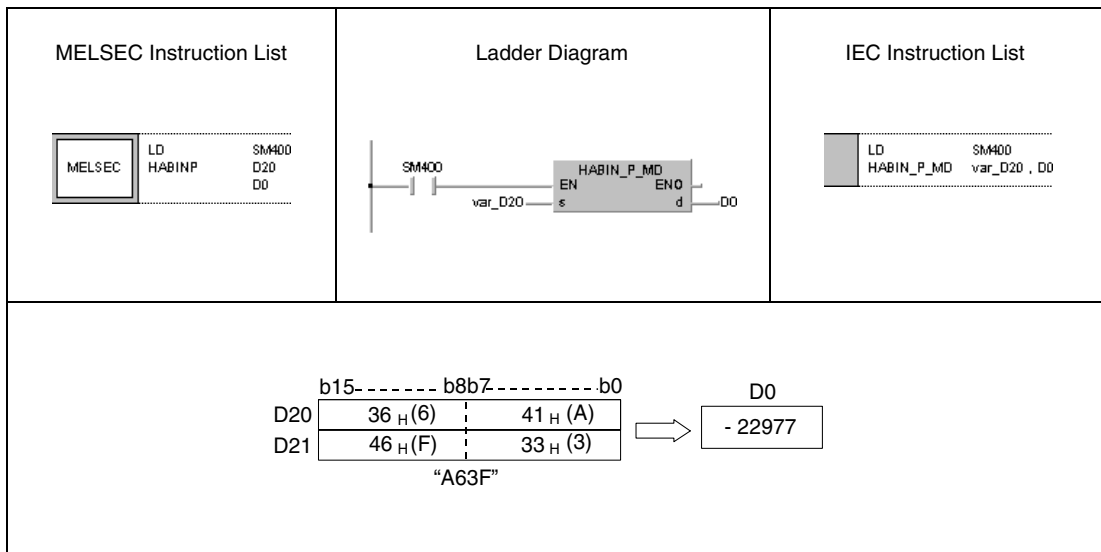
In the following cases an operation error occurs and the error flag is set:

- The ASCII code stored in the area s (Array\_s[1]) through s+3 (Array\_s[4]) exceeds the relevant range of "30H" to "39H" and "41H" to "46H" (error code 4100).



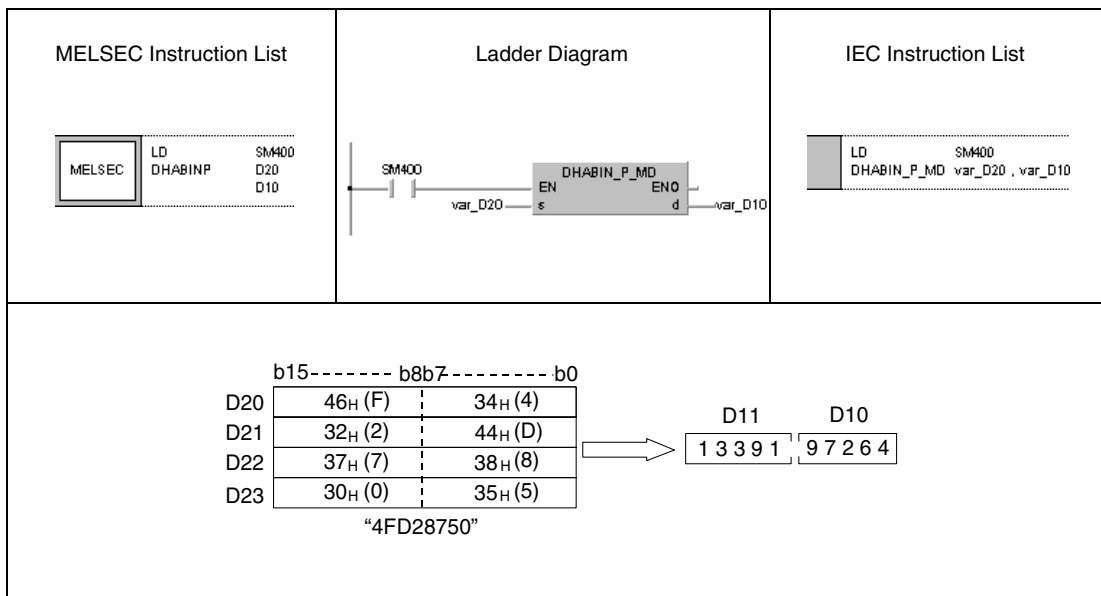
**Program Example 1** HABINP

With leading edge from SM400, the following program converts the 4-digit ASCII value in D20 (var\_D20 Array [0]) through D21 (var\_D20 Array [1]) into a binary value and stores it in D0.



**Program Example 2** DHABINP

With leading edge from SM400, the following program converts the 8-digit ASCII value in D20 (var\_D20 Array [0]) through D23 (var\_D20 Array [3]) into a binary value and stores it in D10 and D11.



**NOTE**

*These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.6 DABCD, DABCDP, DDABCD, DDABCDP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

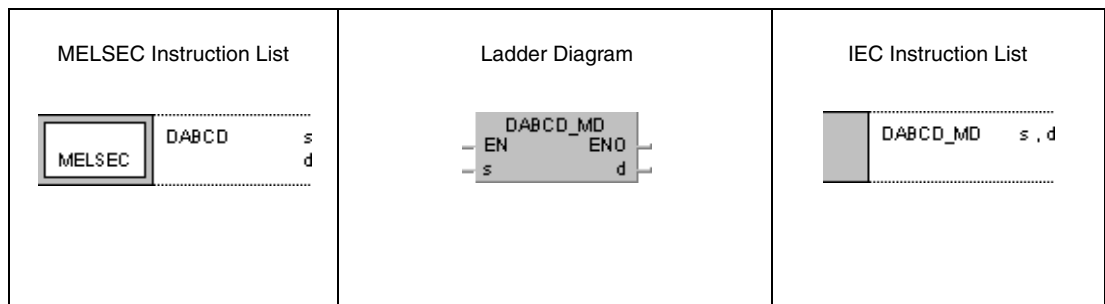
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

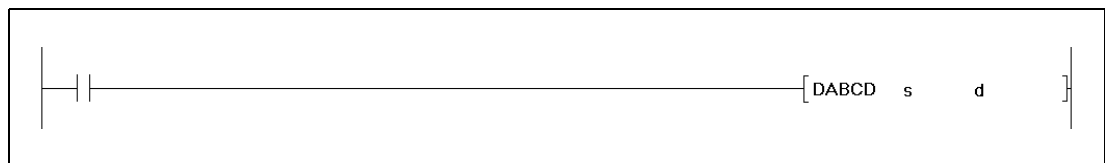
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



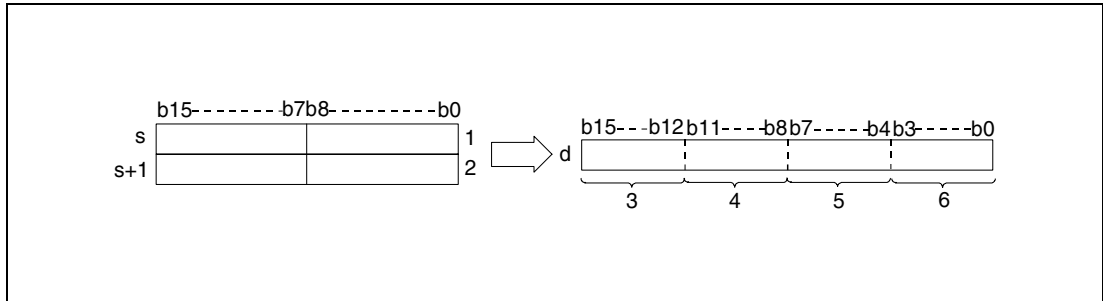
Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	Storage area storing the ASCII data to be converted.	Character string	ANY32/ Array [1..4] of ANY16
d	Storage area storing the conversion result.	4-/8-digit BCD data	ANY16/ 32

**Functions Conversion of decimal ASCII data into 4-/8-digit BCD data**

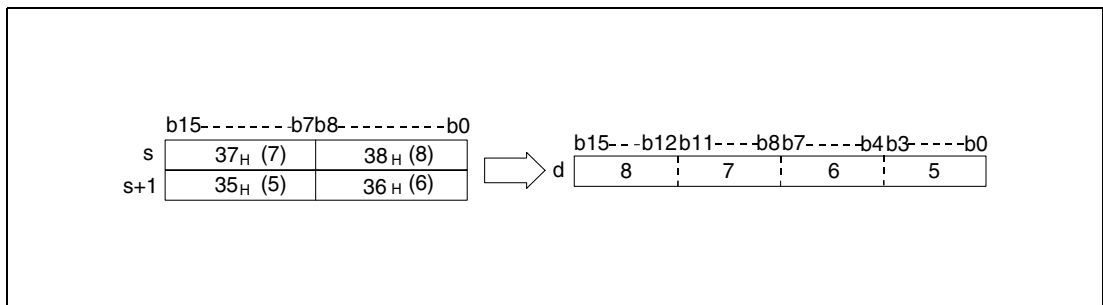
**DABCD Conversion into 4-digit BCD data**

The DABCD instruction converts the decimal ASCII data in s and s+1 into the 4-digit BCD data format and stores it in the devices specified by d.



- <sup>1</sup> ASCII code of the digit of hundreds/ ASCII code of the digit of thousands
- <sup>2</sup> ASCII code of the digit of ones/ ASCII code of the digit of tens
- <sup>3</sup> Digit of thousands
- <sup>4</sup> Digit of hundreds
- <sup>5</sup> Digit of tens
- <sup>6</sup> Digit of ones

The value 8765 specified in s and s+1 is stored in d as follows:



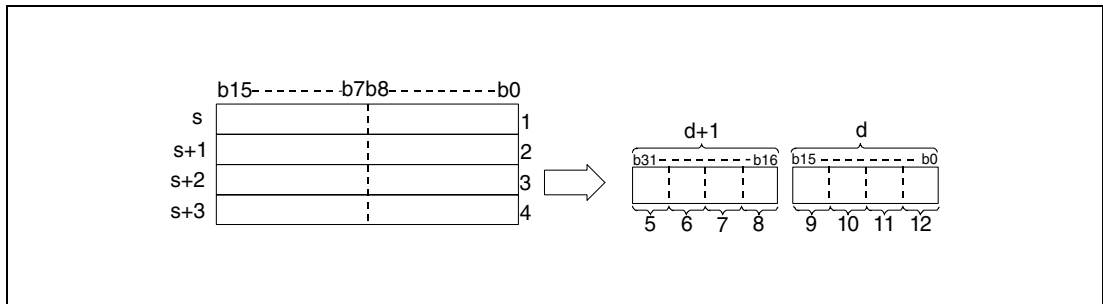
The ASCII value specified in s through s+1 may range from 0 to 9999.

Each stored digit of the ASCII code may range from "30H" to "39H".

If a digit contains the value "20H" or "00H", this value will be overwritten automatically with the value "30H".

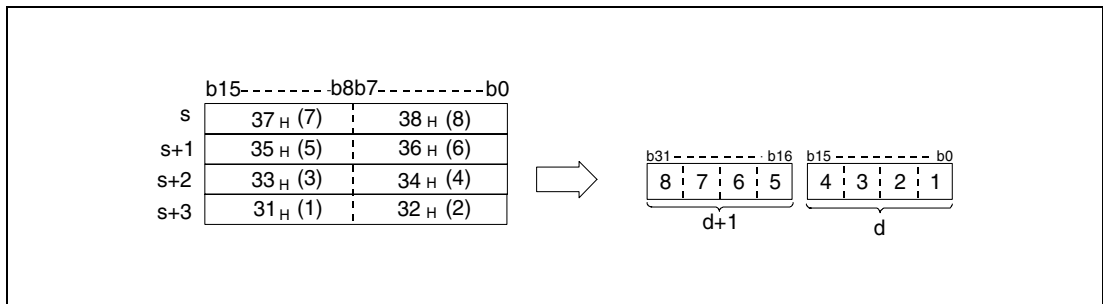
**DDABCD Conversion into 8-digit BCD data**

The DDABCD instruction converts the ASCII data specified in the area s (Array\_s[1]) through s+3 (Array\_s[4]) into the 8-digit BCD format and stores it in the devices specified in d and d+1.



- <sup>1</sup> ASCII code of the digit of millions/ ASCII code of the digit of ten millions
- <sup>2</sup> ASCII code of the digit of ten thousands/ ASCII code of the digit of hundred thousands
- <sup>3</sup> ASCII code of the digit of hundreds/ ASCII code of the digit of thousands
- <sup>4</sup> ASCII code of the digit of ones/ ASCII code of the digit of tens
- <sup>5</sup> Digit of ten millions
- <sup>6</sup> Digit of millions
- <sup>7</sup> Digit of hundred thousands
- <sup>8</sup> Digit of ten thousands
- <sup>9</sup> Digit of thousands
- <sup>10</sup> Digit of hundreds
- <sup>11</sup> Digit of tens
- <sup>12</sup> Digit of ones

The value 87654321 specified in s (Array\_s[1]) through s+3 (Array\_s[4]) is stored in d and d+1 as follows:



The ASCII value specified in s (Array\_s[1]) through s+3 (Array\_s[4]) may range from 0 to 99999999.

Each stored digit of the ASCII code may range from "30H" to "39H".

If a digit contains the value "20H" or "00H", this value will be overwritten automatically with the value "30H".

**Operation Errors**

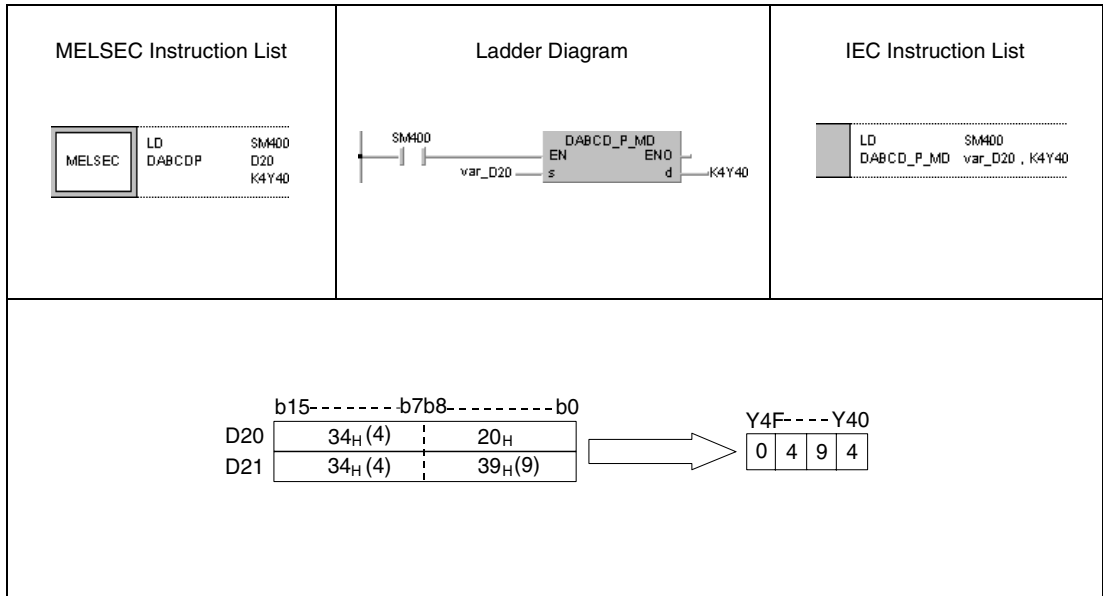
In the following cases an operation error occurs and the error flag is set:

- The ASCII code in the separate registers from s (Array\_s[1]) to s+3 (Array\_s[4]) exceeds the relevant range from "30H" to "39H" (error code 4100).

**Program Example 1**

DABCDP

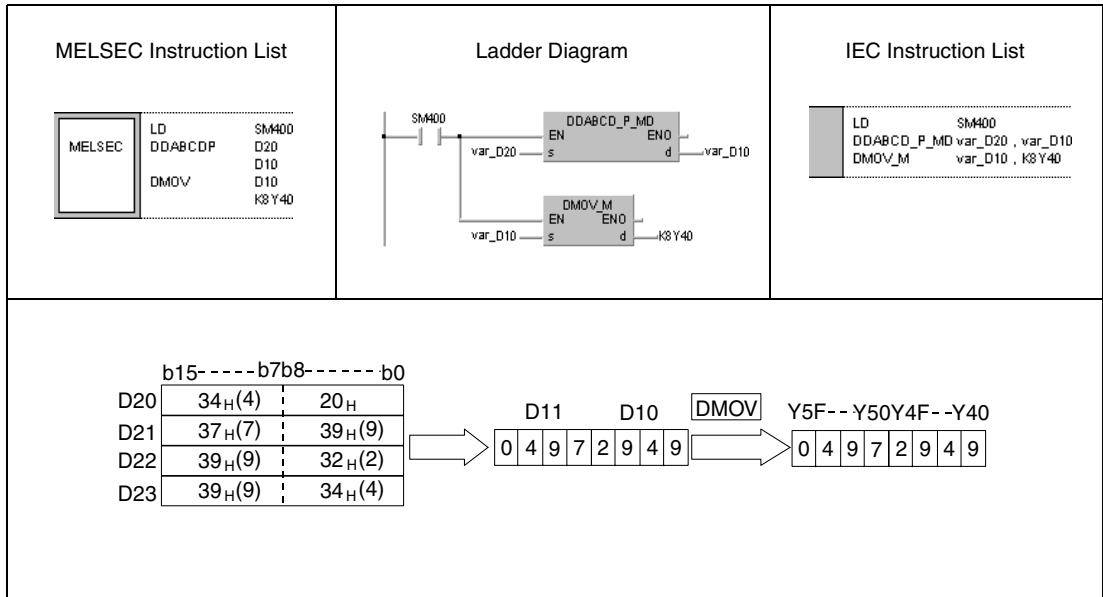
With leading edge from SM400, the following program converts the ASCII value in D20 (var\_D20 Array [0]) through D21 (var\_D20 Array [1]) into a 4-digit BCD value and outputs it at Y40 through Y4F.



**Program Example 2**

**DDABCDP**

With leading edge from SM400, the following program converts the ASCII value in D20 (var\_D20 [0]) through D23 (var\_D20 [3]) into an 8-digit BCD value, stores the result in D10 and D11, and outputs it at Y40 through Y5F.



7.11.7 COMRD, COMRDP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

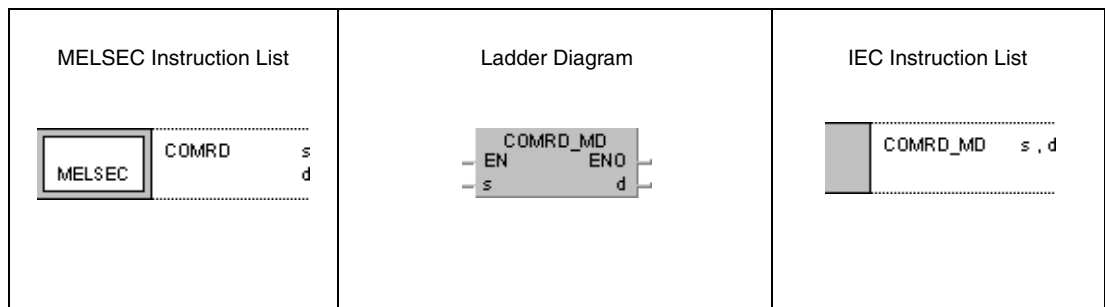
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

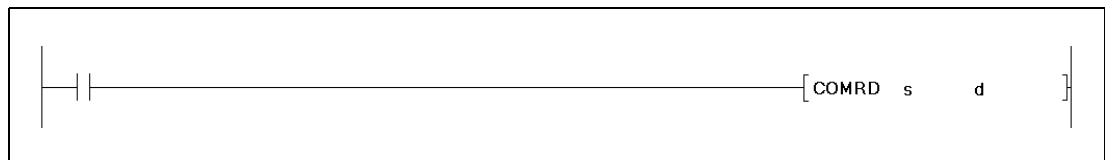
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other BL\S, BL\TR, BL,P, I, J, U		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	—	—	●	SM0	3	
d	—	●	●	—	—	—	—	—			

GX IEC Developer



GX Developer



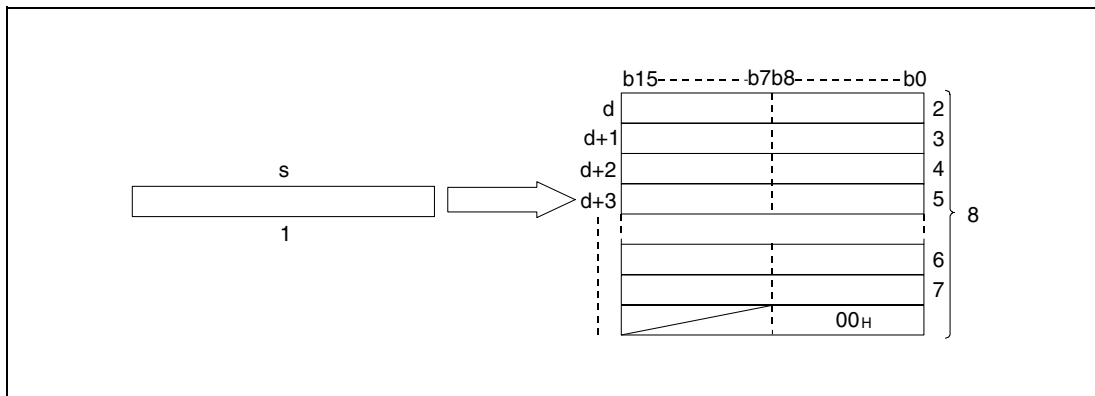
Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	First number of device storing comment data to be read.	Device number	ANY16
d	First number of device to store read comment data.	Character string	Array [1..8] of ANY16

**Functions Reading device comment data**

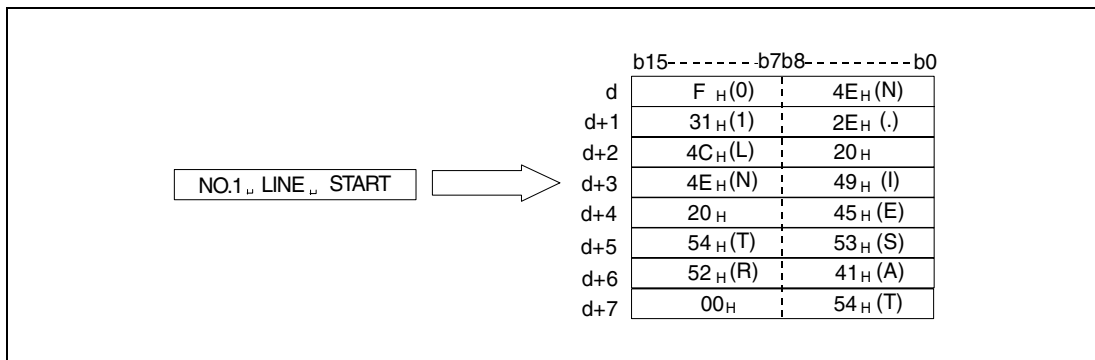
**COMRD Read instruction**

The COMRD instruction reads comment data from the device specified by s and stores it as ASCII code in the area d (Array\_d[1]) through d+7 (Array\_d[8]).



- <sup>1</sup> Comment data
- <sup>2</sup> ASCII code of the 2nd character/ ASCII code of the 1st character
- <sup>3</sup> ASCII code of the 4th character/ ASCII code of the 3rd character
- <sup>4</sup> ASCII code of the 6th character/ ASCII code of the 5th character
- <sup>5</sup> ASCII code of the 8th character/ ASCII code of the 7th character
- <sup>6</sup> ASCII code of the 30th character/ ASCII code of the 29th character
- <sup>7</sup> ASCII code of the 32th character/ ASCII code of the 31th character
- <sup>8</sup> Stores at maximum 32 characters.

The comment data stored in s with the character string "NO.1 LINE START" will be stored from d (Array\_d[1]) on, as follows:



The address area of the devices specified by s must be located within the address area for comment data.

If no comment is specified by s, the characters are converted into blank characters.

A comment must not exceed the maximum length of 32 characters.

The content of the byte following the last character depends on the status of the special relay SM701 as follows:

- If SM701 is not set, a zero is stored
- If SM701 is set, no changes are made.

SM720 is set for one scan after the execution of the COMRD instruction has been finished.



SM721 is ON during the execution of the COMRD instruction. If SM721 is already set, when the COMRD instruction is started, no processing will be performed.

### Operation Errors

In the following cases an operation error occurs and the error flag is set:

- The address area of the device specified by s exceeds the comment data range (error code 4100).
- The COMRD(P) instruction is executed while a comment is written during RUN (error code 4100).
- The designated file does not exist (error code 2410)

### NOTE

A CPU of the System Q completes the processing of the COMRD (P) after several scans. A QnA CPU completes the processing immediately.

The starting signal (command) of the COMRD(P) instruction is disabled when it is turned ON before an other COMRD(P) instruction is completed (SM720 must have been ON).

Two or more file comments cannot be accessed simultaneously.

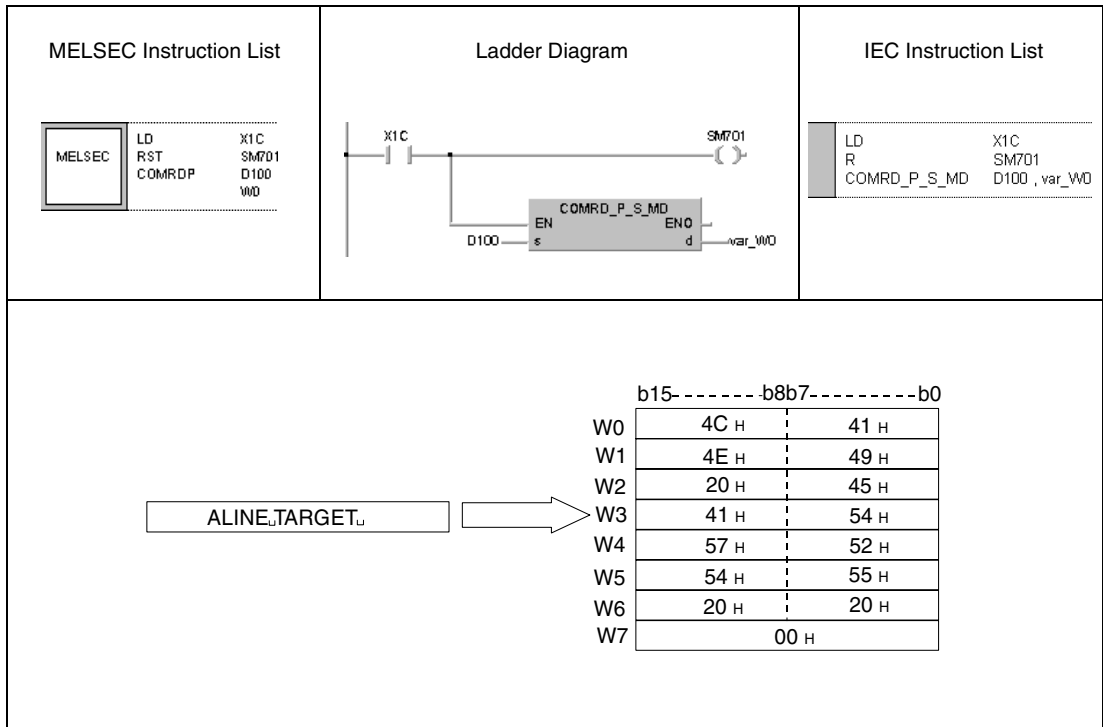
The following instructions cannot be executed simultaneously because the use SM721 in common:

Instruction	ON during execution	ON for one scan after the execution of the instruction is complete	ON after the execution of the instruction is complete with error
S.FREAD S.FWRITE	SM721	Bit designated by instruction	Bit designated by instruction + next Bit
PRC COMRD		SM720	—

**Program Example**

COMRDP

With leading edge from X1C, the following program stores a comment specified in D100, as ASCII code in W0 (var\_W0 Array [0]) through W7 (var\_W0 Array [7]).



**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.8 LEN, LENP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

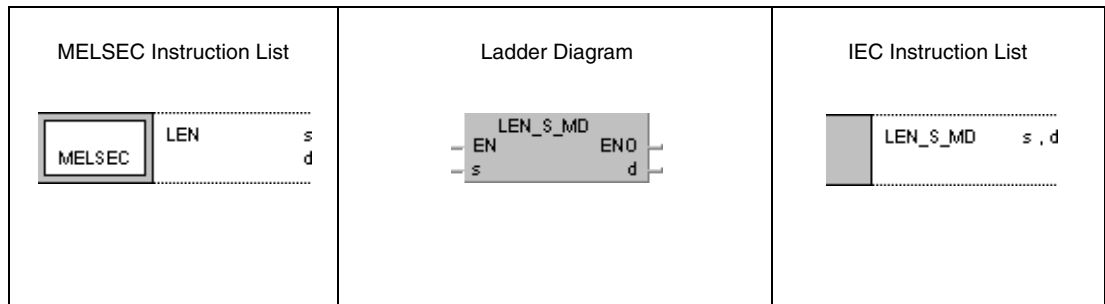
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

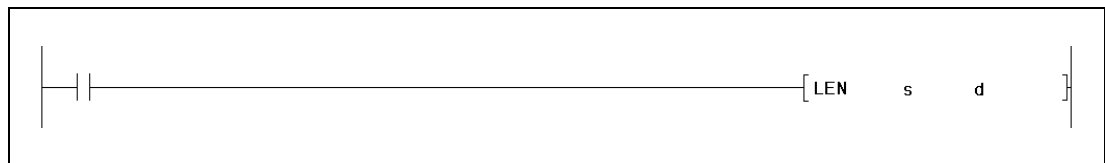
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other U		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



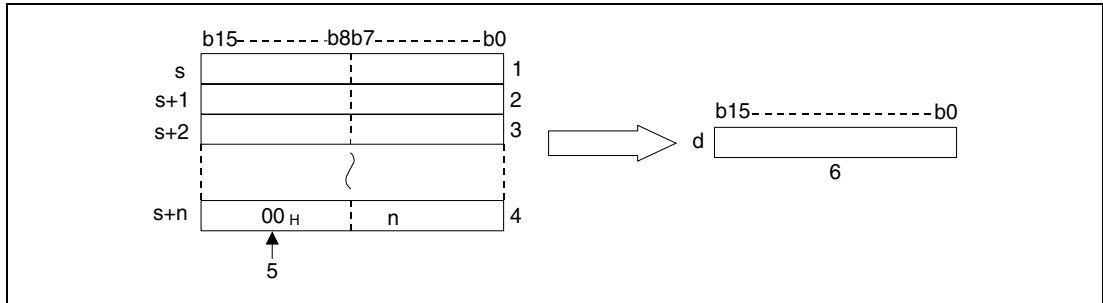
Variables

Set Data	Meaning	Data Type
s	First number of device storing a character string of which the length is to be detected.	Character string
d	Address area storing the detected length of the character string.	BIN 16-bit

**Functions**     **Detecting the length of character strings**

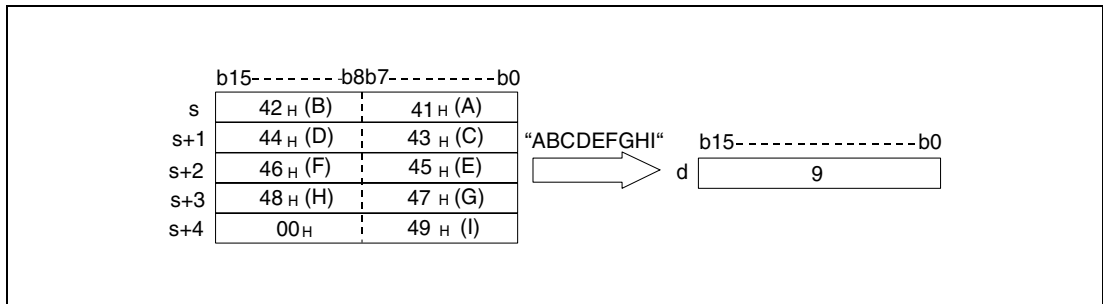
**LEN**     **Length detection**

The length instruction detects the length of a character string specified in s and stores the result in the device specified by d.



- <sup>1</sup> 2nd character/ 1st character
- <sup>2</sup> 4th character/ 3rd character
- <sup>3</sup> 6th character/ 5th character
- <sup>4</sup> nth character
- <sup>5</sup> End of character string
- <sup>6</sup> Length of character string

The character string "ABCDEFGHI" stored in s is stored in d as "9" as follows:



The character string stored in s is being processed until the character code "00H" is read. The result is stored in d.

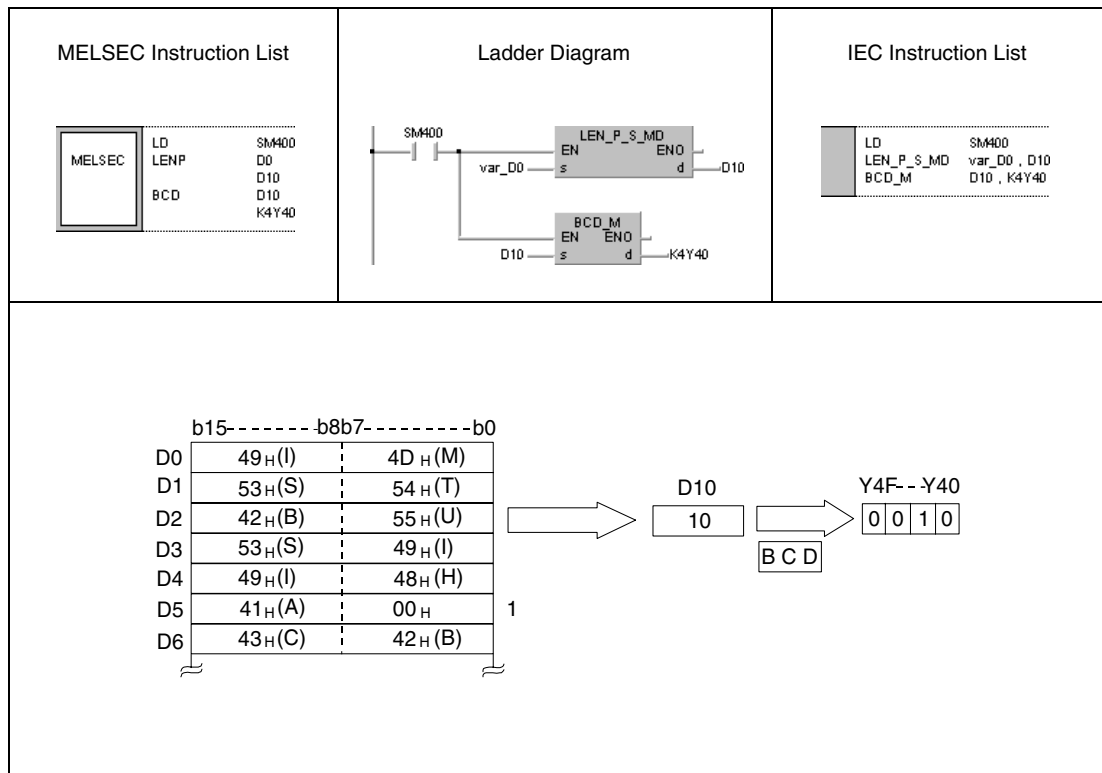
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The character code "00H" is missing in the last byte in s (error code 4101).

**Program Example** LENP

With leading edge from SM400, the following program processes the character string stored in D0, detects its length and outputs the character string as 4-digit BCD data at Y40 through Y4F.



<sup>1</sup> Characters following the character code "00H" are omitted (only the length of the character string "MITSUBISHI" is detected)

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.9 STR, STRP, DSTR, DSTRP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

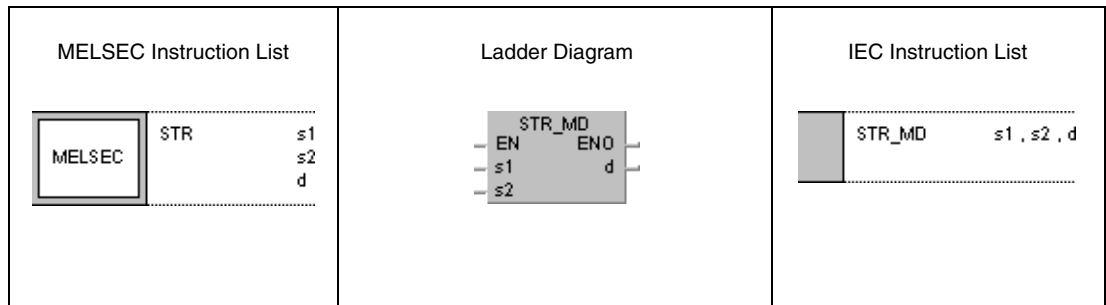
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

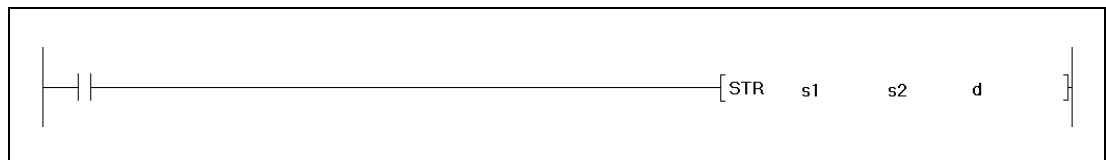
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□□□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	—	—	SM0	4
s2	●	●	●	●	●	●	●	—	—		
d	—	●	●	—	—	—	—	—	—		

GX IEC  
Developer



GX  
Developer



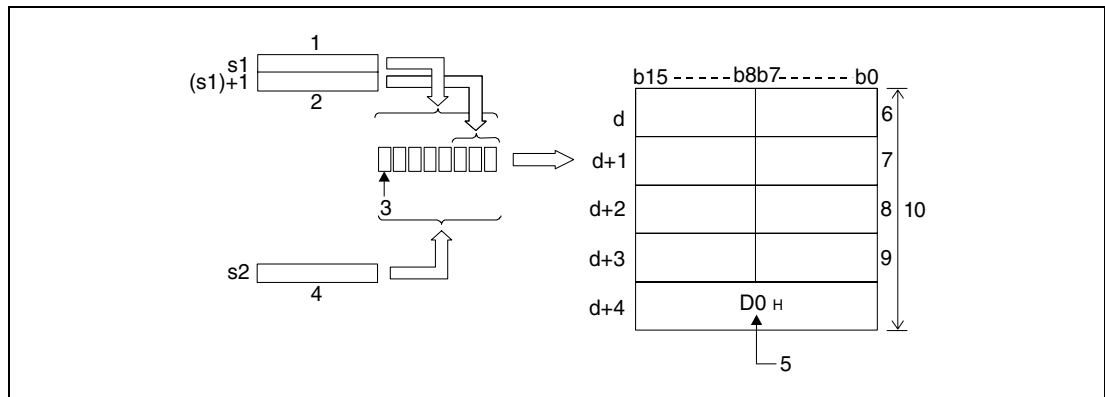
Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s1	First number of device storing the number of digits of the numerical value to be converted.	BIN 16-bit	ANY32
s2	Binary data to be converted.	BIN 16-/32-bit	ANY16/32
d	First number of device storing the converted character string.	Character string	Array [1..5]/ [1..6] of ANY16

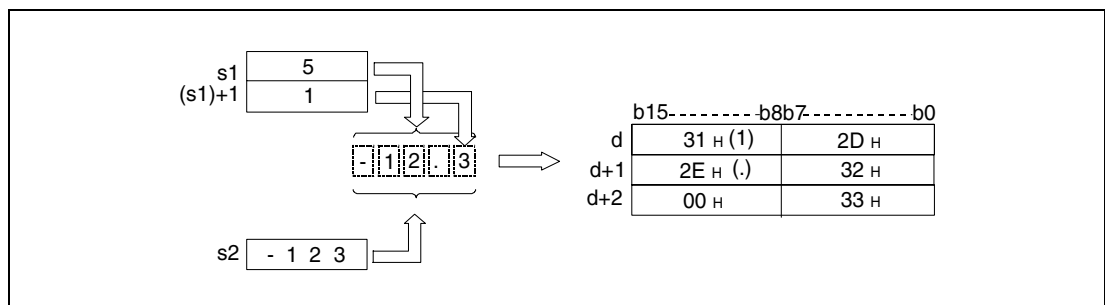
**Functions Conversion of BIN 16-/32-bit binary data into character strings**

**STR Conversion of BIN 16-bit binary data**

The STR instruction adds a decimal point to the BIN 16-bit binary value in the device specified by s2 to the digit specified by the devices s1 and (s1)+1, converts the data into a character string, and stores it in the area of the devices specified by d (Array\_d[1]) through d+4 (Array\_d[5]).



- 1 Total of all digits
- 2 Decimal places
- 3 Sign
- 4 Binary value
- 5 End of character string indication, automatically placed.
- 6 Character position in ASCII; total of digits -1/ ASCII code of the sign
- 7 Character position in ASCII; total of digits -3/ character position in ASCII; total of digits -2
- 8 Character position in ASCII; total of digits -5/ character position in ASCII; total of digits -4
- 9 Character position in ASCII; total of digits -7/ character position in ASCII; total of digits -6
- 10 Total of all digits



The number of digits that can be stored in the device specified by s1 ranges from 2 to 8.

The number of decimal places that can be stored in the devices specified by (s1)+1 ranges from 0 to 5 and must not exceed the number of digits minus 3.

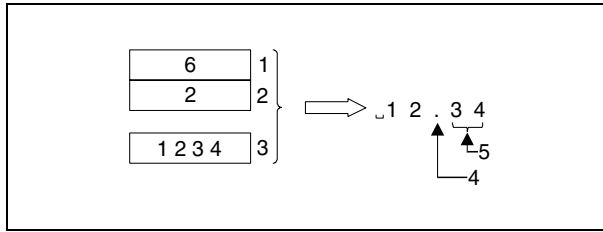
The BIN 16-bit data that can be stored in the device specified by s2 must range from -32768 to 32767.

After the conversion into a character string, the string is stored in the devices specified by d (Array\_d[1]) through d+4 (Array\_d[5]) as follows:

A positive sign of the binary data is stored as ASCII character "20H" (blank).

A negative sign of the binary data is stored as ASCII character "2DH" ("minus"- character).

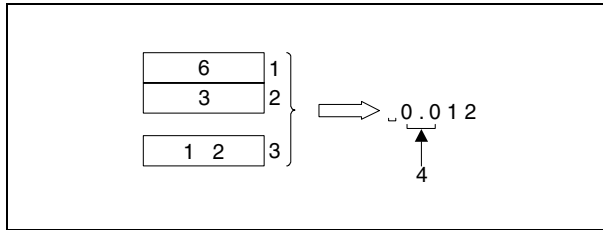
If the number of decimal places is greater than zero, the decimal point "2EH" (.) is placed automatically before the first digit specified.



- <sup>1</sup>Total of all digits
- <sup>2</sup>Number of decimal places
- <sup>3</sup>Binary value
- <sup>4</sup>Decimal point placed automatically
- <sup>5</sup>Decimal places

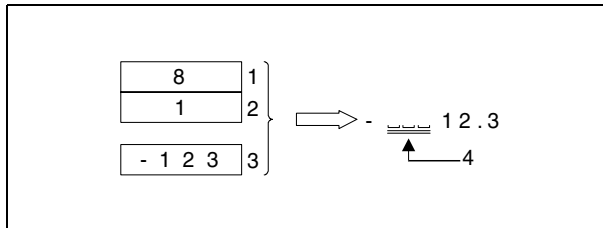
If the number of decimal places equals zero, the decimal point character "2DH" (.) is not placed.

If the number of decimal places is greater than the number of digits of the binary value, the missing digits are replaced by zeroes, the binary value is shifted to the right, and the decimal point is placed accordingly (0.□□□□□).



- <sup>1</sup>Total of all digits
- <sup>2</sup>Number of decimal places
- <sup>3</sup>Binary value
- <sup>4</sup>Zeroes and decimal point placed automatically

If the number of digits, sign and decimal point included, is greater than the number of digits in the binary value, the missing digits between sign and numerical value are replaced by "20H" (blanks) automatically.



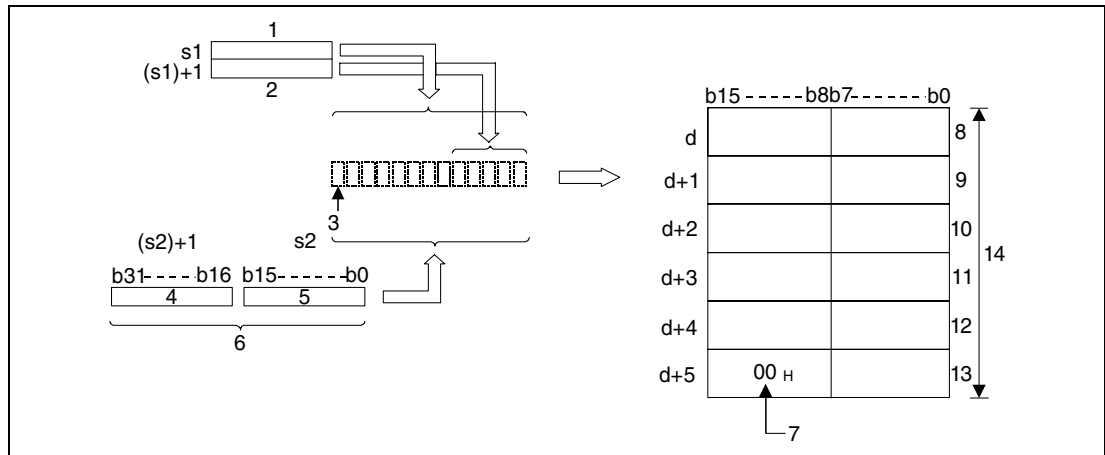
- <sup>1</sup>Total of all digits
- <sup>2</sup>Number of decimal places
- <sup>3</sup>Binary value
- <sup>4</sup>Blank characters placed automatically.

At the end of the converted character string the character code "00H" is stored automatically.

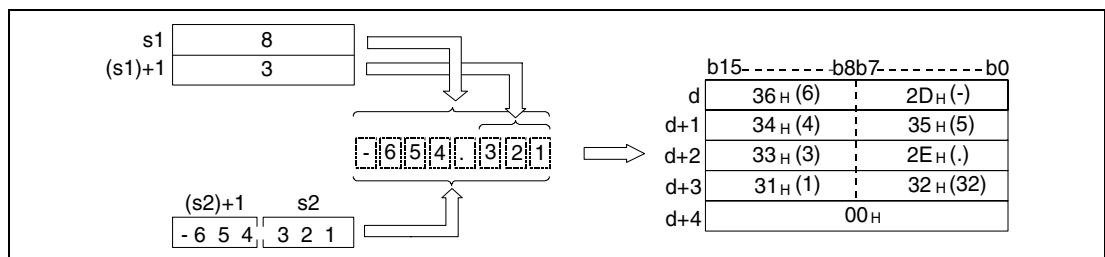


**DSTR Conversion of BIN 32-bit data**

The DSTR instruction adds a decimal point to the BIN 32-bit binary value in the device specified by s2 and (s2)+1 to the digit specified by the devices s1 and (s1)+1, converts the data into a character string, and stores it in the area of the devices specified by d (Array\_d[1]) through d+5 (Array\_d[6]).



- 1 Total of all digits
- 2 Decimal places
- 3 Sign
- 4 Upper 16 Bit
- 5 Lower 16 Bit
- 6 Binary value
- 7 End of character string indication, automatically placed.
- 8 Character position in ASCII; total of digits -1/ ASCII code of the sign
- 9 Character position in ASCII; total of digits -3/ character position in ASCII; total of digits -2
- 10 Character position in ASCII; total of digits -5/ character position in ASCII; total of digits -4
- 11 Character position in ASCII; total of digits -7/ character position in ASCII; total of digits -6
- 12 Character position in ASCII; total of digits -9/ character position in ASCII; total of digits -8
- 13 End of character string indication/ character position in ASCII; total of digits -10
- 14 Total of all digits



The number of digits that can be stored in the device specified by s1 ranges from 2 to 13.

The number of decimal places that can be stored in the devices specified by (s1)+1 ranges from 0 to 10 and must not exceed the number of digits minus 3.

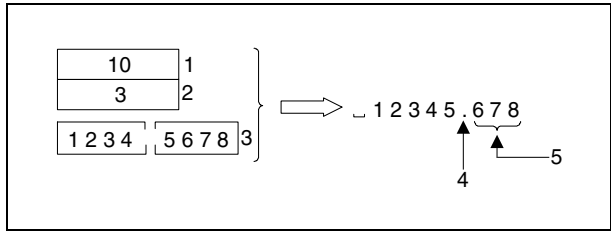
The BIN 32-bit data that can be stored in the device specified by s2 and (s2)+1 must range from -2147483648 and 32147483647.

After the conversion into a character string, the string is stored in the devices specified by d (Array\_d[1]) bis d+5 (Array\_d[6]) as follows:

A positive sign of the binary data is stored as ASCII character "20H" (blank).

A negative sign of the binary data is stored as ASCII character "2DH" ("minus"- character).

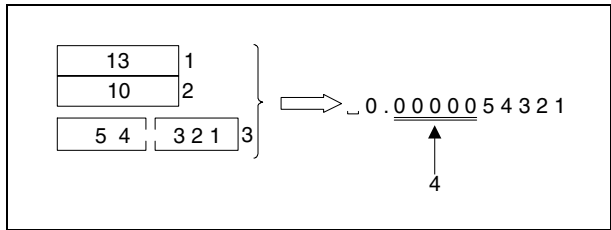
If the number of decimal places is greater than zero, the decimal point "2EH" (.) is placed automatically before the first digit specified.



- <sup>1</sup>Total of all digits
- <sup>2</sup>Number of decimal places
- <sup>3</sup>Binary value
- <sup>4</sup>Decimal point placed automatically
- <sup>5</sup>Decimal places

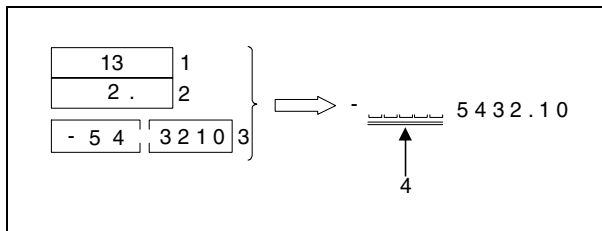
If the number of decimal places equals zero, the decimal point character "2DH" (.) is not placed.

If the number of decimal places is greater than the number of digits of the binary value, the missing digits are replaced by zeroes, the binary value is shifted to the right, and the decimal point is placed accordingly (0.□□□□).



- <sup>1</sup>Total of all digits
- <sup>2</sup>Decimal places
- <sup>3</sup>Binary value
- <sup>4</sup>Zeroes and decimal point placed automatically

If the number of digits, sign and decimal point included, is greater than the number of digits in the binary value, the missing digits between sign and numerical value are replaced by "20H" (blanks) automatically.



<sup>1</sup>Total of all digits

<sup>2</sup>Number of decimal places

<sup>3</sup>Binary value

<sup>4</sup>Blank characters placed automatically.

At the end of the converted character string the character code "00H" is stored automatically.

### Operation Errors

In the following cases an operation error occurs and the error flag is set:

- The number of digits stored in s1 exceeds the range of values specified below (error code 4100):  
Range of values for the STR instruction: 2 to 8  
Range of values for the DSTR instruction: 2 to 13
- The number of decimal places stored in (s1)+1 exceeds the range of values specified below (error code 4100):  
Range of values for the STR instruction: 0 to 5  
Range of values for the DSTR instruction: 0 to 10
- The values stored in s1 and (s1)+1 do not correspond to the following relation:  
The total of all digits minus 3 is greater than or equal to the number of decimal places (error code 4100).
- The number of digits stored in s1 and (s1)+1 is less than the digits of the binary values in s2 and (s2)+1 (error code 4100).
- The area storing the character string specified from d (Array\_d[1]) onwards exceeds the relevant device range (error code 4100).

**Program Example 1**

**STRP**

With leading edge from X0, the following program converts the binary value specified by D10 corresponding to the number of digits specified in D0 and D1. The result is stored in the area from D20 (var\_D20 Array [1]) through D23 (var\_D20 Array [4]).

<p style="text-align: center;"><b>MELSEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10px;">MELSEC</td><td style="width: 10px;">LD</td><td style="width: 10px;">X0</td><td></td></tr> <tr><td></td><td>MOV</td><td>K12672</td><td>D10</td></tr> <tr><td></td><td>MOV</td><td>K6</td><td>D0</td></tr> <tr><td></td><td>MOV</td><td>K0</td><td>D1</td></tr> <tr><td></td><td>STRP</td><td>D0</td><td>D10</td></tr> <tr><td></td><td></td><td>D10</td><td>D20</td></tr> </table>	MELSEC	LD	X0			MOV	K12672	D10		MOV	K6	D0		MOV	K0	D1		STRP	D0	D10			D10	D20	<p style="text-align: center;"><b>Ladder Diagram</b></p>												
MELSEC	LD	X0																																			
	MOV	K12672	D10																																		
	MOV	K6	D0																																		
	MOV	K0	D1																																		
	STRP	D0	D10																																		
		D10	D20																																		
<p style="text-align: center;"><b>IEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10px;">LD</td><td style="width: 10px;">X0</td><td></td></tr> <tr><td>MOV</td><td>M</td><td>12672, D10</td></tr> <tr><td>INT_TO_DWORD</td><td>E</td><td>6, var_Merker</td></tr> <tr><td>INT_TO_DWORD</td><td>E</td><td>0, var_D0</td></tr> <tr><td>SHL</td><td>E</td><td>var_D0, K16, var_D0</td></tr> <tr><td>LD</td><td></td><td>var_Merker</td></tr> <tr><td>OR</td><td></td><td>var_D0</td></tr> <tr><td>ST</td><td></td><td>var_D0</td></tr> <tr><td>LD</td><td></td><td>X0</td></tr> <tr><td>STR_P</td><td>MD</td><td>var_D0, D10, var_D20</td></tr> </table>	LD	X0		MOV	M	12672, D10	INT_TO_DWORD	E	6, var_Merker	INT_TO_DWORD	E	0, var_D0	SHL	E	var_D0, K16, var_D0	LD		var_Merker	OR		var_D0	ST		var_D0	LD		X0	STR_P	MD	var_D0, D10, var_D20							
LD	X0																																				
MOV	M	12672, D10																																			
INT_TO_DWORD	E	6, var_Merker																																			
INT_TO_DWORD	E	0, var_D0																																			
SHL	E	var_D0, K16, var_D0																																			
LD		var_Merker																																			
OR		var_D0																																			
ST		var_D0																																			
LD		X0																																			
STR_P	MD	var_D0, D10, var_D20																																			
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;"></td> <td style="width: 10%; text-align: center;">D10</td> <td style="width: 10%; border: 1px solid black; text-align: center;">12672</td> <td style="width: 10%; text-align: center;">⇒</td> <td style="width: 10%;"></td> <td style="width: 10%; text-align: center;">D20</td> <td style="width: 10%; border: 1px solid black; text-align: center;">31H (1)</td> <td style="width: 10%; border: 1px solid black; text-align: center;">20H</td> <td style="width: 10%;"></td> </tr> <tr> <td></td> <td style="text-align: center;">D0</td> <td style="border: 1px solid black; text-align: center;">6</td> <td></td> <td></td> <td style="text-align: center;">D21</td> <td style="border: 1px solid black; text-align: center;">36H (6)</td> <td style="border: 1px solid black; text-align: center;">32H (2)</td> <td></td> </tr> <tr> <td></td> <td style="text-align: center;">D1</td> <td style="border: 1px solid black; text-align: center;">0</td> <td></td> <td></td> <td style="text-align: center;">D22</td> <td style="border: 1px solid black; text-align: center;">32H (2)</td> <td style="border: 1px solid black; text-align: center;">37H (7)</td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;">D23</td> <td colspan="2" style="border: 1px solid black; text-align: center;">00H</td> <td style="text-align: right;">“ 12672 ”</td> </tr> </table>			D10	12672	⇒		D20	31H (1)	20H			D0	6			D21	36H (6)	32H (2)			D1	0			D22	32H (2)	37H (7)							D23	00H		“ 12672 ”
	D10	12672	⇒		D20	31H (1)	20H																														
	D0	6			D21	36H (6)	32H (2)																														
	D1	0			D22	32H (2)	37H (7)																														
					D23	00H		“ 12672 ”																													

**Program Example 2** DSTRP

With leading edge from X0, the following program converts the binary value specified in D10 and D11 corresponding to the number of digits specified in D0 and D1. The result is stored in the area from D20 (var\_D20 Array [1]) through D26 (var\_D20 Array [7]).

<p style="text-align: center;"><b>MELSEC Instruction List</b></p> <pre> MELSEC LD      X0 DMOV_P M 12345678, var_D11 MOV_P  D11, K6 MOV_P  D2,  D0 MOV_P  K0,  D3 DSTRP  D2,  D11, D30         </pre>	<p style="text-align: center;"><b>Ladder Diagram</b></p>																																																								
<p style="text-align: center;"><b>IEC Instruction List</b></p> <pre> LD      X0 DMOV_P M 12345678, var_D11 INT_TO_DWORD_E 6, var_Marker INT_TO_DWORD_E 0, var_D1 SHL_E   var_D1, K16, var_D1 LD      var_Marker OR      var_D1 ST      var_D1 LD      X0 DSTR_P_MD var_D1, var_D11, var_D30         </pre>	<p style="text-align: center;"><b>Memory Map</b></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"></td> <td style="width: 15%; text-align: center;">D12</td> <td style="width: 15%; text-align: center;">D11</td> <td style="width: 10%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">12345678</td> <td></td> <td style="text-align: center;">⇒</td> <td style="border: 1px solid black; text-align: center;">D30</td> <td style="border: 1px solid black; text-align: center;">30 H (0)</td> <td style="border: 1px solid black; text-align: center;">20 H</td> </tr> <tr> <td style="text-align: center;">D0</td> <td style="border: 1px solid black; text-align: center;">12</td> <td></td> <td></td> <td style="border: 1px solid black; text-align: center;">D31</td> <td style="border: 1px solid black; text-align: center;">30 H (0)</td> <td style="border: 1px solid black; text-align: center;">2E H (.)</td> </tr> <tr> <td style="text-align: center;">D1</td> <td style="border: 1px solid black; text-align: center;">9</td> <td></td> <td></td> <td style="border: 1px solid black; text-align: center;">D32</td> <td style="border: 1px solid black; text-align: center;">32 H (2)</td> <td style="border: 1px solid black; text-align: center;">31 H (1)</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td style="border: 1px solid black; text-align: center;">D33</td> <td style="border: 1px solid black; text-align: center;">34 H (4)</td> <td style="border: 1px solid black; text-align: center;">33 H (3)</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td style="border: 1px solid black; text-align: center;">D34</td> <td style="border: 1px solid black; text-align: center;">36 H (6)</td> <td style="border: 1px solid black; text-align: center;">35 H (5)</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td style="border: 1px solid black; text-align: center;">D35</td> <td style="border: 1px solid black; text-align: center;">38 H (8)</td> <td style="border: 1px solid black; text-align: center;">37 H (7)</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td style="border: 1px solid black; text-align: center;">D36</td> <td colspan="2" style="border: 1px solid black; text-align: center;">00 H</td> </tr> </table> <p style="text-align: right; margin-top: 10px;">“ 0.012345678 ”</p>		D12	D11						12345678		⇒	D30	30 H (0)	20 H	D0	12			D31	30 H (0)	2E H (.)	D1	9			D32	32 H (2)	31 H (1)					D33	34 H (4)	33 H (3)					D34	36 H (6)	35 H (5)					D35	38 H (8)	37 H (7)					D36	00 H	
	D12	D11																																																							
	12345678		⇒	D30	30 H (0)	20 H																																																			
D0	12			D31	30 H (0)	2E H (.)																																																			
D1	9			D32	32 H (2)	31 H (1)																																																			
				D33	34 H (4)	33 H (3)																																																			
				D34	36 H (6)	35 H (5)																																																			
				D35	38 H (8)	37 H (7)																																																			
				D36	00 H																																																				

**NOTE** These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

7.11.10 VAL, VALP, DVAL, DVALP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

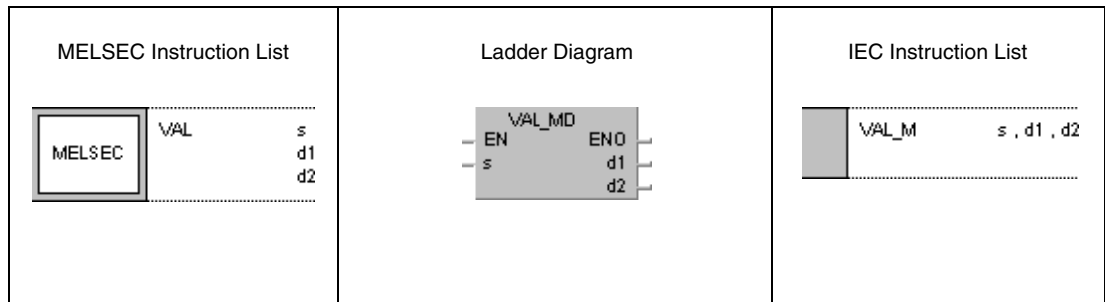
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

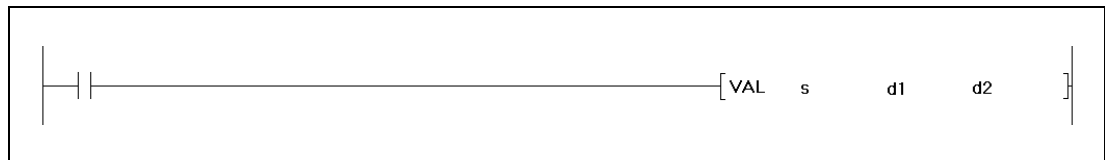
Devices  
MELSEC Q

Usable Devices									Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other		
Bit	Word		Bit	Word						
—	●	●	—	—	—	—	●	—	SM0	4
●	●	●	—	—	—	—	—	—		
●	●	●	●	●	●	●	—	—		

GX IEC  
Developer



GX  
Developer



Variables

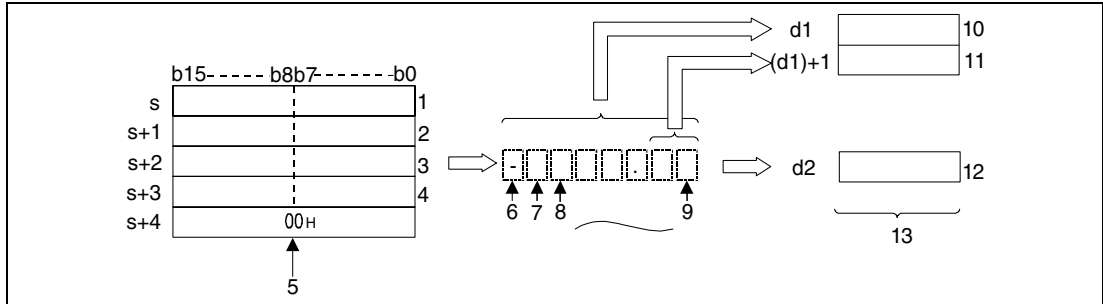
Set Data	Meaning	Data Type	
		MELSEC	IEC
s	First number of device storing the character string of the binary data to be converted.	Character string	Array [1..5]/ [1..7] of ANY16
d1	First number of device storing the number of digits of the binary data after conversion.	BIN 16-bit	ANY32
d2	Initial number of device storing the converted binary data.	BIN 16-/32-bit	ANY16/32

**Functions Conversion of character strings into BIN 16-/32-bit binary data**

**VAL Conversion into BIN 16-bit binary data**

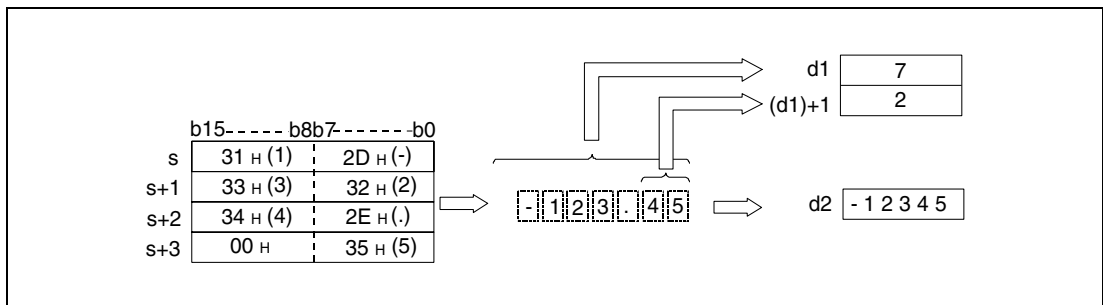
The VAL instruction converts the character strings stored in the area s (Array\_s[1]) through s+4 (Array\_s[5]) into BIN 16-bit data. The number of digits and the binary value are stored in d1, (d1)+1, and d2.

For the conversion into the BIN 16-bit data format all data in the area s (Array\_s[1]) through s+4 (Array\_s[5]) is recognized as character string up to the character code "00H".



- <sup>1</sup> ASCII code for the 1st character/ ASCII code for the sign
- <sup>2</sup> ASCII code for the 3rd character/ ASCII code for the 2nd character
- <sup>3</sup> ASCII code for the 5th character/ ASCII code for the 4th character
- <sup>4</sup> ASCII code for the 7th character/ ASCII code for the 6th character
- <sup>5</sup> Indicates the end of the character string
- <sup>6</sup> Sign character
- <sup>7</sup> 1st character
- <sup>8</sup> 2nd character
- <sup>9</sup> 7th character
- <sup>10</sup> Total of all digits
- <sup>11</sup> Number of decimal places
- <sup>12</sup> Integer value, the decimal point is not processed
- <sup>13</sup> BIN 16-bit

The character string "-123.45" in the area s (Array\_s[1]) through s+4 (Array\_s[5]) is to be converted. The result will be stored in d1, (d1)+1 and d2 as follows:



The number of all characters stored in s (Array\_s[1]) through s+4 (Array\_s[5]) may range from 2 to 8.

The number of possible decimal places stored in the area s (Array\_s[1]) through s+4 (Array\_s[5]) may range from 0 to 5. In general the number of decimal places must not exceed the total of all digits minus 3.

The numerical value of a character string to be converted with the decimal point ignored must range from -32768 to 32767.

The numerical value of the ASCII character string with the sign character and decimal point ignored must range from "30H" and "39H".

A positive sign of the binary data is stored as ASCII character "20H" (blank).

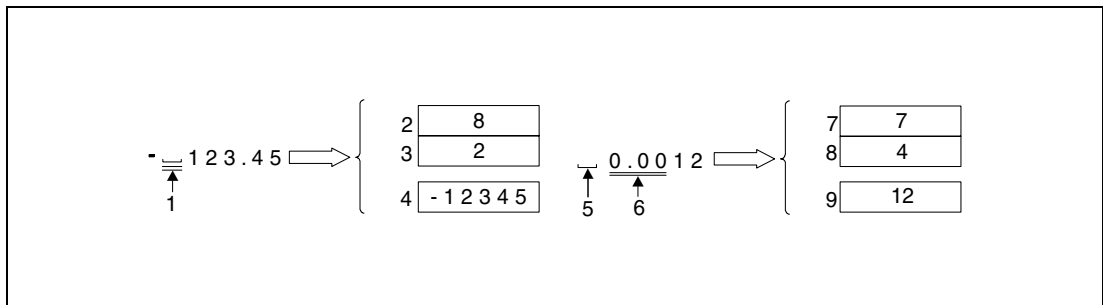
A negative sign of the binary data is stored as ASCII character "2DH" ("minus"- character).

The ASCII character "2EH" is stored as decimal point.

The total of all digits stored in d1, (d1)+1, and d2 contains all characters that represent the numerical value as well as the sign character d1 and the decimal places (d1)+1.

In the binary data stored in d2 after the conversion the decimal point is ignored.

If the characters "20H" (blank) or "30H" (zero) are stored between character sign and first numerical value, these are ignored for the conversion.



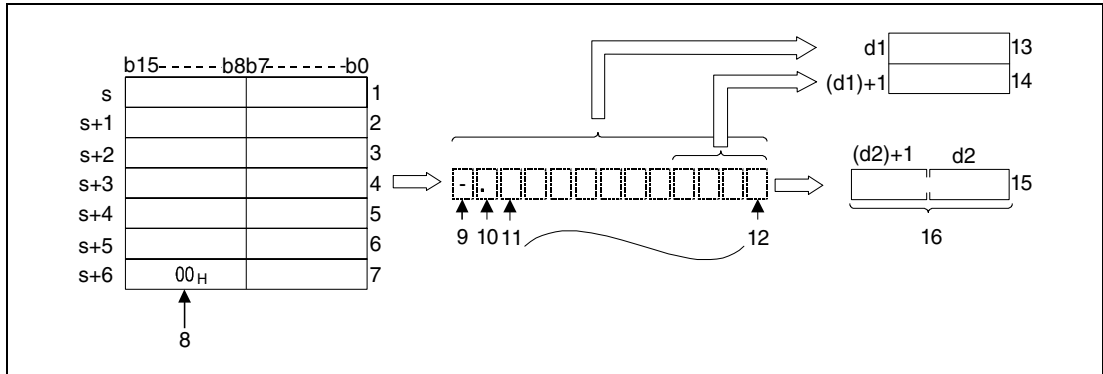
- <sup>1</sup> These characters are not processed
- <sup>2</sup> Total of all digits
- <sup>3</sup> Number of decimal places
- <sup>4</sup> Binary value
- <sup>5</sup> Sign character
- <sup>6</sup> These characters are not processed
- <sup>7</sup> Total of all digits
- <sup>8</sup> Number of decimal places
- <sup>9</sup> Binary value



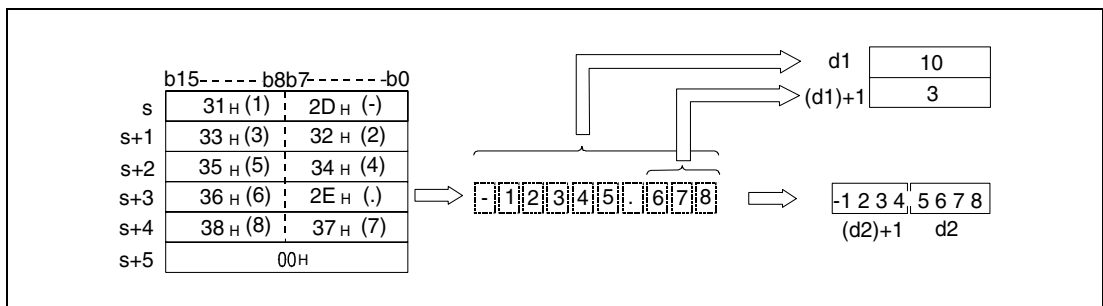
**DVAL Conversion into BIN 32-bit data**

The DVAL instruction converts the character strings stored in s (Array\_s[1]) through s+6 (Array\_s[7]) into BIN 32-bit data. The number of digits and the binary value are stored in d1, (d1)+1, d2 and (d2)+1.

For the conversion into the BIN 32-bit binary format all data in the area s (Array\_s[1]) through s+6 (Array\_s[7]) up to the character code "00H" are recognized as character string.



- 1 ASCII code for the 1st character/ ASCII code for the sign character
- 2 ASCII code for the 3rd character/ ASCII code for the 2nd character
- 3 ASCII code for the 5th character/ ASCII code for the 4th character
- 4 ASCII code for the 7th character/ ASCII code for the 6th character
- 5 ASCII code for the 9th character/ ASCII code for the 8th character
- 6 ASCII code for the 11th character/ ASCII code for the 10th character
- 7 ASCII code for the zero character/ ASCII code for the 12th character
- 8 Indicates the end of the character string
- 9 Sign character
- 10 1st character
- 11 2nd character
- 12 12th character
- 13 Total of all digits
- 14 Number of decimal places
- 15 Integer value, the decimal point is not processed
- 16 BIN 32-bit



The total of all characters stored in s (Array\_s[1]) through s+6 (Array\_s[7]) may range from 2 to 13.

The number of possible decimal places stored in the area s (Array\_s[1]) through s+6 (Array\_s[7]) may range from 0 to 10. In general the number of decimal places must not exceed the total of all digits minus 3.

The numerical value of a character string to be converted with the decimal point ignored must range from -2147483648 to 2147483647.

The numerical value of the ASCII character string with the sign character and decimal point ignored must range from "30H" and "39H".

A positive sign of the binary data is stored as ASCII character "20H" (blank).

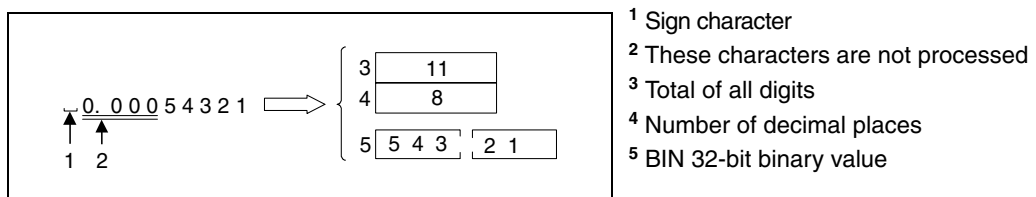
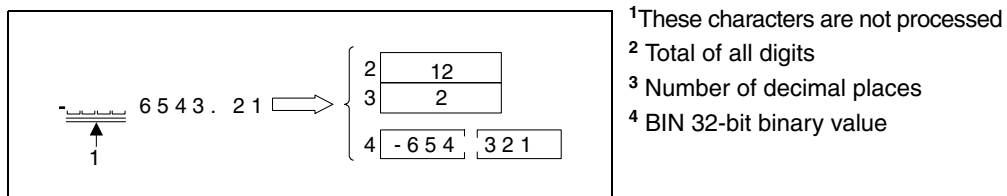
A negative sign of the binary data is stored as ASCII character "2DH" ("minus"- character).

The ASCII character "2EH" is stored as decimal point.

The total of all digits stored in d1, (d1)+1, d2, and (d2)+1 contains all characters that represent the numerical value as well as the sign character d1 and the decimal places (d1)+1.

In the binary data stored in d2 and (d2)+1 after the conversion the decimal point is ignored.

If the characters "20H" (blank) or "30H" (zero) are stored between character sign and first numerical value, these are ignored for the conversion.



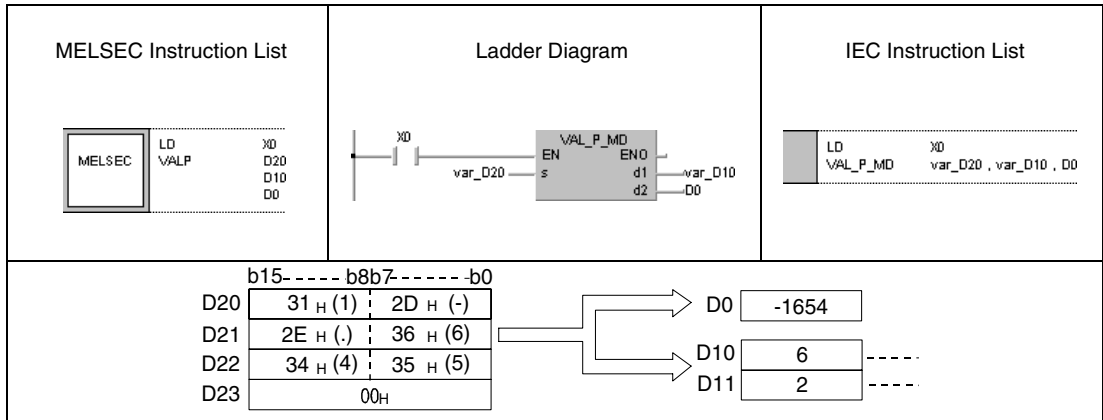
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The total of all digits stored from s (Array\_s[1]) onwards exceeds the range of values from 2 to 8 (VAL) or 2 to 13 (DVAL) respectively (error code 4101).
- The number of decimal places stored in (d1)+1 exceeds the range of values from 0 to 5 (VAL) or 0 to 10 (DVAL) respectively (error code 4100).
- The total of all digits minus 3 is greater than or equal to the number of decimal places (error code 4100).
- Different ASCII characters than "20H" or "2DH" were stored for the character sign (error code 4100).
- Different ASCII characters than "30H", "39H", or "2EH" were stored in a value (error code 4100).
- More than one decimal point is stored in one value (error code 4100).
- The binary value exceeds the range of values from -32768 to 32767 (VAL) or -2147483648 to 2147483647 (DVAL) after the conversion (error code 4100).
- The ASCII character "00H" is placed to the wrong digit (error code 4100).

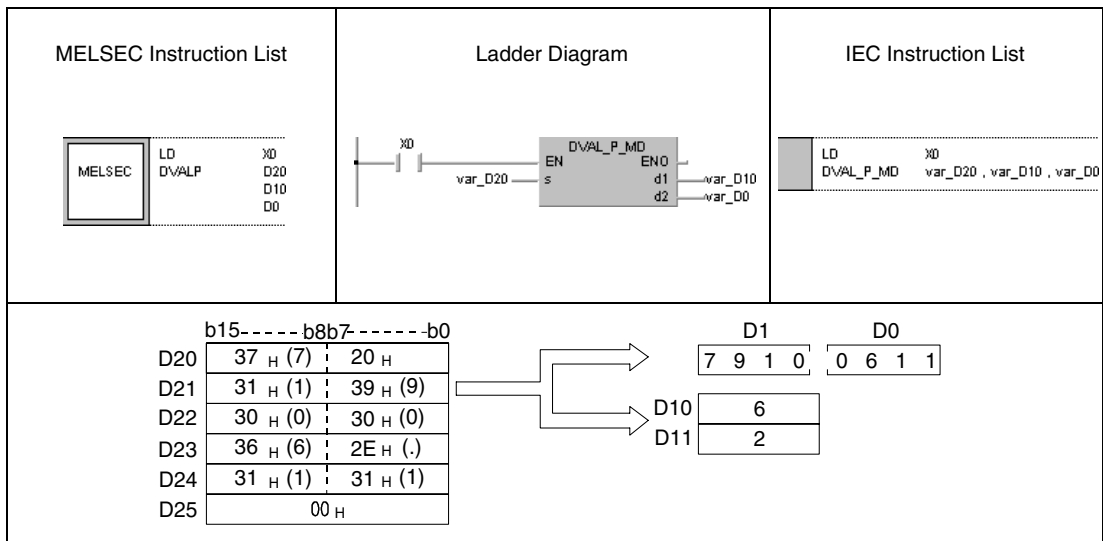
**Program Example 1** VALP

With leading edge from X0, the following program converts the character string stored in the area D20 (var\_ D20 Array [1]) through D23 (var\_ D20 Array [4]) into an integer value, converts this value into a BIN 16-bit binary value, and stores it in D0.



**Program Example 2** DVALP

With leading edge from X0, the following program converts the character string stored in the area D20 (var\_ D20 Array [1]) through D24 (var\_ D20 Array [5]) into an integer value, converts this value into a BIN 32-bit value, and stores it in D0 and D1.



**NOTE** *These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.11 ESTR, ESTRP

CPU


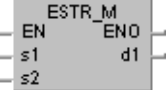
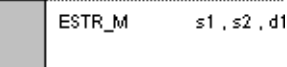
AnS	AnN	AnA, AnAS	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

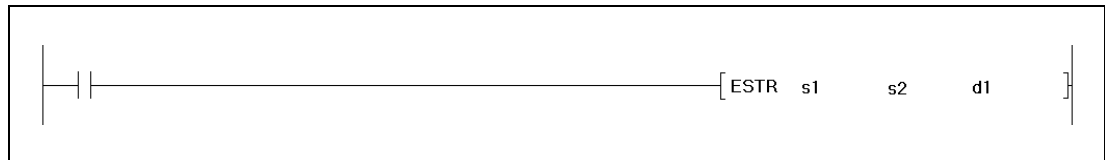
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant E	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	●	●	—	●	—	SM0	4
s2	—	●	●	—	—	—	—	—	—		
d	—	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

GX Developer



Variables

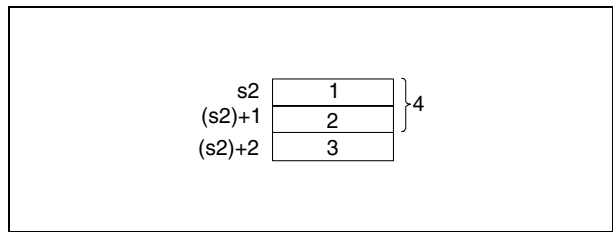
Set Data	Meaning	Data type	
		MELSEC	IEC
s1	Floating point data to be converted or initial number of device storing such data.	Real number	Real number
s2	First number of device storing the data format of the numeric data to be converted.	BIN 16-bit	Array [1..3] of ANY16
d	First number of device storing the converted data.	Character string	Character string

**Functions Conversion of floating point data into character string data**

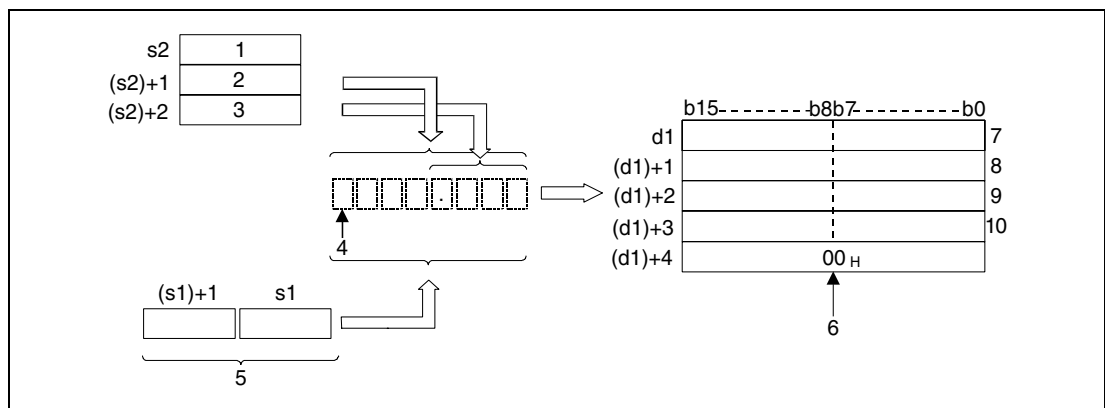
**ESTR Conversion of floating point data**

The ESTR instruction converts the floating point data (real numbers) in s1 and (s1)+1 into character string data. The data format of the character string is specified in s2 (Array\_s2[1]) through (s2)+2 (Array\_s2[3]). The result is stored from d onwards.

The data format after the conversion depends on the data format in s2 (Array\_s2[1]) through (s2)+2 (Array\_s2[3]).



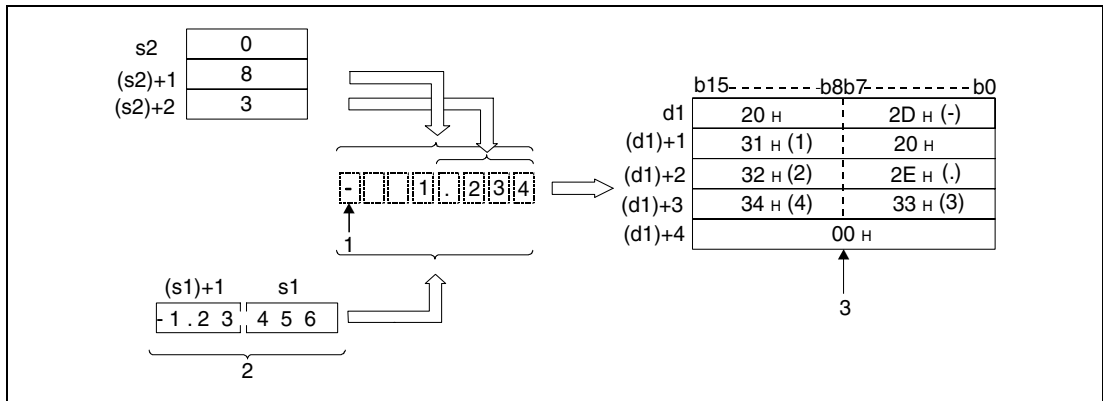
- <sup>1</sup> Data format (decimal format "0"/ exponential format "1")
- <sup>2</sup> Total of all digits
- <sup>3</sup> Number of decimal places



- <sup>1</sup> Data format (decimal format "0"/ exponential format "1")
- <sup>2</sup> Total of all digits
- <sup>3</sup> Number of decimal places
- <sup>4</sup> Sign character
- <sup>5</sup> Floating point data (real number)
- <sup>6</sup> End of character string, placed automatically
- <sup>7</sup> Character position in ASCII; total of digits -1/ ASCII code of the sign
- <sup>8</sup> Character position in ASCII; total of digits -3/ character position in ASCII; total of digits -2
- <sup>9</sup> Character position in ASCII; total of digits -5/ character position in ASCII; total of digits -4
- <sup>10</sup> Character position in ASCII; total of digits -7/ character position in ASCII; total of digits -6

**Decimal format**

The real number -1.23456 is converted into a character string with a total of 8 digits (3 decimal places included). The result is stored from d onwards.



- <sup>1</sup> Sign character
- <sup>2</sup> Floating point data (real number)
- <sup>3</sup> End of character string, automatically placed

The total number of all digits of the number in (s2)+1 (Array\_s2[2]) to be converted is represented as follows:

If the number of decimal places is zero, the total number of digits is >= 2.

If the number of the decimal places is a different value, the total number of all digits is 3 plus the number of decimal places.

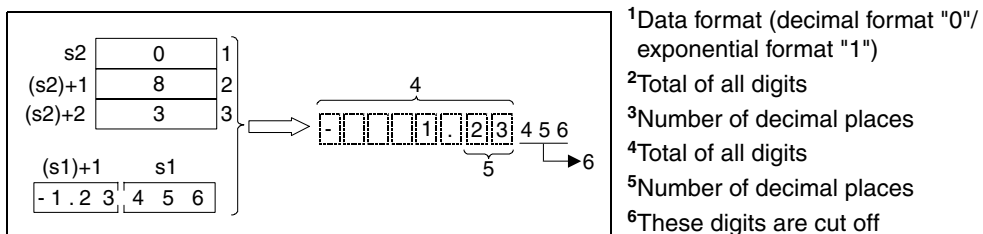
The number of decimal places that has to be specified must range within 0 and 7. In general, the number of decimal places must be less than or equal to the total number of all digits minus 3.

After the conversion the character string in d is stored as follows:

A positive sign of the floating point data is stored as ASCII character "20H" (blank).

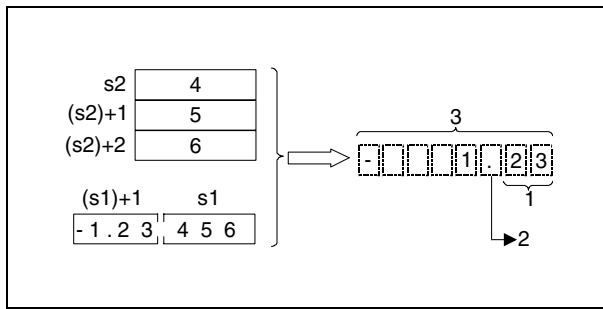
A negative sign of the floating point data is stored as ASCII character "2DH" ("minus"-character).

In cases where the actual number of decimal places of the floating point data exceeds the specified number of decimal places, the surplus digits are cut off.



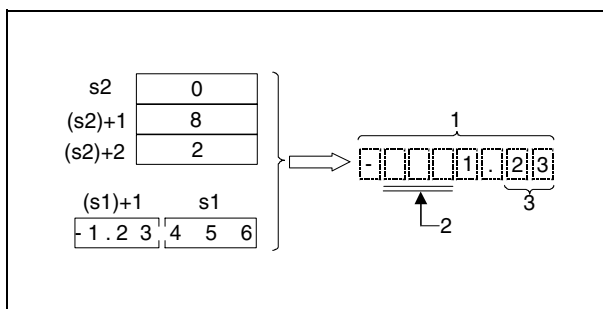
If the number of decimal places is specified a value different from zero, the decimal point "2EH" (.) is placed automatically in the specified digit.

If the number of decimal places is specified zero the decimal point "2EH" (.) is not placed.



- <sup>1</sup> Number of decimal places
- <sup>2</sup> Decimal point is placed and stored automatically
- <sup>3</sup> Total of all digits

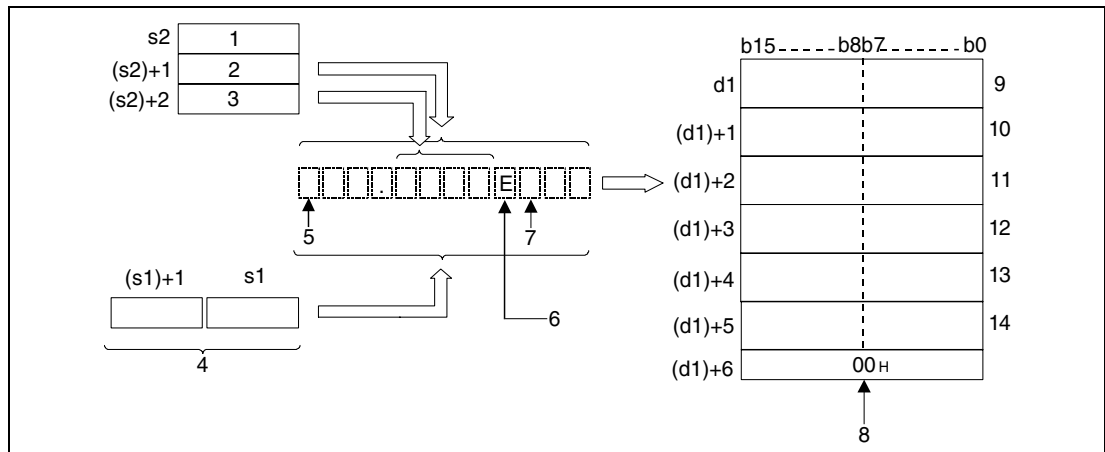
If the total number of all digits to be represented without sign character is less than the number of decimal point and decimal places, the digits between the sign character and the first digit to be represented are replaced by the character codes "20H" (blanks).



- <sup>1</sup> Total of all digits
- <sup>2</sup> Blanks "20H" are stored
- <sup>3</sup> Number of decimal places

The character code "00H" is stored automatically at the end of the character string.

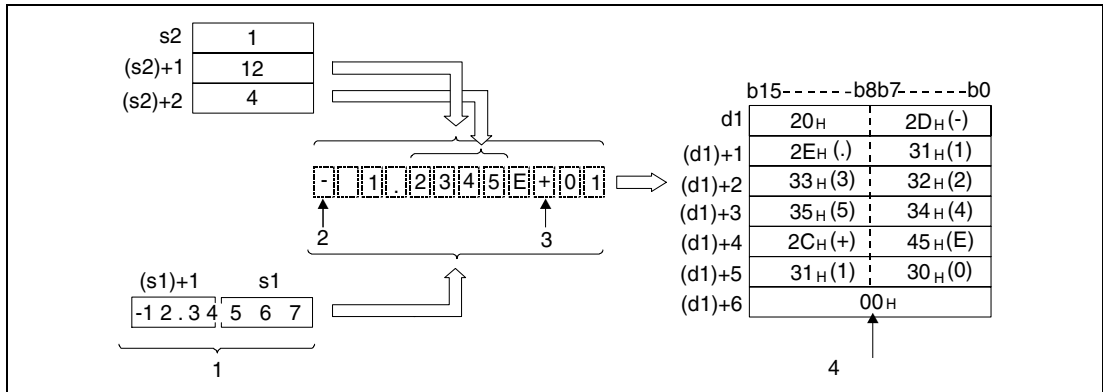
**Exponential format**



- 1 Data format (Exponential format) (1)
- 2 Total number of all digits
- 3 Number of decimal places
- 4 Floating point number (real number)
- 5 Sign of the integer value
- 6 The "E" is placed automatically
- 7 Sign of the exponent
- 8 End of character string indication, placed automatically
- 9 Character position in ASCII; total of digits -1/ ASCII code of the sign
- 10 Character position in ASCII; total of digits -3/ character position in ASCII; total of digits -2
- 11 Character position in ASCII; total of digits -5/ character position in ASCII; total of digits -4
- 12 Character position in ASCII; total of digits -7/ character position in ASCII; total of digits -6
- 13 Sign of the exponent/ 45H (E)
- 14 Character position in ASCII; total of digits -11 (exponent)/ character position in ASCII; total of digits -10 (exponent)



The real number -12.34567 is to be represented in exponential notation. The total number of all digits is 12. The number of decimal digits is specified 4. The result is stored from d onwards.



- <sup>1</sup> Floating point number (real number)
- <sup>2</sup> Sign of the integer value
- <sup>3</sup> Sign of the exponent
- <sup>4</sup> End of character string indication, placed automatically

The total number of all digits of the number in (s2)+1 (Array\_s2[2]) to be converted is represented as follows:

If the number of decimal places is zero, the total number of digits is  $\geq 2$ .

If the number of the decimal places is a different value, the total number of all digits is 7 plus the number of decimal places.

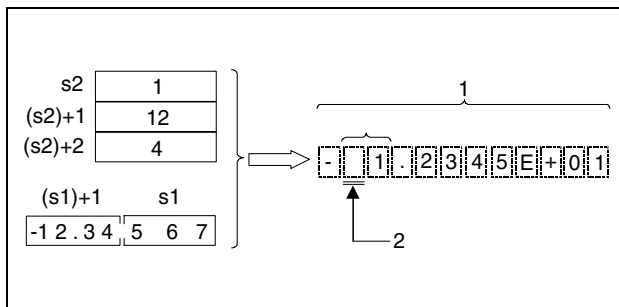
The number of decimal places that has to be specified must range within 0 and 7. In general, the number of decimal places must be less than or equal to the total number of all digits minus 7.

After the conversion the character string in d is stored as follows:

A positive sign of the floating point data is stored as ASCII character "20H" (blank).

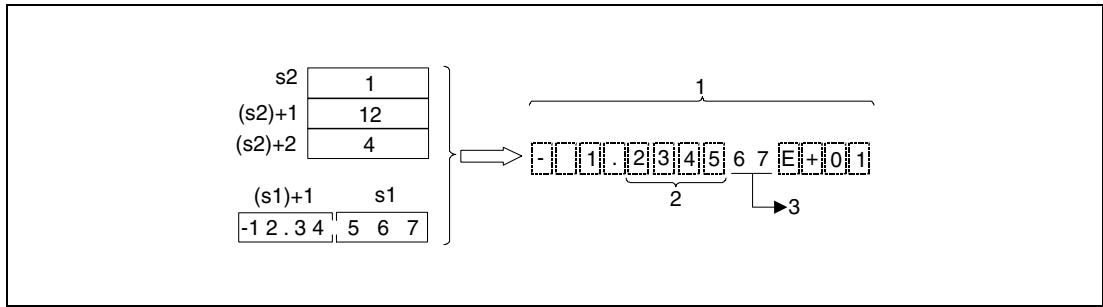
A negative sign of the floating point data is stored as ASCII character "2DH" ("minus"-character).

The integer range is fixed to 2 digits. If the integer range contains one digit only, a blank in ASCII code is placed and stored between the sign character and the integer digit.



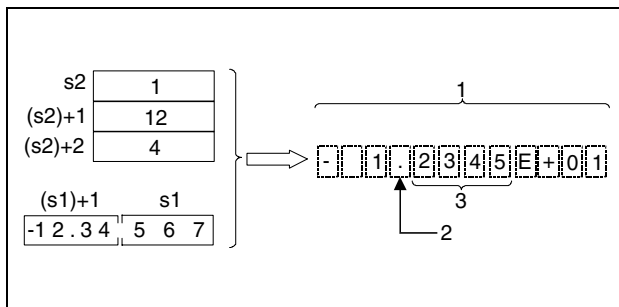
- <sup>1</sup> Total of all digits (12)
- <sup>2</sup> Becomes a blank

If the floating point value of the decimal range is longer than the relevant storage range, the digits that cannot be stored are cut off.



- <sup>1</sup> Total of all digits (12)
- <sup>2</sup> Number of digits in the decimal range (4)
- <sup>3</sup> These digits are cut off

If the number of decimal places is specified a value different from zero, the decimal point "2EH" (.) is placed automatically in the specified digit.



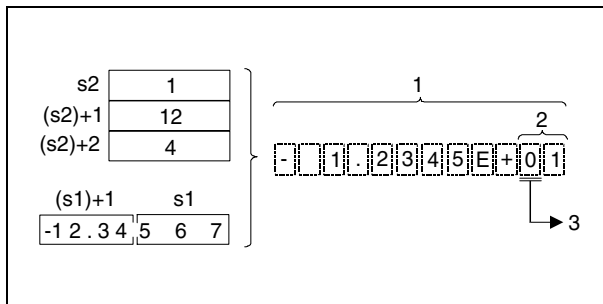
- <sup>1</sup> Total of all digits (12)
- <sup>2</sup> Is placed automatically
- <sup>3</sup> Number of digits in the decimal range (4)

If the number of decimal places is specified zero the decimal point "2EH" (.) is not placed.

The ASCII code "2CH" (+) is placed and stored for a positive exponent.

The ASCII code "2DH" (-) is placed and stored for a negative exponent.

The exponential range is fixed to 2 digits. If the exponential range contains one digit only, the ASCII code "30H" (0) is placed and stored between the exponent sign and the exponent.



- <sup>1</sup> Total of all digits (12)
- <sup>2</sup> Is fixed to 2 digits
- <sup>3</sup> Is set to zero automatically

The character code "00H" is stored automatically at the end of the character string.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The devices specified in s1 and (s1)+1 are not zero or exceed the range of values of  $\pm 2^{-127} \leq s1 < \pm 2^{129}$  (error code 4100).
- The format in s2 (Array\_s2[1]) is neither 0 nor 1 (error code 4100).
- The total number of digits in (s2)+1 (Array\_s2[2]) exceeds the range of values (error code 4100):

For the decimal format

- The number of decimal places is zero (total number of digits  $\geq 2$ ).
- The number of decimal places is different from zero (total number of digits  $\geq$  (number of decimal places + 3)).

For the exponential format

- The number of decimal places is zero (total number of digits  $\geq 2$ ).
- The number of decimal places is different from zero (total number of digits  $\geq$  (number of decimal places + 7)).

- The number of digits in (s2)+2 (Array\_s2[3]), forming the decimal part exceeds the range of values (error code 4100):

For the decimal format

The number of digits forming the decimal part is less than or equal to the total number of digits minus 3.

For the exponential format

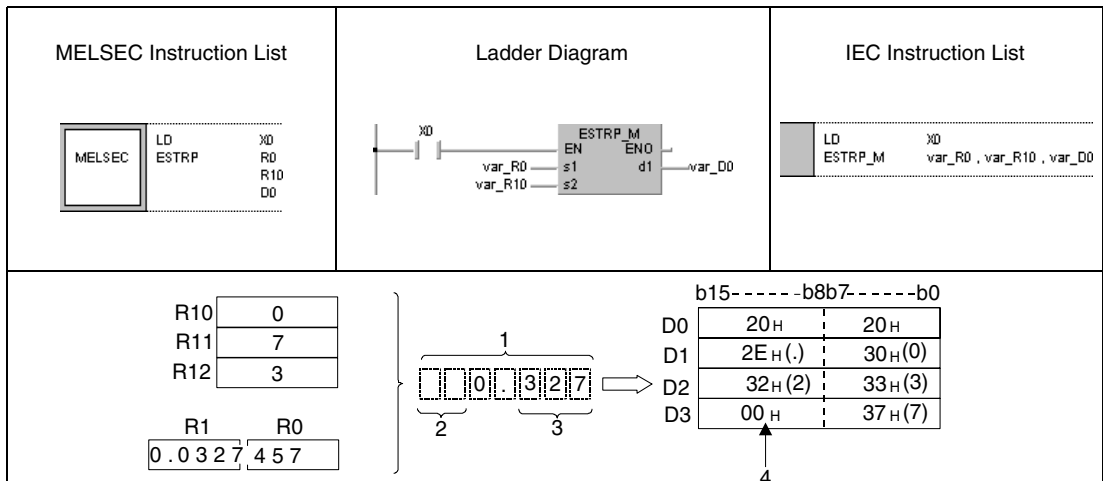
The number of digits forming the decimal part is less than or equal to the total number of digits minus 7.

- The storage range in d exceeds the relevant storage device range (error code 4101).

**Program Example 1**

**ESTRP**

With leading edge from X0, the following program converts a floating point value (real number) specified by the devices R0 and R1 into the format specified by R10 (var\_R10 Array [1]) through R12 (var\_R10 Array [3]) and stores the result in D0 through D3.



1 Total number of digits

2 Blanks

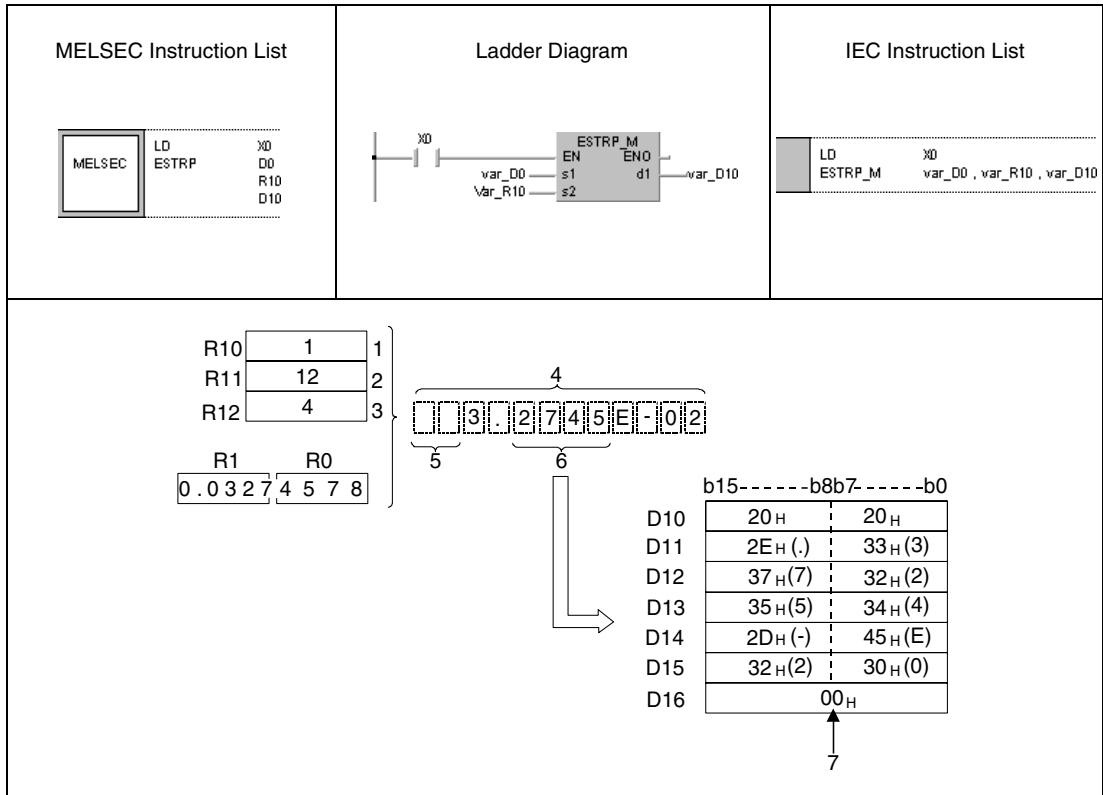
3 Number of decimal places

4 Is stored automatically

**Program Example 2**

**ESTRP**

With leading edge from X0, the following program converts a floating point value (real number) specified by D0 and D1 into the format specified by R10 (var\_R10 Array [1]) through R12 (var\_R10 Array [3]) and stores the result in D10 through D16.



b15-----b8	20 H	20 H
b7-----b0	2E H (.)	33 H (3)
	37 H (7)	32 H (2)
	35 H (5)	34 H (4)
	2D H (-)	45 H (E)
	32 H (2)	30 H (0)
	00 H	

- <sup>1</sup> Data format (Exponential representation) (1)
- <sup>2</sup> Total number of all digits
- <sup>3</sup> Number of decimal places
- <sup>4</sup> Total number of all digits
- <sup>5</sup> Blanks
- <sup>6</sup> Number of decimal places in the decimal part
- <sup>7</sup> Is stored automatically

7.11.12 EVAL, EVALP

CPU

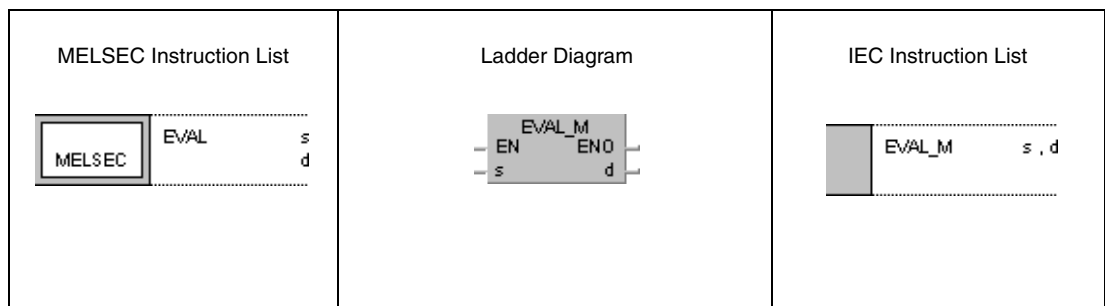
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

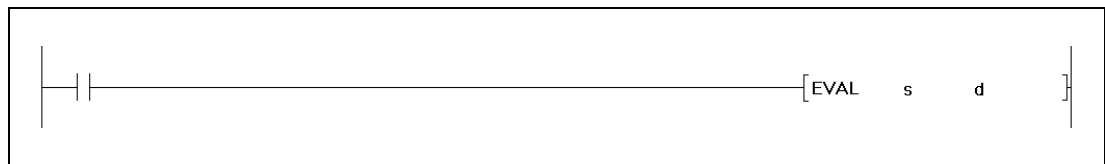
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		

GX IEC  
Developer



GX  
Developer



Variables

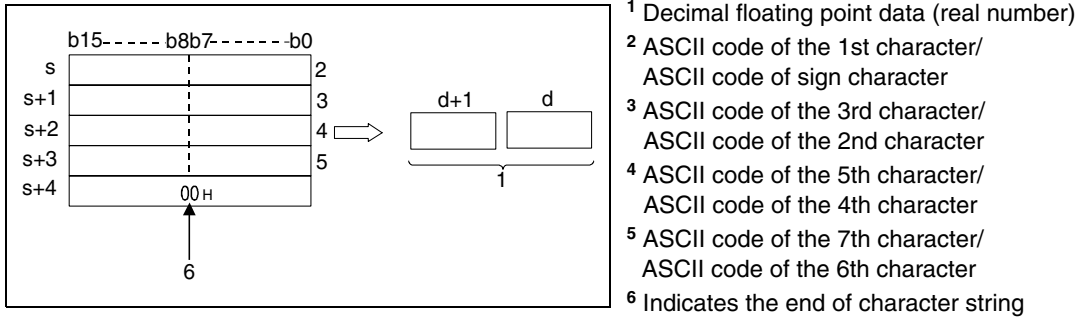
Set Data	Meaning	Data Type
s	Character string data to be converted into a floating point number (real number) or initial number of device storing such data.	Character string
d	First number of device storing the converted decimal floating point number (real number).	Real number

**Functions Conversion of character string data into decimal floating point data**

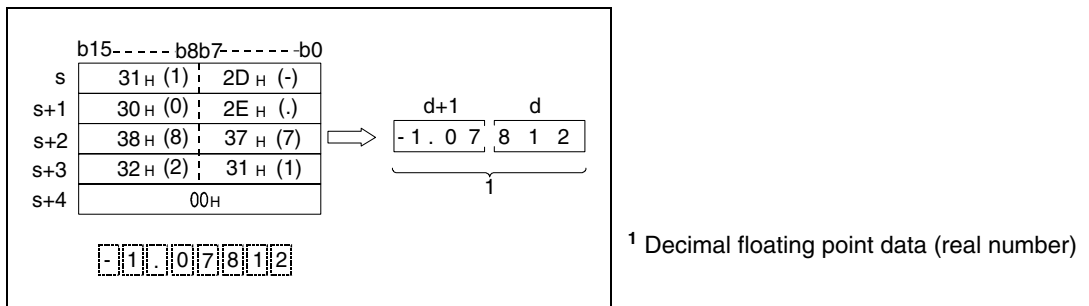
**EVAL Conversion of character strings**

The EVAL instruction converts the character string in s through s+4 into a decimal floating point number (real number). The result is stored in d.

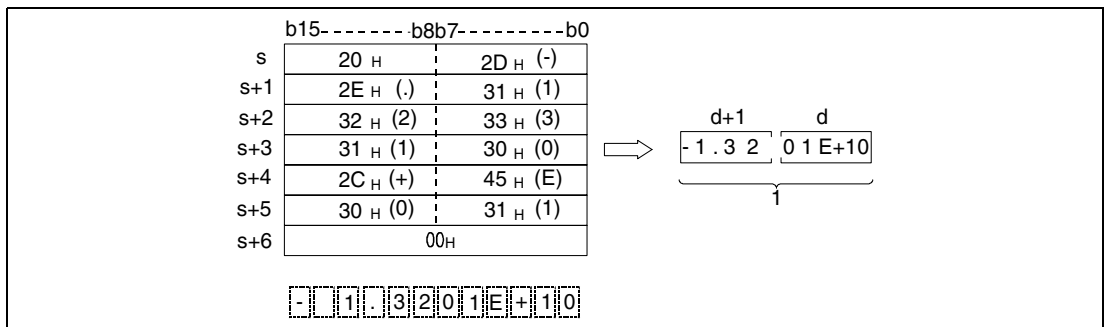
The character string can be converted into decimal floating point format as well as into the exponential format.



**Decimal format**

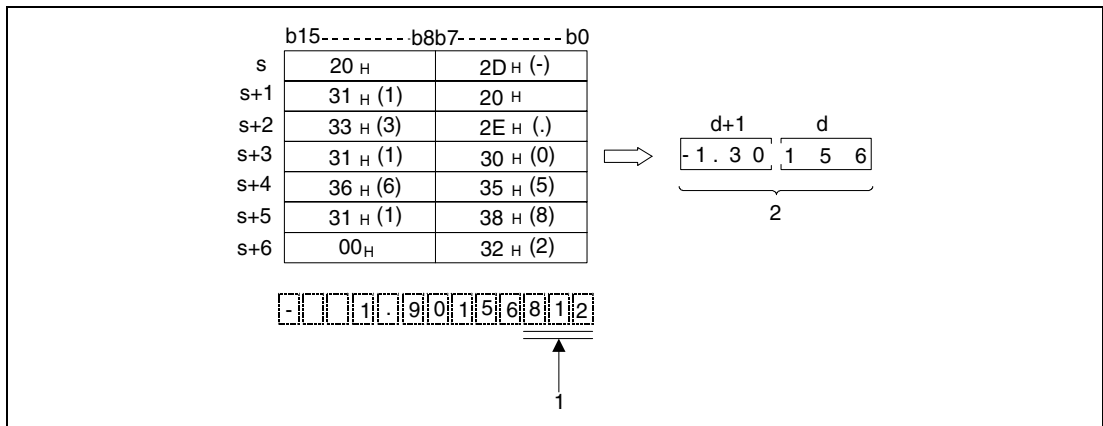


**Exponential format**



In the example below, six digits (without sign, decimal point, and exponent digits of the result) of the character string from s onwards are converted into a decimal floating point number. The digits from the 7th digit on are cut off from the result.

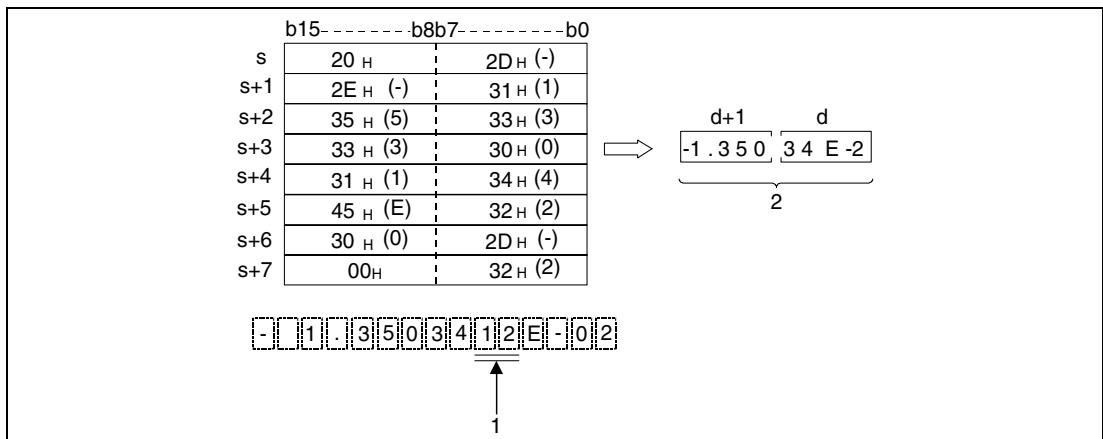
Decimal format



<sup>1</sup> These digits are omitted

<sup>2</sup> Decimal floating point data (real number)

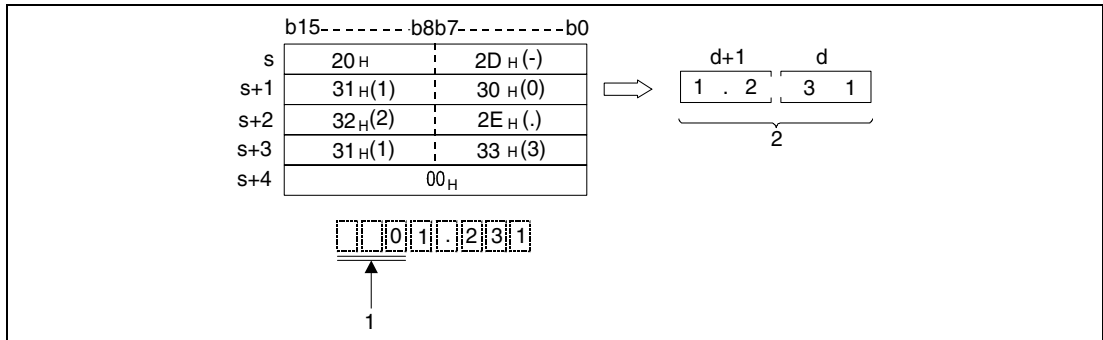
Exponential format



<sup>1</sup> These digits are omitted

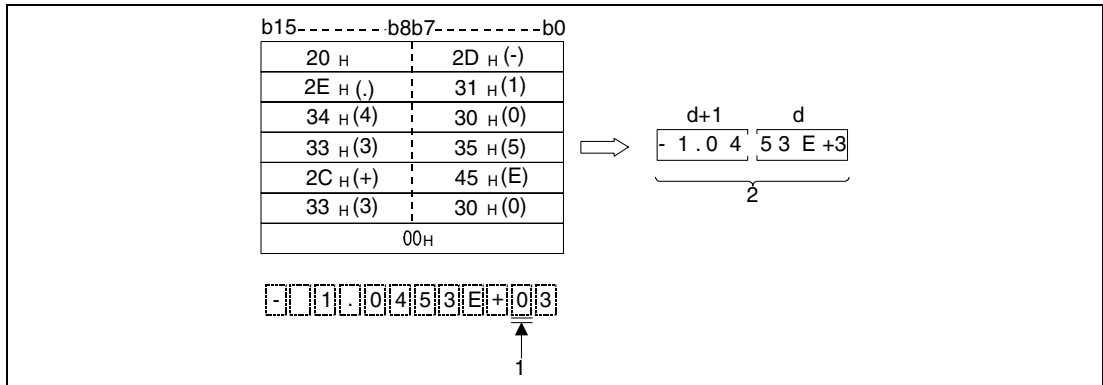
<sup>2</sup> Decimal floating point data (real number)

Leading blanks (ASCII code "20H") or zeroes (ASCII code "30H") in the character string from s onwards are ignored by the conversion, except for the initial zero (e.g. 0.123).



- <sup>1</sup> These characters are ignored by the conversion
- <sup>2</sup> Decimal floating point data (real number)

If the ASCII code "30H" (zero) is placed between the character "E" and the character string for the exponential format, this character is ignored by the conversion.



- <sup>1</sup> These characters are ignored by the conversion
- <sup>2</sup> Decimal floating point data (real number)

A character string to be converted may contain a maximum of 24 characters.

**Operation Errors**

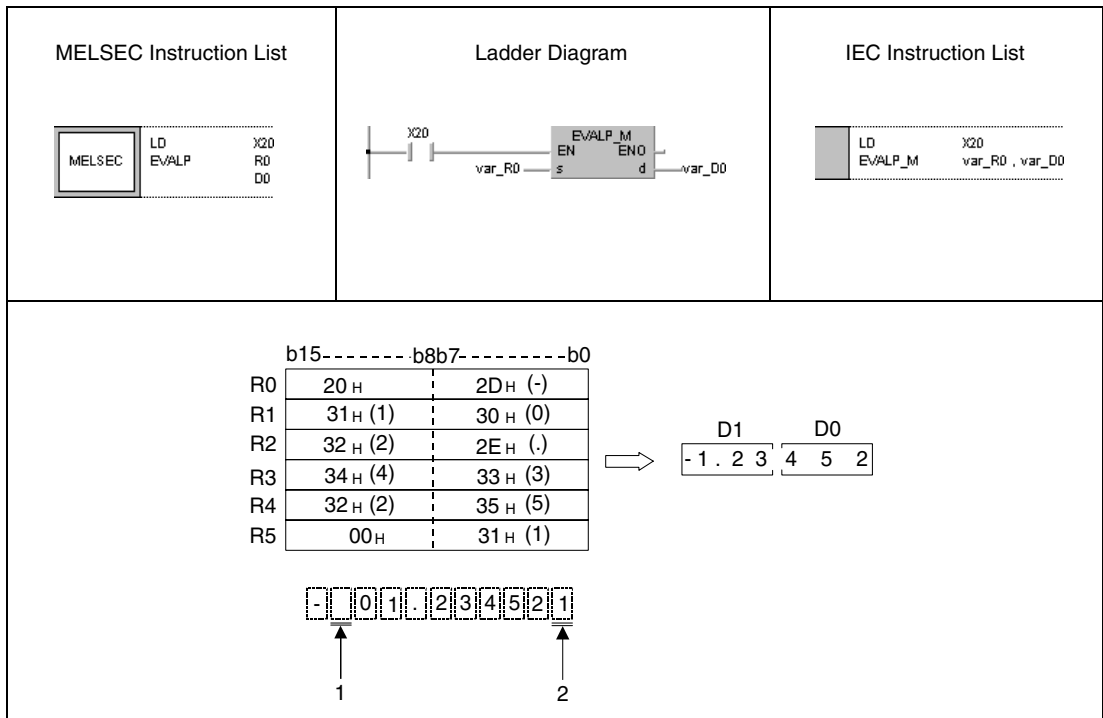
In the following cases an operation error occurs and the error flag is set:

- The character string does not begin with the character "20H" (blank) or "2DH" (minus) (error code 4100).
- The digits prior to the decimal point or the decimal places contain characters exceeding the range of values from "30H" (0) to "39H" (9) (error code 4100).
- The character "2EH" is used more than once within the character string (error code 4100).
- The exponent part contains characters different from "45H (E), 2CH (+)" or "45H (E), 2DH (-)". More than one exponent is used (error code 4100).
- The value is 0 or exceeds the relevant range of values from  $1.0 \times 2^{-127}$  to  $1.0 \times 2^{129}$  (error code 4100).
- The end of string indicator "00H" exceeds the relevant storage device range (error code 4100).
- The number of characters in the string is 0 or greater than 24.



**Program Example 1** EVALP

With leading edge from X20, the following program converts the character string specified in R0 through R5 into a decimal floating point number (real number) and stores the result in D0 and D1.

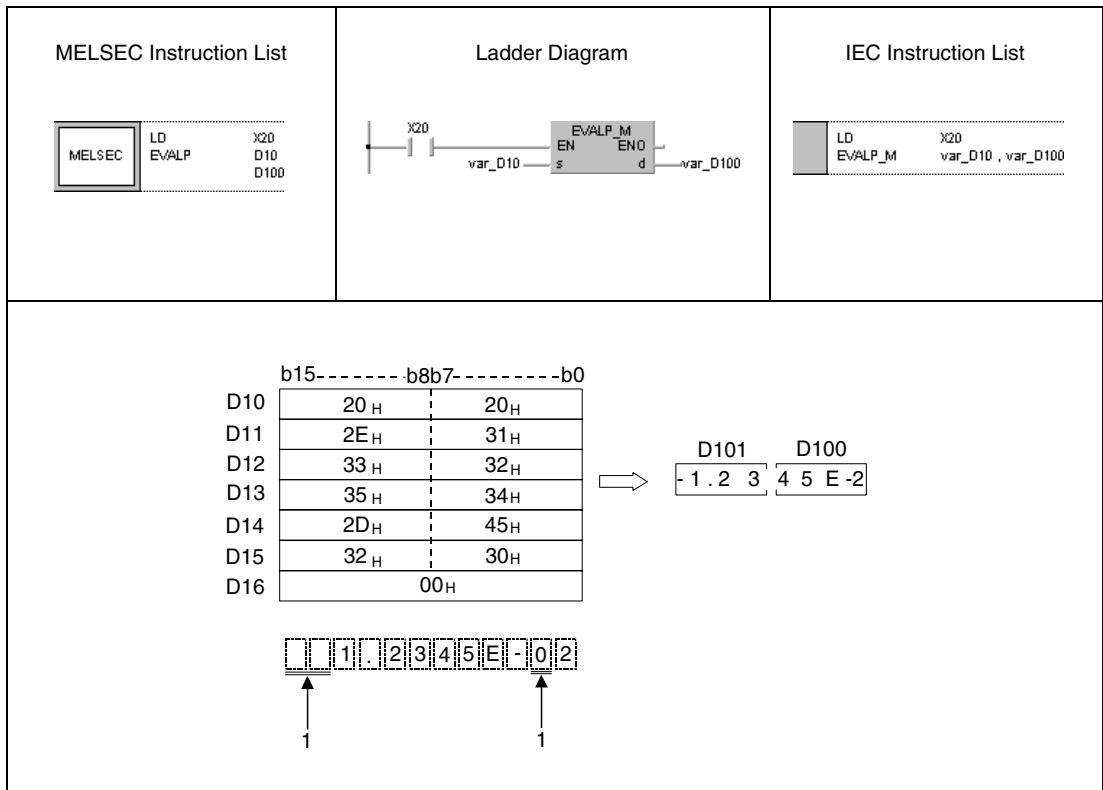


<sup>1</sup> This digit is not processed

<sup>2</sup> This number is cut off

**Program Example 2** EVALP

With leading edge from X20, the following program converts the character string specified in D10 through D16 into a floating point number (real number) and stores the result in D100 and D101.



<sup>1</sup> These digits are not processed

**NOTE**

*These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.13 ASC, ASCP (Q series and System Q)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

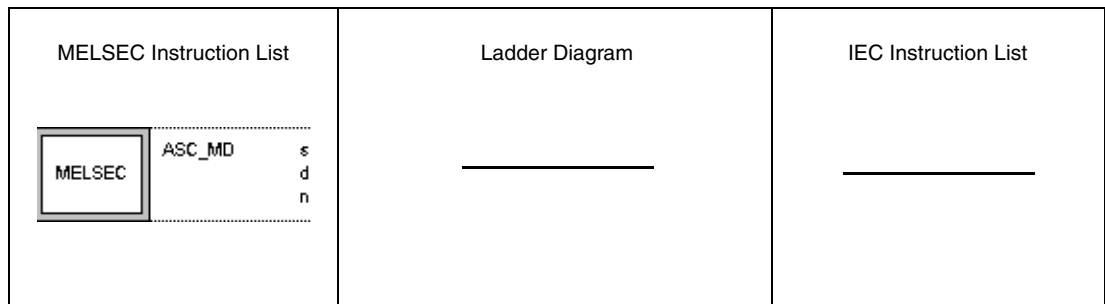
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices  
MELSEC Q

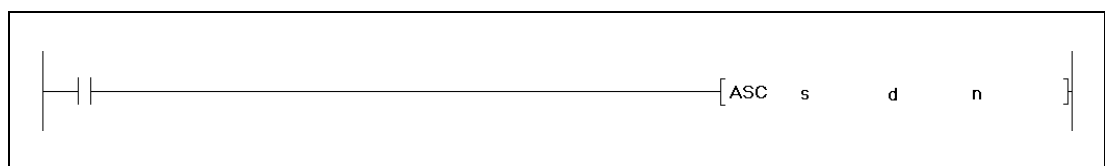
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC  
Developer



The ASC and the ASCP instructions dont work with the IEC editors. The only way to program these instructions is by using the MELSEC instruction list.  
Remedy: Move the hexadecimal ASCII format direct into the target registers.

GX  
Developer



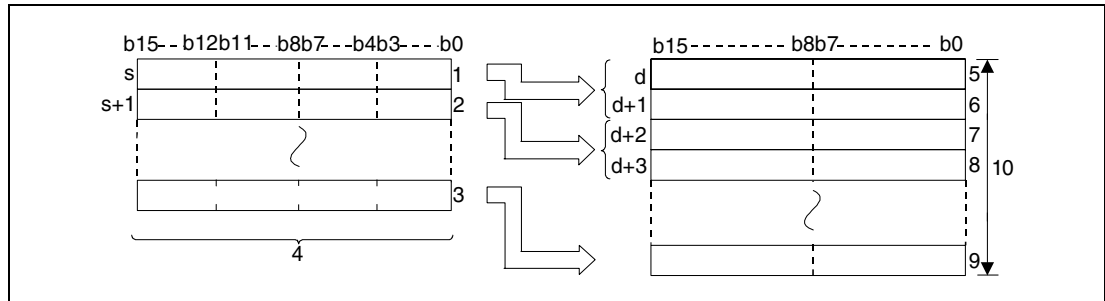
Variables

Set Data	Meaning	Data Type
s	First number of device storing the character string to be converted into the binary format.	Character string
d	First number of device storing converted binary data.	
n	Number of characters to be converted.	

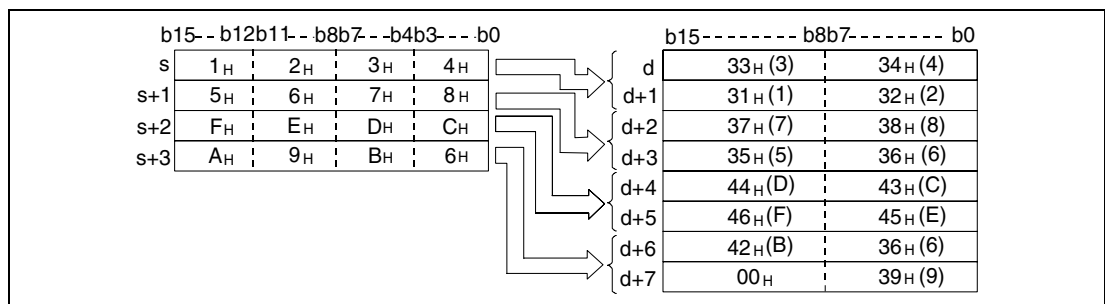
**Functions Conversion of BIN 16-bit data into ASCII code**

**ASC/ASCP Conversion instruction**

The ASCII instruction converts the 16-bit binary data stored from s onwards into the hexadecimal ASCII format and stores the result considering the number of characters specified by n from d onwards.

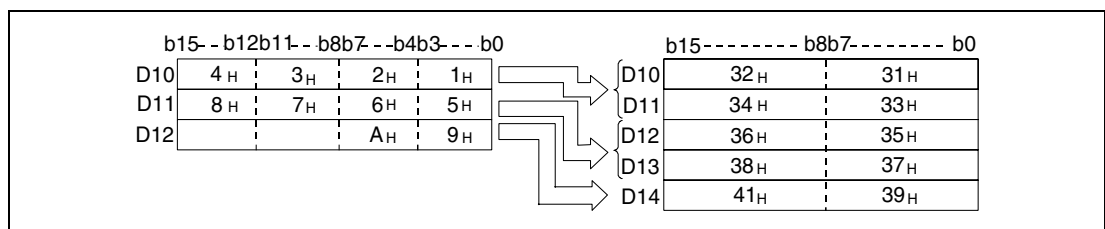


- 1 First digit/ second digit/ third digit/ fourth digit
- 2 First digit/ second digit/ third digit/ fourth digit
- 3 First digit/ second digit/ third digit/ fourth digit
- 4 Binary data
- 5 ASCII code of the 1st digit/ ASCII code of the 2nd digit
- 6 ASCII code of the 3rd digit/ ASCII code of the 4th digit
- 7 ASCII code of the 5th digit/ ASCII code of the 6th digit
- 8 ASCII code of the 7th digit/ ASCII code of the 8th digit
- 9 ASCII code of the 9th digit/ ASCII code of the 10th digit
- 10 Number of digits specified in n

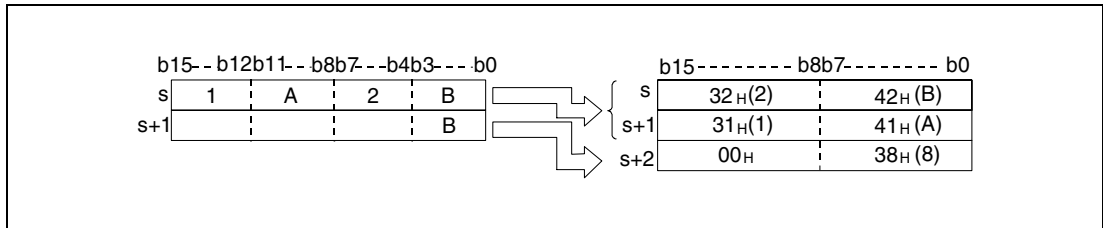


The number of characters specified in n determines the ranges of values of the devices specified from s and d onwards. The devices specified from s onwards contain the binary data to be converted. The converted character string is stored in the devices specified from d onwards.

The program is even processed accurately and without an error message, if the storage area of the binary data to be converted overlaps with that of the converted ASCII data.



If n specifies an odd number of characters, the ASCII character "00H" is placed automatically into the upper 8 bits of the highest address of the area, storing the character string.



If the number of characters specified by n is zero, the program will not be executed.

**Operation Errors**

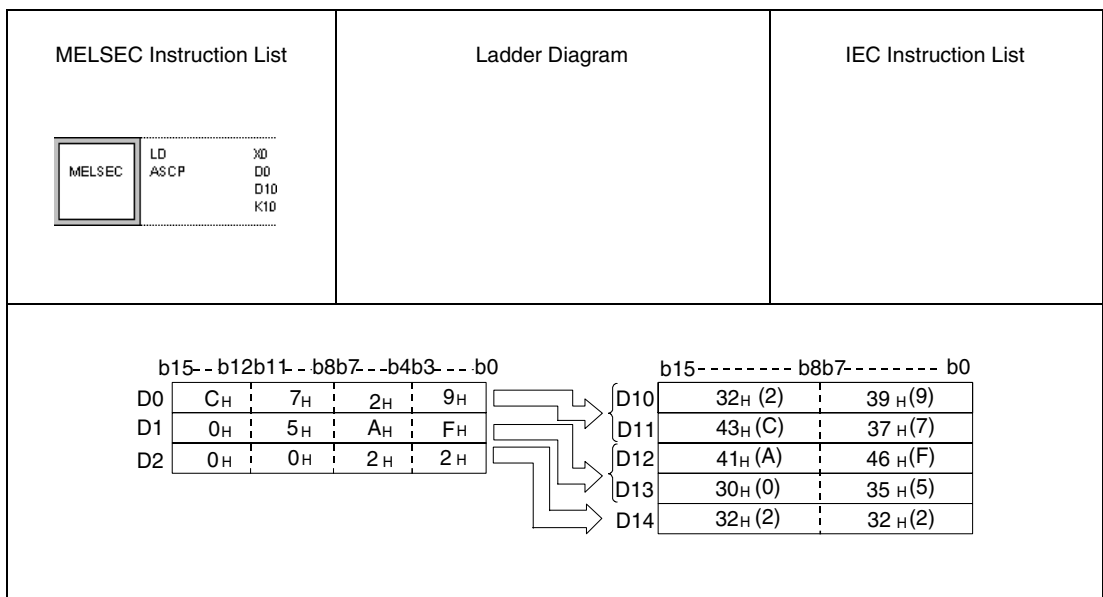
In the following cases an operation error occurs and the error flag is set:

- The number of characters specified by n and therefore the required number of registers from s onwards exceeds the relevant storage device range (error code 4101).
- The number of characters specified by n and therefore the required number of registers from d onwards exceeds the relevant storage device range (error code 4101).

**Program Example**

ASCP

With leading edge from X0, the following program reads in the binary data stored in D0 as hexadecimal values and converts it into a character string. The result is stored in D10 through D14.



7.11.14 ASC (A series)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

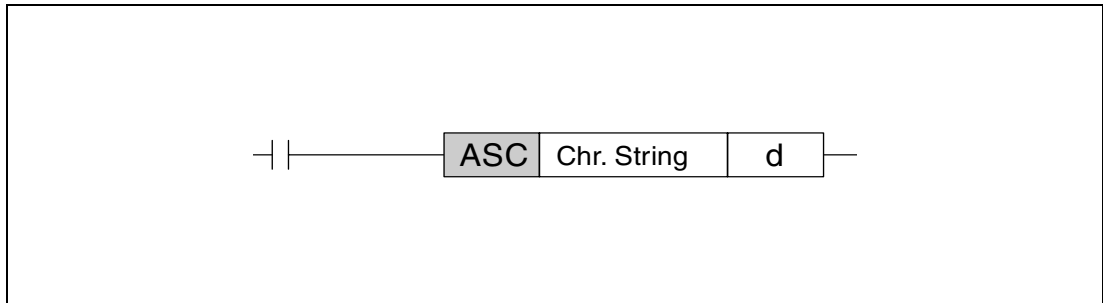
Devices  
MELSEC A

Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag						
Bit Devices						Word Devices (16-bit)						Constant		Pointer					Level		M9012	M9010 M9011				
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I	N			
d							●	●	●	●	●												13	●		●

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

GX IEC  
Developer

GX  
Developer



Variables

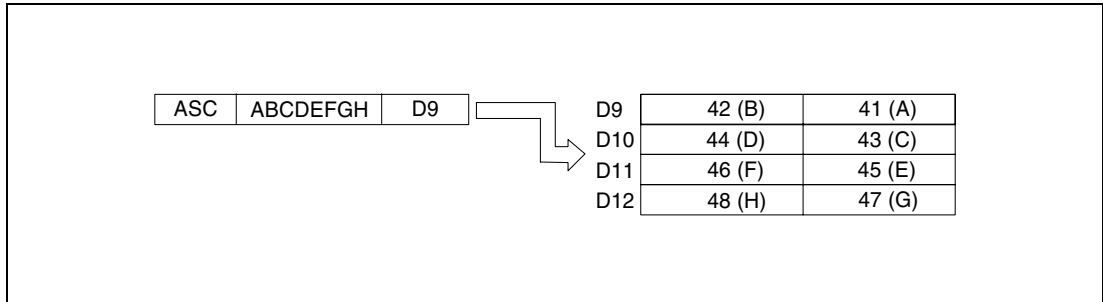
Device	Meaning	Data type
d	Device storing the converted characters.	BIN 16-bit

**Functions Conversion of character string data into ASCII code**

**ASC Conversion of alphanumerical character strings**

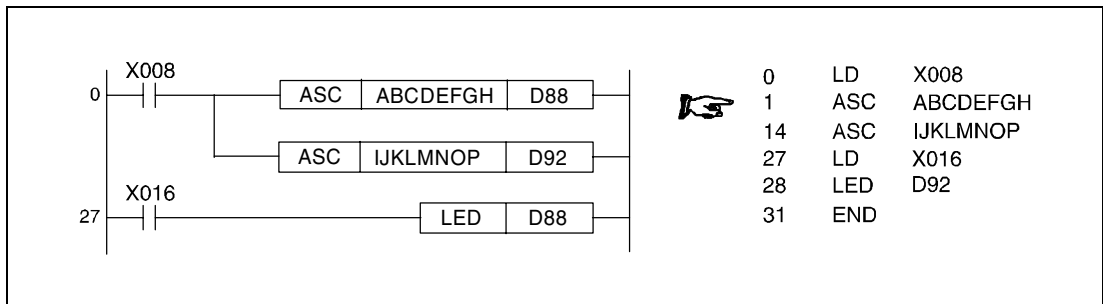
The ASC instruction converts alphanumerical character strings with up to 8 characters into the ASCII code. The result is stored from d onwards.

The stored ASCII code can be printed out via the PR/ PRC instruction and displayed on the LED display of a suitable CPU via the LED instruction.



**Program Example ASCP**

After X8 is set, the following program converts the character string "ABCDEFGHIJKLMNOP" into ASCII code and stores the result in D88 through D91 and D92 through D95. After X16 is set, the ASCII data in D88 through D95 is displayed on the LEDs on the front panel of the CPU.



7.11.15 HEX, HEXP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

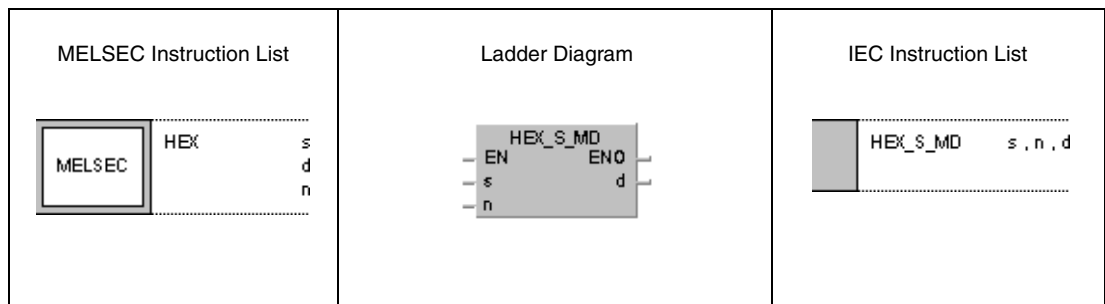
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

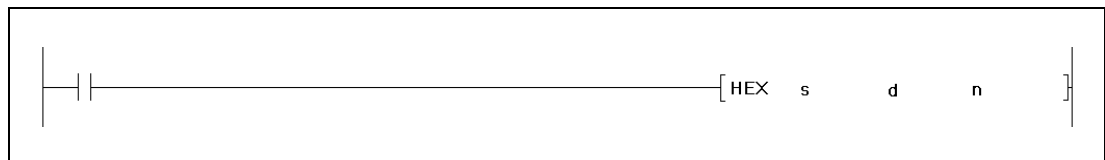
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		

GX IEC  
Developer



GX  
Developer



Variables

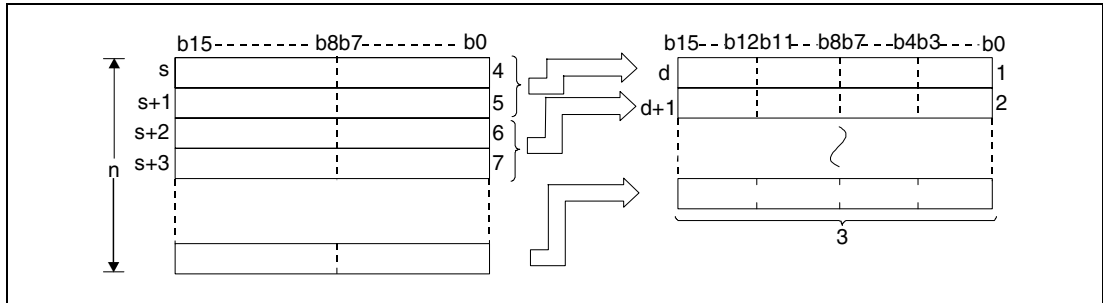
Set Data	Meaning	Data Type
s	First number of device storing binary data to be converted.	Character string
d	First address of area storing the converted binary data.	BIN 16-bit
n	Number of characters to be converted.	



**Functions Conversion of hexadecimal ASCII values into binary values**

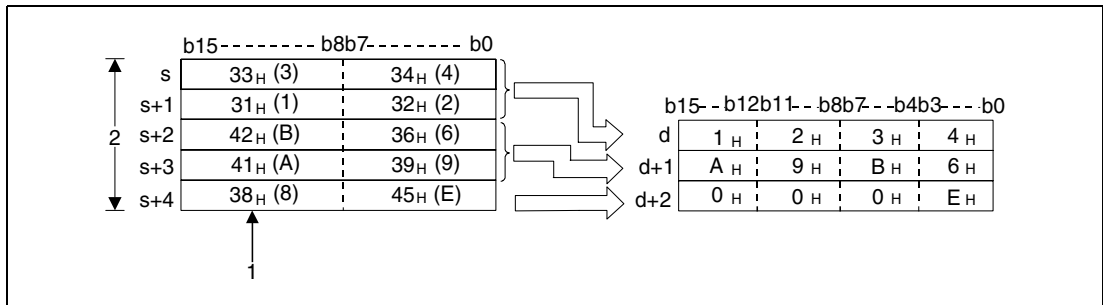
**HEX Conversion of hexadecimal ASCII values**

The HEX instruction converts the hexadecimal ASCII characters from s onwards into binary values. The result is stored from d onwards.



- <sup>1</sup> 4th digit, 3rd digit, 2nd digit, 1st digit
- <sup>2</sup> Binary data
- <sup>3</sup> ASCII code of the 2nd digit/ ASCII code of the 1st digit
- <sup>4</sup> ASCII code of the 4th digit/ ASCII code of the 3rd digit
- <sup>5</sup> ASCII code of the 2nd digit/ ASCII code of the 1st digit
- <sup>6</sup> ASCII code of the 4th digit/ ASCII code of the 3rd digit

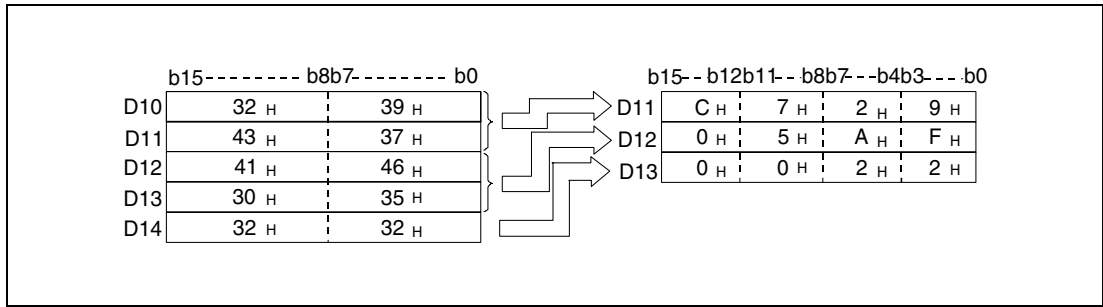
The number of characters in n is 9.



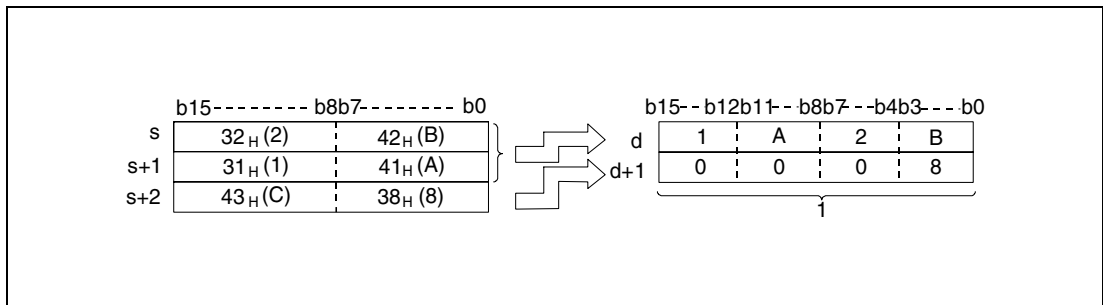
- <sup>1</sup> Since the character string contains 9 characters, the "38H" is not changed or moved.
- <sup>2</sup> n = 9

The number of characters specified in n determines the range of values of the character string from s and of the binary data from d onwards automatically.

Although the range of values of the ASCII code to be converted and that of the converted binary values overlap, this instruction processes the data accurately.



If the number of characters in n is not divisible by 4, a zero is written after the specified number of characters automatically to the highest registers storing the converted binary values.



<sup>1</sup> The value zero is stored automatically

If the number of characters in n is zero, the conversion will not be executed.

The ASCII code from s onwards may range from "30H" through "39H" and "41H" through "46H".

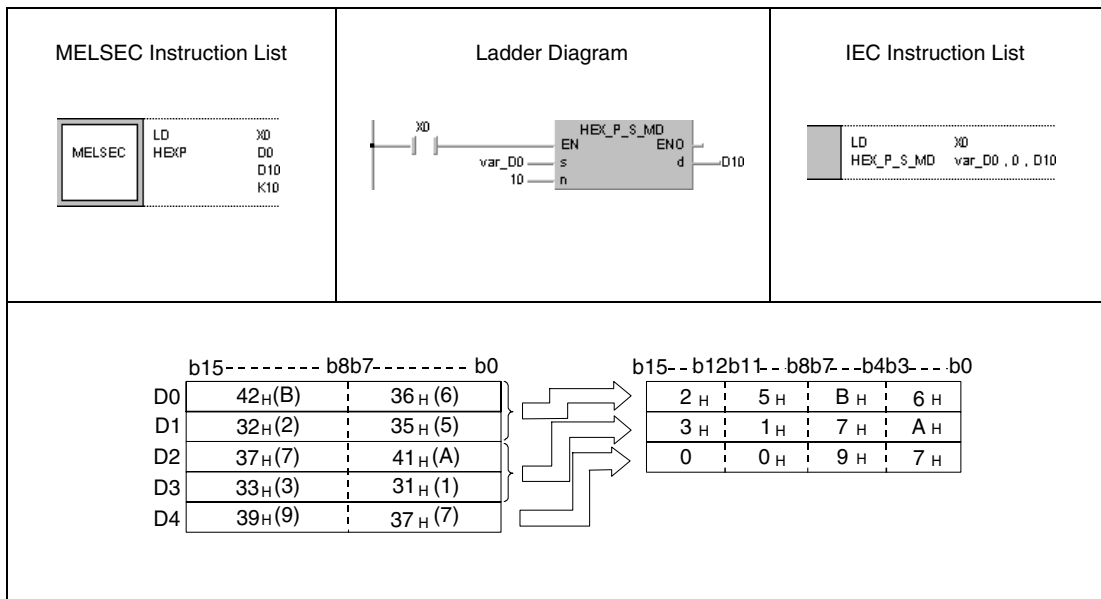
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The devices specified from s onwards contain characters exceeding the ranges "30H" through "39H", "41H", and "46H" (error code 4100).
- The number of characters specified by n and therefore the required number of registers from s onwards exceeds the relevant storage device range (error code 4101).
- The number of characters specified by n and therefore the required number of registers from d onwards exceeds the relevant storage device range (error code 4101).
- The value n is negative.

**Program Example**      HEXP

With leading edge from X0, the following program converts the character string "6B52A71379" stored in D0 through D4 into binary data. The result is stored in D10 through D14.



**NOTE**      *This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.16 RIGHT, RIGHTP, LEFT, LEFTP

CPU

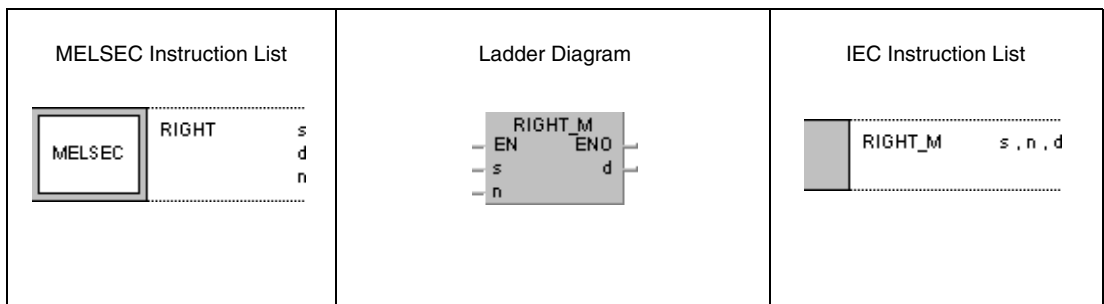
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

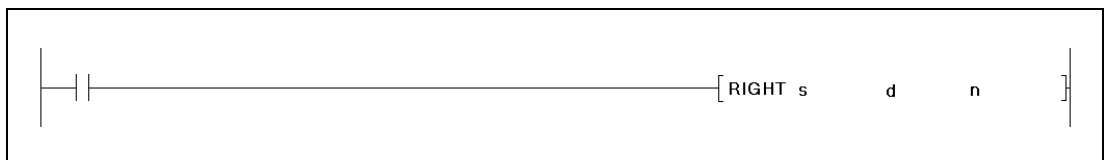
Devices  
MELSEC Q

	Usable Devices										Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant		Other		
	Bit	Word		Bit	Word			K, H (16#)	\$			
s	—	●	●	—	—	—	—	—	●	—	SMO	4
d	—	●	●	—	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



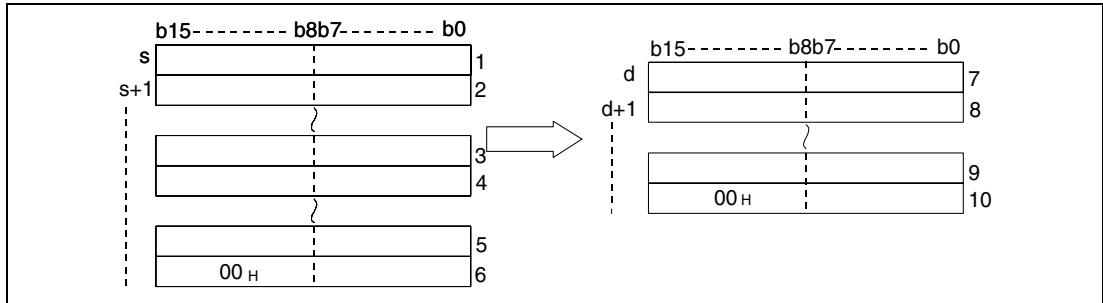
Variables

Device	Meaning	Data Type
s	First number of device storing the character string.	Character string
d	First number of device area storing the determined characters of the character string.	
n	Number of characters stored on the left or on the right.	BIN 16-bit

**Functions Extraction of character string data from the right or from the left**

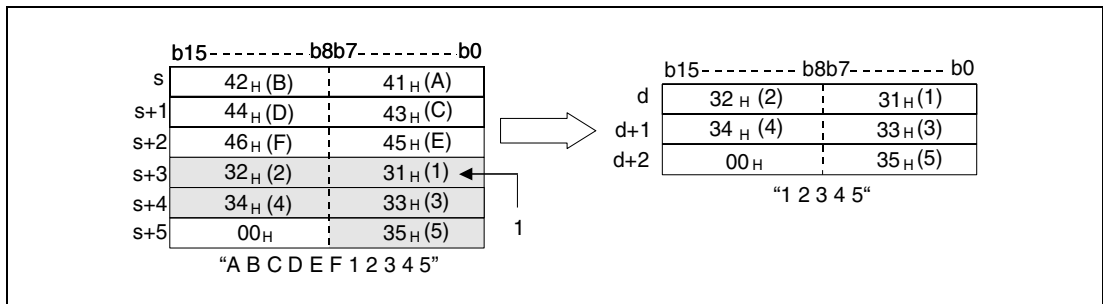
**RIGHT Extract character string data from the right**

The RIGHT instruction stores n characters from the right side of the character string (end of character string) from s onwards. The characters are stored from d onwards.



- <sup>1</sup> ASCII code of the 2nd characters/ ASCII code of the 1st character
- <sup>2</sup> ASCII code of the 4th character/ ASCII code of the 3rd character
- <sup>3</sup> ASCII code of the last character minus n+2/ ASCII code of the last character minus n+1
- <sup>4</sup> ASCII code of the last character minus n+4/ ASCII code of the last character minus n+3
- <sup>5</sup> ASCII code of the last character minus 1/ ASCII code of the last character minus 2
- <sup>6</sup> "00H"/ ASCII code of the last character
- <sup>7</sup> ASCII code of the last character minus n+2/ ASCII code of the last character minus n+1
- <sup>8</sup> ASCII code of the last character minus n+4/ ASCII code of the last character minus n+3
- <sup>9</sup> ASCII code of the last character minus 1/ ASCII code of the last character minus 2
- <sup>10</sup> "00H"/ ASCII code of the last character

With n = 5

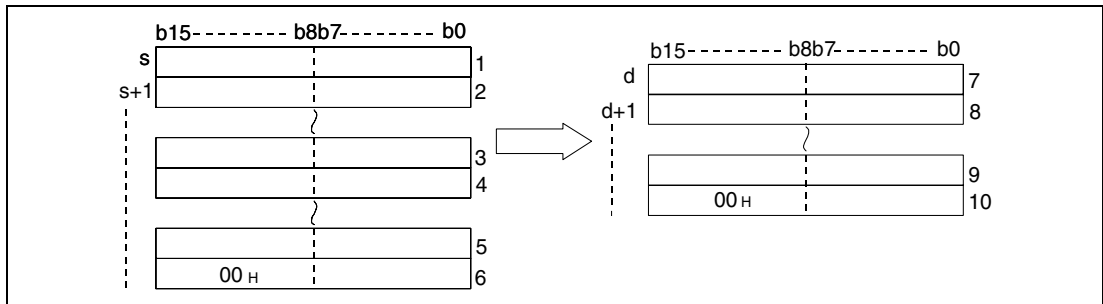


- <sup>1</sup> ASCII code for the 5th character

If the number of characters in n is zero, the character code "00H" is stored from d onward.

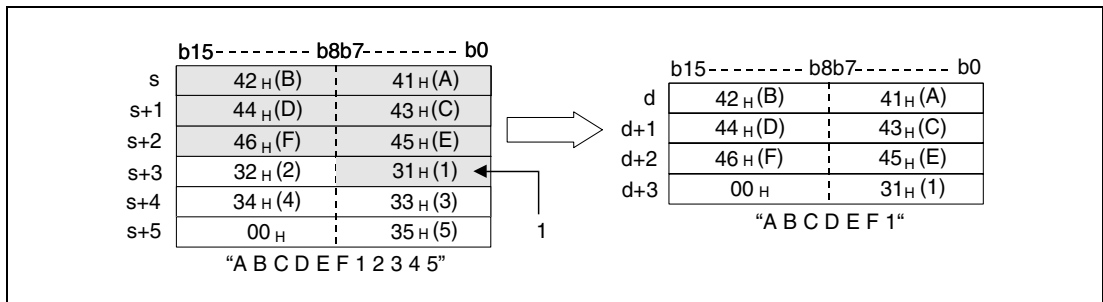
**LEFT Extract character string data from the left**

The LEFT instruction stores n characters from the left side of the character string (beginning of character string) from s onwards. The characters are stored from d onwards.



- <sup>1</sup> ASCII code of the 2nd character/ ASCII code of the 1st character
- <sup>2</sup> ASCII code of the 4th character/ ASCII code of the 3rd character
- <sup>3</sup> ASCII code of the character n-1/ ASCII code of the character n-2
- <sup>4</sup> ASCII code of the character n+1/ ASCII code of the nth character
- <sup>5</sup> "00H"/ ASCII code of the last character
- <sup>6</sup> ASCII code of the 2nd character/ ASCII code of the 1st character
- <sup>7</sup> ASCII code of the 4th character/ ASCII code of the 3rd character
- <sup>8</sup> ASCII code of the character n-1/ ASCII code of the character n- 2
- <sup>9</sup> "00H"/ ASCII code of the nth character

With n=7



- <sup>1</sup> ASCII code of the 7th character

If the number of characters in n is zero, the character code "00H" is stored from d onwards.

**Operation Errors**

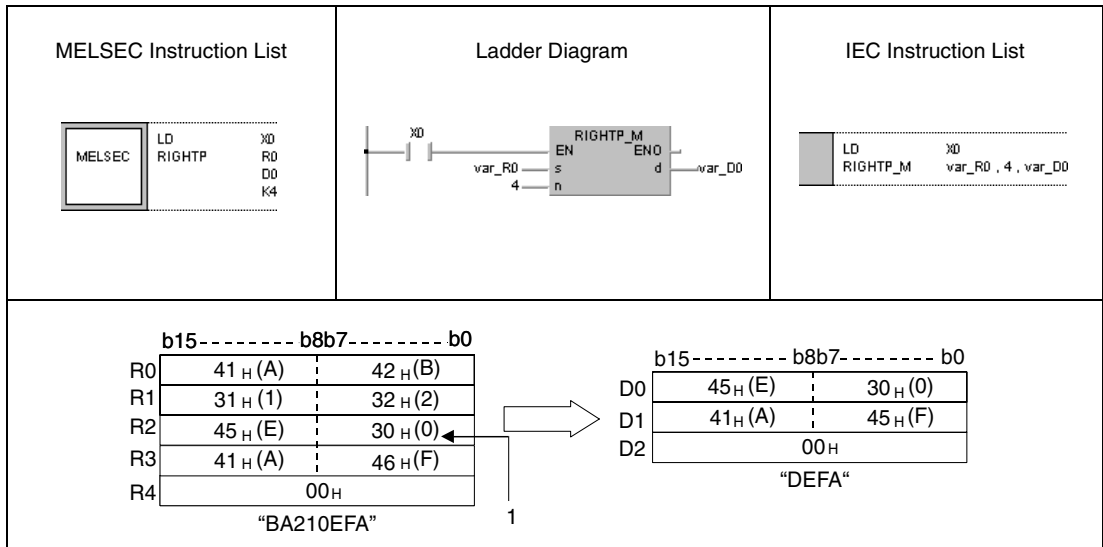
In the following cases an operation error occurs and the error flag is set:

- The value in n exceeds the number of existing characters stored from s onwards (error code 4101).
- The area specified by n exceeds the relevant device range of the device specified by d (error code 4101).

**Program Example 1**

**RIGHTP**

With leading edge from X0, the following program extracts 4 characters of the data from the right side of the character string stored in R0 through R4 and stores it in D0 through D2.

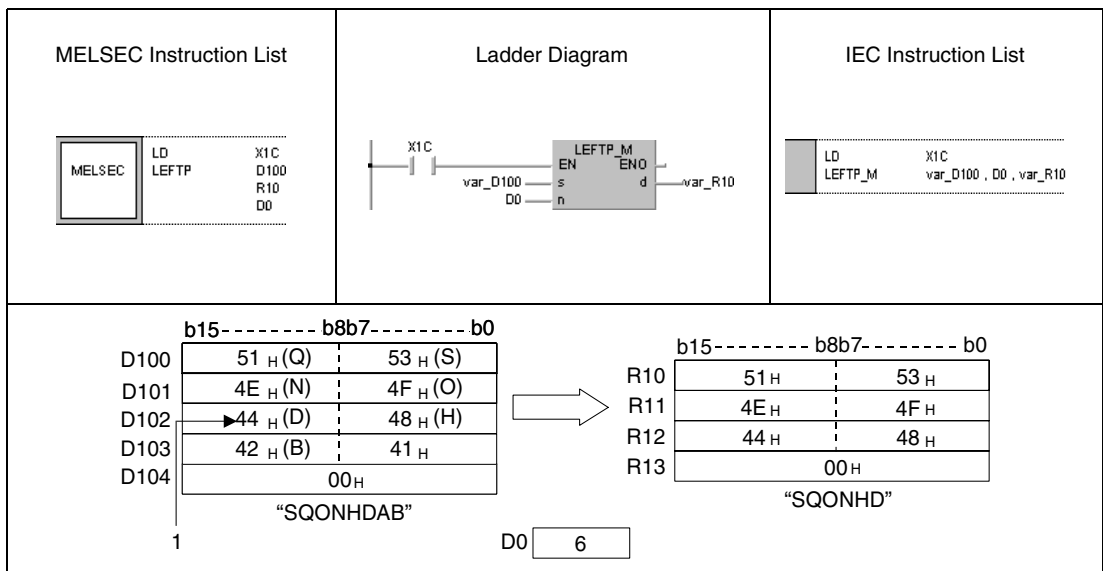


<sup>1</sup> ASCII code of the 4th character

**Program Example 2**

**LEFTP**

With leading edge from X1C, the following program extracts the number of characters specified in D0 from the left side of the character string specified in D100 through D104. The result is stored in R10 through R13.



<sup>1</sup> ASCII code of the 6th character

**NOTE**

*These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.17 MIDR, MIDRP, MIDW, MIDWP

CPU


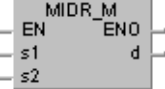
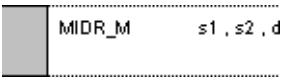
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

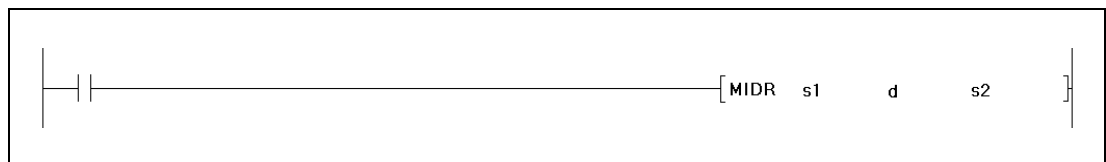
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
s2	●	●	●	●	●	●	●	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

GX Developer



Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s1	First number of device storing character string data.	Character string	Character string
d	First number of device storing the operation result.		
s2	First number of device storing the 1st character and the number of characters. (s2)+0: Register of the 1st character (s2)+1: Number of characters	BIN 16-bit	Array [1..2] of ANY16



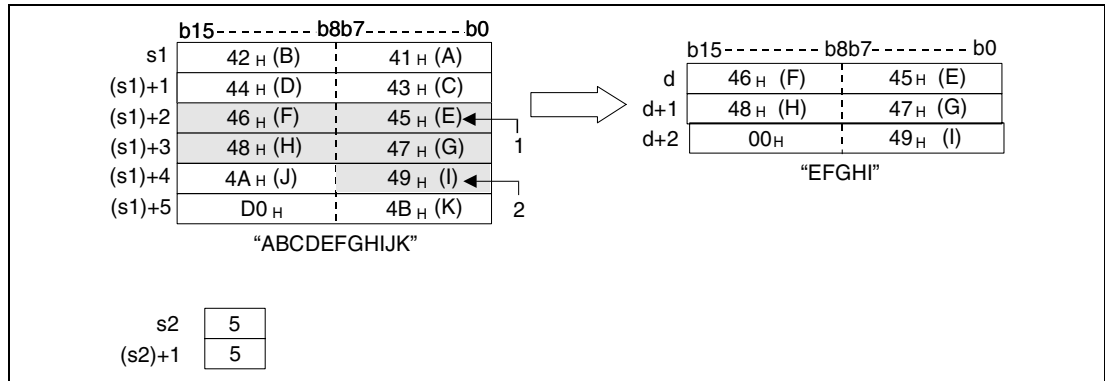
**Functions Storing and moving parts of character strings**

**MIDR Storing specified parts of character strings**

The MIDR instruction stores a part specified from s onwards of the character string stored from d onwards.

The first character of part to be stored is specified in s2 (Array\_s[1]) and is counted beginning from the left part of the character string (lower byte of s1).

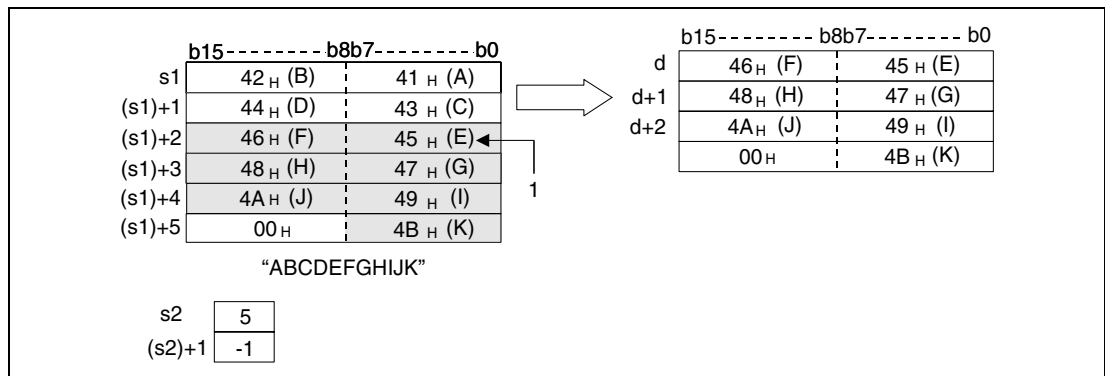
The length of the part to be stored is specified in s2+1 (Array\_s[2]).



<sup>1</sup> Position of the 5th character (s2)

<sup>2</sup> Position of the last character to be stored

No operation is processed, if the number of characters in (s2)+1 (Array\_s[2]) is zero.



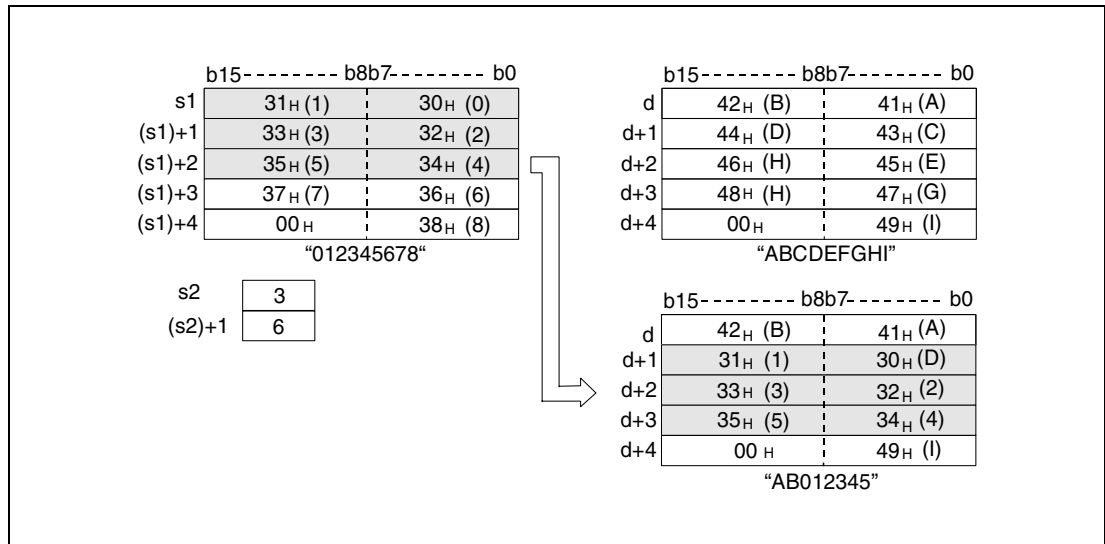
<sup>1</sup> Position of the 5th character (s2)

**MIDW Moving parts of character string to a defined area**

The MIDW instruction stores a part of specified length of the character string stored from s1 onwards in the area specified in d and d+1.

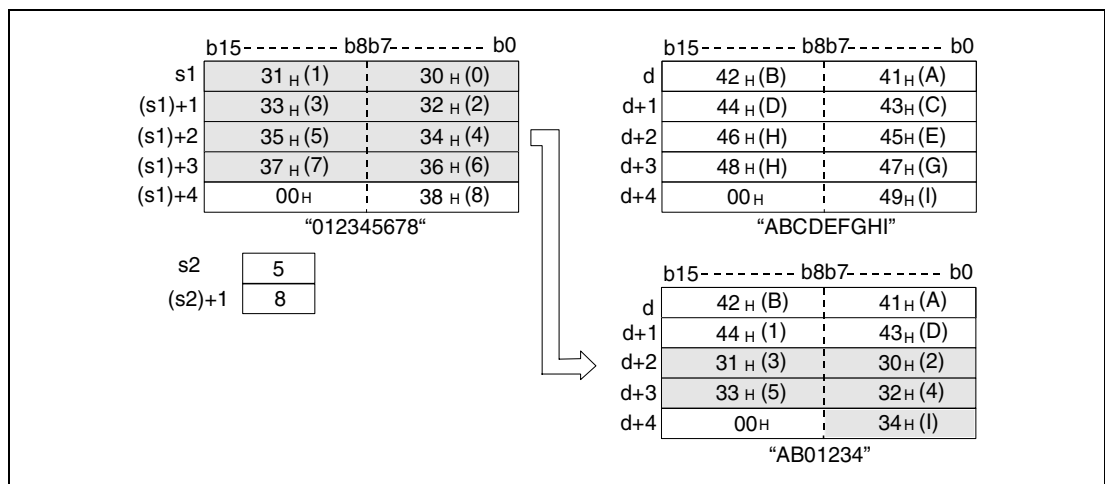
The first address of the storage area in d through d+n is specified in s2 (Array\_s2[1]) and is counted beginning from the left part of the character string (lower byte of d).

The length of the part of string to be stored is specified in s2+1 (Array\_s2[2]).

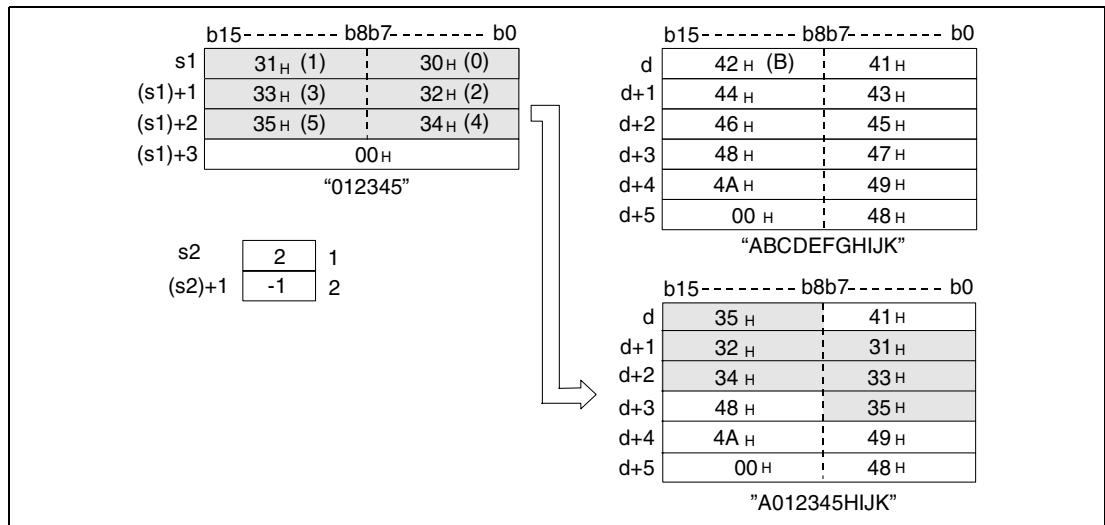


No operation is processed, if the number of characters in (s2)+1 (Array\_s2[2]) is zero.

If the number of characters specified in (s2)+1 (Array\_s2[2]) exceeds the storage area specified from d onwards, the remaining characters are cut off. In the following diagram the characters "35H" through "37H" are not stored.



If the value -1 is stored in (s2)+1 (Array\_s2[2]), the characters are stored from s1 onwards.



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

For the MIDR instruction

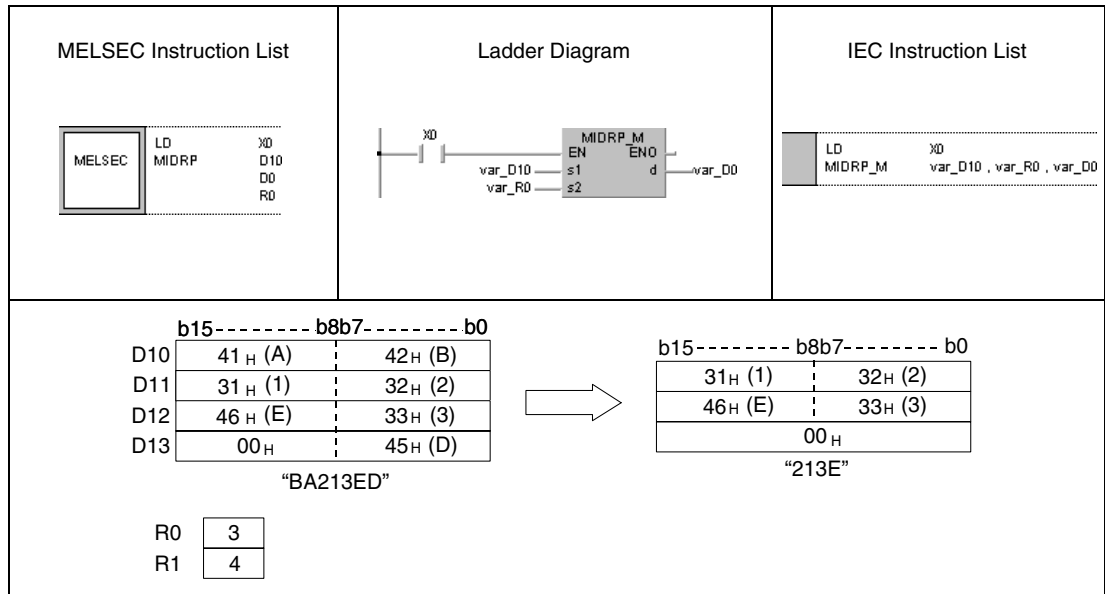
- The initial device number of the characters to be stored specified in s2 (Array\_s2[1]) exceeds the range from s1 to (s1)+n (error code 4101).
- The initial device number of the characters to be stored specified in (s2)+1 (Array\_s2[2]) exceeds the range from d to d+n (error code 4101).

For the MIDW instruction

- The initial device number of the characters to be stored specified in (s2) (Array\_s2[1]) exceeds the range from d to d+n (error code 4101).
- The initial device number of the characters to be stored specified in (s2)+1 (Array\_s2[2]) exceeds the storage range in s1 through (s1)+n (error code 4101).

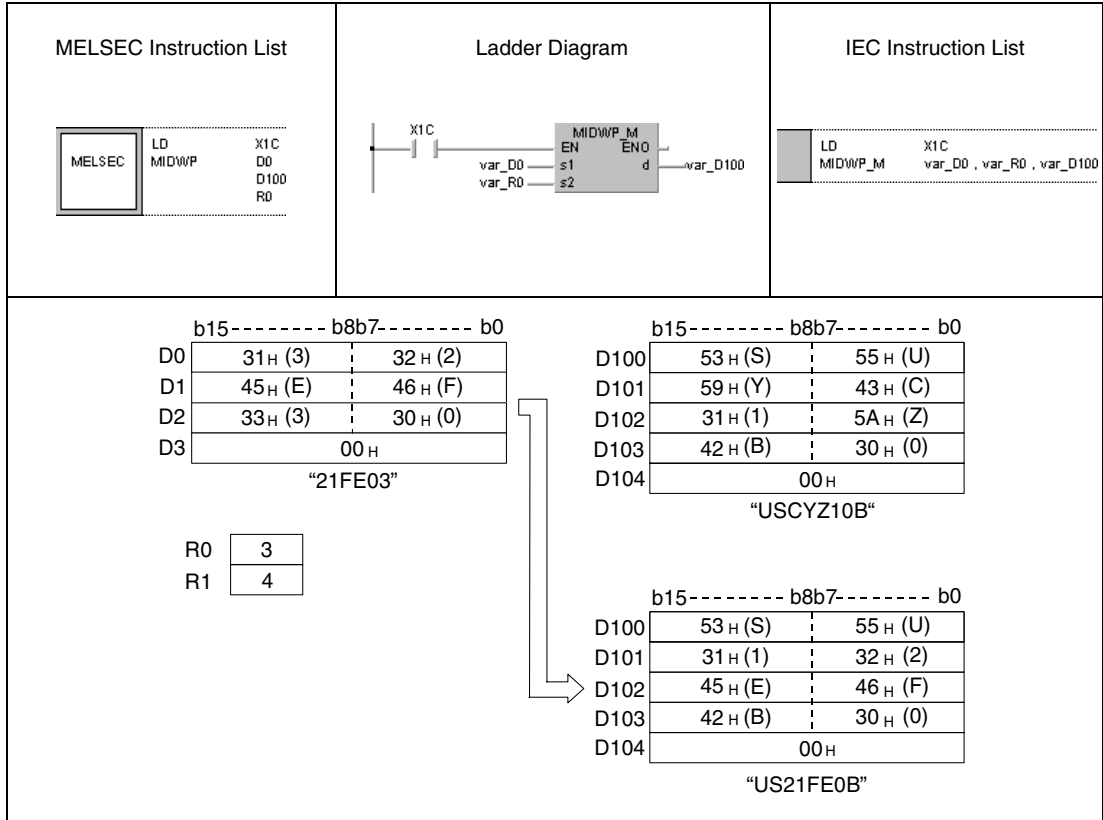
**Program Example 1** MIDRP

With leading edge from X0, the following program stores characters in D0 through D2 from a character string in D10 through D13. The number of characters to be stored is specified in R1 (var\_R0 Array [2]). The starting position within the source string is specified in R0 (var\_R0 Array [1]).



**Program Example 2** MIDWP

With leading edge from X1C, the following program stores characters in D100 through D104 from the beginning of a character string in D0 through D3. The number of characters to be stored is specified in R1 (var\_R0 Array [2]). The starting position where the characters are stored is specified by R0 (var\_R0 Array [1]).



**NOTE** *These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.18 INSTR, INSTRP

CPU

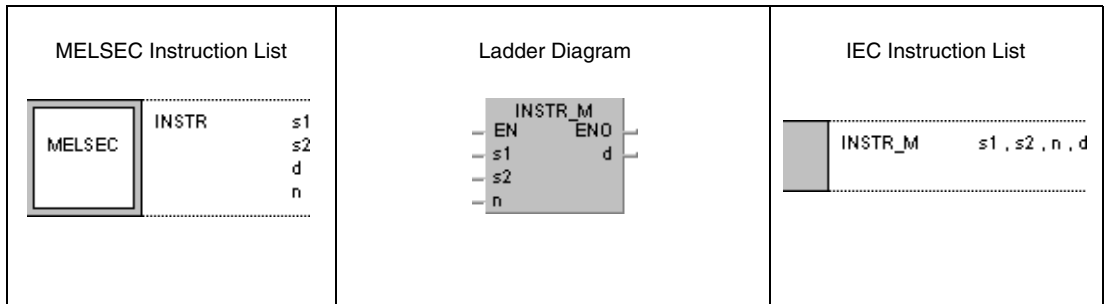
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

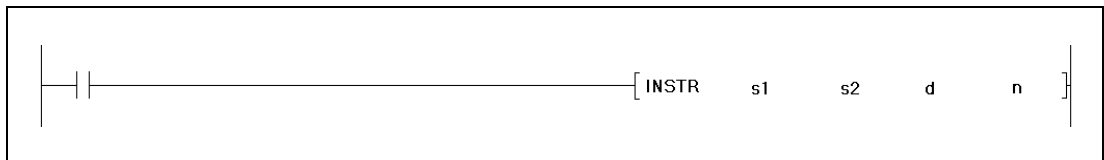
Devices  
MELSEC Q

	Usable Devices										Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant		Other		
	Bit	Word		Bit	Word			K, H (16#)	\$			
s1	—	●	●	—	—	—	—	—	●	—	SMO	5
s2	—	●	●	—	—	—	—	—	●	—		
d	●	●	●	●	●	●	●	—	—	—		
n	●	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s1	First number of device storing the character string to be searched for.	Character string
s2	First number of device storing the character string data to be searched through.	
d	Initial number of device storing the search result.	BIN 16-bit
n	Initial position where data is searched.	

**Functions Search for character strings**

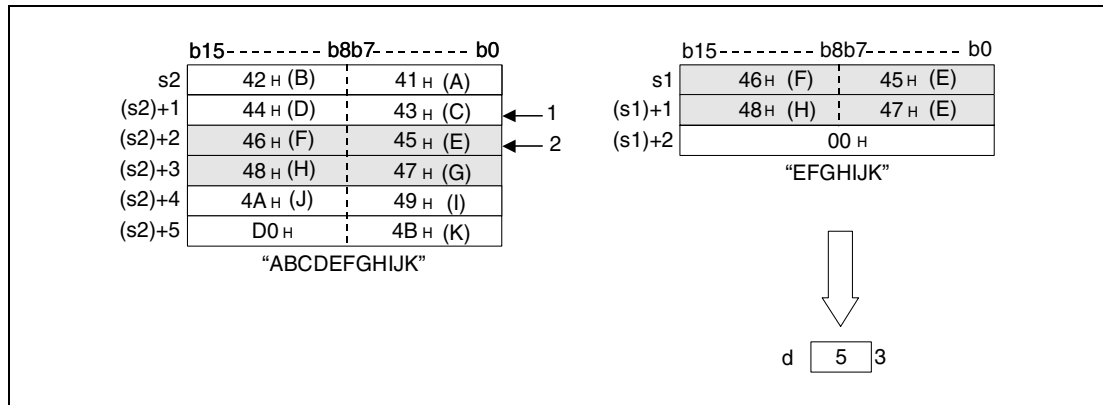
**INSTR Search for character strings**

The INSTR instruction searches the character string specified in s1 through (s1)+n within the character string data specified by s2 through (s2)+n.

The search begins with the character specified in n.

The first matching character is stored in d. The character is counted beginning from the left part of the character string (lower byte of s2).

For n=3



- <sup>1</sup> The search starts from the 3rd character
- <sup>2</sup> First character of the searched character string
- <sup>3</sup> Search result

If no matching character string is found, a zero is stored in d.

In case the value specified in n is negative or zero, no operation is processed.

**Operation Errors**

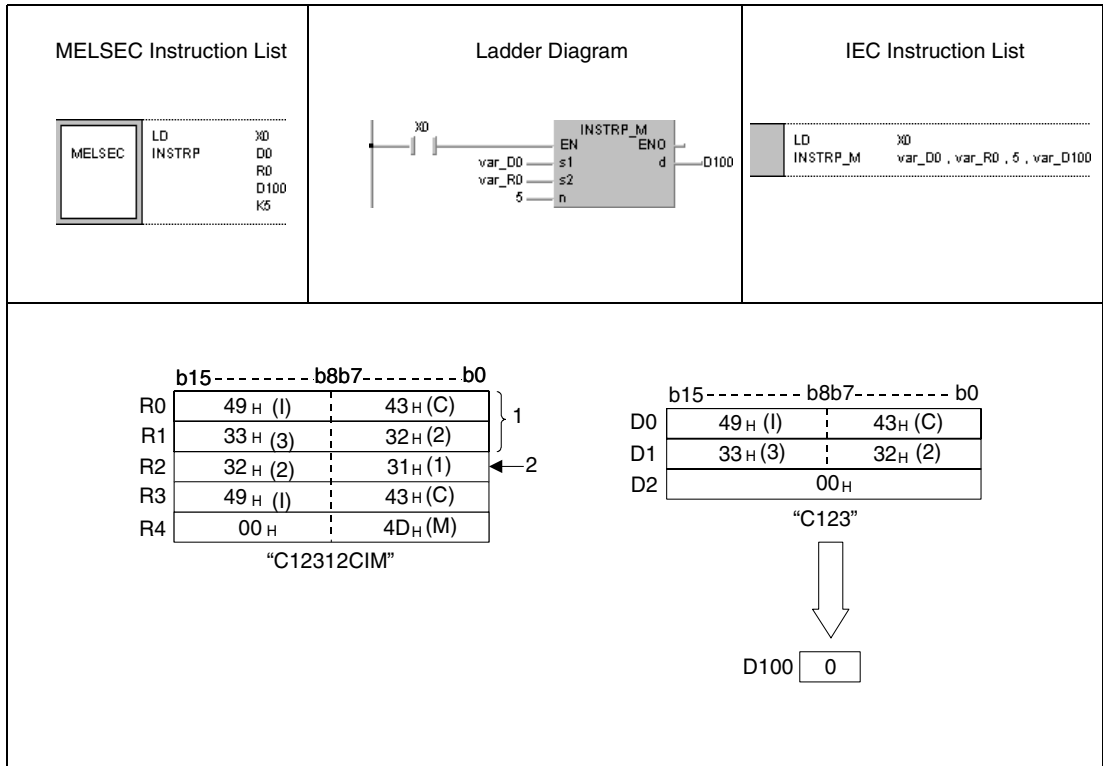
In the following cases an operation error occurs and the error flag is set:

- The initial search position stored in n exceeds the range of (s2) through (s2)+n (error code 4100).

**Program Example 1**

**INSTRP**

With leading edge from X0, the following program searches in R0 onwards beginning with the 5th character for the character string specified in D0 through D2. The result (0) is stored in D100.

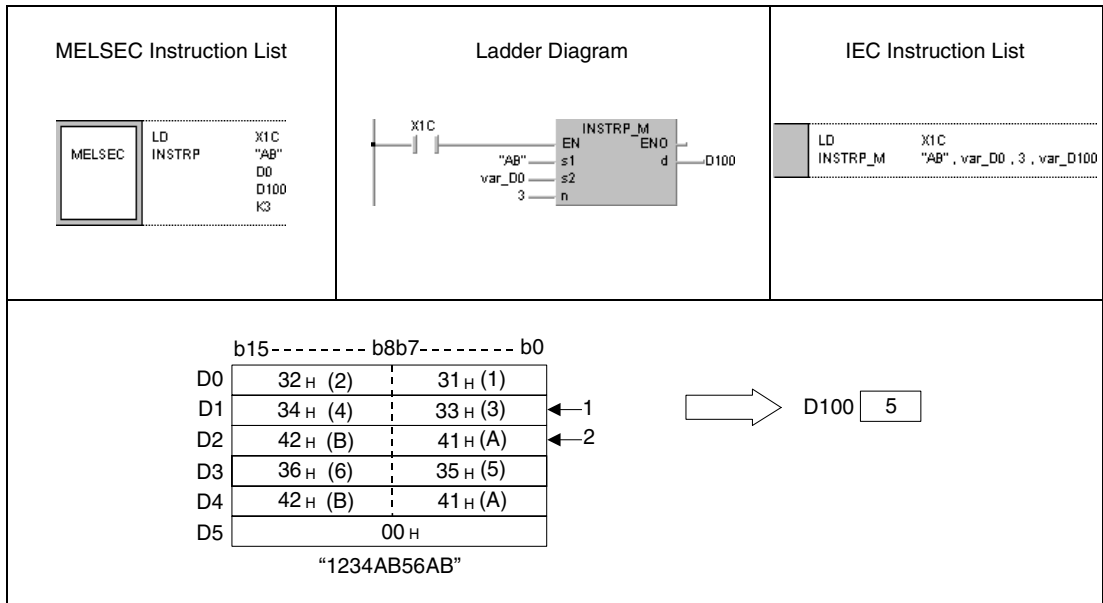


<sup>1</sup> This area is not searched through.  
<sup>2</sup> The search begins with the 5th character.



**Program Example 2** INSTRP

With leading edge from X0, the following program searches in D0 onwards beginning with the 3rd character for the character string "AB". The search result (5) is stored in D100.



- <sup>1</sup> The search begins with the 3rd character.
- <sup>2</sup> The searched character string begins at the 5th character.

**NOTE**

*These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.19 EMOD, EMODP

CPU

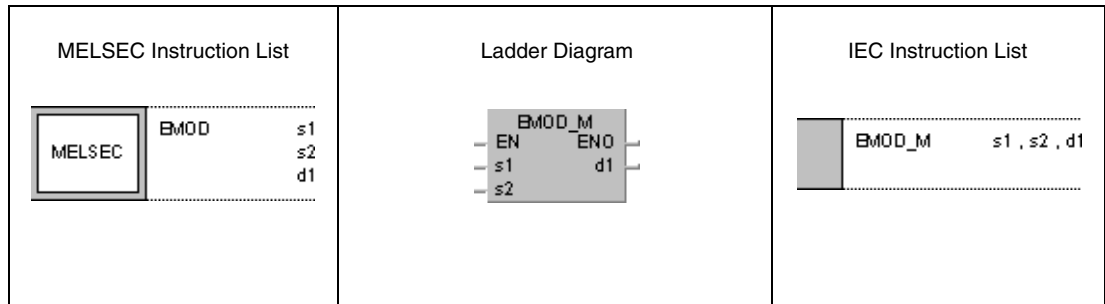
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

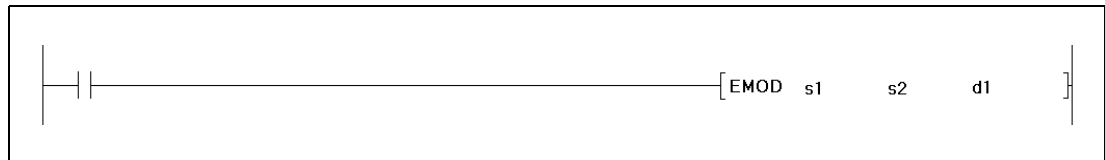
Devices  
MELSEC Q

	Usable Devices										Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant		Other		
	Bit	Word		Bit	Word			K, H (16#)	E			
s1	—	●	●	—	●	●	—	—	●	—	SM0	4
s2	●	●	●	●	●	●	●	●	—	—		
d1	—	●	●	—	—	—	—	—	—	—		

GX IEC Developer



GX Developer



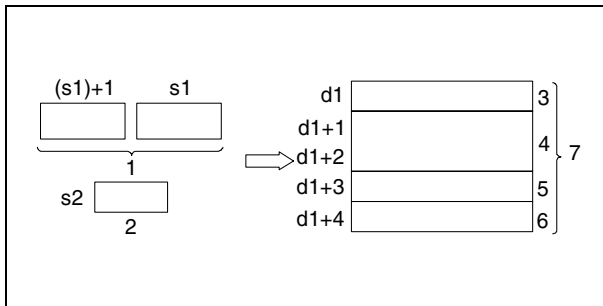
Variables

Set Data	Meaning	Data Type
s1	Floating point data (real number) or first number of device storing the floating point data.	Real number
s2	Number of digits the floating point is moved to the right or first number of device storing such data.	BIN 16-bit
d1	First number of device storing the floating point number in BCD data format.	

**Functions Conversion of floating point number into the BCD format**

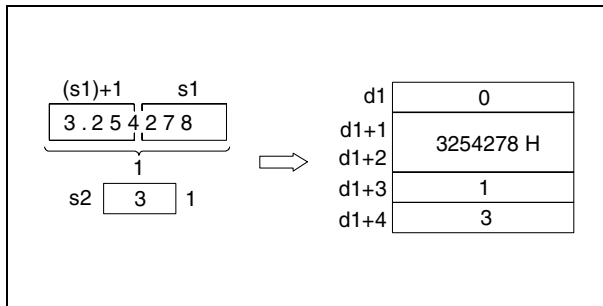
**EMOD Conversion into the BCD format**

The EMOD instruction calculates the BCD format from the floating point number (real number) in s1 and (s1)+1 considering the decimal point shift to the right specified in s2. The result is stored in d1 through (d1)+4.

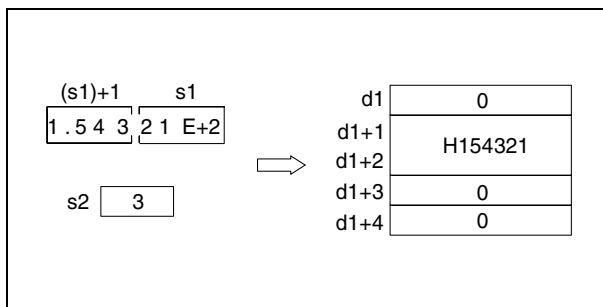
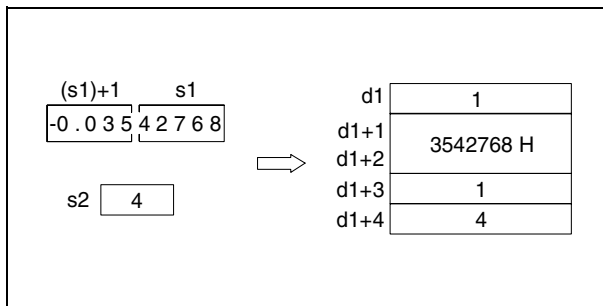


- <sup>1</sup> Floating point data (real number)
- <sup>2</sup> Shift of the decimal point to the right
- <sup>3</sup> Sign bit (0 = positive / 1 = negative)
- <sup>4</sup> 7 BCD digits
- <sup>5</sup> Exponent sign (0 = positive / 1 = negative)
- <sup>6</sup> BCD exponent (Value range 0 to 38)
- <sup>7</sup> Floating point number in BCD data format

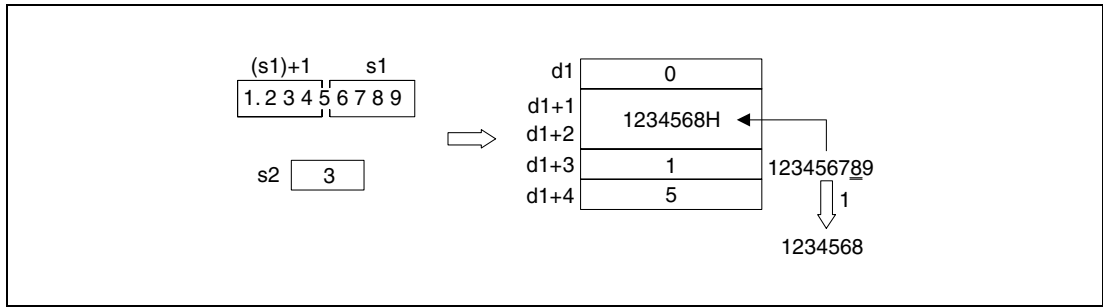
The following diagrams show conversion examples.



- <sup>1</sup> Floating point data (real number)



The floating point number in s1 and (s1)+1 is rounded up to 7 digits and stored in (d1)+1 and (d1)+2.



<sup>1</sup> Rounded up

**Operation Errors**

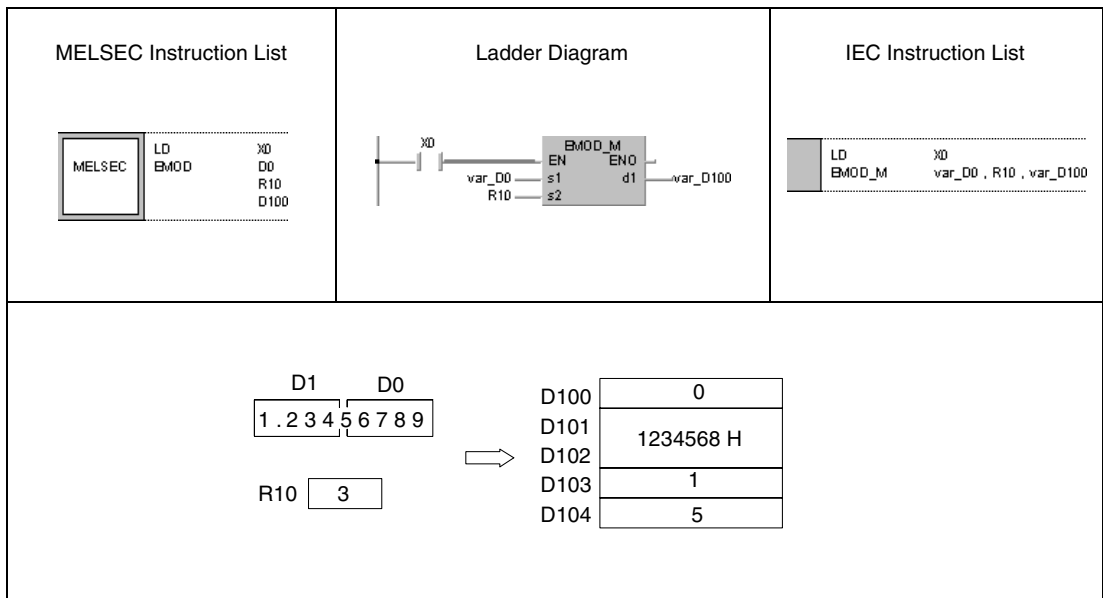
In the following cases an operation error occurs and the error flag is set:

- The number of digits of the decimal point shift (s2) exceeds the range of 0 to 7 (error code 4100).
- The value entered in d1 through (d1)+4 exceeds the relevant storage device area (error code 4101).

**Program Example**

EMOD

While X0 is set, the following program converts the floating point data (real number) specified in D0 and D1 considering the decimal point shift specified in R10. The result is stored in D100 through D104.



**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.11.20 EREXP, EREXPP

CPU


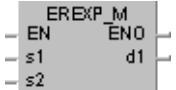
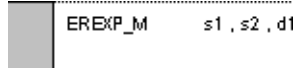
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

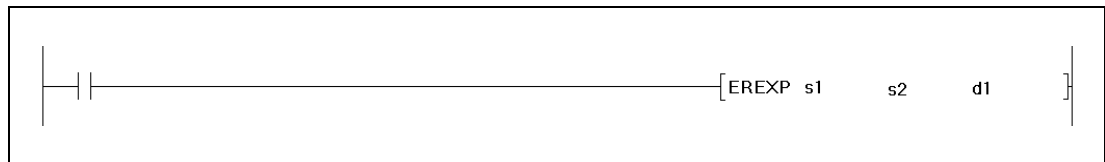
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other U		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	3
s2	●	●	●	●	●	●	●	●	—		
d1	—	●	●	—	●	●	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

GX Developer



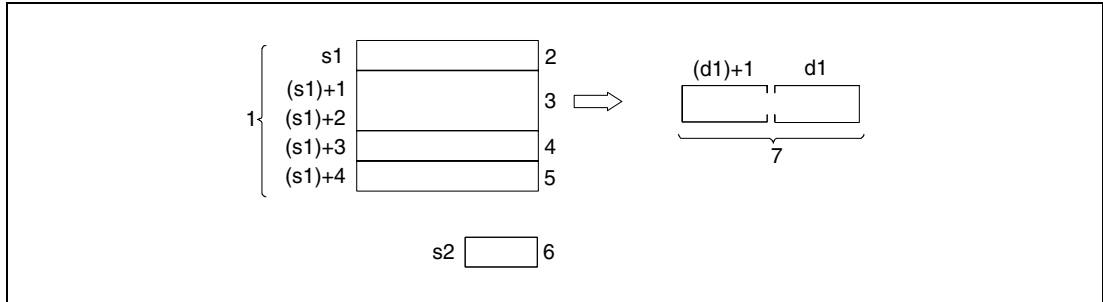
Variables

Set Data	Meaning	Data Type
s1	First number of device storing floating point data in BCD data format.	BIN 16-bit
s2	Specification of decimal places or device storing such data.	
d1	Device storing floating point data (real number).	Real number

**Functions Conversion of floating point data into the decimal format**

**EREXP Conversion into the decimal format**

The EREXP instruction calculates the decimal format of the floating point data (real number) from the floating point data in BCD format in s1 through (s1)+4, considering the decimal places specified in s2. The result is stored in d1 and (d1)+1.

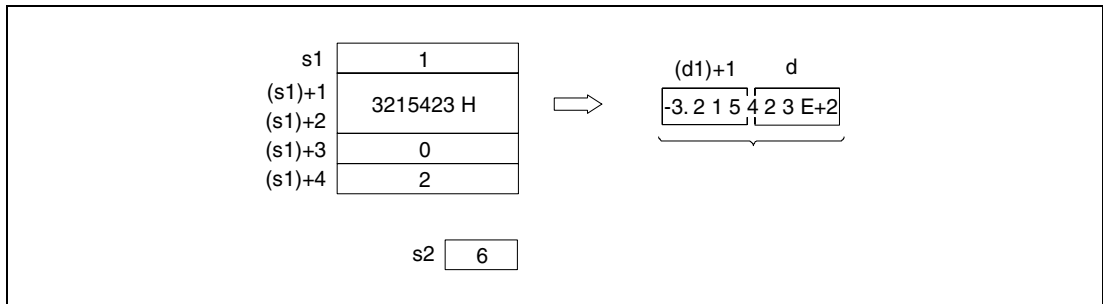


- <sup>1</sup> Floating point data in BCD data format
- <sup>2</sup> Sign bit (0 = positive / 1 = negative)
- <sup>3</sup> 7 BCD digits
- <sup>4</sup> Exponent sign (0 = positive / 1 = negative)
- <sup>5</sup> BCD exponent (value range 0 to 38)
- <sup>6</sup> Number of decimal places (value range 0 to 7)
- <sup>7</sup> Floating point data (real number)

The sign in s1 and the sign of the exponent in (s1)+3 is set to 0 for a positive value. For a negative value the sign bit is 1.

The value of the BCD exponent (s1)+4 may range from 0 to 7.

The decimal places in s2 may range from 0 to 7.



**Operation Errors**

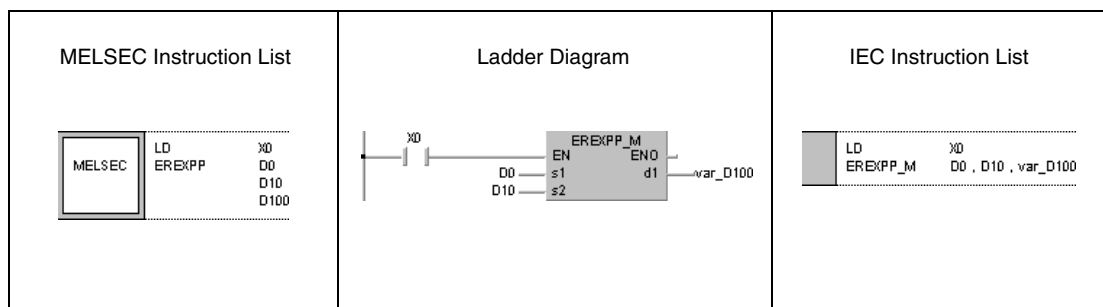
In the following cases an operation error occurs and the error flag is set:

- The sign designation in s1 is not 0 or 1 (error code 4100).
- The BCD data in (s1)+1 and (s1)+2 contains more than 8 digits (error code 4100).
- The exponent sign in (s1)+3 is not 0 or 1 (error code 4100).
- The exponent data in (s1)+4 exceeds the range from 0 to 38 (error code 4100).
- The number of decimal places in s2 exceeds the range of 0 to 7 (error code 4101).

**Program Example**

**EREXPP**

With leading edge from X0, the following program calculates the floating point value (real number) in decimal format from the floating point value in BCD format specified in D0 through D4 considering the decimal places specified in D10. The result is stored in D100 and D101.



## 7.12 Special functions

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Sine calculation	SIN	SIN_MD
		SIN_E_MD
	SINP	SIN_P_MD
		SIN_P_E_MD
Cosine calculation	COS	COS_MD
		COS_E_MD
	COSP	COS_P_MD
		COS_P_E_MD
Tangent calculation	TAN	TAN_MD
		TAN_E_MD
	TANP	TAN_P_MD
		TAN_P_E_MD
Arcus sine calculation	ASIN	ASIN_MD
		ASIN_E_MD
	ASINP	ASIN_P_MD
		ASIN_P_E_MD
Arcus cosine calculation	ACOS	ACOS_MD
		ACOS_E_MD
	ACOSP	ACOS_P_MD
		ACOS_P_E_MD
Arcus tangent calculation	ATAN	ATAN_MD
		ATAN_E_MD
	ATANP	ATAN_P_MD
		ATAN_P_E_MD
Conversion from degrees into radian	RAD	RAD_MD
		RAD_E_MD
	RADP	RAD_P_MD
		RAD_P_E_MD
Conversion from radian into degree	DEG	DEG_MD
		DEG_E_MD
	DEGP	DEG_P_MD
		DEG_P_E_MD
Square root	SQR	SQR_MD
		SQR_E_MD
	SQRP	SQR_P_MD
		SQR_P_E_MD
Floating point value as exponent of e	EXP	EXP_MD
		EXP_E_MD
	EXPP	EXP_P_MD
		EXP_P_E_MD



Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Logarithm (natural) calculation	LOG	LOG_MD
		LOG_E_MD
	LOGP	LOG_P_MD
		LOG_P_E_MD
Randomize value	RND	RND_M
	RNDP	RNDP_M
Update random values	SRND	SRND_M
	SRNDP	SRNDP_M
Square root calculation from 4-digit BCD data	BSQR	BSQR_MD
		BSQR_K_MD
	BSQRP	BSQR_P_MD
		BSQR_K_P_MD
Square root calculation from 8-digit BCD data	BDSQR	BDSQR_MD
		BDSQR_K_MD
	BDSQRP	BDSQR_P_MD
		BDSQR_K_P_MD
Sine calculation from BCD data	BSIN	BSIN_MD
		BSIN_K_MD
	BSINP	BSIN_P_MD
		BSIN_K_P_MD
Cosine calculation from BCD data	BCOS	BCOS_MD
		BCOS_K_MD
	BCOSP	BCOS_P_MD
		BCOS_K_P_MD
Tangent calculation from BCD data	BTAN	BTAN_MD
		BTAN_K_MD
	BTANP	BTAN_P_MD
		BTAN_K_P_MD
Arcus sine calculation from BCD data	BASIN	BASIN_MD
	BASINP	BASIN_P_MD
Arcus cosine calculation from BCD data	BACOS	BACOS_MD
	BACOSP	BACOS_P_MD
Arcus tangent calculation from BCD data	BATAN	BATAN_MD
	BATANP	BATAN_P_MD

**NOTE**      *Within the IEC editors please use the IEC instructions.*

7.12.1 SIN, SINP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

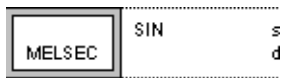
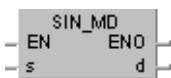
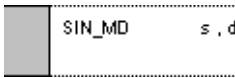
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

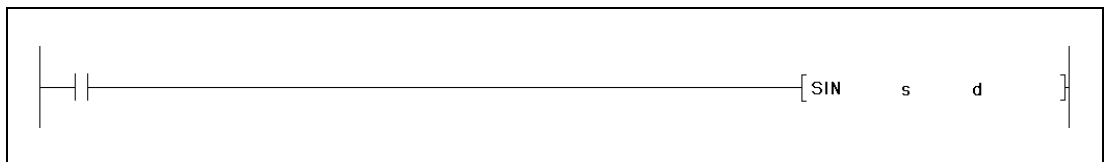
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	3	
d	—	●	●	—	●	●	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	--

GX Developer

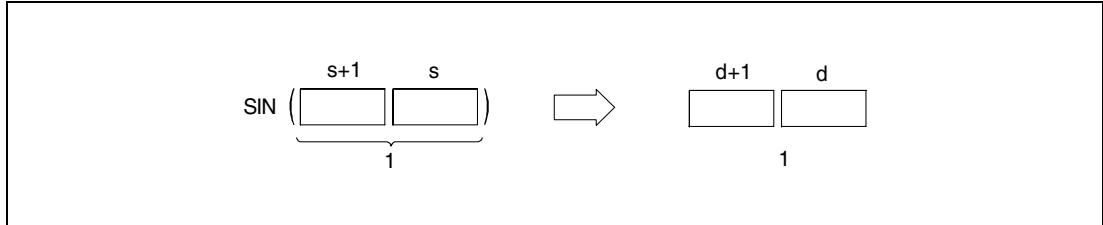


Variables

Set Data	Meaning	Data Type
s	First number of device storing angle data for the SIN instruction (sine).	Real number
d	First number of device storing the operation result.	

**Functions**    **Sine calculation from floating point values****SIN**    **Sine calculation**

The SIN instruction calculates the sine value from angle data in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The angle in s and s+1 must be specified in radian measure (degrees  $\times \pi/180$ ). The conversion from degrees into radian is described in the sections on the RAD and DEG instructions.

**Operation Error**

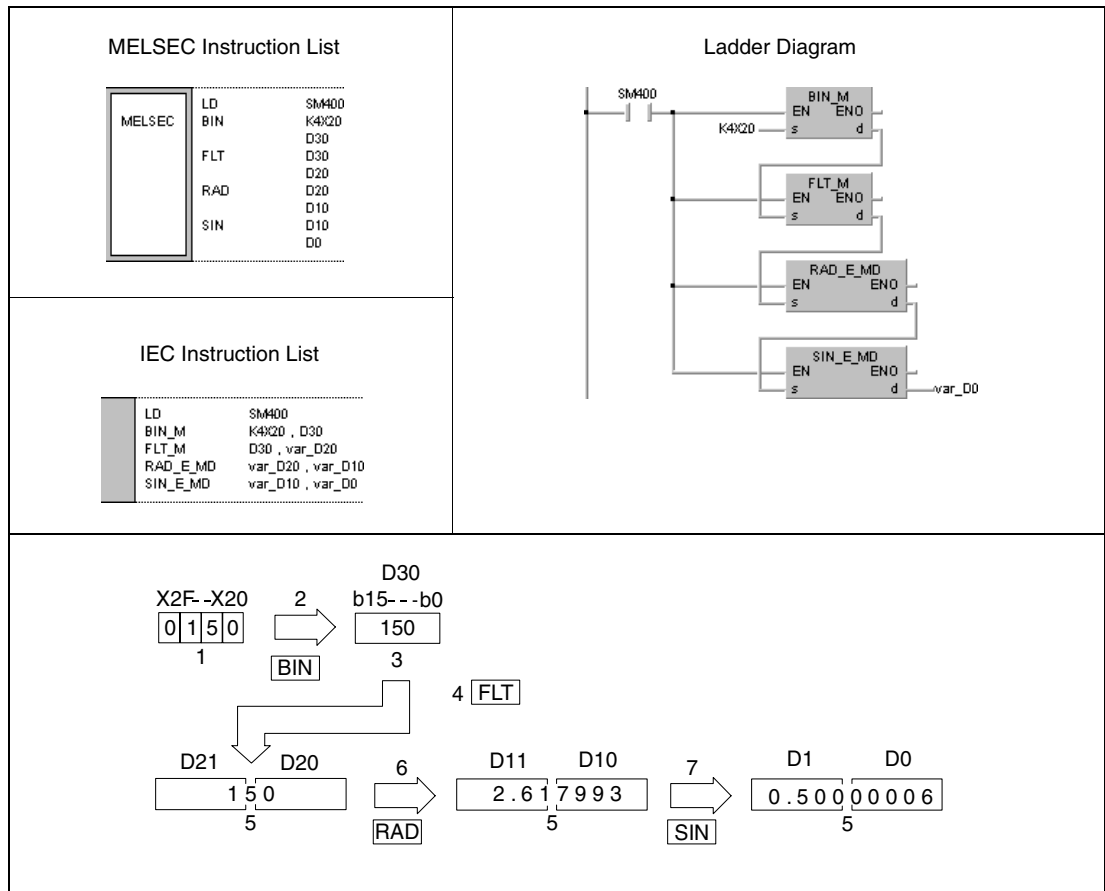
In the following cases an operation error occurs and the error flag is set:

- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

**SIN**

While SM400 is set, the following program calculates the sine value from the 4-digit BCD angle specification in X20 through X2F. The result is stored as floating point value (real number) in D0 and D1.



- 1 BCD value
- 2 Conversion into the BIN format
- 3 BIN value
- 4 Conversion into the floating point format
- 5 Floating point value (real number)
- 6 Conversion into the radian measure
- 7 Calculation of the sine value

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 7.12.2 COS, COSP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

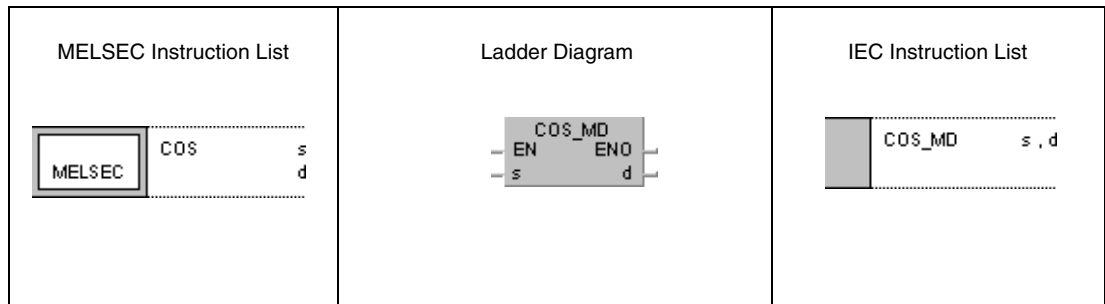
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

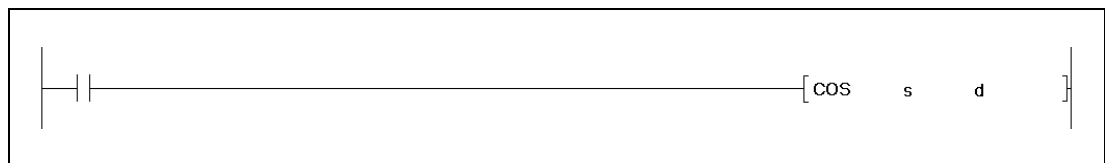
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	3	
d	—	●	●	—	●	●	—	—	—		

**GX IEC  
Developer**



**GX  
Developer**



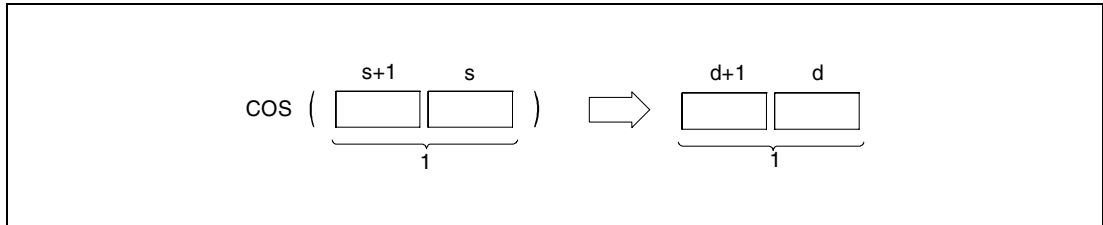
**Variables**

Set Data	Meaning	Data Type
s	First number of device storing angle data for the COS instruction (cosine).	Real number
d	First number of device storing the operation result.	

**Functions**      **Cosine calculation from floating point values**

**COS**      **Cosine calculation**

The COS instruction calculates the cosine value from angle data in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The angle in s and s+1 must be specified in radian measure (degrees x  $\pi/180$ ). The conversion from degrees into radian is described in the sections on the RAD and DEG instructions.

**Operation Error**

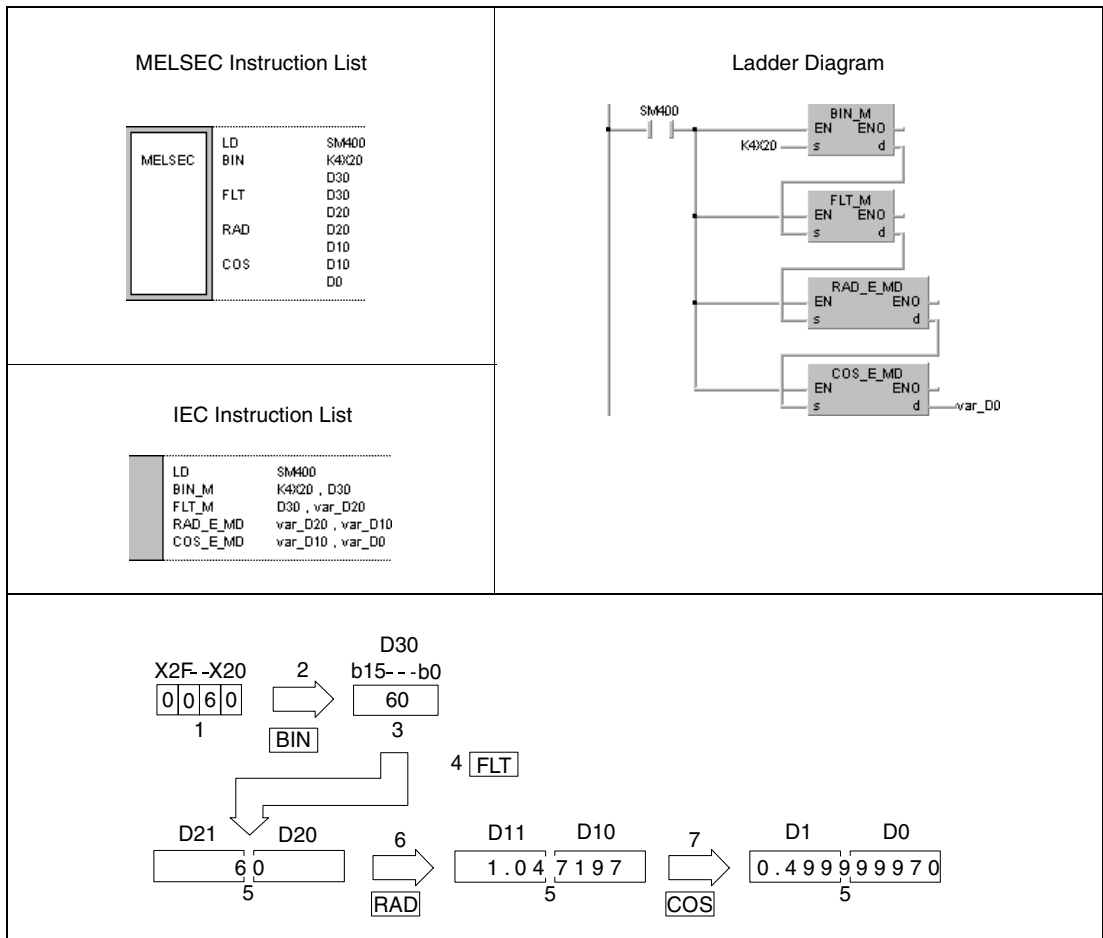
In the following cases an operation error occurs and the error flag is set:

- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

**COS**

While SM400 is set, the following program calculates the cosine value from the 4-digit BCD angle specification in X20 through X2F. The result is stored as floating point value (real number) in D0 and D1.



- 1 BCD value
- 2 Conversion into the BIN format
- 3 Binary value
- 4 Conversion into the floating point format
- 5 Floating point value (real number)
- 6 Conversion into the radian measure
- 7 Calculation of the cosine value

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.3 TAN, TANP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

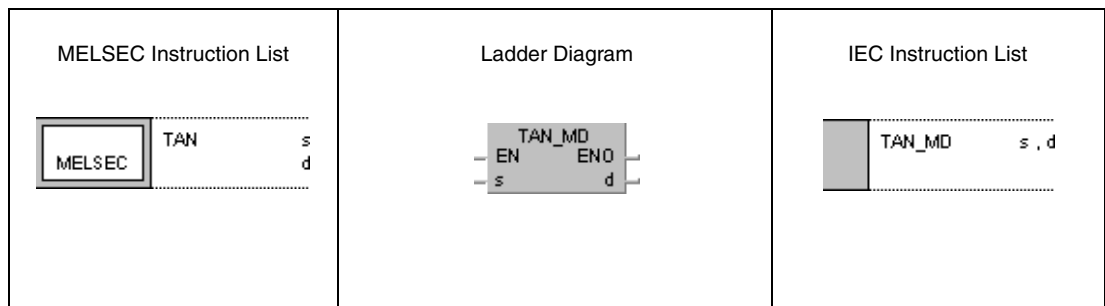
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

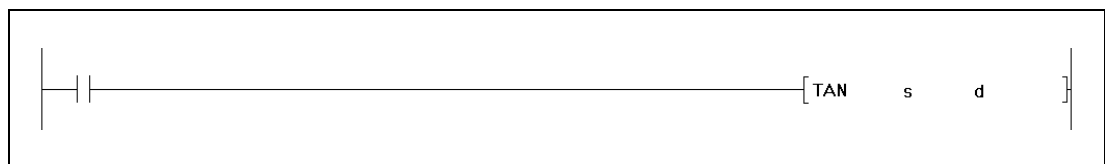
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		

GX IEC Developer



GX Developer



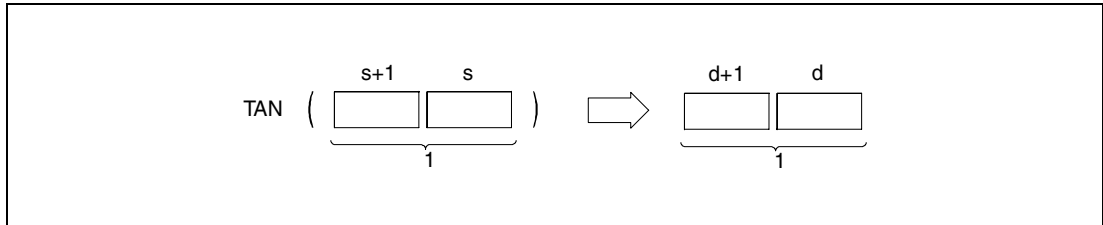
Variables

Set Data	Meaning	Data Type
s	First number of device storing angle data for the TAN instruction (tangent).	Real number
d	First number of device storing the operation result.	



**Functions**    **Tangent calculation from floating point values****TAN**    **Tangent calculation**

The TAN instruction calculates the tangent value from angle data in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The angle in s and s+1 must be specified in radian measure (degrees  $\times \pi/180$ ). The conversion from degrees into radian is described in the sections on the RAD and DEG instructions.

If the angle in s and s+1 retains the values  $\pi/2$  rad or  $(3/2)\times\pi$  rad, an error message is returned from the radian measure calculation.

**Operation Errors**

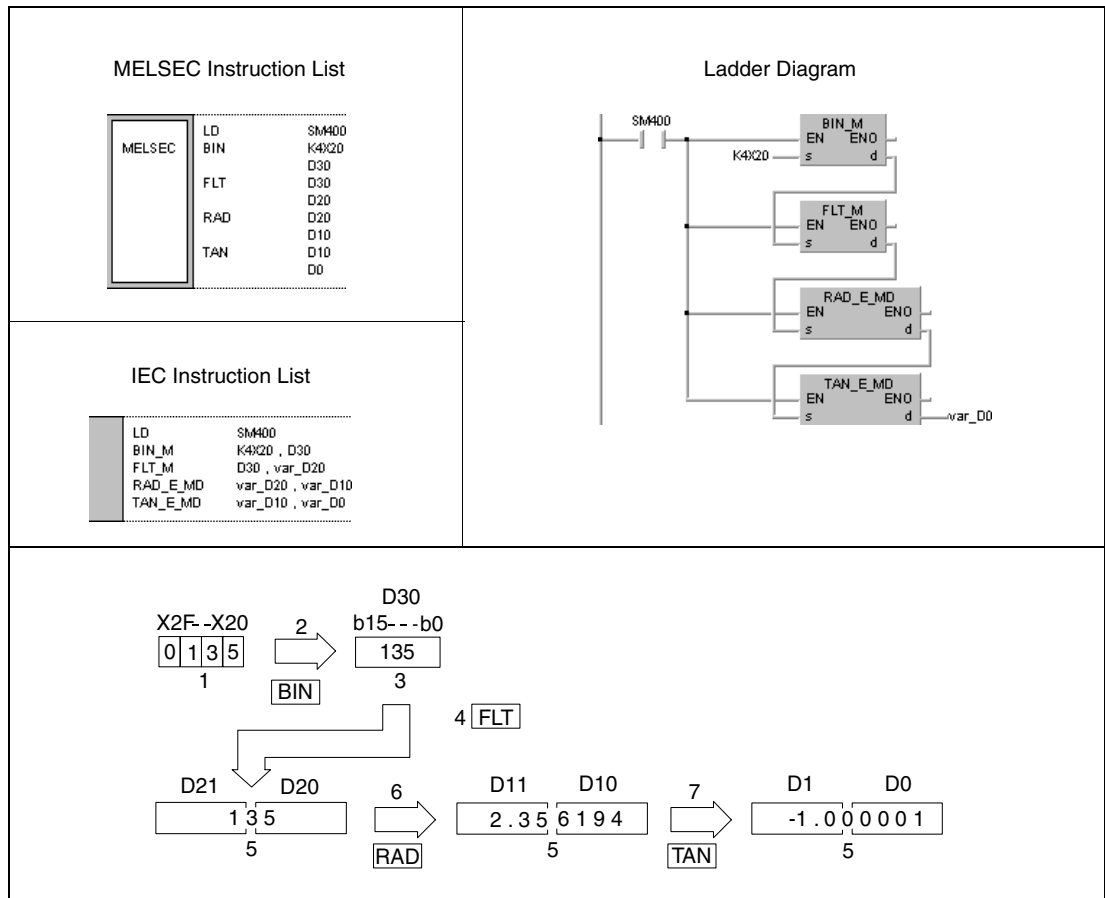
In the following cases an operation error occurs and the error flag is set:

- The operation result is zero or does not range from  $\pm 2^{-127}$  to  $\pm 2^{129}$  (error code 4100).
- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

**TAN**

With leading edge from SM400, the following program calculates the tangent value from the 4-digit BCD angle specification in X20 through X2F. The result is stored as floating point value (real number) in D0 and D1.



- <sup>1</sup> BCD value
- <sup>2</sup> Conversion into the BIN format
- <sup>3</sup> Binary value
- <sup>4</sup> Conversion into the floating point format
- <sup>5</sup> Floating point value (real number)
- <sup>6</sup> Conversion into the radian measure
- <sup>7</sup> Calculation of the tangent value

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 7.12.4 ASIN, ASINP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

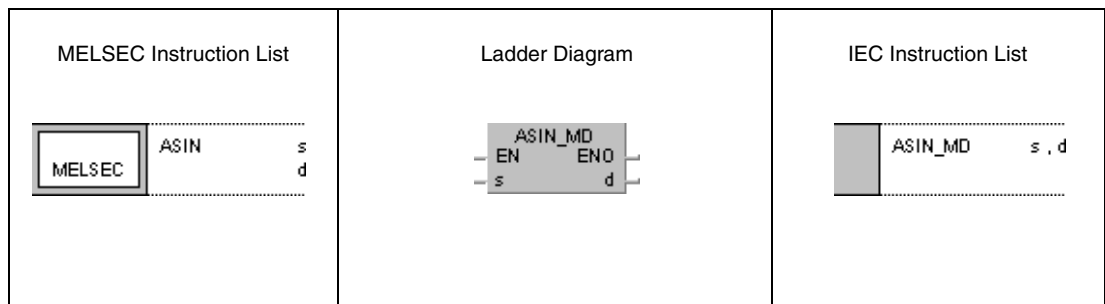
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

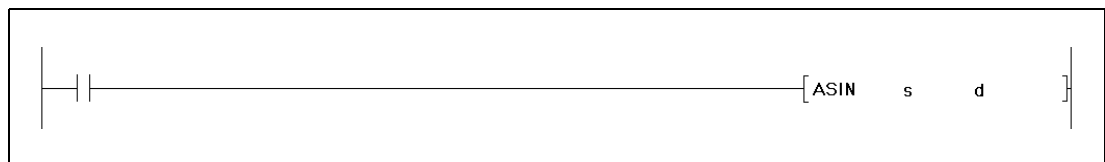
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		

**GX IEC Developer**



**GX Developer**

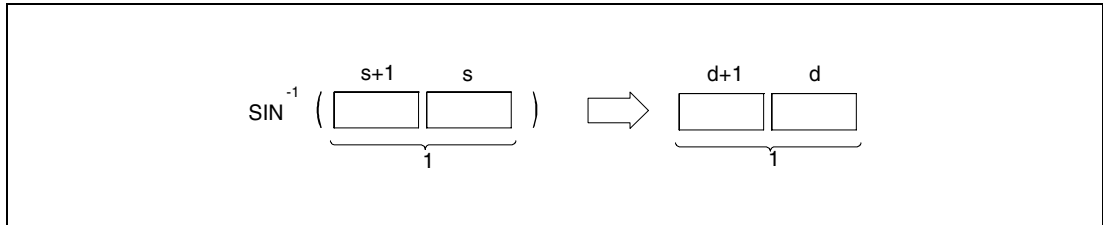


**Variables**

Set Data	Meaning	Data Type
s	First number of device storing sine value for the calculation of the arcus sine.	Real number
d	First number of device storing the operation result.	

**Functions**    **Arcus sine calculation of floating point values****ASIN**    **Arcus sine calculation**

The ASIN instruction calculates the angle from the sine value in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The sine value in s and s+1 may range within the value range of -1 to 1.

The angle in s and s+1 must be specified in radian measure (degrees  $\times \pi/180$ ). The conversion from degrees into radian is described in the sections on the RAD and DEG instructions.

**Operation Errors**

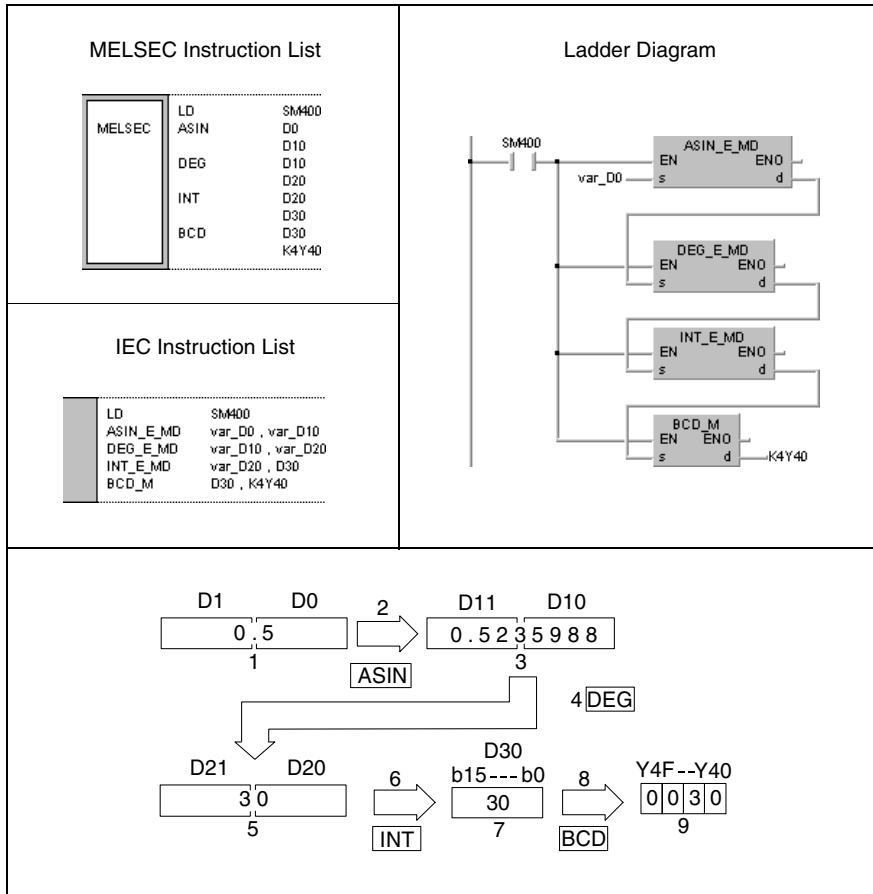
In the following cases an operation error occurs and the error flag is set:

- The value in s and s+1 exceeds the value range of -1 to 1 (error code 4100).
- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

ASIN

While SM400 is set, the following program calculates the arcus sine value from the floating point data (real number) in D0 and D1. The resulting angle in radian measure is output at Y20 through Y4F as 4-digit BCD value.



- 1 Floating point value (real number)
- 2 Arcus sine calculation
- 3 Floating point value (real number)
- 4 Conversion of the angle measures
- 5 Floating point value (real number)
- 6 Conversion into the BIN format
- 7 Binary value
- 8 Conversion into the BCD format
- 9 BCD value

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.5 ACOS, ACOSP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

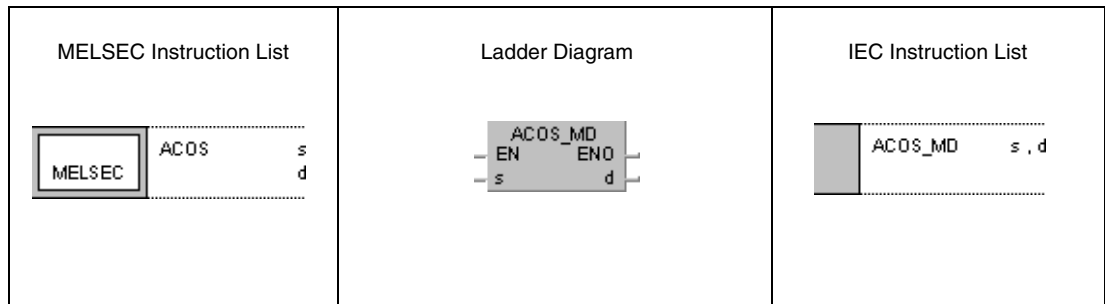
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

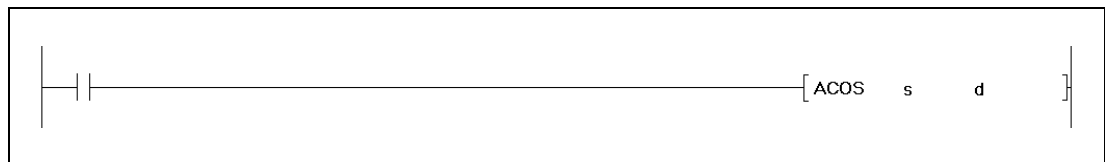
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant E	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		

GX IEC  
Developer



GX  
Developer

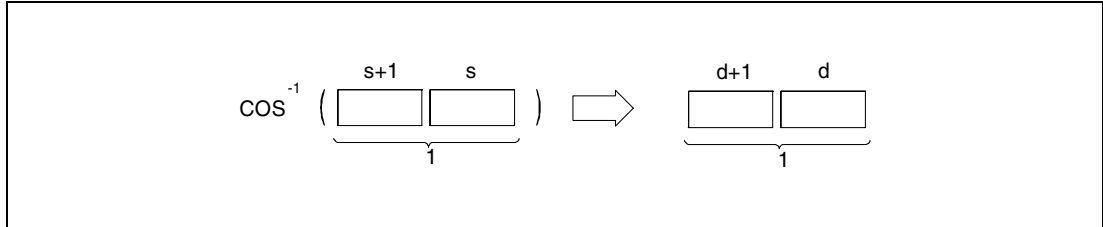


Variables

Set Data	Meaning	Data Type
s	First number of device storing cosine value for the calculation of the arcus cosine.	Real number
d	First number of device storing the operation result.	

**Functions**    **Arcus cosine calculation of floating point values****ACOS**    **Arcus cosine calculation**

The ACOS instruction calculates the angle from the cosine value in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The cosine value in s and s+1 may range within the value range of -1 to 1.

The angle in s and s+1 must be specified in radian measure (degrees  $\times \pi/180$ ). The conversion from degrees into radian is described in the sections on the RAD and DEG instructions.

**Operation Errors**

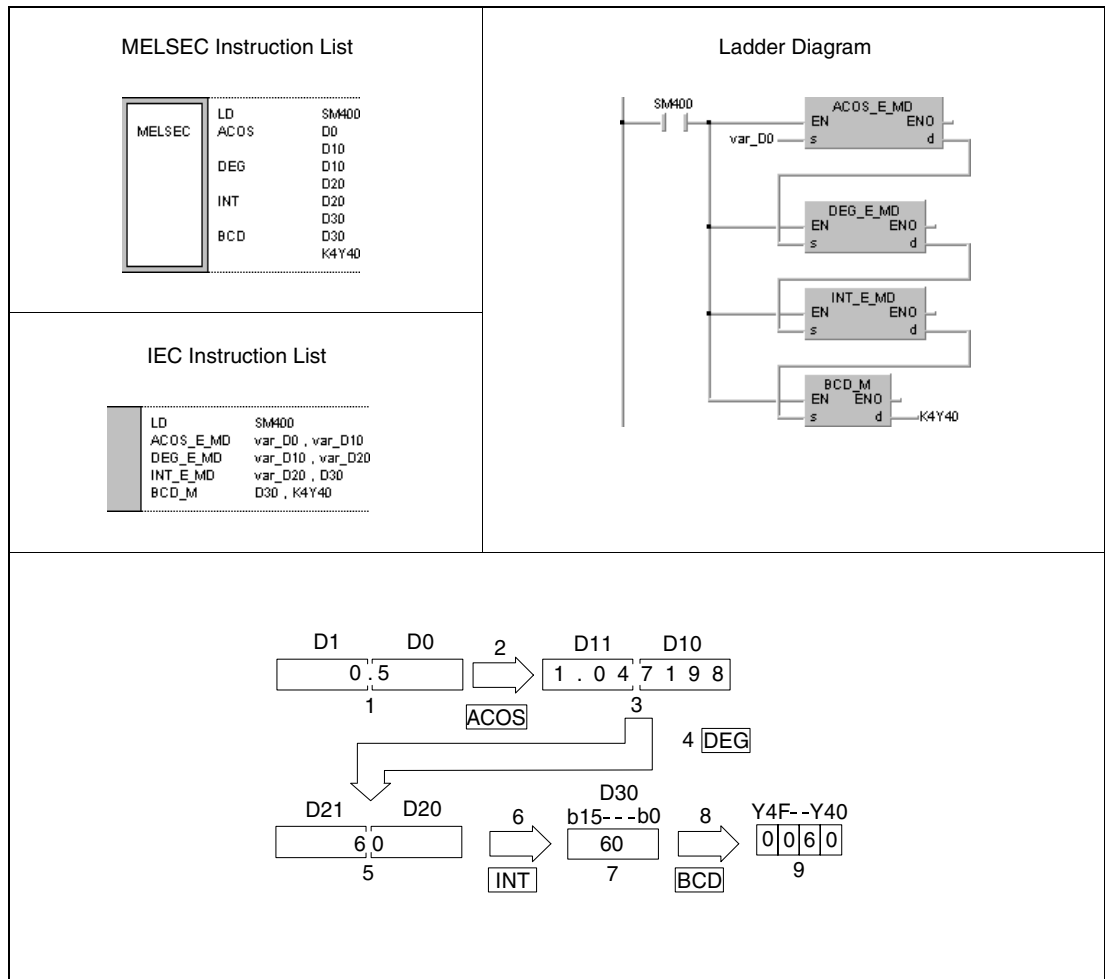
In the following cases an operation error occurs and the error flag is set:

- The value in s and s+1 exceeds the value range of -1 to 1 (error code 4100).
- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

**ACOS**

While SM400 is set, the following program calculates the arcus cosine value from the floating point data (real number) in D0 and D1. The resulting angle in radian measure is output at Y20 through Y4F as 4-digit BCD value.



- 1 Floating point value (real number)
- 2 Arcus cosine calculation
- 3 Floating point value (real number)
- 4 Conversion of the angle measures
- 5 Floating point value (real number)
- 6 Conversion into the BIN format
- 7 Binary value
- 8 Conversion into the BCD format
- 9 BCD value

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



7.12.6 ATAN, ATANP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

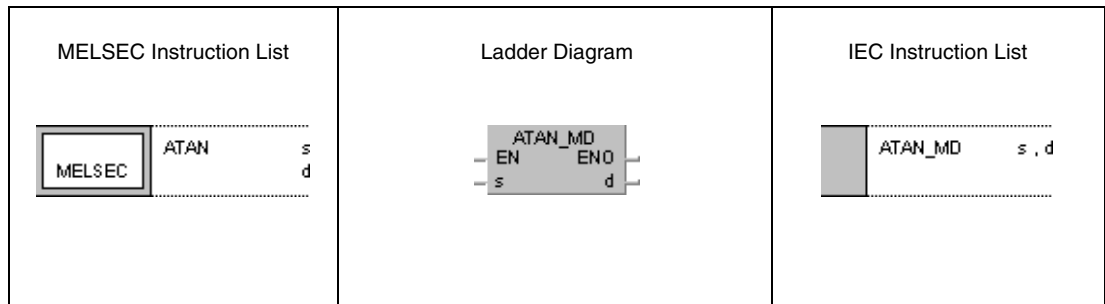
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

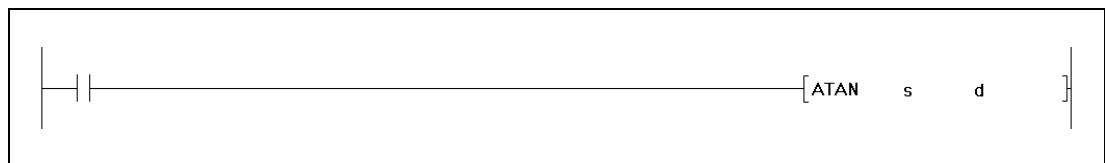
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	3	
d	—	●	●	—	●	●	—	—	—		

GX IEC Developer



GX Developer



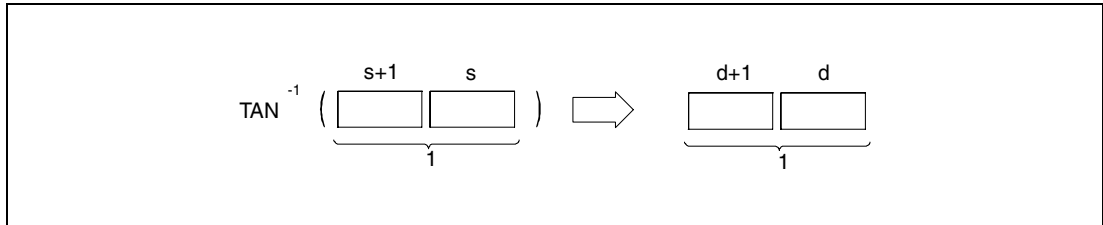
Variables

Set Data	Meaning	Data Type
s	First number of device storing tangent value for the calculation of the arcus tangent.	Real number
d	First number of device storing the operation result.	

**Functions**     **Arcus tangent calculation of floating point values**

**ATAN**     **Arcus tangent calculation**

The ATAN instruction calculates the angle from the cosine value in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The angle in s and s+1 must be specified in radian measure (degrees x  $\pi/180$ ). The conversion from degrees into radian is described in the sections on the RAD and DEG instructions.

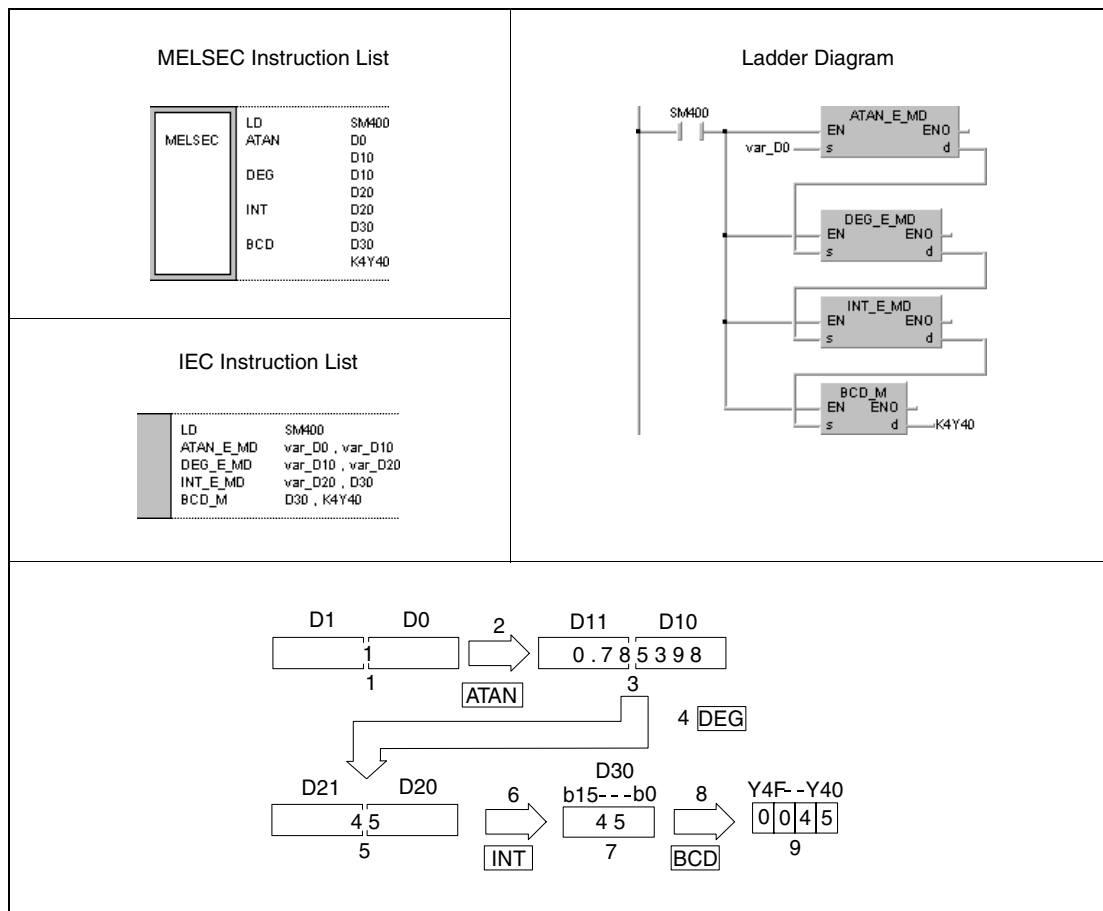
**Operation Error**

In the following cases an operation error occurs and the error flag is set:

- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example** ATAN

While SM400 is set, the following program calculates the arcus tangent value from the floating point data (real number) in D0 and D1. The resulting angle in radian measure is output at Y20 through Y4F as 4-digit BCD value.



- 1 Floating point value (real number)
- 2 Arcus tangent calculation
- 3 Floating point value (real number)
- 4 Conversion of the angle measures
- 5 Floating point value (real number)
- 6 Conversion into the BIN format
- 7 Binary value
- 8 Conversion into the BCD format
- 9 BCD value

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.7 RAD, RADP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

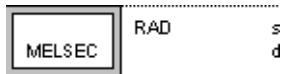
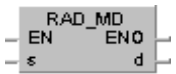
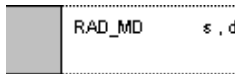
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

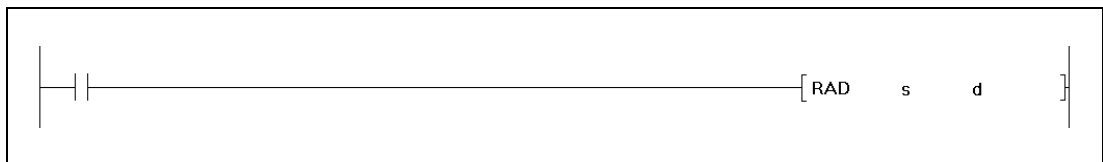
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	3	
d	—	●	●	—	●	●	—	—	—		

GX IEC  
Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	--

GX  
Developer

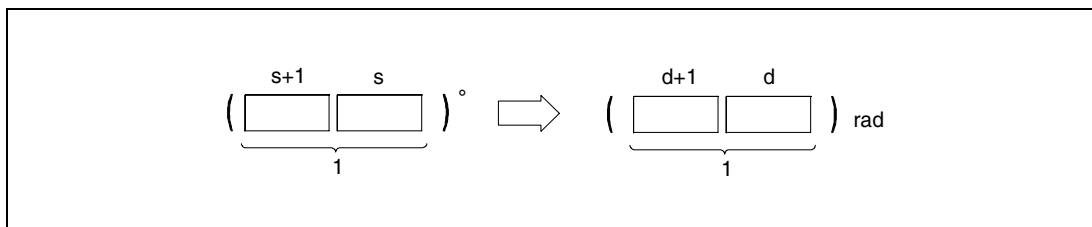


Variables

Set Data	Meaning	Data Type
s	First number of device storing degree value to be converted into radiant value.	Real number
d	First number of device storing conversion result.	

**Functions**    **Conversion from degrees into radian as floating point value****RAD**    **Conversion from degrees into radian**

The RAD instruction calculates the radian value (rad) from the degree value (°) in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The conversion from degrees into radian applies to the following equation:

$$\text{Radian value} = \text{degree value} \times \pi / 180$$

**Operation Error**

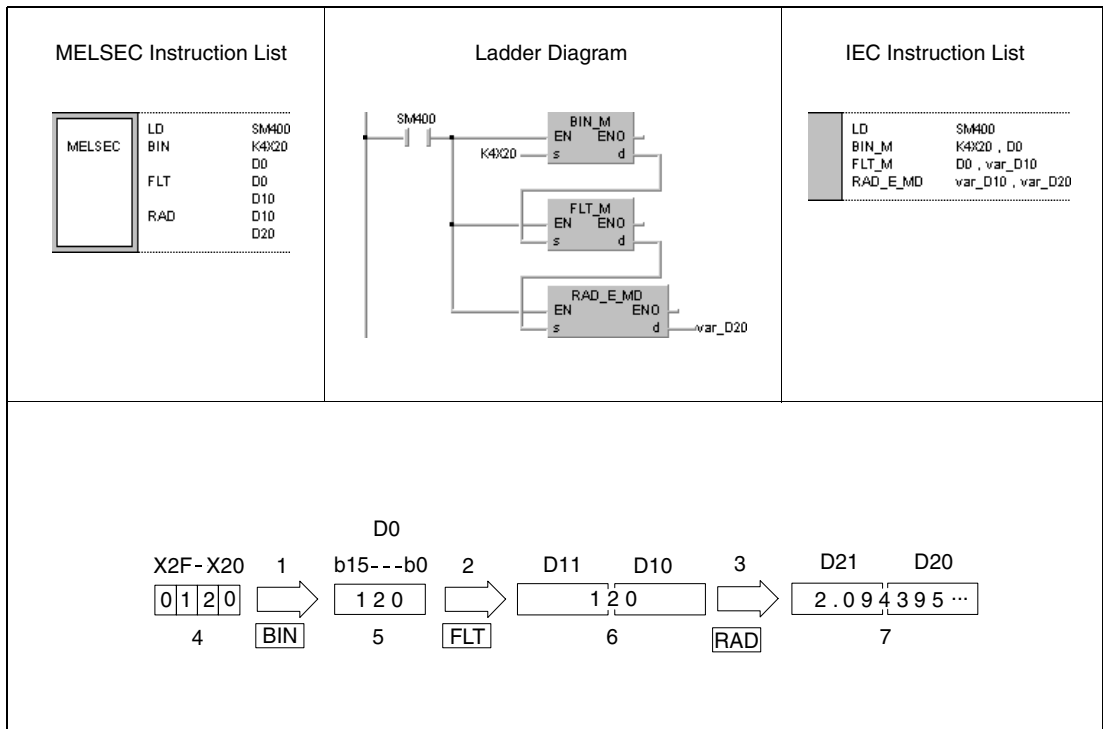
In the following cases an operation error occurs and the error flag is set:

- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

**RAD**

While SM400 is set, the following program calculates the radian value from the degree value of the 4-digit BCD value in X20 through X2F. The result is stored in D20 and D21 as floating point value.



- 1 Conversion into the BIN format
- 2 Conversion into the floating point format
- 3 Conversion into radian measure
- 4 BCD value
- 5 Binary value
- 6 Floating point value (real number)
- 7 Floating point value (real number)

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.8 DEG, DEGP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

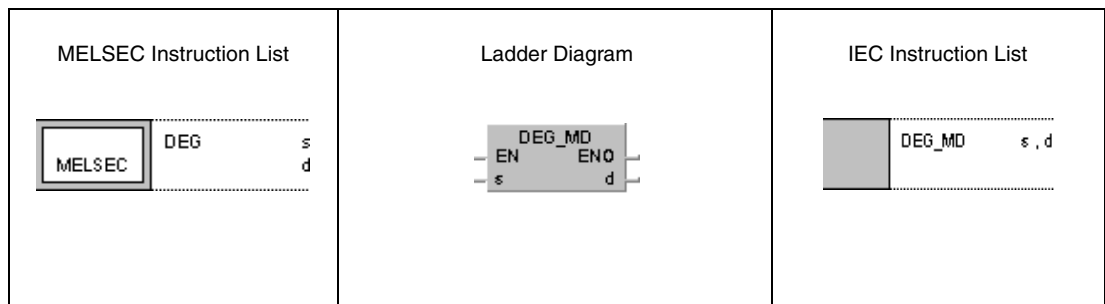
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

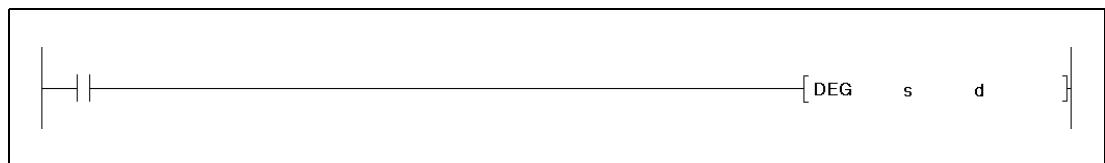
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	3	
d	—	●	●	—	●	●	—	—	—		

GX IEC  
Developer



GX  
Developer



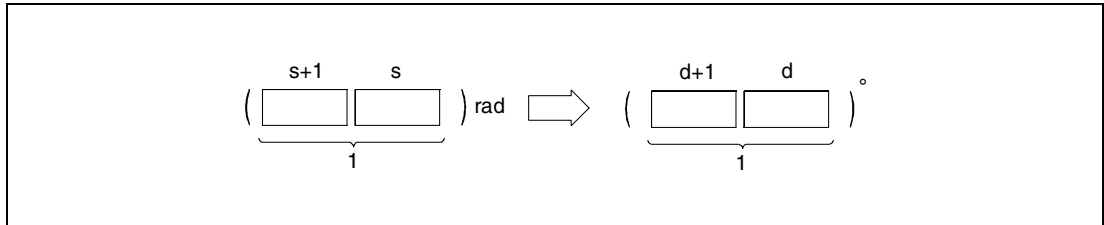
Variables

Set Data	Meaning	Data Type
s	First number of device storing radiant value to be converted into degree value.	Real number
d	First number of device storing conversion result.	

**Functions**      **Conversion from radian in floating point format into degrees**

**DEG**      **Conversion from radian into degrees**

The DEG instruction calculates the degree value (°) from the radian value (rad) in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The conversion from radian into degrees applies to the following equation:

$$\text{Degree value} = \text{radian value} \times 180 / \pi$$

**Operation Error**

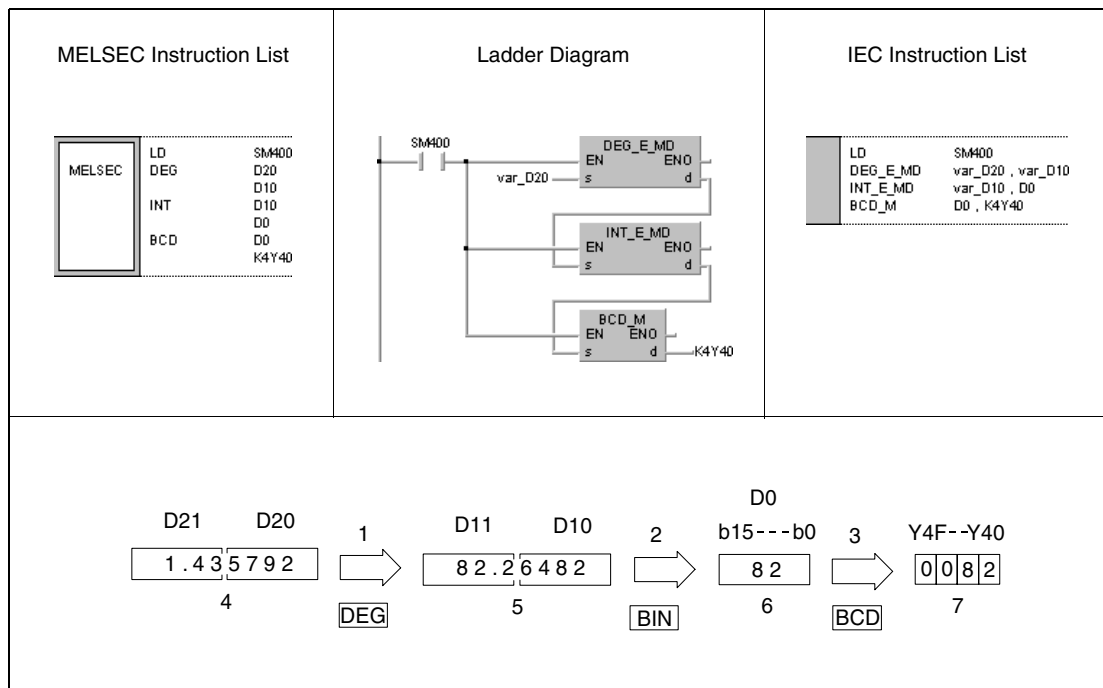
In the following cases an operation error occurs and the error flag is set:

- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).



**Program Example** DEG

While SM400 is set, the following program calculates the degree value from the radian value stored in D20 and D21 in 4-digit BCD format. The result is stored in D20 and D21 as floating point value.



- 1 Conversion into degrees
- 2 Conversion into the BIN format
- 3 Conversion into the BCD format
- 4 Floating point value (real number)
- 5 Floating point value (real number)
- 6 Binary value
- 7 BCD value

**NOTE** This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

7.12.9 SQR, SQRP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

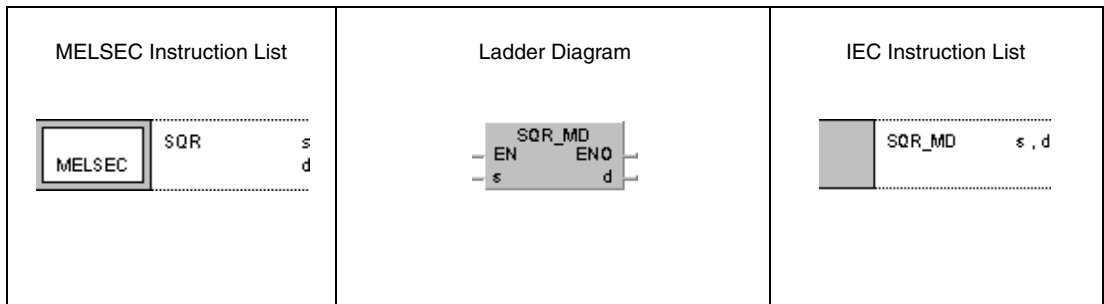
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

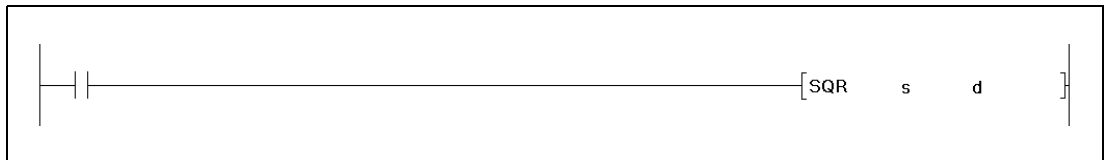
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		

GX IEC Developer



GX Developer

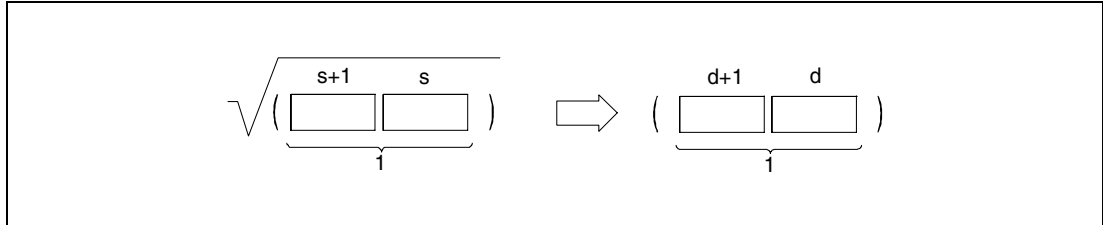


Variables

Set Data	Meaning	Data Type
s	First number of device storing the value for the square root calculation.	Real number
d	First number of device storing the square root result.	

**Functions**    **Square root calculation of floating point values****SQR**    **Square root calculation**

The SQR instruction calculates the square root of the floating point value in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

Only positive values may be stored in s and s+1.  
(Negative values cannot be processed).

**Operation Errors**

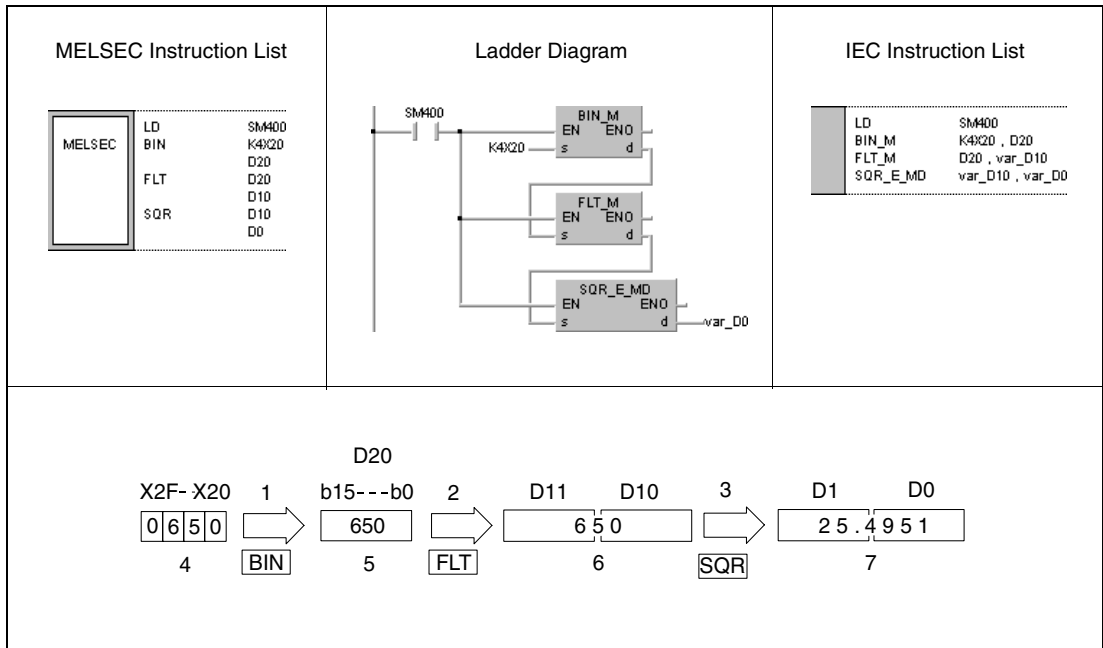
In the following cases an operation error occurs and the error flag is set:

- The value entered in s is negative.
- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

**SQR**

While SM400 is set, the following program calculates the square root of the 4-digit BCD value in X20 through X2F. The result is stored in D0 and D1.



- 1 Conversion into the BIN format
- 2 Conversion into the floating point format
- 3 Square root calculation
- 4 BCD value
- 5 Binary value
- 6 Floating point value (real number)
- 7 Floating point value (real number)

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.10 EXP, EXPP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

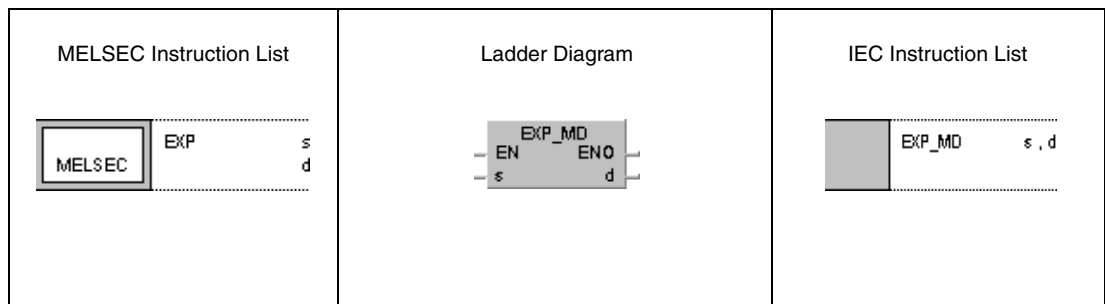
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

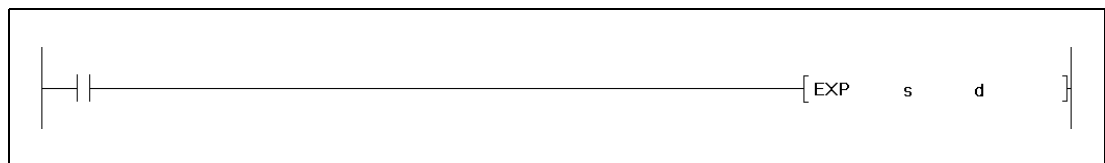
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		

GX IEC  
Developer



GX  
Developer



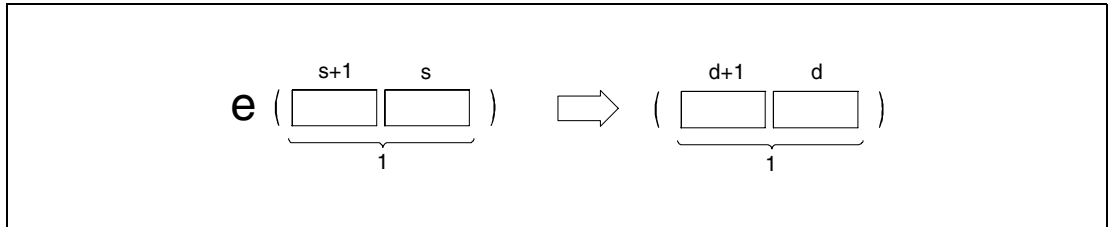
Variables

Set Data	Meaning	Data Type
s	First number of device storing the value for the EXP instruction.	Real number
d	First number of device storing the operation result.	

**Functions** Floating point values as exponent of the base e

**EXP Exponent of e**

The EXP instruction calculates the corresponding exponent to the base e from the floating point value in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

The calculation is based on the Euler's constant: "e = 2.718281828".

**Operation Errors**

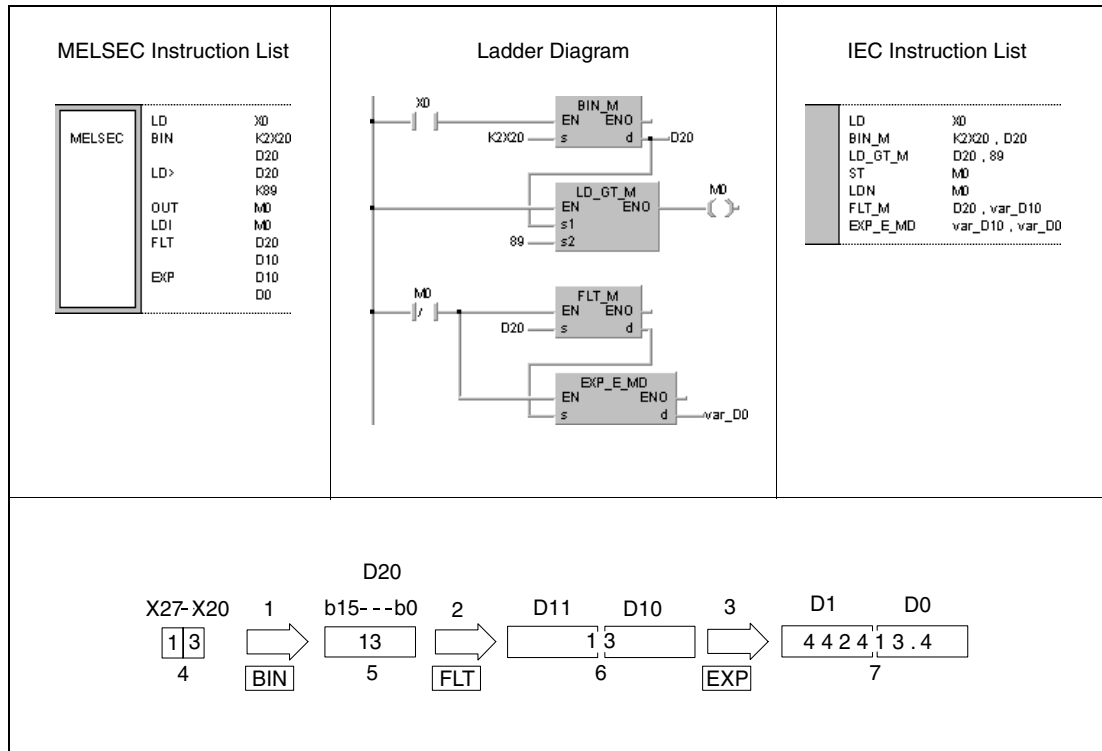
In the following cases an operation error occurs and the error flag is set:

- The calculation result exceeds the value range from  $2^{-127}$  to  $2^{129}$  (error code 4100).
- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

EXP

The following program calculates the result of the exponential function to the base e with the 2-digit BCD value at X20 through X27. The result is stored in D0 and D1 in floating point format.



- 1 Conversion into the BIN format
- 2 Conversion into the floating point format
- 3 Exponential calculation
- 4 BCD value
- 5 Binary value
- 6 Floating point value (real number)
- 7 Floating point value (real number)

**NOTE**

The calculation result must not exceed  $2^{129}$  ln = 89.41598. If the BCD value exceeds the value 90, an error message is returned from SM0.

This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

7.12.11 LOG, LOGP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

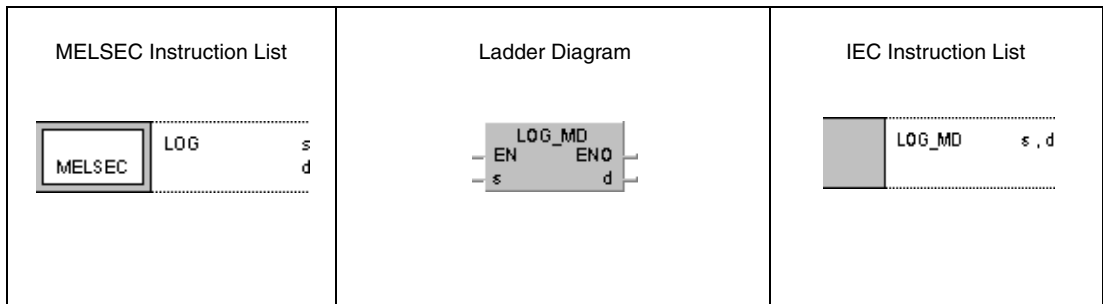
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

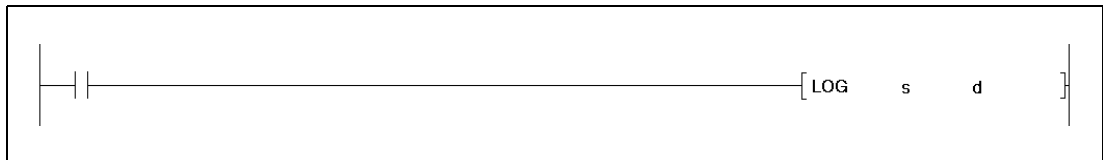
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant E			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	●	●	—	●	—	SM0	3
d	—	●	●	—	●	●	—	—	—		

GX IEC  
Developer



GX  
Developer



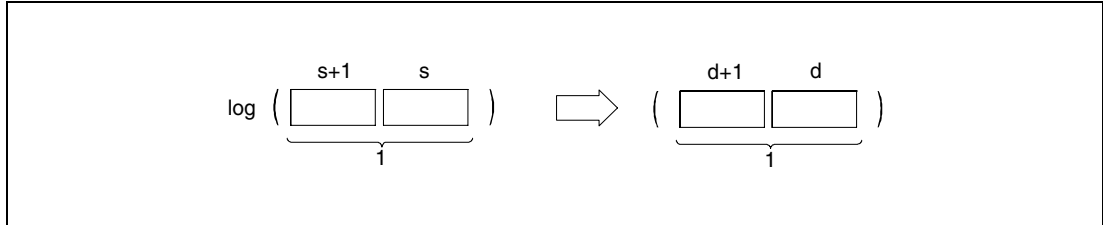
Variables

Set Data	Meaning	Data Type
s	First number of device storing the value for the LOG instruction.	Real number
d	First number of device storing the operation result.	



**Functions**    **Logarithm (ln) calculation from floating point values****LOG**    **Logarithm (ln) calculation**

The LOG instruction calculates the natural logarithm from the floating point number in s and s+1. The result is stored in d and d+1.



<sup>1</sup> Floating point value (real number)

Only positive values can be specified in s and s+1. Negative values cannot be calculated.

**Operation Errors**

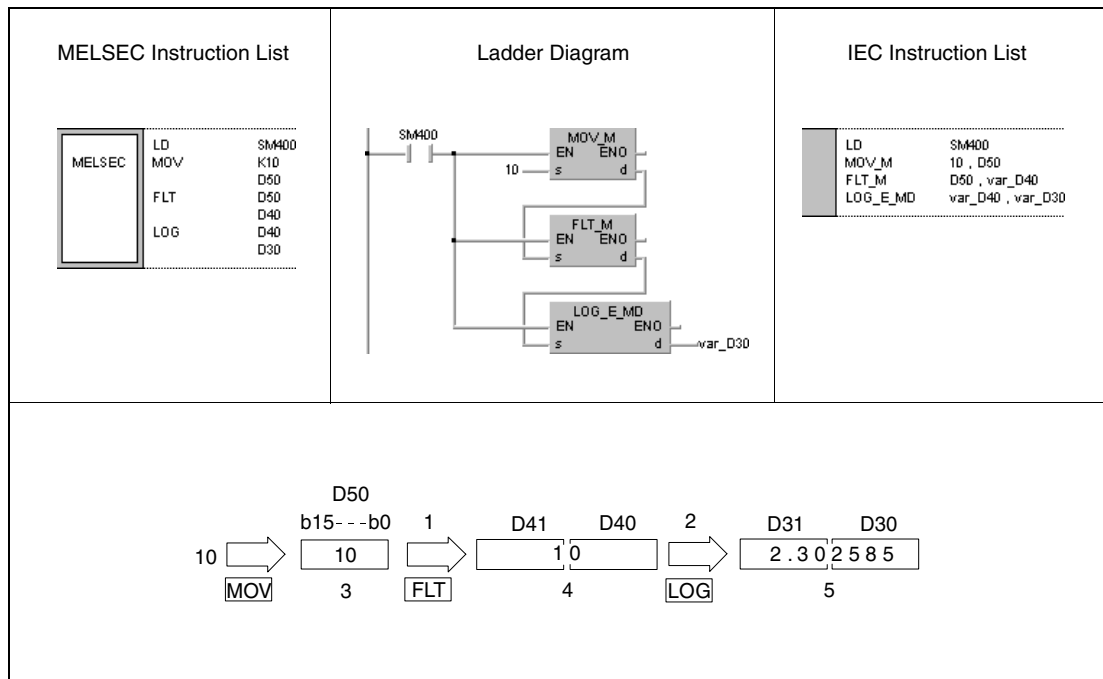
In the following cases an operation error occurs and the error flag is set:

- The value specified in s is negative (error code 4100).
- The calculation result exceeds the value range from  $2^{-127}$  to  $2^{129}$  (error code 4100).
- For Q4AR CPU only: When SM707 is OFF and the specified device (s) contains -0 (error code 4100).

**Program Example**

**LOG**

The following program calculates the natural logarithm from the value 10. The result is stored in D30 through D31.



- 1 Conversion into the floating point format
- 2 Logarithm calculation
- 3 Binary value
- 4 Floating point value (real number)
- 5 Floating point value (real number)

**NOTE** The LOG instruction calculates the natural logarithm (base e). The following formula converts the natural logarithm to normal logarithm (base 10):

$$\log_{10} X = 0.43429 \times \log_e X$$

**NOTE** This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

7.12.12 RND, RNDP, SRND, SRNDP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

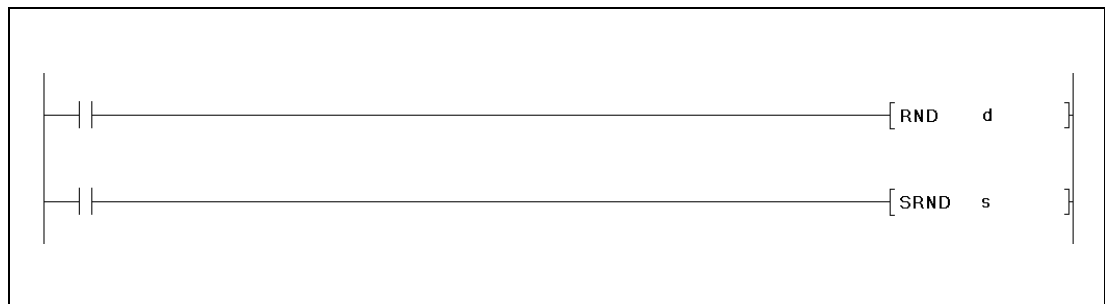
Devices  
MELSEC Q

Usable Devices										
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H, 16#	Other	Error Flag	Number of steps
Bit	Word		Bit	Word						
s							—	—	—	3

GX IEC  
Developer

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="border: 2px solid black; padding: 2px;">MELSEC</td> <td>RND</td> <td>d</td> </tr> <tr> <td></td> <td>SRND</td> <td>s</td> </tr> </table>	MELSEC	RND	d		SRND	s	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="background-color: #cccccc;"></td> <td>RND_M</td> <td>d</td> </tr> <tr> <td style="background-color: #cccccc;"></td> <td>SRND_M</td> <td>s</td> </tr> </table>		RND_M	d		SRND_M	s
MELSEC	RND	d												
	SRND	s												
	RND_M	d												
	SRND_M	s												

GX  
Developer



Variables

Set Data	Meaning	Data Type
d	First number of device storing the randomized value.	BIN 16-bit
s	Random value series or first number of device storing such data.	

**Functions Randomizing values and series update**

**RND Randomizing values**

The RND instruction generates a random value ranging from 0 to 32767 and stores it in d.

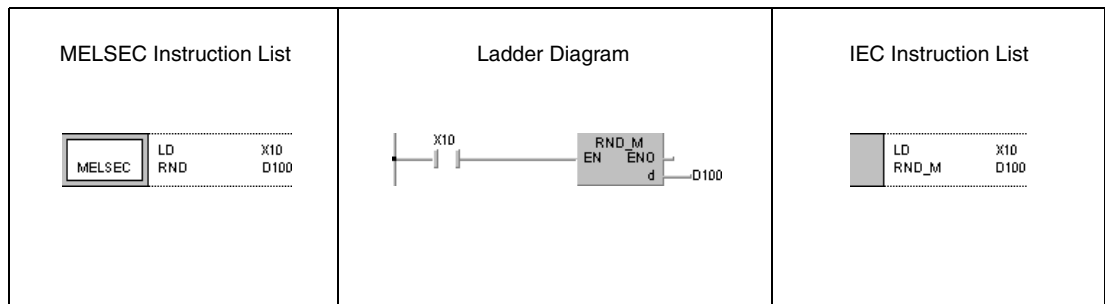
**SRND Updating series of random values**

The SRND instruction updates the series of random values stored in s.

**Program Example 1**

RND

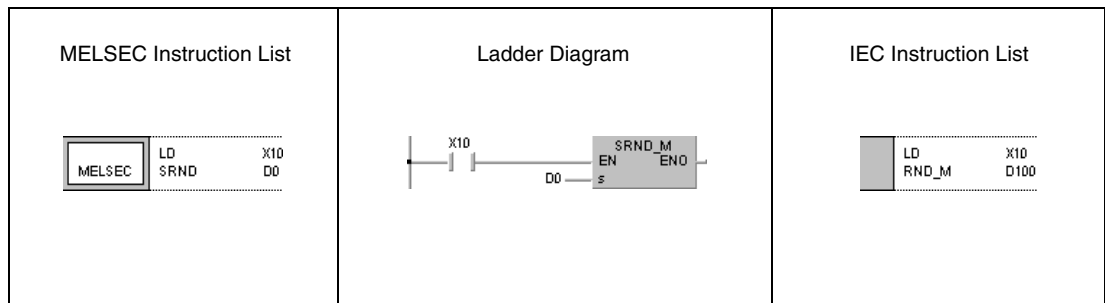
While X10 is set, the following program stores the generated random value in D100.



**Program Example 2**

SRND

While X10 is set, the following program updates the series of random values in D0.



**7.12.13 BSQR, BSQRP, BDSQR, BDSQRP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

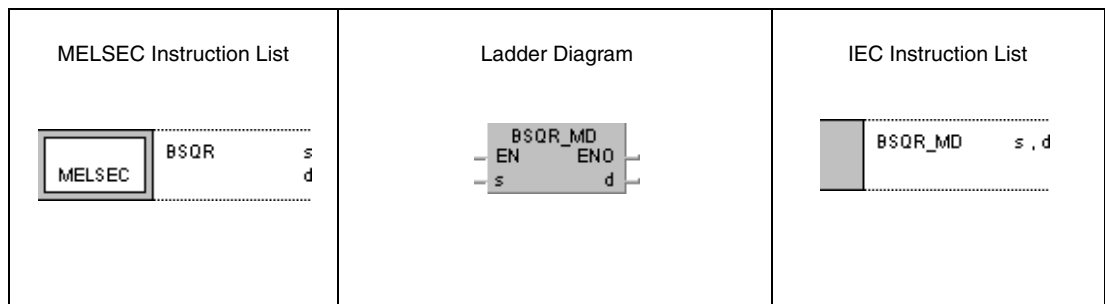
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

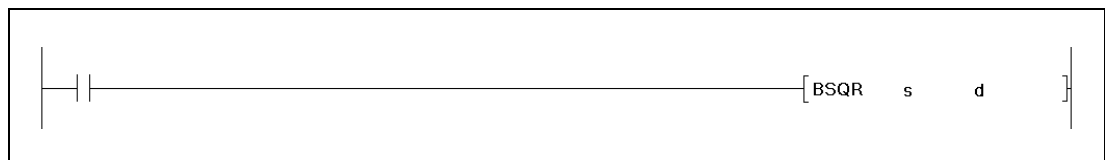
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H, (16#)			Other
	Bit	Word		Bit	Word						
s					●			●	—	SM0	3
d					●			—	—		

**GX IEC  
Developer**



**GX  
Developer**



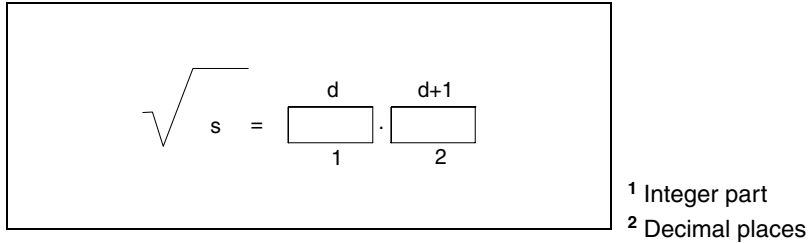
**Variables**

Set Data	Meaning	Data Type
s	Data for the square root calculation or first number of device storing such data.	BCD 4-/ 8-digit
d	First number of device storing the found square root.	BCD 4-digit

**Functions Square root calculation from 4-digit or 8-digit BCD data**

**BSQR Square root calculation from 4-digit BCD data**

The BSQR instruction calculates the square root of s and stores the result in d and d+1.



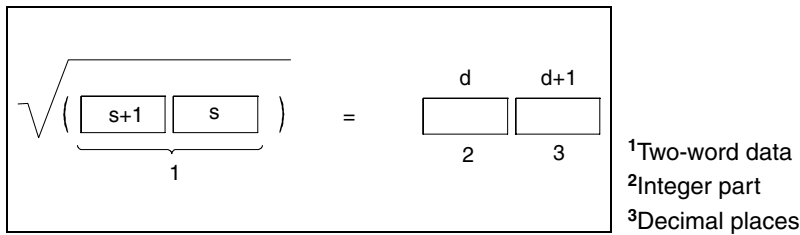
The data in s must be a BCD value with at maximum 4 digits. The value range from 0 to 9999 must not be exceeded.

The calculation result stored in d and d+1 must not exceed the value range from 0 to 9999.

The result is calculated with a 5-digit accuracy and rounded to a 4-digit value.

**BDSQR Square root calculation from 8-digit BCD data**

The BDSQR instruction calculates the square root of s and s+1 and stores the result in d and d+1.



The data in s and s+1 must be a BCD value with at maximum 8 digits. The value range from 0 to 99999999 must not be exceeded.

The calculation result stored in d and d+1 must not exceed the value range from 0 to 9999.

The result is calculated with a 5-digit accuracy and rounded up to a 4-digit value.

**Operation Errors**

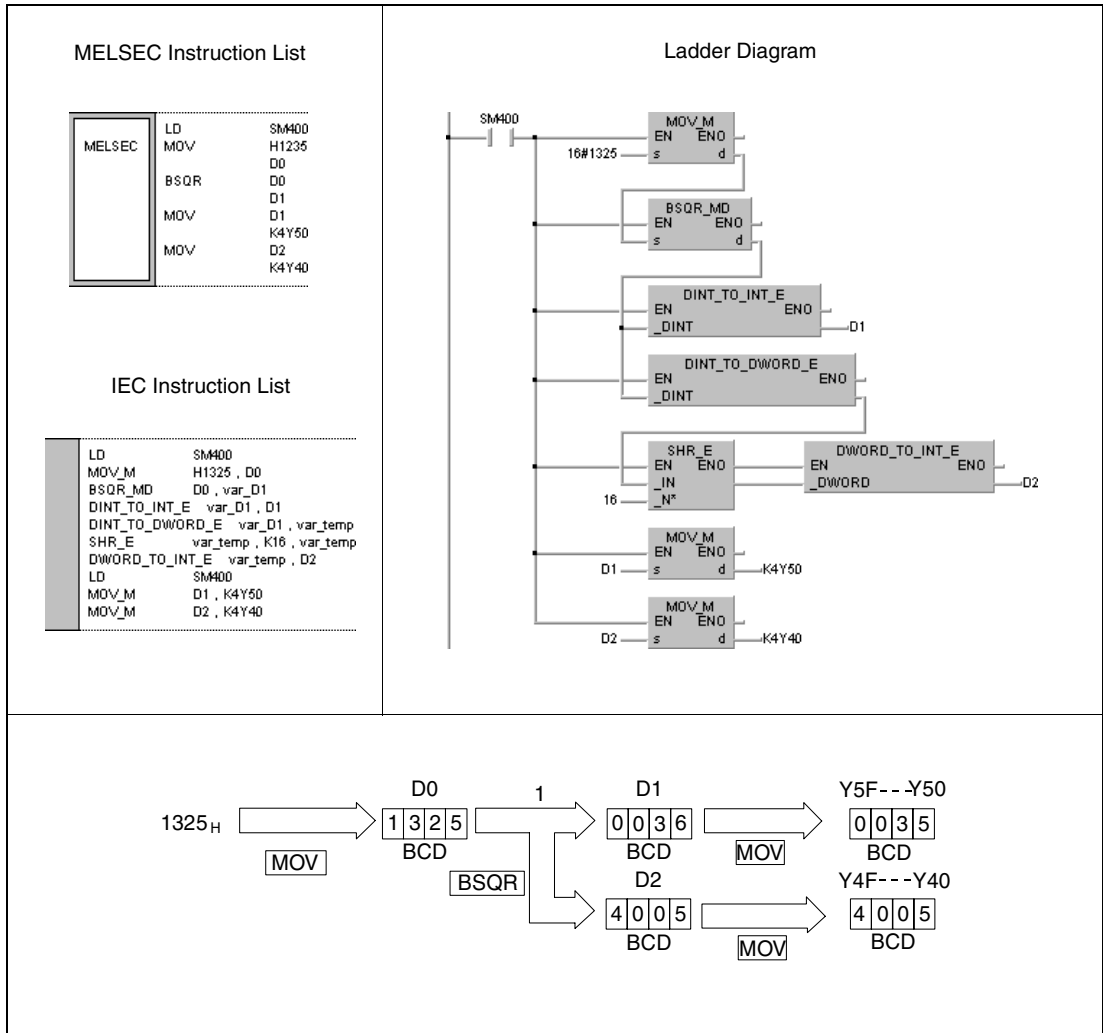
In the following cases an operation error occurs and the error flag is set:

- The data stored in s (s+1) is no BCD data (error code 4100).

**Program Example 1**

**BSQR**

While SM400 is set, the following program calculates the square root of the BCD value 1325 and outputs the integer part of the result as 4-digit BCD value at Y50 through Y5F. The decimal places are output as 4-digit BCD value at Y40 through Y4F.

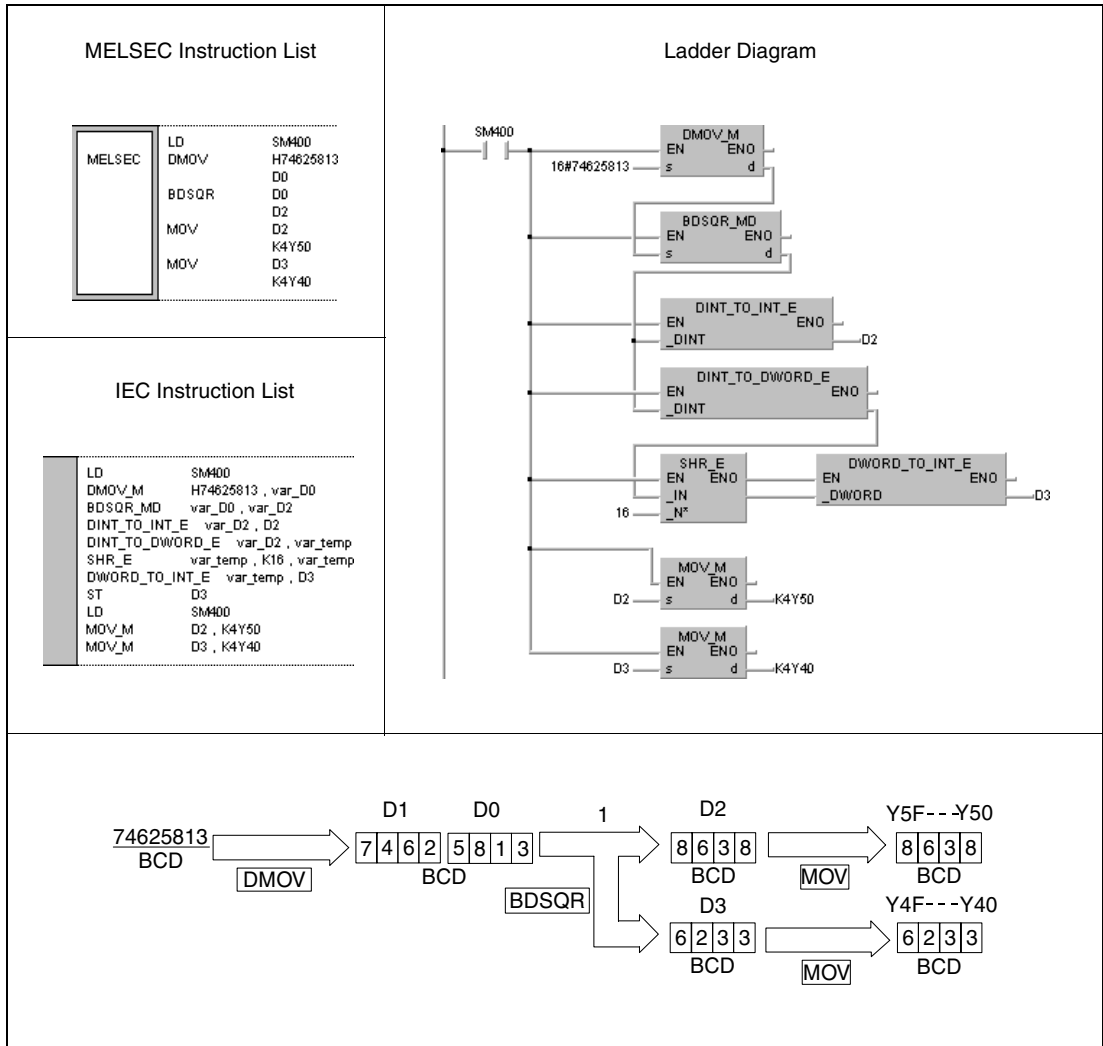


<sup>1</sup> Square root calculation

**Program** BDSQR

**Example 2**

While SM400 is set, the following program calculates the square root of the BCD value 74625813 and outputs the integer part of the result as 4-digit BCD value at Y5F through Y5D. The decimal places are output as 4-digit BCD value at Y4F through Y4D.



<sup>1</sup> Square root calculation

**NOTE**

*These program examples will not run without variable definition in the header of the program organization unit (POU). They would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*



7.12.14 BSIN, BSINP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

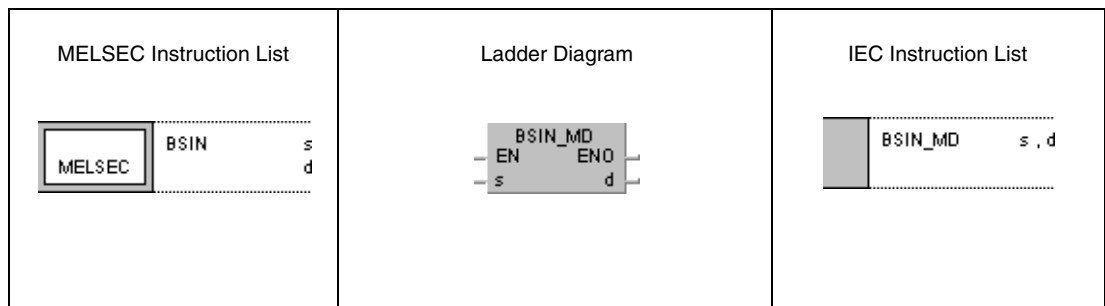
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

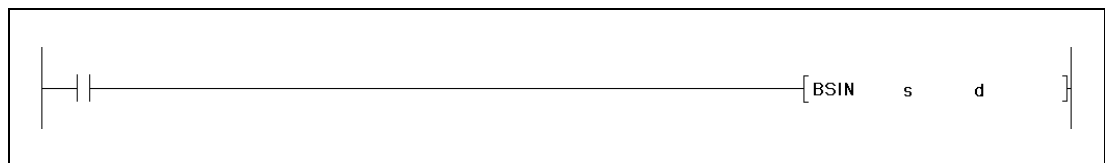
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H, (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	—	●	●	—	—	—	—	—	—		

GX IEC  
Developer



GX  
Developer



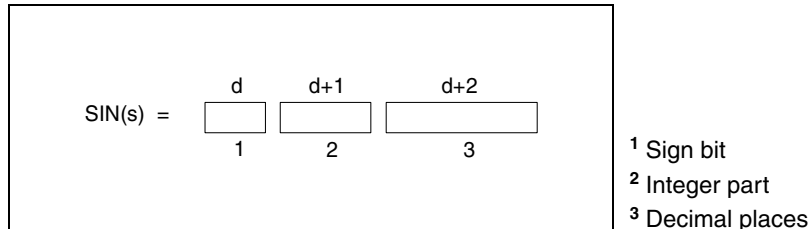
Variables

Set Data	Meaning	Data Type
s	First number of device storing angle data for the BSIN instruction (sine).	4-digit BCD value
d	First number of device storing the calculation result.	

**Functions**      **Sine calculation from BCD data**

**BSIN**      **Sine calculation**

The BSIN instruction calculates the sine value from the angle data in s. The sign character of the result is stored in d. The value of the result is stored in d+1 and d+2.



The value s must be a BCD value ranging from 0° to 360°.

The sign of the result in d is 0 for a positive value and 1 for a negative value.

The result in d+1 and d+2 may range from -1.000 to 1.000 in BCD format.

The calculation result will be rounded from the 5th digit on.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

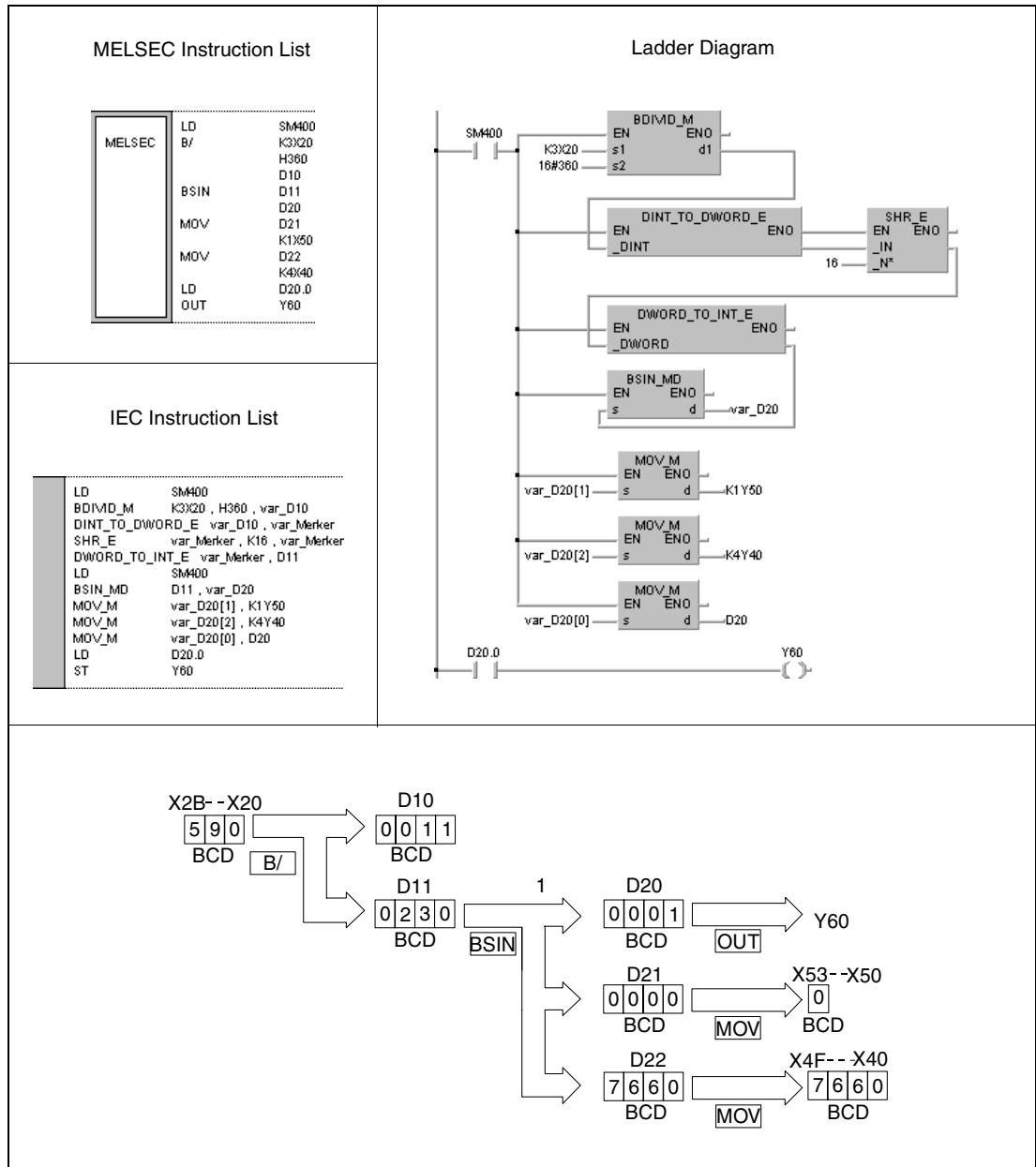
- The data specified in s is no BCD data (error code 4100).
- The data specified in s exceeds the value range from 0° to 360° (error code 4100).

**Program Example**

**BSIN**

While SM400 is set, the following program calculates the sine value of the 3-digit BCD value at X20 through X2B. If the value at X20 through X2B exceeds 360 (degrees), it will be corrected to suit the required value range of 0° to 360°.

The sign is output at Y60. The integer part is output at Y50 through Y53 as 1-digit BCD value. The decimal places are output at Y40 through Y4F as 4-digit BCD value.



<sup>1</sup> Sine calculation

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.15 BCOS, BCOSP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>


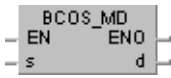
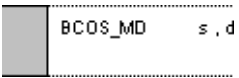
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

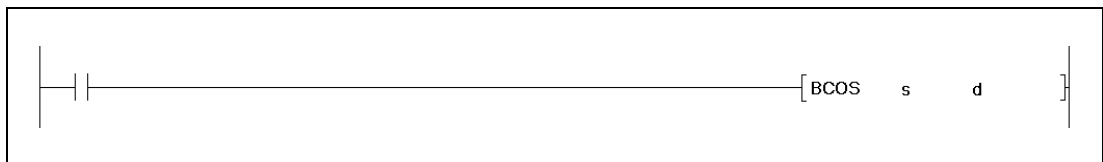
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H, (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	—	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	--

GX Developer

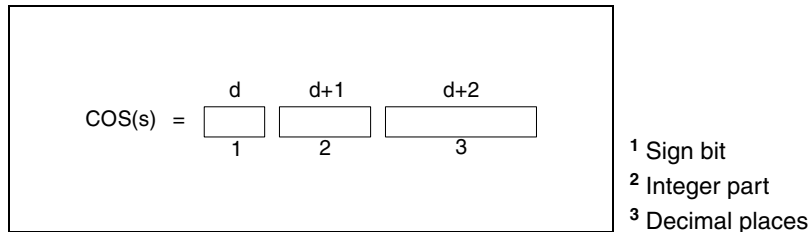


Variables

Set Data	Meaning	Data Type
s	First number of device storing angle data for the BCOS instruction (cosine).	4-digit BCD value
d	First number of device storing the calculation result.	

**Functions**    **Cosine calculation from BCD data****BCOS**    **Cosine calculation**

The BCOS instruction calculates the cosine value from the angle data in s. The sign character of the result is stored in d. The value of the result is stored in d+1 and d+2.



The value s must be a BCD value ranging from 0° to 360°.

The sign of the result in d is 0 for a positive value and 1 for a negative value.

The result in d+1 and d+2 may range from -1.000 to 1.000 in BCD format.

The calculation result will be rounded from the 5th digit on.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

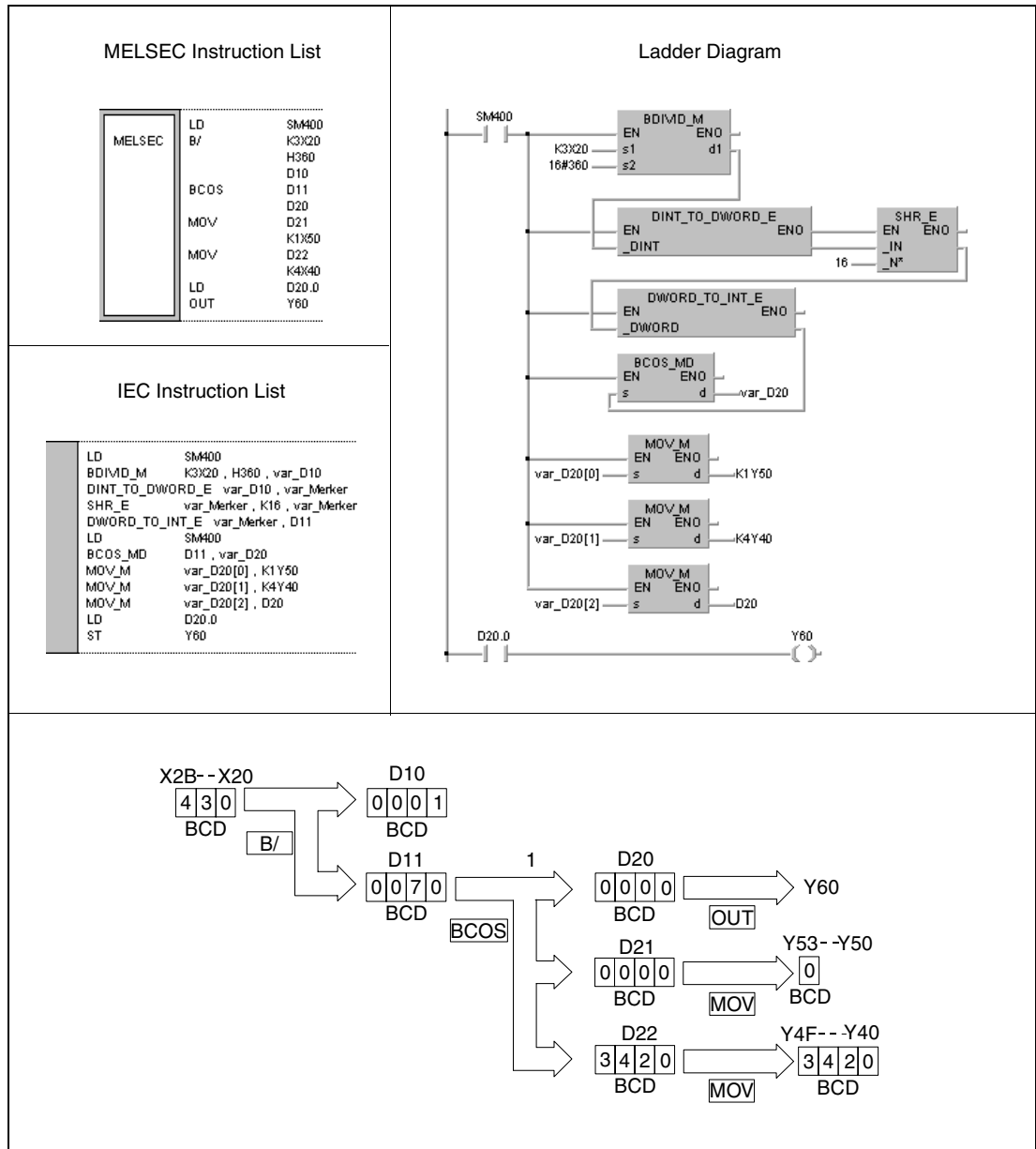
- The data specified in s is no BCD data (error code 4100).
- The data specified in s exceeds the value range from 0° to 360° (error code 4100).

**Program Example**

**BCOS**

While SM400 is set, the following program calculates the cosine value of the 3-digit BCD value at X20 through X2B. If the value at X20 through X2B exceeds 360 (degrees), it will be corrected to suit the required value range of 0° to 360°.

The sign is output at Y60. The integer part is output at Y50 through Y53 as 1-digit BCD value. The decimal places are output at Y40 through Y4F as 4-digit BCD value.



<sup>1</sup> Cosine calculation

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.16 BTAN, BTANP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

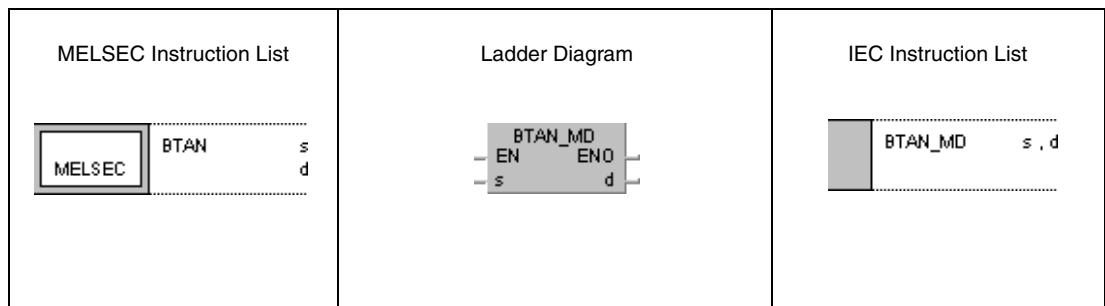
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

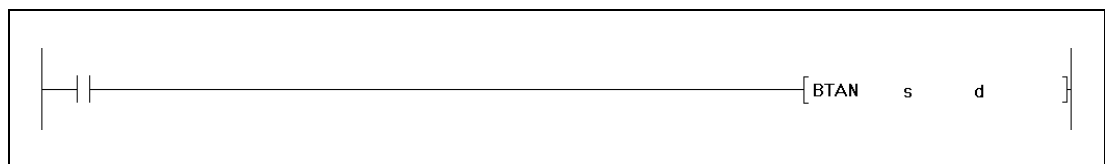
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H, (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	●	—	SM0	3
d	—	●	●	—	—	—	—	—	—		

GX IEC  
Developer



GX  
Developer

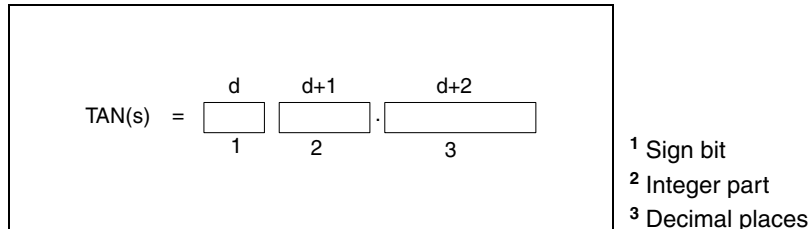


Variables

Set Data	Meaning	Data Type
s	First number of device storing angle data for the BTAN instruction (tangent).	4-digit BCD value
d	First number of device storing the calculation result.	

**Functions**    **Tangent calculation from BCD data****BTAN**    **Tangent calculation**

The BTAN instruction calculates the tangent value from the angle data in s. The sign character of the result is stored in d. The value of the result is stored in d+1 and d+2.



The value s must be a BCD value ranging from 0° to 360°.

The sign of the result in d is 0 for a positive value and 1 for a negative value.

The result in d+1 and d+2 may range from -57.2900 to 57.2900 in BCD format.

The calculation result will be rounded from the 5th digit on.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

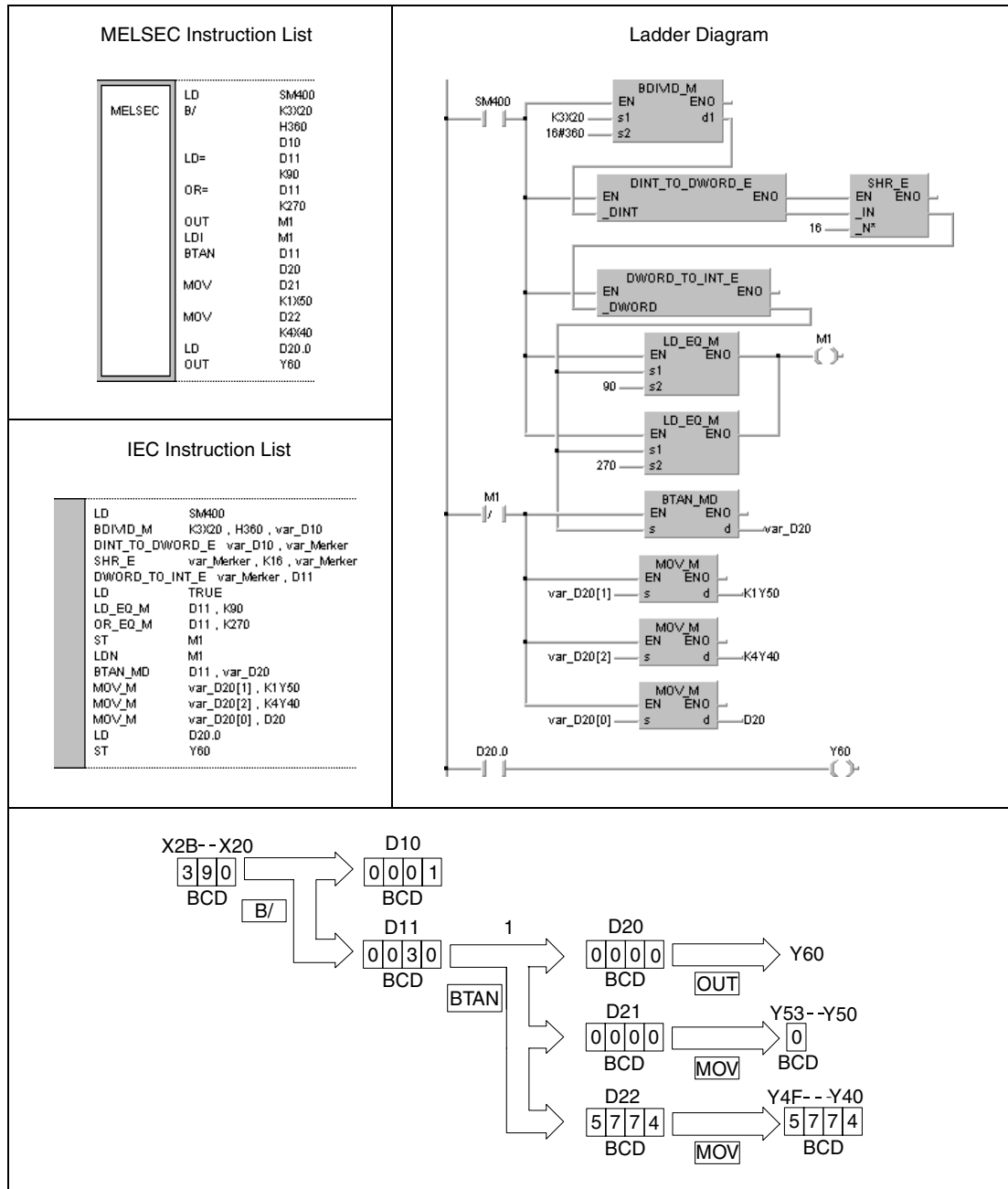
- The data specified in s is no BCD data (error code 4100).
- The data specified in s exceeds the value range from 0° to 360° (error code 4100).
- The value in s is 90° or 270° (error code 4100).



**Program Example** BTAN

While SM400 is set, the following program calculates the tangent value of the 3-digit BCD value at X20 through X2B. If the value at X20 through X2B exceeds 360 (degrees), it will be corrected to suit the required value range of 0° to 360°.

The sign is output at Y60. The integer part is output at Y50 through Y53 as 1-digit BCD value. The decimal places are output at Y40 through Y4F as 4-digit BCD value.



<sup>1</sup> Tangent calculation

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.17 BASIN, BASINP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

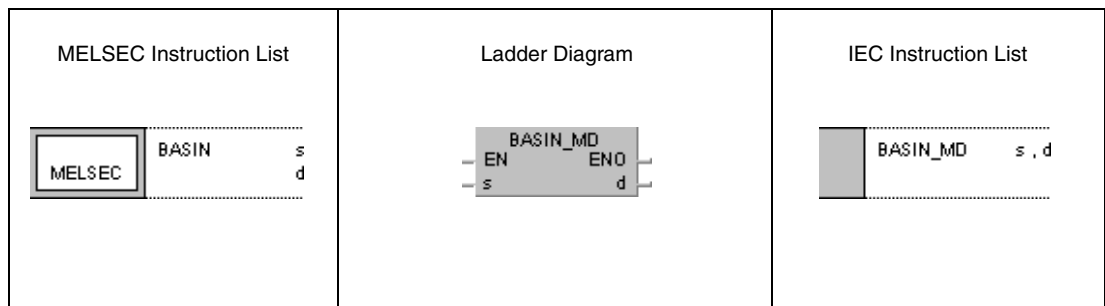
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

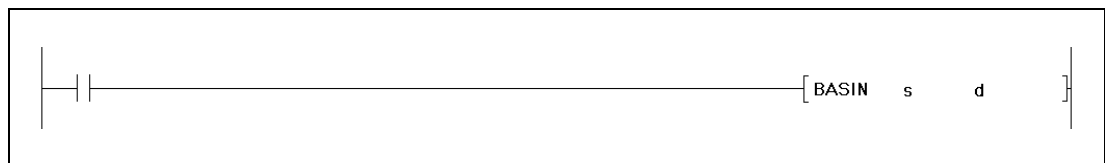
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H, (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	3
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer



GX Developer

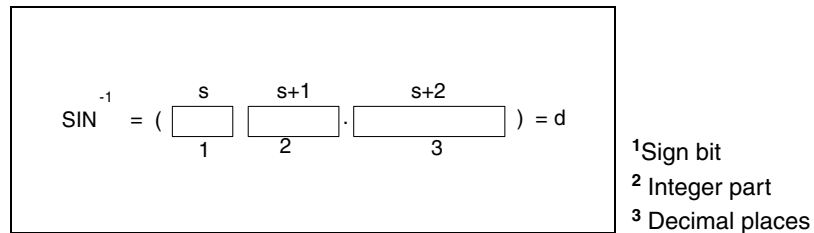


Variables

Set Data	Meaning	Data Type
s	First number of device storing the sine value for the BASIN instruction (arcus sine).	4-digit BCD value
d	First number of device storing the calculation result.	

**Functions**     **Arcus sine calculation from BCD data****BASIN**     **Arcus sine calculation**

The BASIN instruction calculates the angle data from the sine value in s, s+1, and s+2. The result is stored in d.



The sign of the result in s is 0 for a positive value and 1 for a negative value.

The integer part prior to the decimal point and the decimal places must be BCD values ranging from 0 to 10000.

The value or the result in d must be a BCD value ranging from 0° to 90° or from 270° to 360°.

The calculation result will be rounded from the 5th digit on.

**Operation Errors**

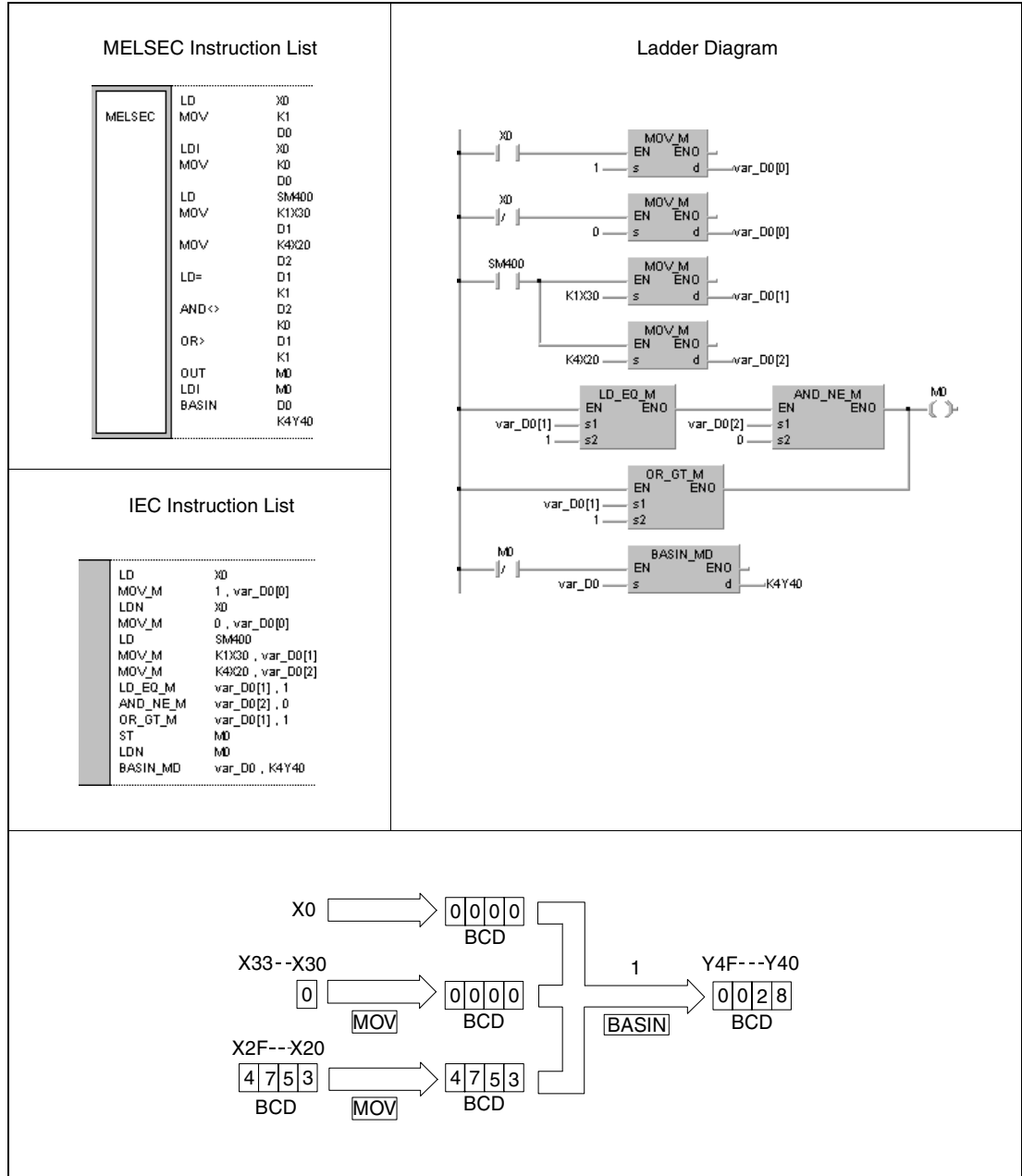
In the following cases an operation error occurs and the error flag is set:

- The data specified in s through s+2 is no BCD data (error code 4100).
- The data specified in s through s+2 exceeds the value range from -1.0000 to 1.0000 (error code 4100).

**Program Example**

**BASIN**

While SM400 is set, the following program calculates the arcus sine value from the sign bit at X0 (1 = positive, 0 = negative), the 1-digit BCD integer part at X30 through X33, and the decimal places of the 4-digit BCD value at X20 through X2F. The resulting angle value is output in 4-digit BCD format at Y40 through Y4F.



<sup>1</sup> Arcus sine calculation

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

**7.12.18 BACOS, BACOSP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

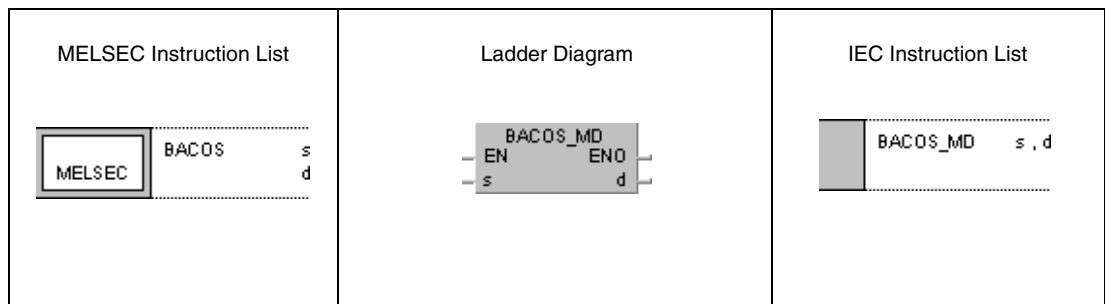
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

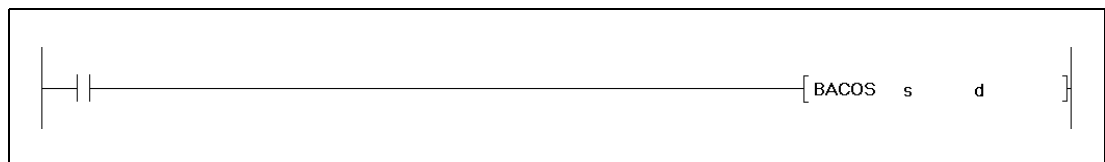
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H, (16#)			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	SM0	3	
d	●	●	●	●	●	●	—	—			

**GX IEC  
Developer**



**GX  
Developer**

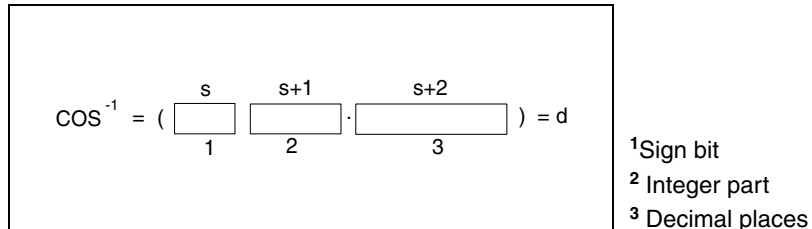


**Variables**

Set Data	Meaning	Data Type
s	First number of device storing the sine value for the BACOS instruction (arcus cosinus).	4-digit BCD value
d	First number of device storing the calculation result.	

**Functions**     **Arcus cosine calculation from BCD data****BACOS Arcus cosine calculation**

The BACOS instruction calculates the angle data from the cosine value in s, s+1, and s+2. The result is stored in d.



The sign of the result in s is 0 for a positive value and 1 for a negative value.

The integer part prior to the decimal point and the decimal places must be BCD values ranging from 0 to 10000.

The value or the result in d must be a BCD value ranging from 0° to 180.

The calculation result will be rounded from the 5th digit on.

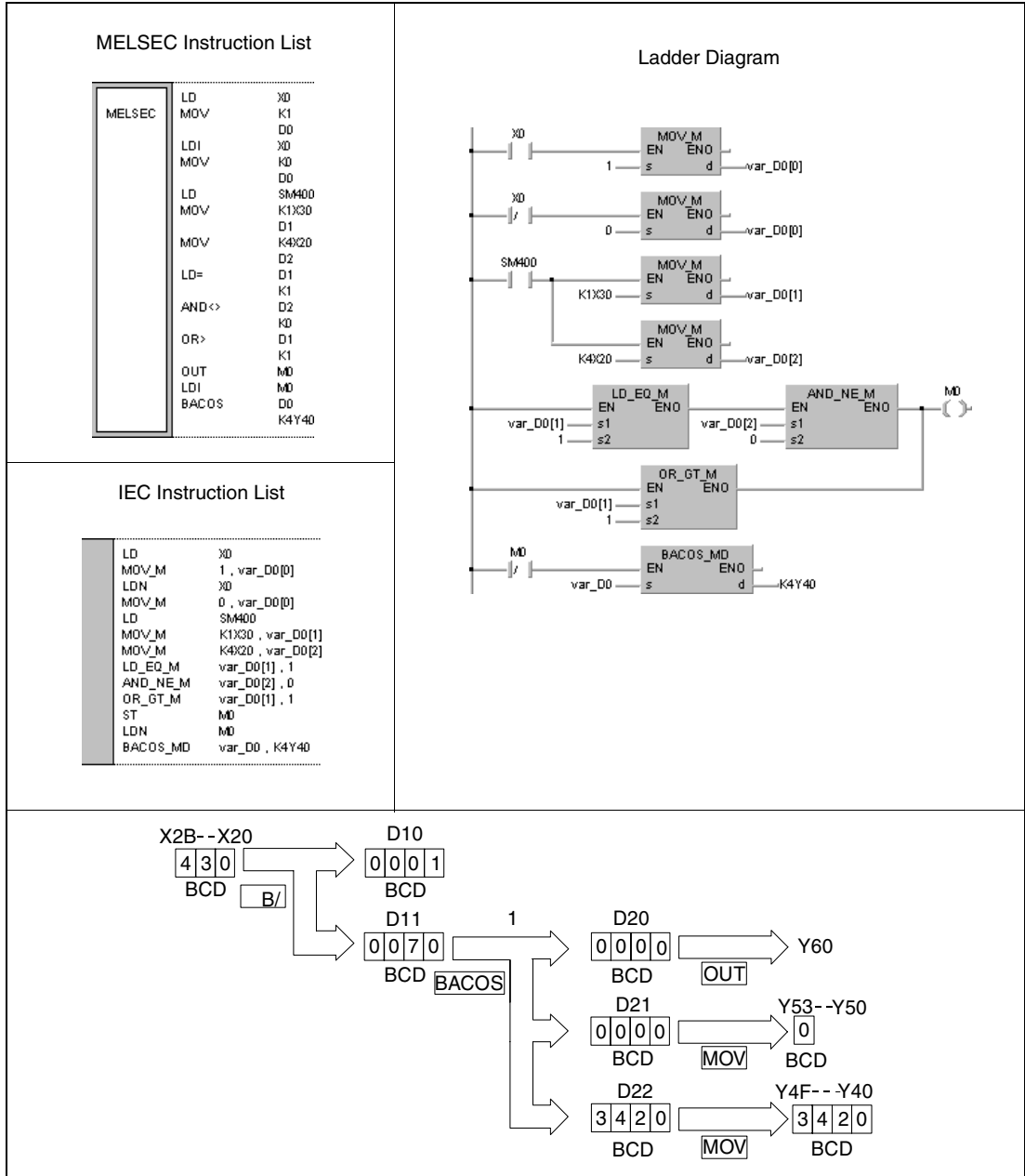
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The data specified in s through s+2 is no BCD data (error code 4100).
- The data specified in s through s+2 exceeds the value range from -1000 to 1000 (error code 4100).

**Program Example** BACOS

While SM400 is set, the following program calculates the arcus cosine value from the sign bit at X0 (1 = positive, 0 = negative), the 1-digit BCD integer part at X30 through X33, and the decimal places of the 4-digit BCD value at X20 through X2F. The resulting angle value is output in 4-digit BCD format at Y40 through Y4F.



<sup>1</sup> Arcus cosine calculation

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

7.12.19 BATAN, BATANP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

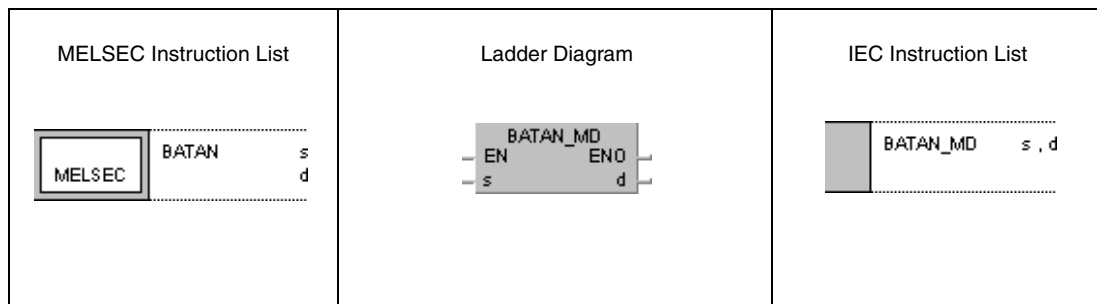
<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

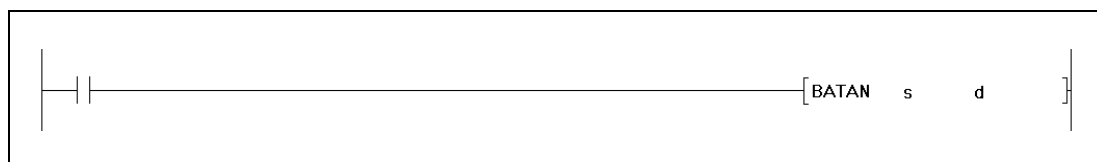
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	3
d	●	●	●	●	●	●	—	—			

GX IEC Developer



GX Developer



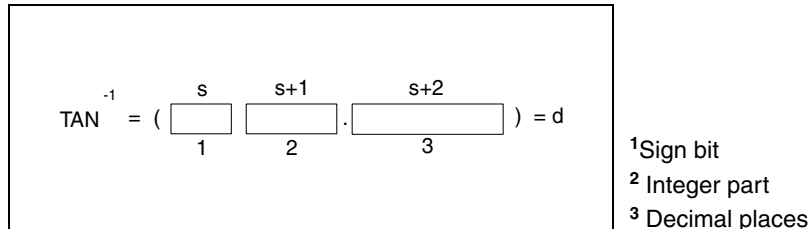
Variables

Set Data	Meaning	Data Type
s	First number of device storing the tangent value for the BATAN instruction (arcus tangent).	4-digit BCD value
d	First number of device storing the calculation result.	



**Functions**     **Arcus tangent calculation from BCD data****BATAN**   **Arcus tangent calculation**

The BATAN calculates the angle data from the tangent value in s, s+1, and s+2. The result is stored in d.



The sign bit of the result in s is 0 for a positive value and 1 for a negative value.

The integer part prior to the decimal point and the decimal places must be BCD values ranging from 0 to 99999999.

The value of the result in d must be a BCD value ranging from 0° to 90° or 270° or from 270° and 360°.

The calculation result will be rounded from the 5th digit on.

**Operation Errors**

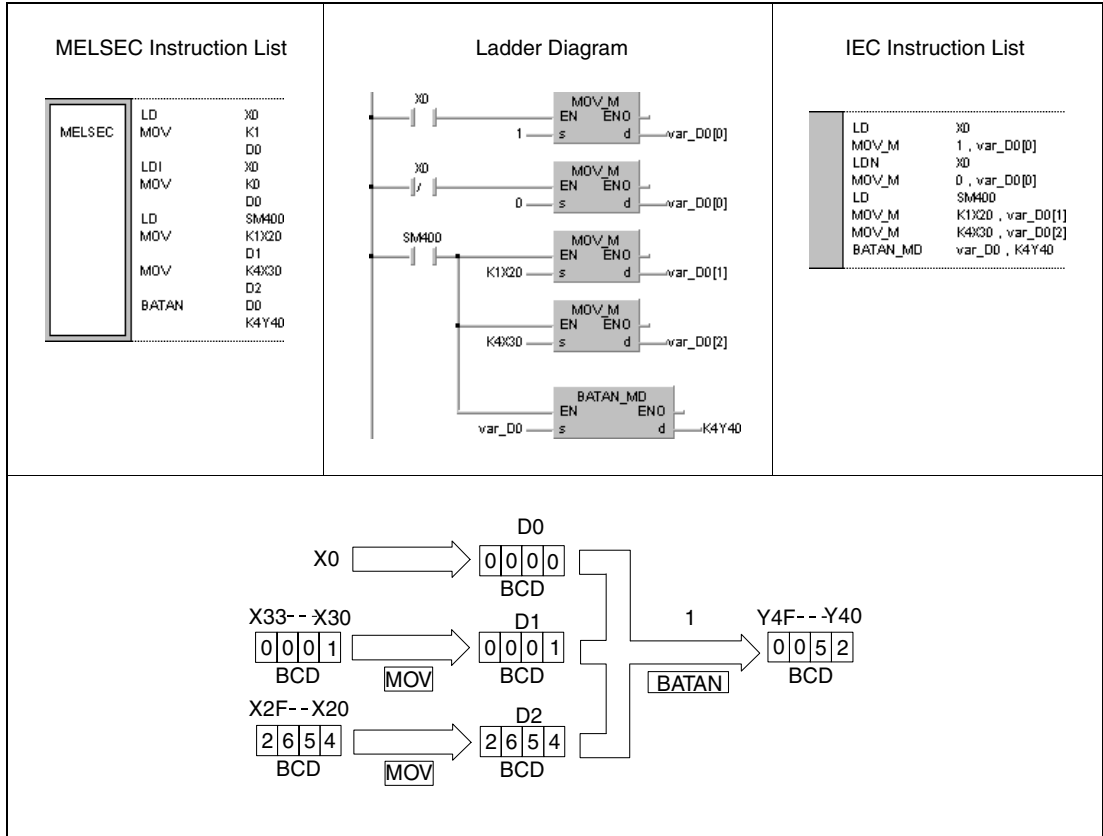
In the following cases an operation error occurs and the error flag is set:

- The data specified in s through s+2 is no BCD data (error code 4100).

**Program Example**

**BATAN**

While SM400 is set, the following program calculates the arcus tangent value from the sign bit at X0 (1 = positive, 0 = negative), the 1-digit BCD integer part at X20 through X23, and the decimal places of the 4-digit BCD value at X30 through X3F. The resulting angle value is output in 4-digit BCD format at Y40 through Y4F.



<sup>1</sup> Arcus tangent calculation

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 7.13 Data control instructions

The data control instructions include input and output devices. The 16-bit and 32-bit data of the input devices are output to the output devices via parameters controlling the upper and lower limits, the dead band, and the zone.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Upper and lower limit controls for BIN 16-/32-bit data	LIMIT	LIMIT_MD
	LIMITP	LIMIT_P_MD
	DLIMIT	DLIMIT_MD
	DLIMITP	DLIMIT_P_MD
Dead band controls for BIN 16-/32-bit data	BAND	BAND_MD
	BANDP	BAND_P_MD
	DBAND	DBAND_MD
	DBANDP	DBAND_P_MD
Zone control for BIN 16-/32-bit data	ZONE	ZONE_MD
	ZONEP	ZONE_P_MD
	DZONE	DZONE_MD
	DZONEP	DZONE_P_MD

**NOTE**

*Within the IEC editors please use the IEC instructions.*

7.13.1 LIMIT, LIMITP, DLIMIT, DLIMITP

CPU

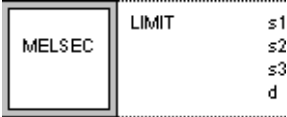
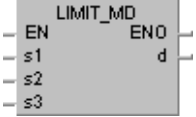
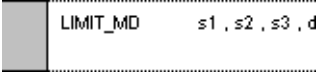
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

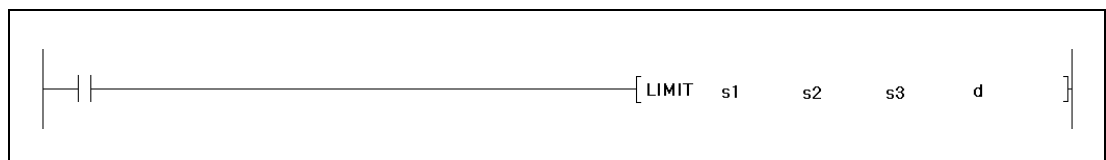
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	SM0	5
s2	●	●	●	●	●	●	●	●	—		
s3	●	●	●	●	●	●	●	●	—		
d	●	●	●	●	●	●	●	—	—		

GX IEC  
Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	--

GX  
Developer



Variables

Set Data	Meaning	Data Type
s1	Lower limit value (minimum output threshold value).	BIN 16-bit
s2	Upper limit value (maximum output threshold value).	
s3	Input value to be limited.	
d	First number of device storing limited output value.	

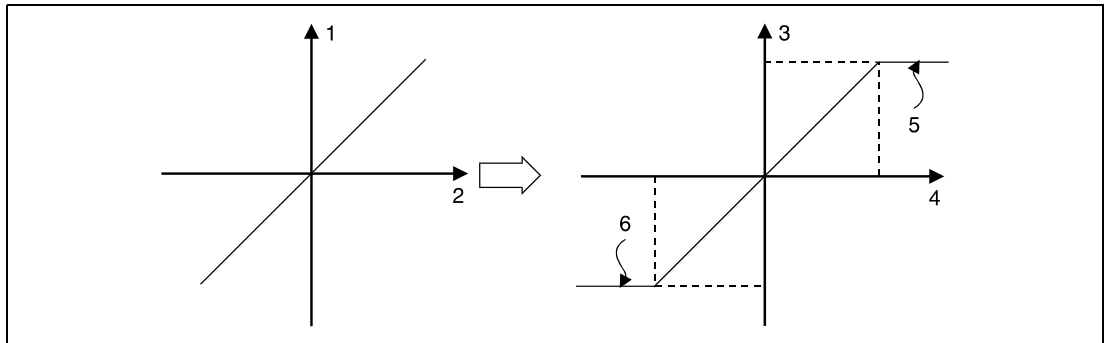
**Functions    Limitation of output values for BIN 16-bit and BIN 32-bit data****LIMIT    Limitation instruction for BIN 16-bit data**

The LIMIT instruction controls whether data in the device specified by s3 ranges within the lower limits specified by s1 and the upper limits specified by s2. Depending on the control operation result the values are stored as follows in the device specified by d:

If the data value in s3 is less than the lower limit value in s1, the lower limit value is stored in d.

If the data value in s3 is greater than the upper limit value in s2, the upper limit value is stored in d.

If the data value in s3 ranges within the lower and the upper limit value, the data value is stored in d.



- <sup>1</sup> Output value
- <sup>2</sup> Input value
- <sup>3</sup> Output value (d)
- <sup>4</sup> Input value (s3)
- <sup>5</sup> Upper limit value (s2)
- <sup>6</sup> Lower limit value (s1)

The values specified by s1, s2, and s3 have to range within -32768 and 32767.

If only the upper limit value is to be checked, the lower limit value in s1 has to be set to -32768.

If only the lower limit value is to be checked, the upper limit value in s2 has to be set to 32767.

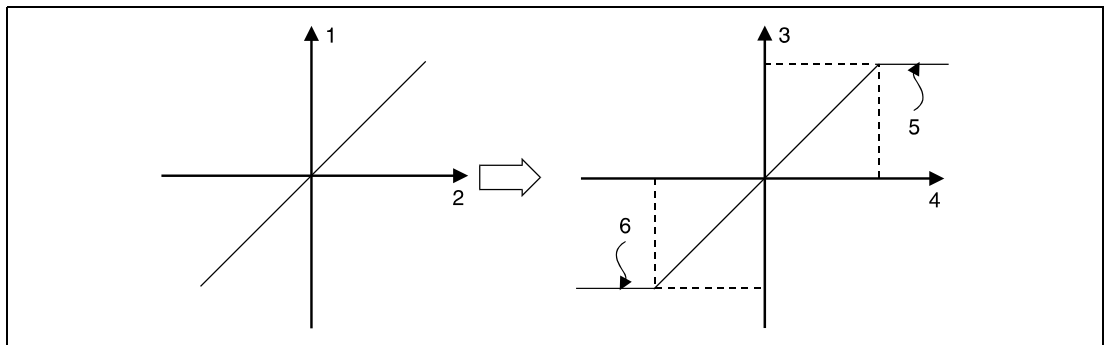
**DLIMIT Limitation instruction for BIN 32-bit data**

The DLIMIT instruction controls whether data in the devices specified by s3 and (s3)+1 range within the lower limits specified by s1 and (s1)+1 and the upper limits specified by s2 and (s2)+1. Depending on the control operation result the values are stored as follows in the device specified by d:

If the data value in s3 and (s3)+1 is less than the lower limit value in s1 and (s1)+1, the lower limit value is stored in d and d+1.

If the data value in s3 and (s3)+1 is greater than the upper limit value in s2 and (s2)+1, the upper limit value is stored in d and d+1.

If the data value in s3 and (s3)+1 ranges within the lower and the upper limit value, the data value is stored in d and d+1.



<sup>1</sup> Output value

<sup>2</sup> Input value

<sup>3</sup> Output value (d+1, d)

<sup>4</sup> Input value ((s3)+1, s3)

<sup>5</sup> Upper limit value ((s2)+1, s2)

<sup>6</sup> Lower limit value ((s1)+1, s1)

The values specified by s1 and (s1)+1, s2 and (s2)+1, and s3 and (s3)+1 have to range within -2147483648 and 2147483647.

If only the upper limit value is to be checked, the lower limit value in s1 and (s1)+1 has to be set to -2147483648.

If only the lower limit value is to be checked, the upper limit value in s2 and (s2)+1 has to be set to 2147483647.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The value in s1 ((s1)+1) is greater than that in s2 ((s2)+1) (error code 4100).

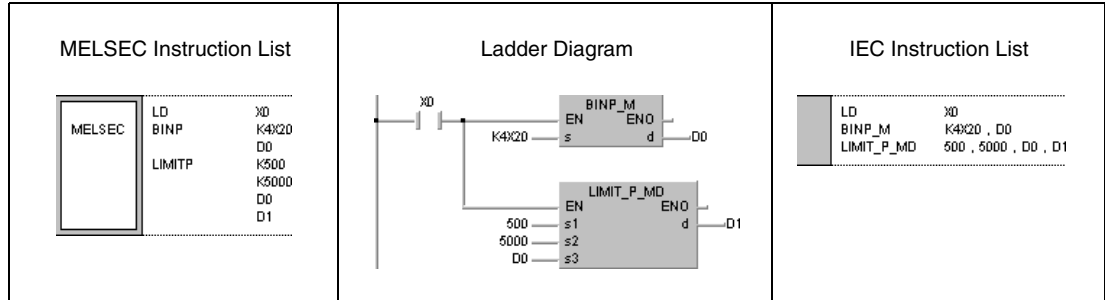
**Program Example 1**      **LIMITP**

With leading edge from X0, the following program controls whether BCD data at X20 through X2F ranges between the lower limit of 500 and the upper limit of 5000. The result of the control operation is stored in D1.

If the value in D0 is greater than 5000, the value 5000 is stored in D1.

If the value in D0 is less than 500, the value 500 is stored in D1.

If the value ranges within 500 and 5000, the data value is stored in D1.



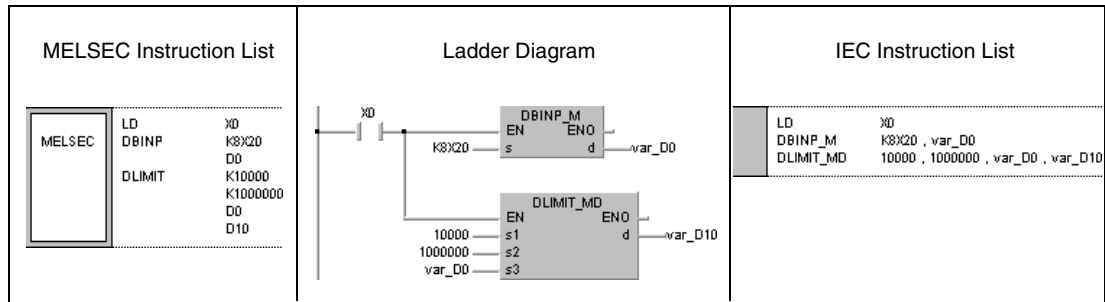
**Program Example 2**      **DLIMIT**

With leading edge from X0, the following program controls whether BCD data at X20 through X3F ranges within the lower limit of 10000 and the upper limit of 1000000. The result of the control operation is stored in D10 and D11.

If the value in D0 and D1 is greater than 1000000, the value 1000000 is stored in D10 and D11.

If the value in D0 and D1 is less than 10000, the value 10000 is stored in D10 and D11.

If the value ranges within 10000 and 1000000, the data value is stored in D10 and D11.



7.13.2 BAND, BANDP, DBAND, DBANDP

CPU

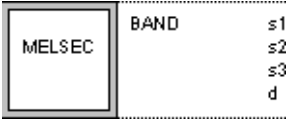
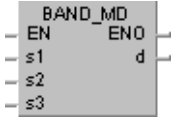
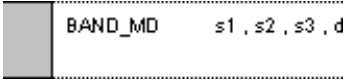
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

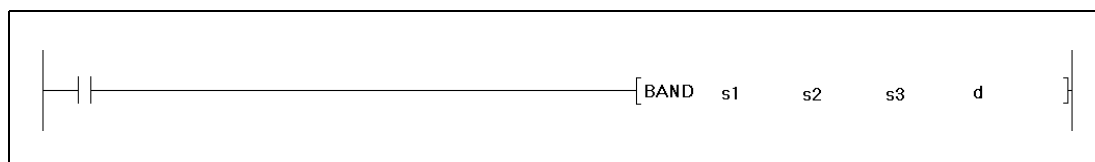
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	SM0	5
s2	●	●	●	●	●	●	●	●	—		
s3	●	●	●	●	●	●	●	●	—		
d	●	●	●	●	●	●	●	—	—		

GX IEC Developer

MELSEC Instruction List	Ladder Diagram	IEC Instruction List
		

GX Developer



Variables

Set Data	Meaning	Data Type
s1	Lower limit value of dead band (output value = 0).	BIN 16-bit
s2	Upper limit value of dead band (output value = 0).	
s3	Input value to be controlled via dead band control.	
d	First number of device storing subtraction result of input value minus limit value.	



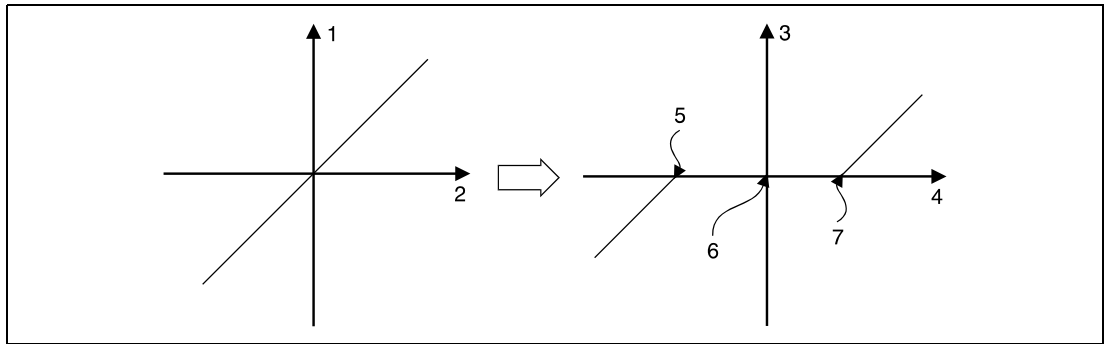
**Functions BIN 16-bit and 32-bit dead band control****BAND Dead band control of BIN 16-bit data**

The BAND instruction subtracts a lower (negative) and an upper (positive) limit value from a BIN 16-bit value in a device specified by s3. The lower limit value is specified by s1; the upper limit value is specified by s2. The result is stored depending on the input value in the device specified by d as follows:

If the data value in s3 is less than the lower limit value in s1, the result of the subtraction s3-s1 is stored in the device specified by d.

If the data value in s3 is greater than the upper limit value in s2, the result of the subtraction s3-s2 is stored in the device specified by d.

If the data value in s3 ranges within the limit values, the value 0 is stored in the device specified by d.



- 1 Output value
- 2 Input value
- 3 Output value (d)
- 4 Input value (s3)
- 5 Lower (negative) limit value (s1)
- 6 Output value = 0
- 7 Upper (positive) limit value (s2)

The values in s1, s2, and s3 have to range within -32768 and 32767.

If the subtraction result leaves the relevant device range of -32768 and 32767 the output value is controlled as follows:

If the value -32768 is fallen below, the remaining subtraction is proceeded beginning from 32767. For example, if s3 stores the value -32768 and the value 10 in s1 is subtracted, the result is

$$-32768 - 10 = 8000_{\text{H}} - A_{\text{H}} = 7\text{FF}6_{\text{H}} = 32758.$$

If the value 32760 is exceeded, the remaining subtraction is proceeded beginning from -32768.

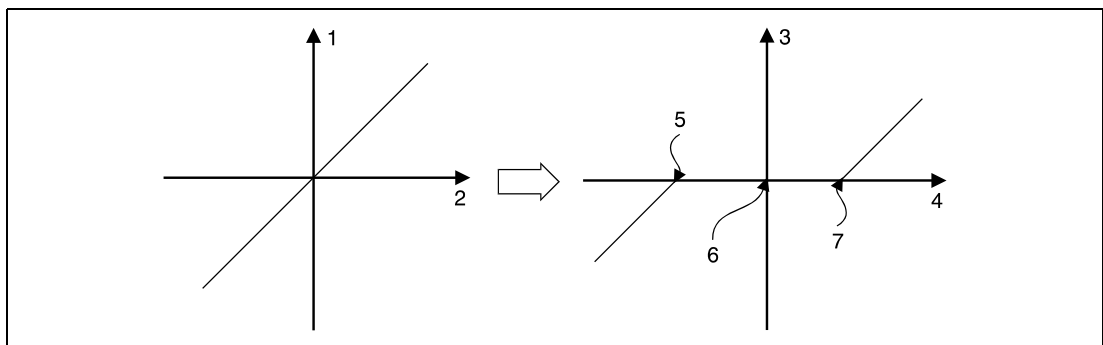
**DBAND Dead band control of BIN 32-bit data**

The DBAND instruction subtracts a lower (negative) and an upper (positive) limit value from a BIN 32-bit value in a device specified by s3 and (s3)+1. The lower limit value is specified by s1 and (s1)+1; the upper limit value is specified by s2 and (s2)+1. The result is stored depending on the input value in the device specified by d and d+1 as follows:

If the data value in s3 and (s3)+1 is less than the lower limit value in s1 and (s1)+1, the result of the subtraction s3, (s3)+1 - s1, (s1)+1 is stored in the device specified by d and d+1.

If the data value in s3 and (s3)+1 is greater than the upper limit value in s2 and (s2)+1, the result of the subtraction s3, (s3)+1 - s2, (s2)+1 is stored in the device specified by d and d+1.

If the data value in s3 and (s3)+1 ranges within the limit values, the value 0 is stored in the device specified by d and d+1.



- 1 Output value
- 2 Input value
- 3 Output value (d+1, d)
- 4 Input value ((s3)+1, s3)
- 5 Lower (negative) limit value ((s1)+1, s1)
- 6 Output value = 0
- 7 Upper (positive) limit value ((s2)+1, s2)

The values in s1 and (s1)+1, s2 and (s2)+1, and s3 and (s3)+1 have to range within -2147483648 and 2147483647.

If the subtraction result leaves the relevant device range of -2147483648 and 2147483647 the output value is controlled as follows:

If the value -2147483648 is fallen below, the remaining subtraction is proceeded beginning from 2147483647. For example, if s3 and (s3)+1 store the value -2147483648 and the value 1000 in s1 is subtracted, the result is

$$-2147483648 - 1000 = 80000000H - 3E8H = 7FFFFC18H = 2147482648.$$

If the value 2147483647 is exceeded, the remaining subtraction is proceeded beginning from -2147483648.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The value in s1 ((s1)+1) is greater than that in s2 ((s2)+1) (error code 4100).

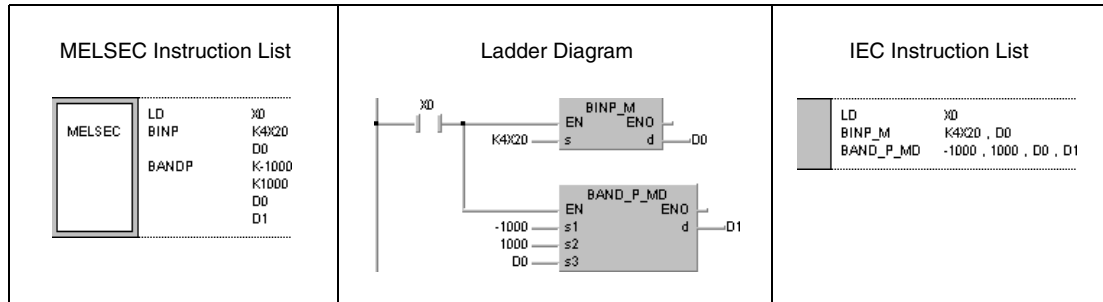
**Program Example 1** BANDP

With leading edge from X0, the following program subtracts the lower (negative) limit value -1000 and the upper (positive) limit value 1000 from the BCD data at X20 through X2F. The result is stored in D1.

If the value in D0 is greater than 1000, the value D0 - 1000 is stored in D1.

If the value in D0 is less than -1000, the value D0 - (-1000) is stored in D1.

If the value in D0 ranges within -1000 and 1000, the value 0 is stored in D1.



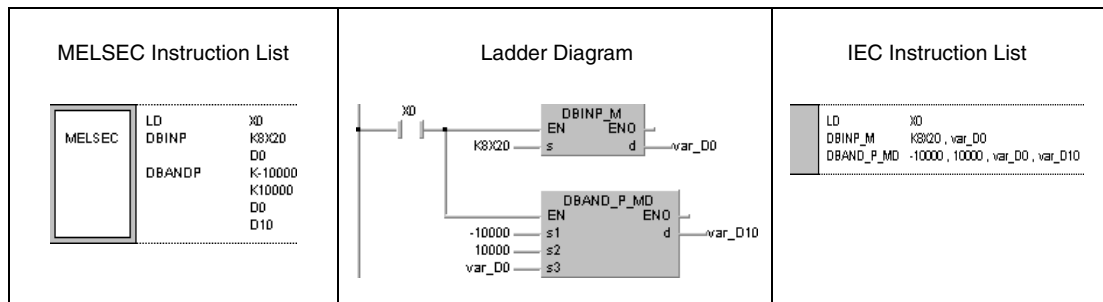
**Program Example 2** DBANDP

With leading edge from X0, the following program subtracts the lower (negative) limit value -10000 and the upper (positive) limit value 10000 from the BCD data at X20 through X3F. The result is stored in D10 and D11.

If the value in D0 and D1 is greater than 10000, the value D0, D1 - 1000 is stored in D10 and D11.

If the value in D0 and D1 is less than -10000, the value D0, D1 - (-10000) is stored in D10 and D11.

If the value in D0 and D1 ranges within -10000 and 10000, the value 0 is stored in D10 and D11.



7.13.3 ZONE, ZONEP, DZONE, DZONEP

CPU

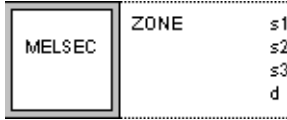
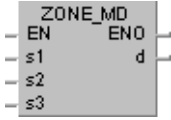

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

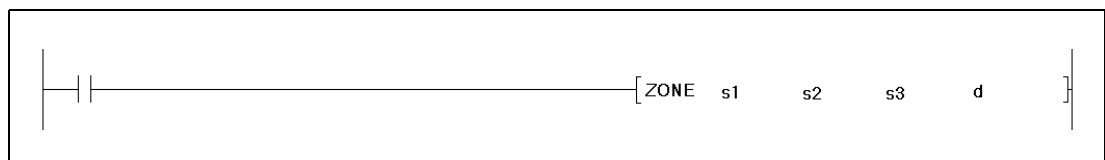
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	●	●	●	●	●	●	●	●	—	—	5
s2	●	●	●	●	●	●	●	●	—		
s3	●	●	●	●	●	●	●	●	—		
d	●	●	●	●	●	●	●	—	—		

GX IEC  
Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	--

GX  
Developer



Variables

Set Data	Meaning	Data Type
s1	Negative zone control value to be added to the input value.	BIN 16-Bit
s2	Positive zone control value to be added to the input value.	
s3	Input value to be controlled via zone control.	
d	First number of device storing total of input value and zone control value.	

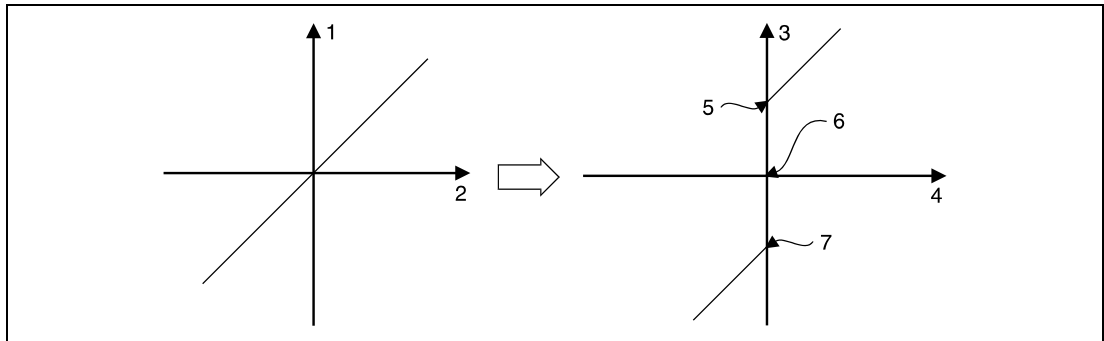
**Functions BIN 16-bit and 32-bit zone control****ZONE Zone control of BIN 16-bit data**

The ZONE instruction adds a negative and a positive control value to a BIN 16-bit value in a device specified by s3. The negative control value is stored in s1; the positive control value is stored in s2. The result is stored depending on the input value in the device specified by d as follows:

If the data value in s3 is less than 0, the result of the addition  $s3 + s1$  is stored in the device specified by d.

If the data value in s3 is greater than 0, the result of the addition  $s3 + s2$  is stored in the device specified by d.

If the data value in s3 is equal to 0, the value 0 is stored in the device specified by d.



- <sup>1</sup> Output value
- <sup>2</sup> Input value
- <sup>3</sup> Output value (d)
- <sup>4</sup> Input value (s3)
- <sup>5</sup> Upper (positive) zone control value (s2)
- <sup>6</sup> Input value = 0
- <sup>7</sup> Lower (negative) zone control value (s1)

The values in s1, s2, and s3 have to range within -32768 and 32767.

If the addition result leaves the relevant device range of -32768 and 32767, the output value is controlled as follows:

If the value -32768 is fallen below, the remaining addition is proceeded beginning from 32767. For example, if s3 stores the value -32768 and the value -100 in s1 is added, the result is

$$-32768 + (-100) = 8000H + FF9CH = 7F9CH = 32668.$$

If the value 32767 is exceeded, the remaining addition is proceeded beginning from -32768.

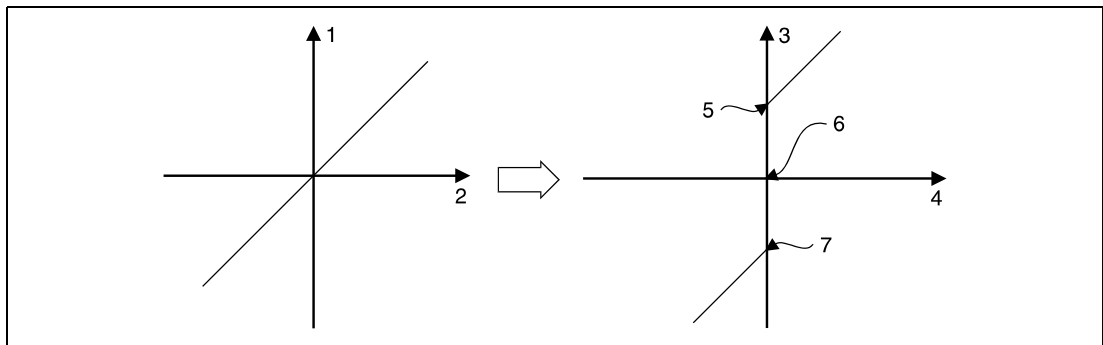
**DZONE Zone control of BIN 32-bit data**

The DZONE instruction adds a negative and a positive control value to a BIN 32-bit value in a device specified by s3 and (s3)+1. The negative control value is stored in s1 and (s1)+1; the positive control value is stored in s2 and (s2)+1. The result is stored depending on the input value in the device specified by d and d+1 as follows:

If the data value in s3 and (s3)+1 is less than 0, the result of the addition s3, (s3)+1 + s1, (s1)+1 is stored in the device specified by d and d+1.

If the data value in s3 and (s3)+1 is greater than 0, the result of the addition s3, (s3)+1 + s2, (s2)+1 is stored in the device specified by d+1.

If the data value in s3 and (s3)+1 is equal to 0, the value 0 is stored in the device specified by d and d+1.



- 1 Output value
- 2 Input value
- 3 Output value (d+1, d)
- 4 Input value ((s3)+1, s3)
- 5 Upper (positive) zone control value ((s2)+1, s2)
- 6 Input value = 0
- 7 Lower (negative) zone control value ((s1)+1, s1)

The values in s1 and (s1)+1, s2 and (s2)+1, and s3 and (s3)+1 have to range within -2147483648 and 2147483647.

If the addition result leaves the relevant device range of -2147483648 and 2147483647 the output value is controlled as follows:

If the value -2147483648 is fallen below, the remaining addition is proceeded beginning from 2147483647. For example, if s3 and (s3)+1 store the value -2147483648 and the value -1000 in s1 is added, the result is

$$-2147483648 + (-1000) = 80000000H + FFFFC18H = 7FFFC18H = 2147482648.$$

If the value 2147483647 is exceeded, the remaining addition is proceeded beginning from -2147483648.

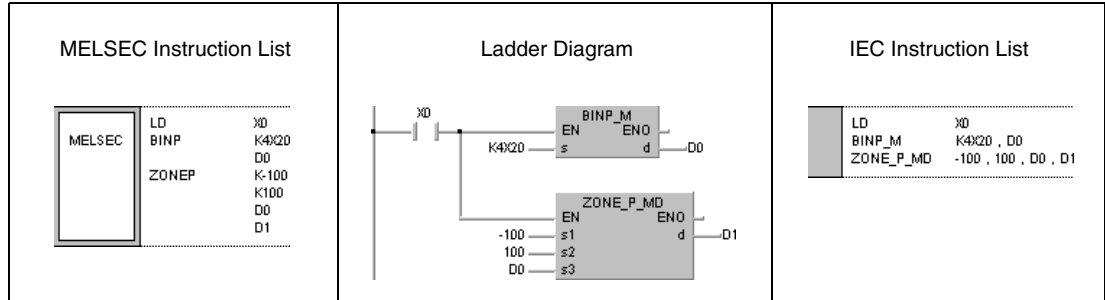
**Program Example 1** ZONEP

With leading edge from X0, the following program adds the negative zone control value -100 and the positive zone control value 100 to BCD data at X20 through X2F. The result is stored in D1.

If the value in D0 is greater than 0, the value  $D0 + 100$  is stored in D1.

If the value in D0 is less than 0, the value  $D0 + (-100)$  is stored in D1.

If the value D0 is equal to 0, the value 0 is stored in D1.



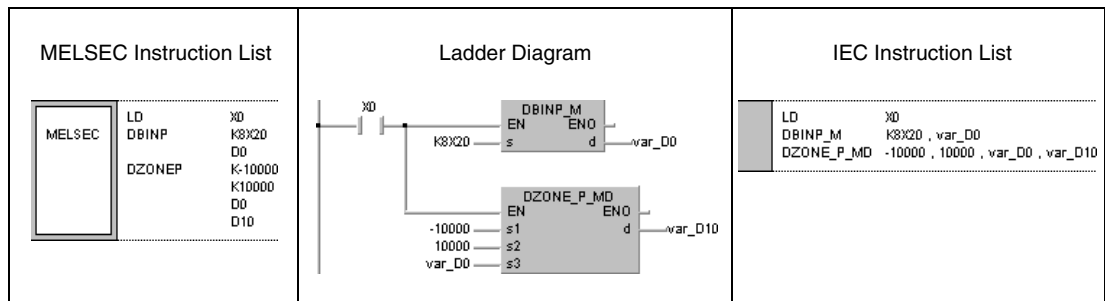
**Program Example 2** DZONEP

With leading edge from X0, the following program adds the negative zone control value -10000 and the positive zone control value 10000 to BCD data at X20 through X3F. The result is stored in D10 and D11.

If the value in D0 and D1 is greater than 0, the value  $D0, D1 + 10000$  is stored in D10 and D11.

If the value in D0 and D1 is less than 0, the value  $D0, D1 + (-10000)$  is stored in D10 and D11.

If the value D0 and D1 is equal to 0, the value 0 is stored in D10 and D11.



## 7.14 File register switching instructions

The switching instructions enable switching between file register blocks and between file names in file registers. The table below gives an overview of the instructions.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Setting file register blocks	RSET	RSET_MD
		RSET_K_MD
	RSETP	RSET_P_MD
		RSET_K_P_MD
Setting file register files	QDRSET	QDRSET_M
	QDRSETP	QDRSET_P_MD
Setting comment files	QCDSET	QCDSET_M
	QCDSET	QCDSET_P_MD



**7.14.1 RSET, RSETP**

**CPU**

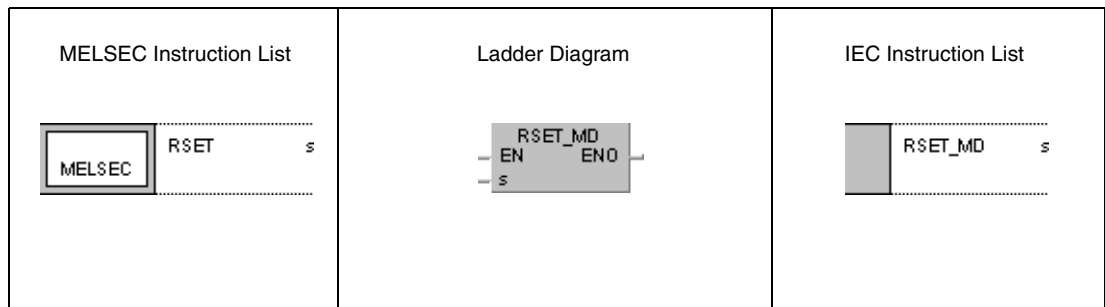
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

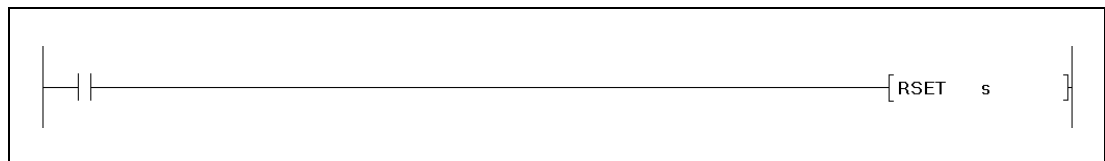
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	●	●	●	●	—	SM0	2	

**GX IEC Developer**



**GX Developer**

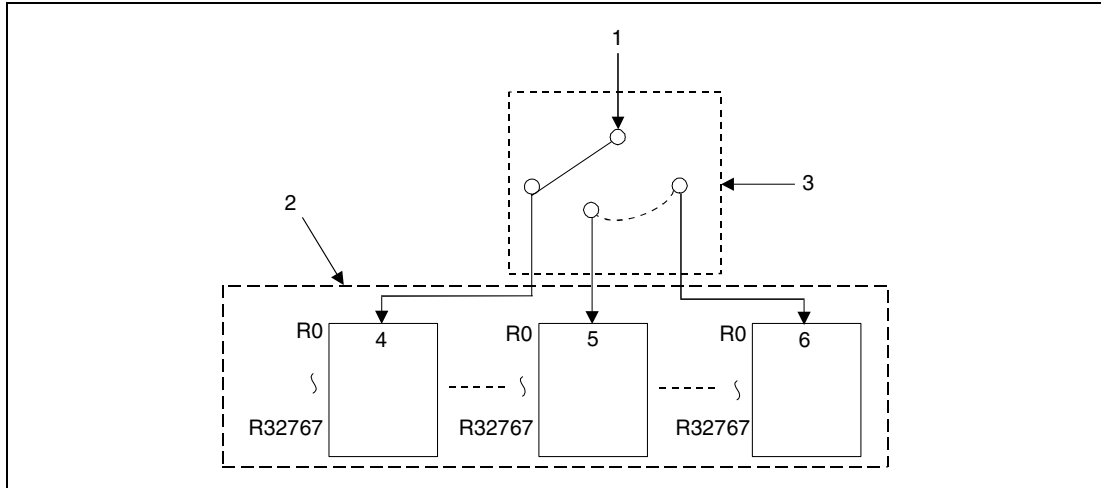


**Variables**

Set Data	Meaning	Data Type
s	Number of file register block or first number of device storing this number.	BIN 16-bit

**Functions**    **Setting file register blocks****RSET**    **Switch instruction for file register blocks**

The RSET instruction switches from a file register block being in use by a program to a file register block with the number specified by s. After switching over, the sequence program exclusively accesses file registers (R0 - R32767) in the specified block.



<sup>1</sup> Processing with file register access

<sup>2</sup> File used by program

<sup>3</sup> Number of file register block (s)

<sup>4</sup> Block 0

<sup>5</sup> Block 1

<sup>6</sup> Block n

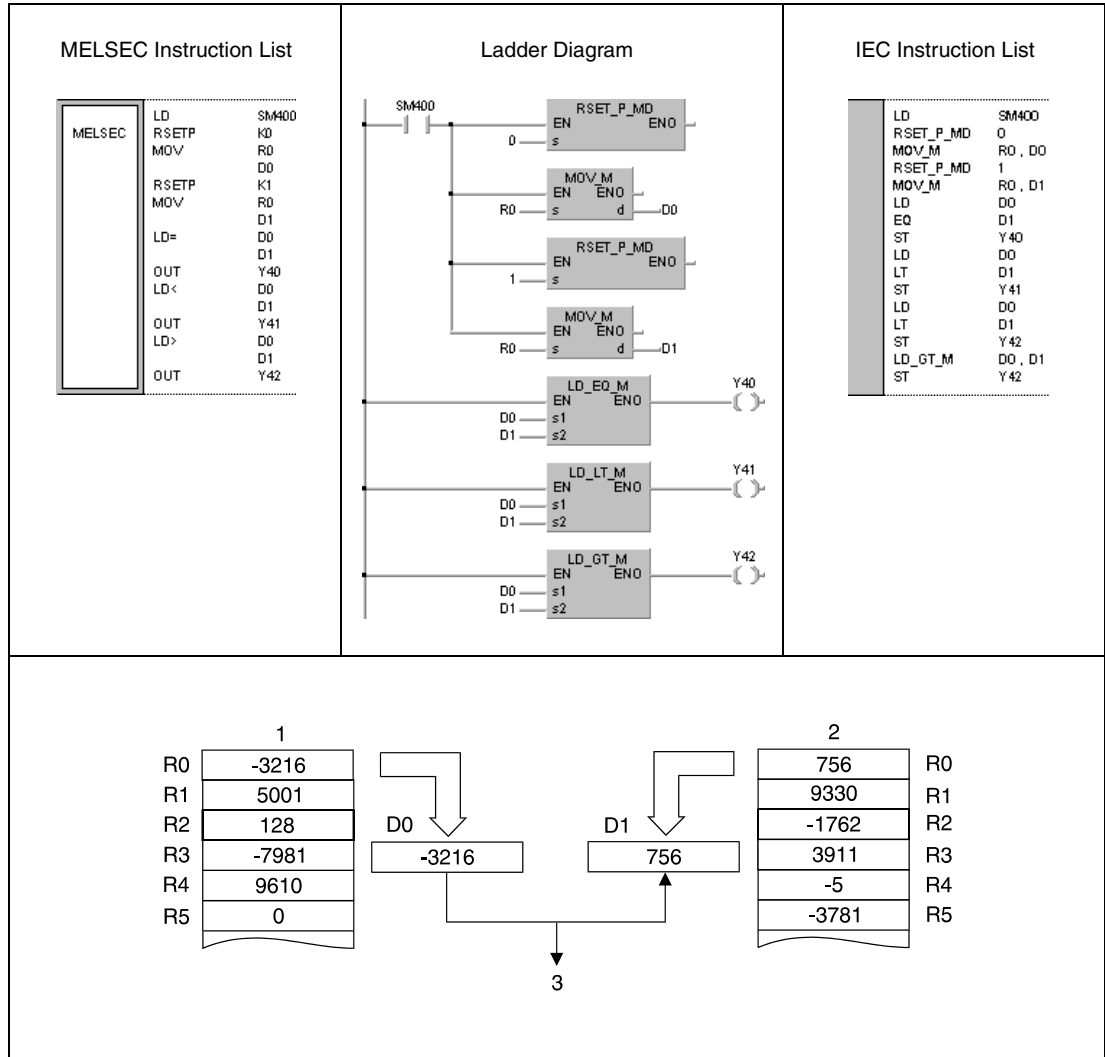
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The block number specified by s does not exist (error code 4100).
- There are no file registers in the block specified by s (error code 4101).

**Program Example** RSETP

With leading edge from SM400, the following program compares the file register R0 in register block 0 to the file register R0 in register block 1. The file register blocks 0 and 1 are addressed via the RSET instruction. Both file registers R0 are read via the MOV instruction. If the value in R0 (block 0) is equal to the value in R0 (block 1), the output Y40 is set. If the value in R0 (block 0) is less than the value in R0 (block 1), the output Y41 is set. If the value in R0 (block 0) is greater than the value in R0 (block 1), the output Y42 is set.



- <sup>1</sup> Block 0
- <sup>2</sup> Block 1
- <sup>3</sup> Y41 is set because D0 is less than D1

**7.14.2 QDRSET, QDRSETP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

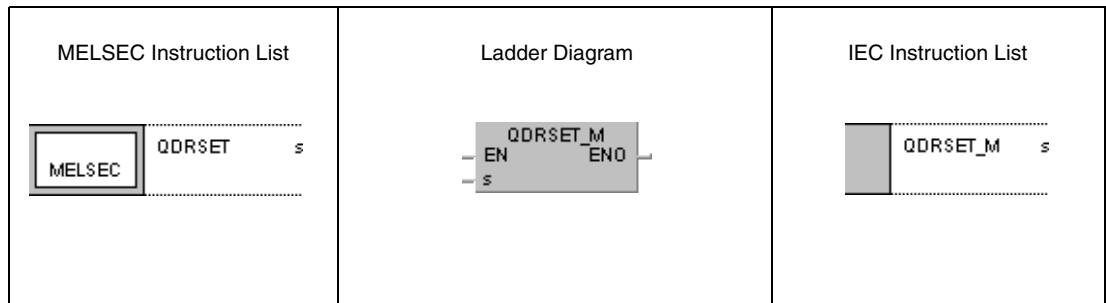
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

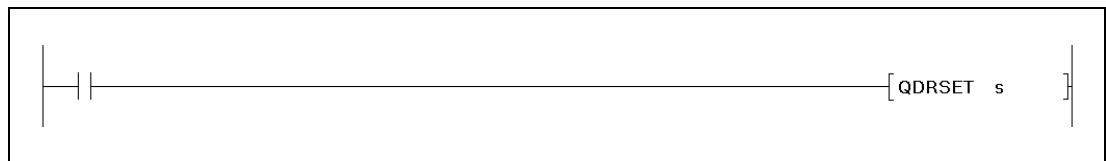
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	2 + n <sup>1)</sup>

<sup>1</sup> n = (number of program name characters)/2 = Number of additional steps (Decimal fractions are rounded up)

**GX IEC  
Developer**



**GX  
Developer**

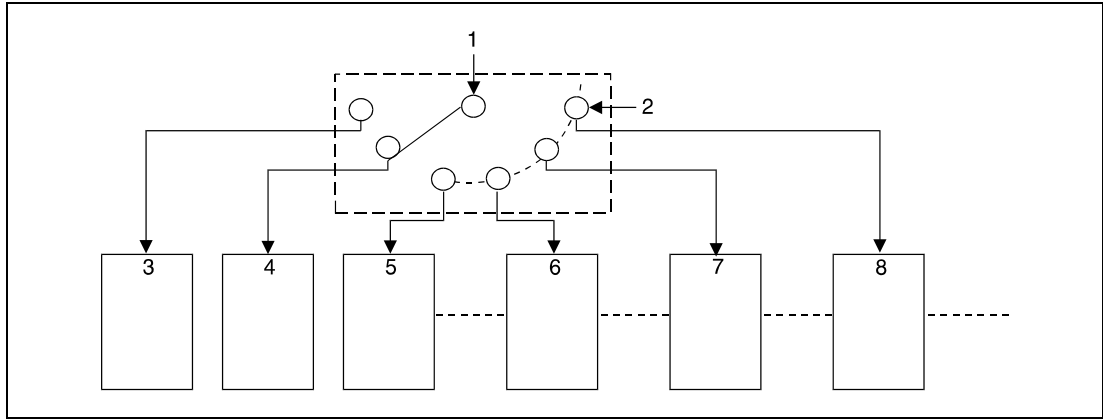


**Variables**

Set Data	Meaning	Data Type
s	Drive number and file name of file register file to be switched to or first number of device storing such data.	Character string

**Functions**     **Setting file register files****QDRSET**     **Switch instruction for file register files**

The QDRSET instruction switches from a file register file being in use by a program to a file register file specified by *s*. After switching over, the sequence program exclusively accesses file registers (R0 - R32767) in block 0 of the specified file register file. The file register blocks are selected via the RSET instruction.



- 1 Processing with file register access
- 2 Setting the drive and file(s)
- 3 Drive 1, file A
- 4 Drive 1, file B
- 5 Drive 1, file C
- 6 Drive 2, file A
- 7 Drive 3, file A
- 8 Drive 4, file A

In total, 4 drives can be assigned (1-4). The drive number 0 cannot be assigned; this range is reserved for internal memory.

The extension .QDR is not needed to be entered for file specification.

A file name setting can be cleared by specifying the NULL character (00H) for the file name.

File register files selected by the QDRSET instruction are given priority even if a drive number and file name were specified by the parameters.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The file register file does not exist on the drive specified by *s* (error code 2410).

**Program Example**

**QDRSET/QDRSETP**

With leading edge from X0, the following program switches to the file register file ABC.QDR on drive 1. While X1 is set, the file register file DEF.QDR on drive 3 is accessed.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 10%;">MELSEC</td> <td style="width: 10%; border-bottom: 1px dotted black;">LD</td> <td style="width: 10%;">X0</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">QDRSETP</td> <td style="border-bottom: 1px dotted black;">"1:ABC"</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">X1</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">QDRSET</td> <td style="border-bottom: 1px dotted black;">"3:DEF"</td> </tr> </table>	MELSEC	LD	X0		QDRSETP	"1:ABC"		LD	X1		QDRSET	"3:DEF"	<pre> graph LR     X0((X0)) --- R1(( ))     R1 --- QDRSETP_M[QDRSETP_M]     QDRSETP_M --- EN1[EN]     QDRSETP_M --- s1[s]          X1((X1)) --- R2(( ))     R2 --- QDRSET_M[QDRSET_M]     QDRSET_M --- EN2[EN]     QDRSET_M --- s2[s]     </pre>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%; border-bottom: 1px dotted black;">LD</td> <td style="width: 10%;">X0</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">QDRSETP_M</td> <td style="border-bottom: 1px dotted black;">"1:ABC"</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">LD</td> <td style="border-bottom: 1px dotted black;">X1</td> </tr> <tr> <td></td> <td style="border-bottom: 1px dotted black;">QDRSET_M</td> <td style="border-bottom: 1px dotted black;">"3:DEF"</td> </tr> </table>		LD	X0		QDRSETP_M	"1:ABC"		LD	X1		QDRSET_M	"3:DEF"
MELSEC	LD	X0																								
	QDRSETP	"1:ABC"																								
	LD	X1																								
	QDRSET	"3:DEF"																								
	LD	X0																								
	QDRSETP_M	"1:ABC"																								
	LD	X1																								
	QDRSET_M	"3:DEF"																								

**7.14.3 QCDSET, QCDSETP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

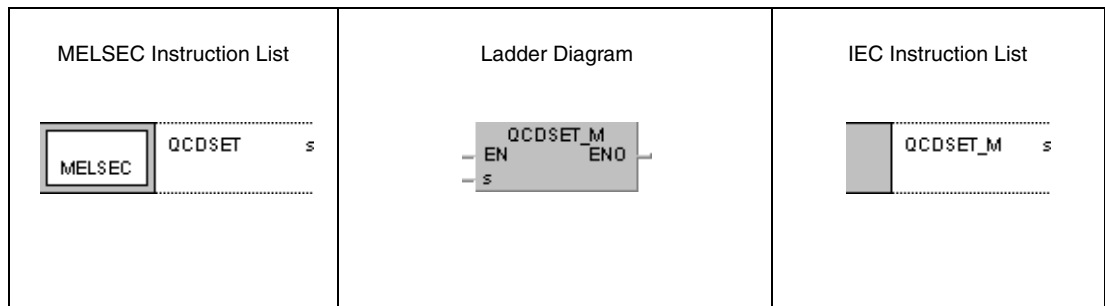
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	2 + n <sup>1)</sup>

<sup>1</sup> n = (number of program name characters)/2 = Number of additional steps (Decimal fractions are rounded up)

**GX IEC  
Developer**



**GX  
Developer**



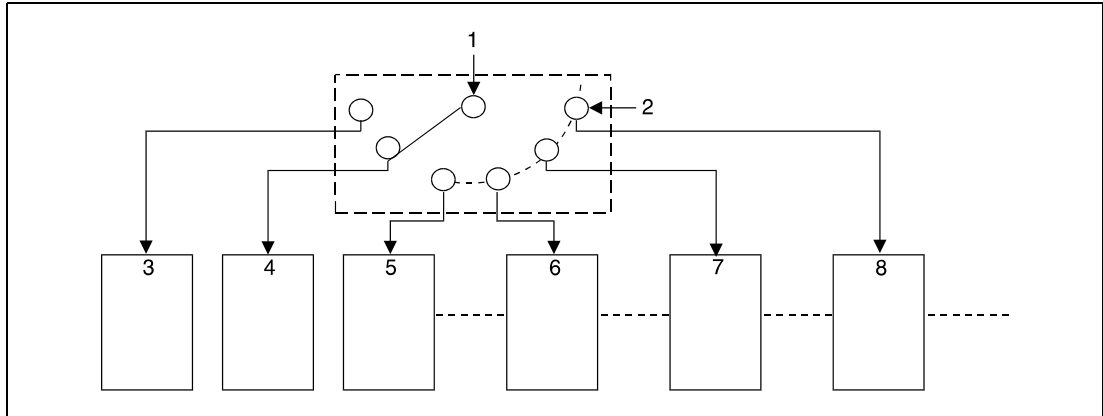
**Variables**

Set Data	Meaning	Data Type
s	Drive number and file name of comment file to be switched to or first number of device storing such data.	Character string

**Functions      Setting comment files**

**QCDSET      Switch instruction for comment files**

The QCDSET instruction switches from a comment file being in use by a program to a comment file specified by s. After switching over, the sequence program exclusively accesses comment data of the specified comment file.



- 1 Processing with comment data access
- 2 Setting the drive and comment file(s)
- 3 Drive 1, file A
- 4 Drive 1, file B
- 5 Drive 1, file C
- 6 Drive 2, file A
- 7 Drive 3, file A
- 8 Drive 4, file A

In total, 4 drives can be assigned (1-4). The drive number 0 cannot be assigned; this range is reserved for internal memory.

The extension .QCD is not needed to be entered for file specification.

A file name setting can be cleared by specifying the NULL character (00H) for the file name.

Comment files selected by the QCDSET instruction are given priority even if a drive number and file name were specified by the parameters.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The comment file does not exist on the drive specified by s (error code 2410).



**Program Example**

**QCDSET/QCDSETP**

With leading edge from X0, the following program switches to the comment file ABC.QCD on drive 1. While X1 is set, the comment file DEF.QCD on drive 3 is accessed.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List				
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: center; vertical-align: middle;"><b>MELSEC</b></td> <td style="padding: 5px;"> <pre>LD      X0 QCDSETP  "1:ABC" LD      X1 QCDSET   "3:DEF"</pre> </td> </tr> </table>	<b>MELSEC</b>	<pre>LD      X0 QCDSETP  "1:ABC" LD      X1 QCDSET   "3:DEF"</pre>		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="padding: 5px;"> <pre>LD      X0 QCDSETP_M  "1:ABC" LD      X1 QCDSET_M   "3:DEF"</pre> </td> </tr> </table>		<pre>LD      X0 QCDSETP_M  "1:ABC" LD      X1 QCDSET_M   "3:DEF"</pre>
<b>MELSEC</b>	<pre>LD      X0 QCDSETP  "1:ABC" LD      X1 QCDSET   "3:DEF"</pre>					
	<pre>LD      X0 QCDSETP_M  "1:ABC" LD      X1 QCDSET_M   "3:DEF"</pre>					

## 7.15 Clock instructions

The clock instructions read and write, add and subtract, and change the data format of clock data. The table below gives an overview of these instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Reading clock data	DATERD	DATERD_MD
	DATERDP	DATERD_P_MD
Writing clock data	DATEWR	DATEWR_MD
	DATEWRP	DATEWR_P_MD
Adding clock data	DATE+	DATEPLUS_M
	DATE+P	DATEPLUSP_M
Subtracting clock data	DATE-	DATEMINUS_M
	DATE-P	DATEMINUSP_M
Changing clock data format from hh:mm:ss to seconds	SECOND	SECOND_M
	SECONDP	SECONDP_M
Changing clock data format from seconds to hh:mm:ss	HOUR	HOUR_M
	HOURP	HOURP_M

**7.15.1 DATERD, DATERDP**

**CPU**

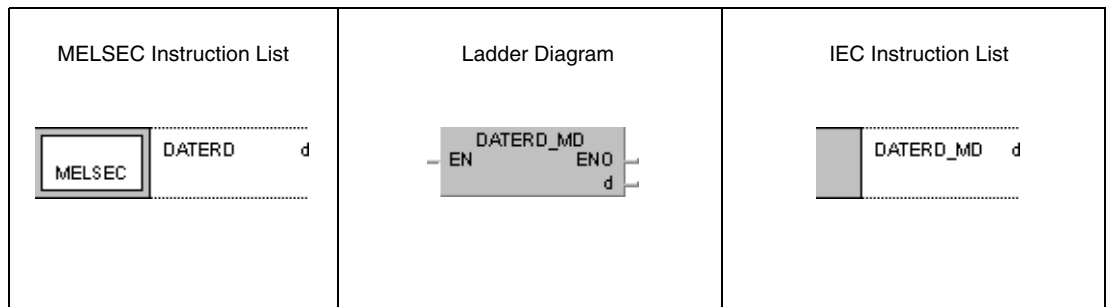
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

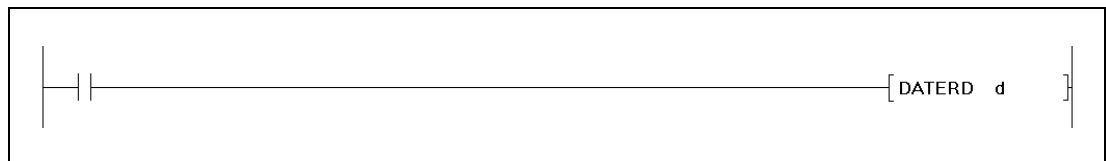
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant			Other
	Bit	Word		Bit	Word						
d	—	●	●	—	—	—	—	—	—	2	

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
d	First number of device storing clock data being read.	BIN 16-bit	Array [0..6] of ANY16

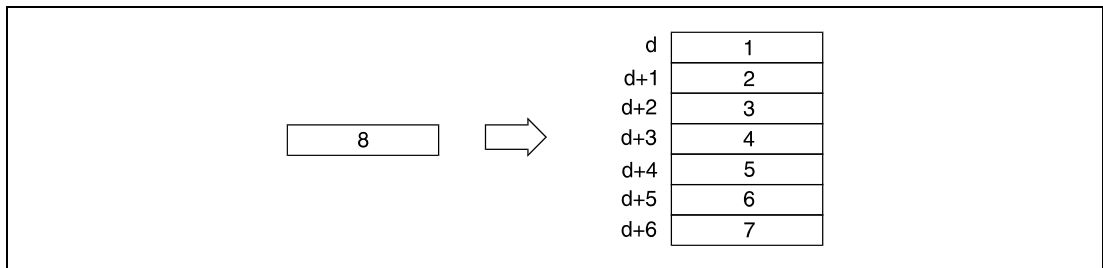
**Functions      Reading clock data**

**DATERD      Read instruction**

The DATERD instruction reads year, month, day, hour, minute, second, and weekday from the internal QnA CPU clock and stores the clock data in binary format in the devices specified by d+0 (Array\_d[0]) through d+6 (Array\_d[6]). The assignment of registers to clock data is illustrated below:

- d+0, array\_d[0] = year (1)
- d+1, array\_d[1] = month (January = 1, December = 12) (2)
- d+2, array\_d[2] = day (3)
- d+3, array\_d[3] = hour (24 hour format) (4)
- d+4, array\_d[4] = minute (5)
- d+5, array\_d[5] = second (6)
- d+6, array\_d[6] = day of the week(7)

The QnA clock is indicated 8.



The following table contains the value range of clock data in d+0 through d+6:

Clock data	Year	Month	Day	Hour	Minute	Second	Day of the week
Input range	1)	1 - 12	1 - 31	0 - 23	0 - 59	0 - 59	0 - 6
Devices	d+0 (array_d[0])	d+1 (array_d[1])	d+2 (array_d[2])	d+3 (array_d[3])	d+4 (array_d[4])	d+5 (array_d[5])	d+6 (array_d[6])

<sup>1</sup> 0 to 99 for QnA CPU, 1980 to 2079 for a System Q CPU

The "year" is stored in a QnA CPU as a two-digit number. Only the ones and tens is stored (eg. 1998 = 98).

When a System Q CPU is used, the "year" is stored as four-digit indication.

The day of the week stored in d+6 (Array\_d[6]) is indicated from 0 to 6. The table below shows the assignment of weekdays:

Weekday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Storage value	0	1	2	3	4	5	6

Leap years are calculated automatically by the CPU clock.

**Program Example** DATERD (QnA CPU)

While SM400 is set, the following program reads clock data from the internal CPU clock and outputs it in BCD format at the outputs Y47 through Y67 as follows:

Y60 - Y67 = month

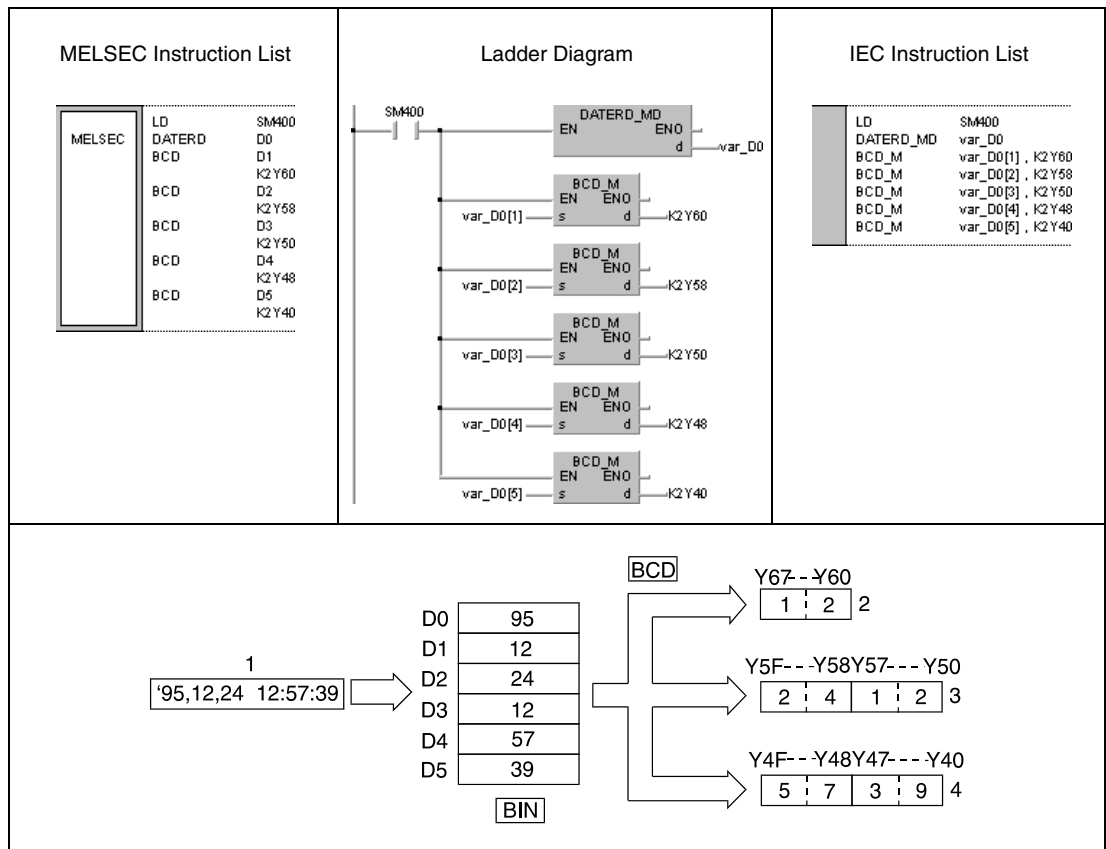
Y58 - Y5F = day

Y50 - Y57 = hour

Y48 - Y4F = minute

Y40 - Y47 = second

Clock data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	D0 (var_D0[0])	D1 (var_D0[1])	D2 (var_D0[2])	D3 (var_D0[3])	D4 (var_D0[4])	D5 (var_D0[5])	D6 (var_D0[6])



- <sup>1</sup> Clock data
- <sup>2</sup> Month
- <sup>3</sup> Day, hour
- <sup>4</sup> Minute, second

**NOTE** This program example will not run without variable definition in the header of the program or organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

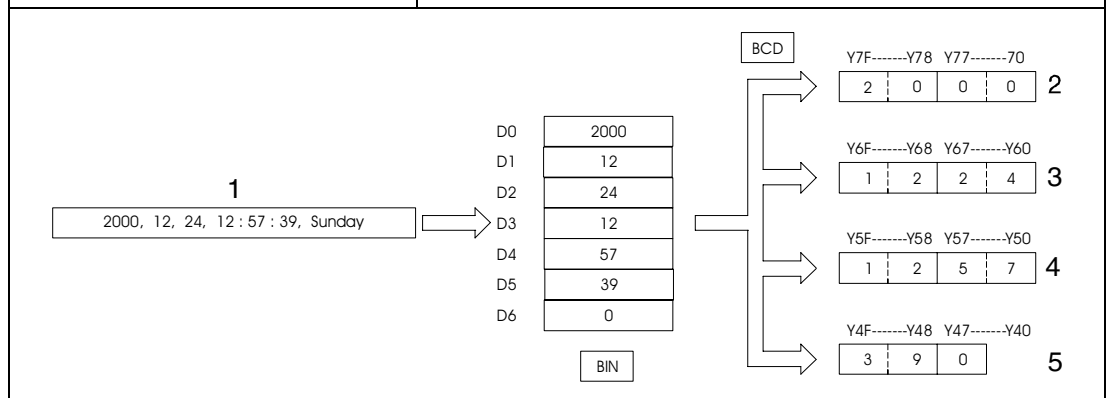
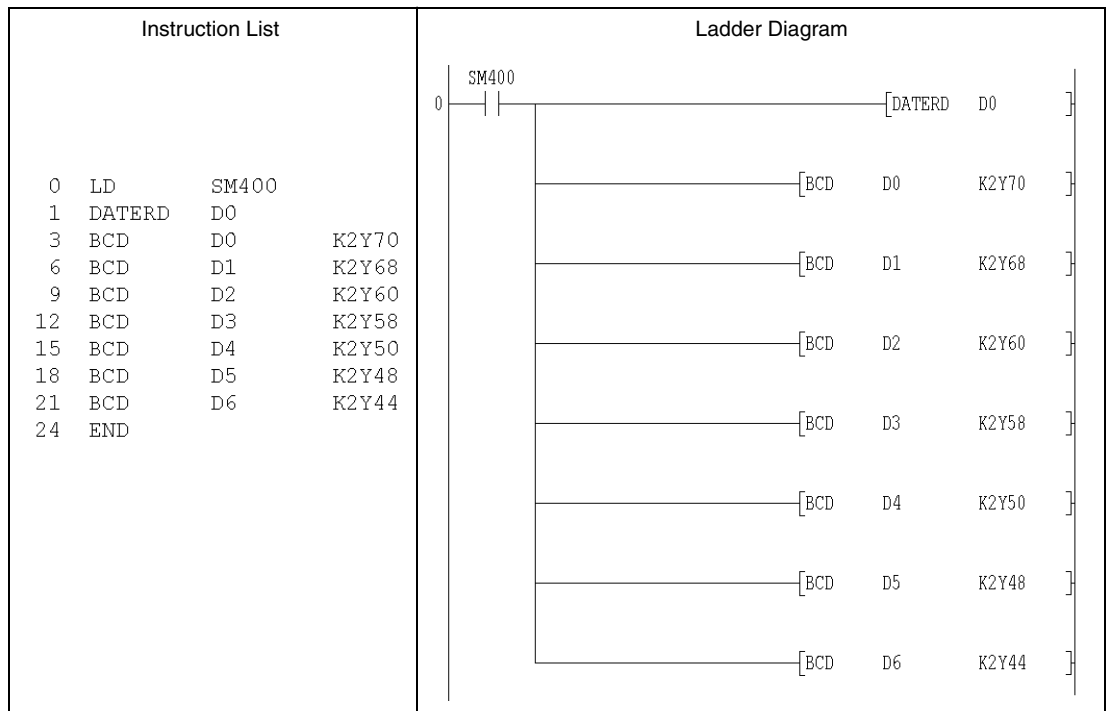
**Program Example**

DATERD (System Q CPU)

While SM400 is set, the following program reads clock data from the internal CPU clock and outputs it in BCD format at the outputs Y47 through Y67 as follows:

- Y70 - Y7F = year                      Y68 - Y6F = month
- Y60 - Y67 = day                      Y58 - Y5F = hour
- Y50 - Y57 = minute                  Y48 - Y4F = seconds
- Y44 - Y47 = day of the week

Clock data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	D0	D1	D2	D3	D4	D5	D6



- <sup>1</sup> Clock data
- <sup>2</sup> Year
- <sup>3</sup> Month, day
- <sup>4</sup> Hour, minute
- <sup>5</sup> Second, day of the week

**7.15.2 DATEWR, DATEWRP**

**CPU**

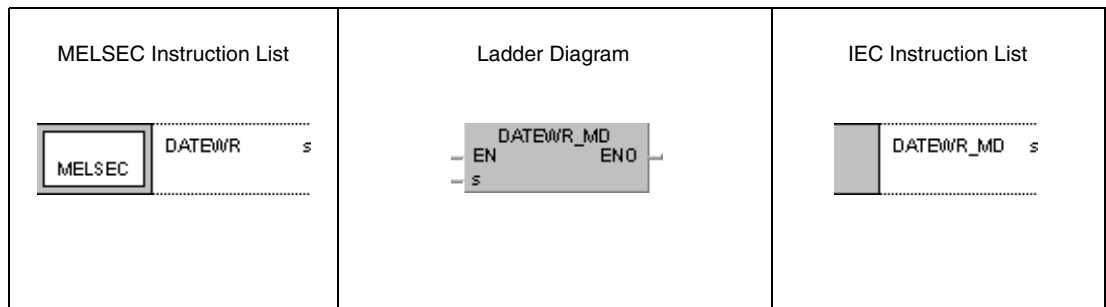
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	●

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

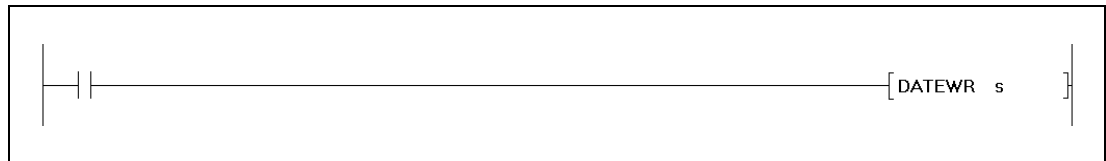
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other DY
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	2

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	First number of device storing the data to be written to the internal CPU clock.	BIN 16-bit	Array [0..6] of ANY16

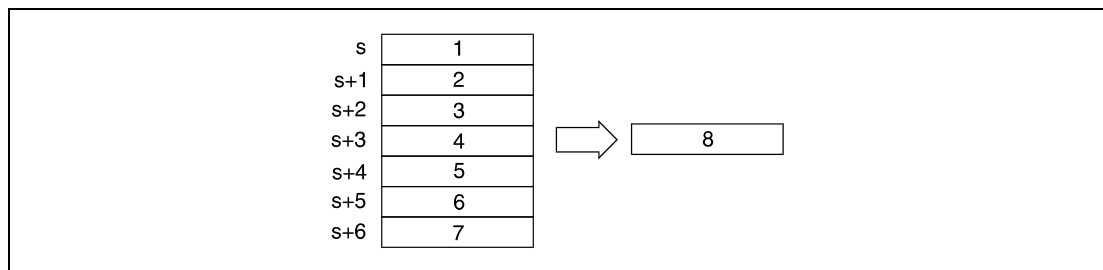
**Functions Writing clock data**

**DATEWR Write instruction**

The DATEWR instruction writes clock data of year, month, day, hour, minute, second, and weekday stored in the devices specified by d+0 (Array\_d[0]) through d+6 (Array\_d[6]) to the internal CPU clock. The clock data are stored in binary format. The assignment of registers to clock data is illustrated below:

- s+0, array\_s[0] = year (1)
- s+1, array\_s[1] = month (January = 1, December = 12) (2)
- s+2, array\_s[2] = day (3)
- s+3, array\_s[3] = hour (24 hour format, 0 to 23 hours) (4)
- s+4, array\_s[4] = minute (5)
- s+5, array\_s[5] = second (6)
- s+6, array\_s[6] = weekday (7)

The QnA clock is indicated 8.



The following table contains the value range of clock data in d+0 through d+6:

Clock data	Year	Month	Day	Hour	Minute	Second	Day of the week
Input range	1)	1 - 12	1 - 31	0 - 23	0 - 59	0 - 59	0 - 6
Devices	s+0 (array_s[0])	s+1 (array_s[1])	s+2 (array_s[2])	s+3 (array_s[3])	s+4 (array_s[4])	s+5 (array_s[5])	s+6 (array_s[6])

<sup>1</sup> 0 to 99 for QnA CPU, 1980 to 2079 for a System Q CPU

The "year" is stored in a QnA CPU as a two-digit number. Only the ones and tens is stored (eg. 1998 = 98). In a System Q CPU the "year" is stored as four-digit indication.

The weekday stored in s+6 (Array\_s[6]) is indicated from 0 to 6. The table below shows the assignment of weekdays:

Weekday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Storage value	0	1	2	3	4	5	6

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The clock data specified in s+0 (Array\_s[0]) through s+6 (Array\_s[6]) exceed the relevant value range (error code 4100).



**Program Example**

DATEWRP (QnA CPU)

With leading edge from X40, the following program writes the clock data in binary format at the inputs X0 through X2F to the internal CPU clock. The inputs are assigned to the clock data as follows:

X28 - X2F = year

X20 - X27 = month

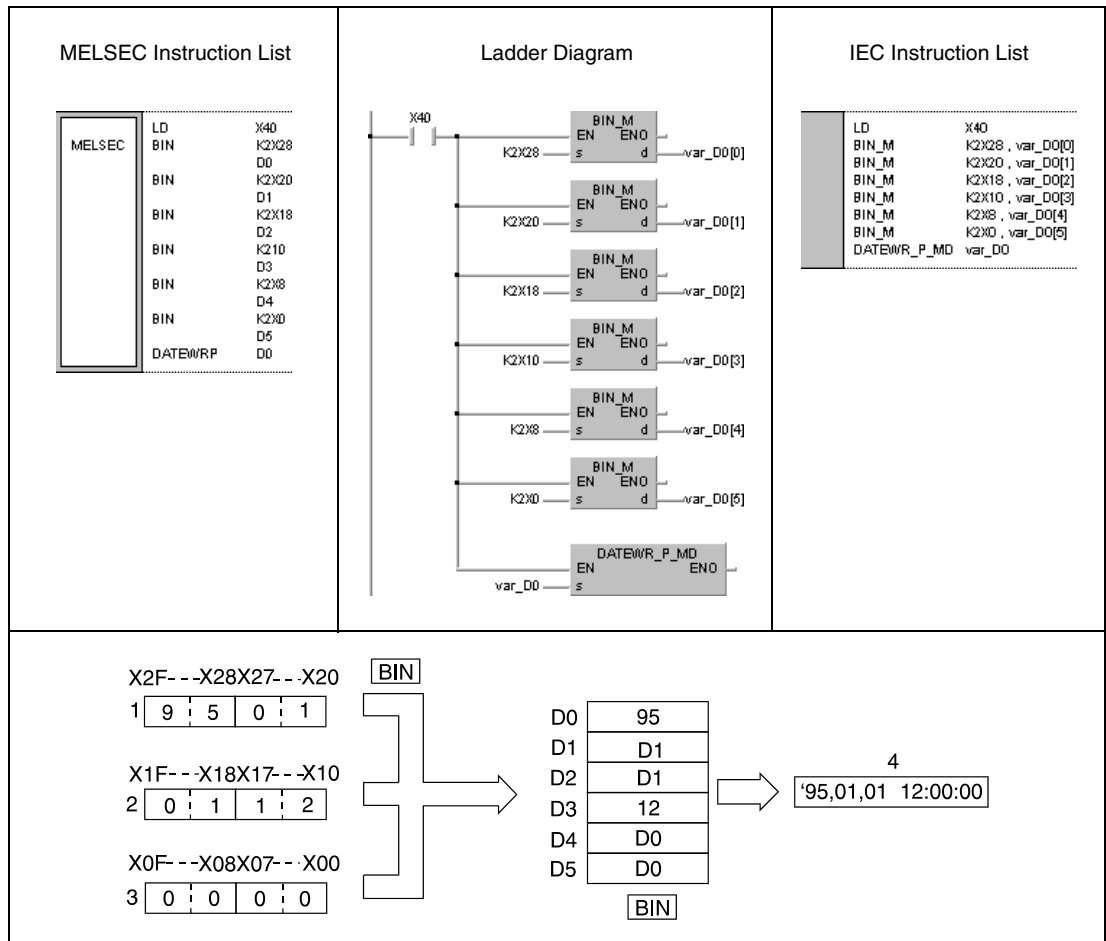
X18 - X1F = day

X10 - X17 = hour

X8 - XF = minute

X0 - X7 = second

Clock data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	D0 (var_D0[0])	D1 (var_D0[1])	D2 (var_D0[2])	D3 (var_D0[3])	D4 (var_D0[4])	D5 (var_D0[5])	D6 (var_D0[6])



- 1 Year, month
- 2 Day, hour
- 3 Minute, second
- 4 Clock data

**NOTE**

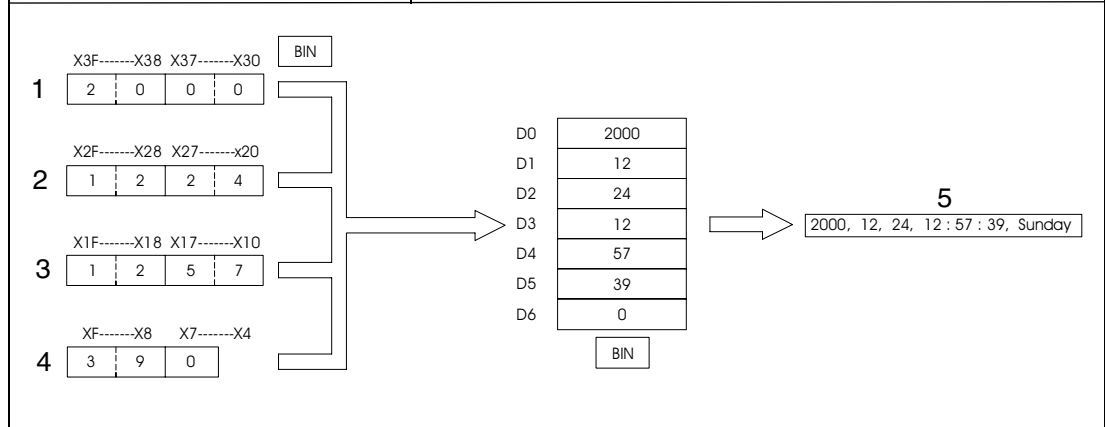
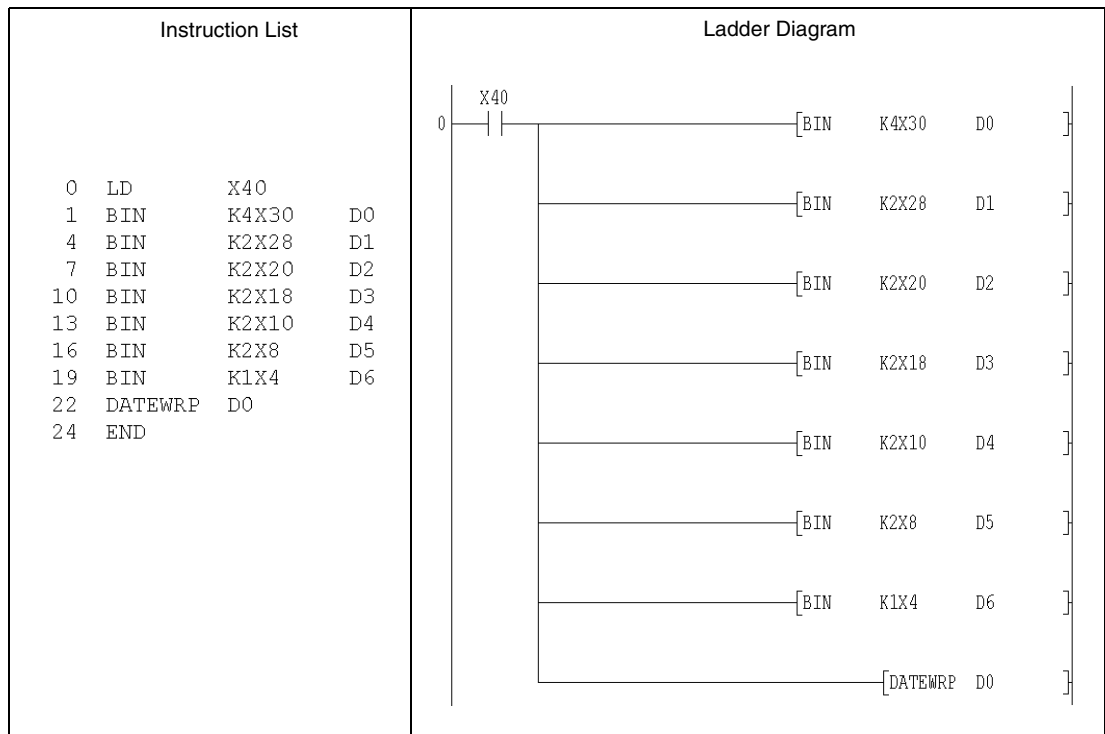
This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

**Program Example** DATEWRP (System Q CPU)

With leading edge from X40, the following program writes the clock data in binary format at the inputs X0 through X2F to the internal CPU clock. The inputs are assigned to the clock data as follows:

- X30 - X3F = year
- X28 - X2F = month
- X20 - X27 = Day
- X18 - X1F = hour
- X10 - X17 = minute
- X8 - XF = seconds

Clock data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	D0	D1	D2	D3	D4	D5	D6



- 1 Year
- 2 Month, day
- 3 Hour, minute
- 4 Seconds, Day of the week
- 5 Clock data

**7.15.3 DATE+, DATE+P**

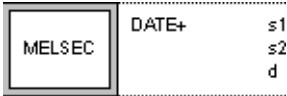
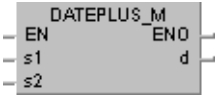
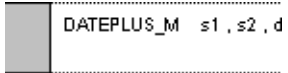
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

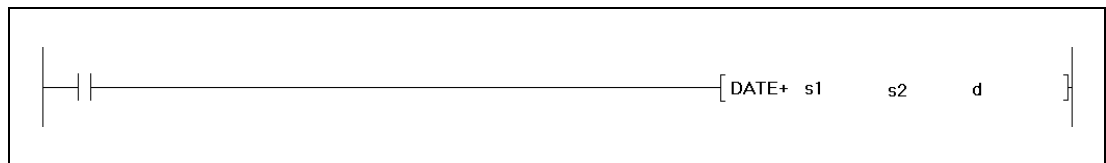
**Devices  
MELSEC Q**

	UsableDevices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	SM0	4	
s2	—	●	●	—	—	—	—	—			
d	—	●	●	—	—	—	—	—			

**GX IEC Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	--	---

**GX Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
s1	Clock data to be added to.	BIN 16-bit	Array [0..2] of ANY16
s2	Clock data to be added.		
d	First number of device storing the clock data of the operation result.		

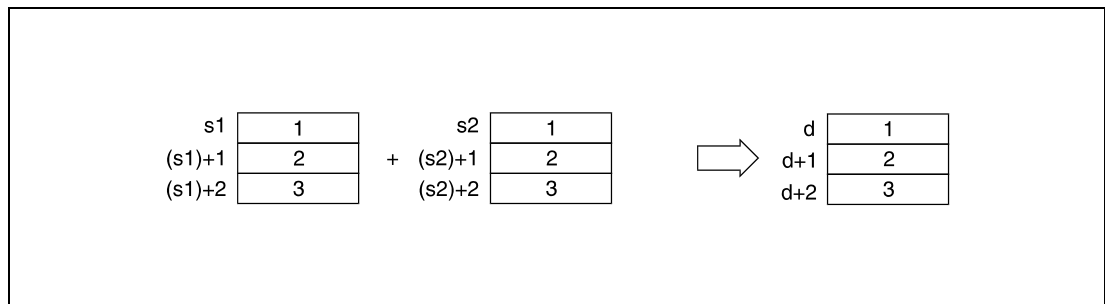
**Functions Adding clock data**

**DATE+ Addition instruction**

The DATE+ instruction adds the clock data stored in the devices specified from s2 on to the clock data stored in the devices specified from s1 on. The clock data of the operation result is stored in the devices specified from d.

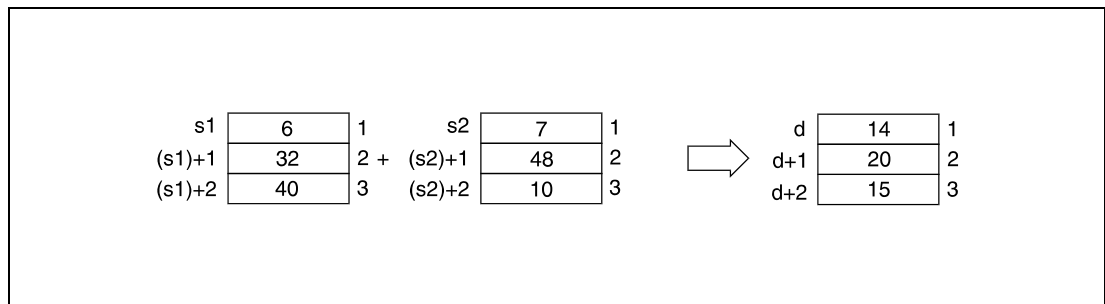
The following table contains the value range of clock data in (s1)+0 through (s1)+2, (s2)+0 through (s2)+2, and d+0 through d+2):

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Input range	—	—	—	0 - 23	0 - 59	0 - 59	—
Devices	—	—	—	s1+0 (array_s1[0])	s1+1 (array_s1[1])	s1+2 (array_s1[2])	—
Devices	—	—	—	s2+0 (array_s2[0])	s2+1 (array_s2[1])	s2+2 (array_s2[2])	—
Devices	—	—	—	d+0 (array_d[0])	d+1 (array_d[1])	d+2 (array_d[2])	—



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

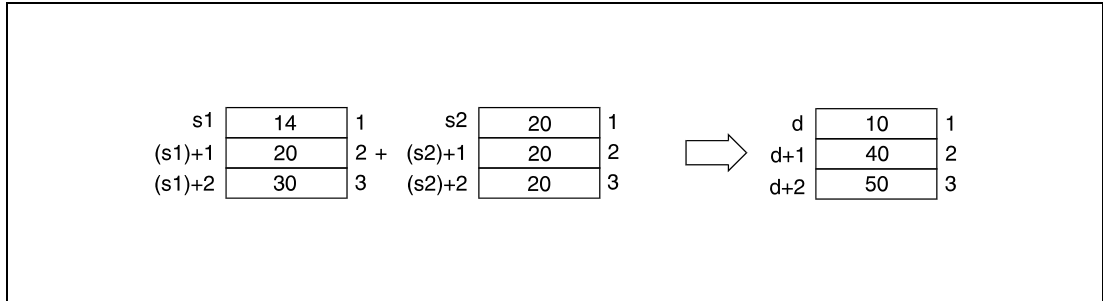
In the following diagram the clock data 6 hours, 32 minutes, 40 seconds ((s1)+0 through (s1)+2) is added the clock data 7 hours, 48 minutes, 10 seconds ((s2)+0 through (s2)+2). The result 14 hours, 20 minutes, 50 seconds is stored in d+0 through d+2.



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

If the addition result of clock data exceeds 24 hours, 24 hours are subtracted automatically to achieve a correct time value.

The following diagram illustrates the addition of 14 hours, 20 minutes, and 30 seconds to 20 hours, 20 minutes, and 20 seconds. The result would be 34 hours, 40 minutes, and 50 seconds. Since this result is not a correct time format, after the subtraction of 24 hours, the correct result is 10 hours, 40 minutes, and 50 seconds (10:40:50 the next day).



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

**NOTE** Refer to section "Writing clock data" for further information on that topic.

**Operation Errors** In the following cases an operation error occurs and the error flag is set:

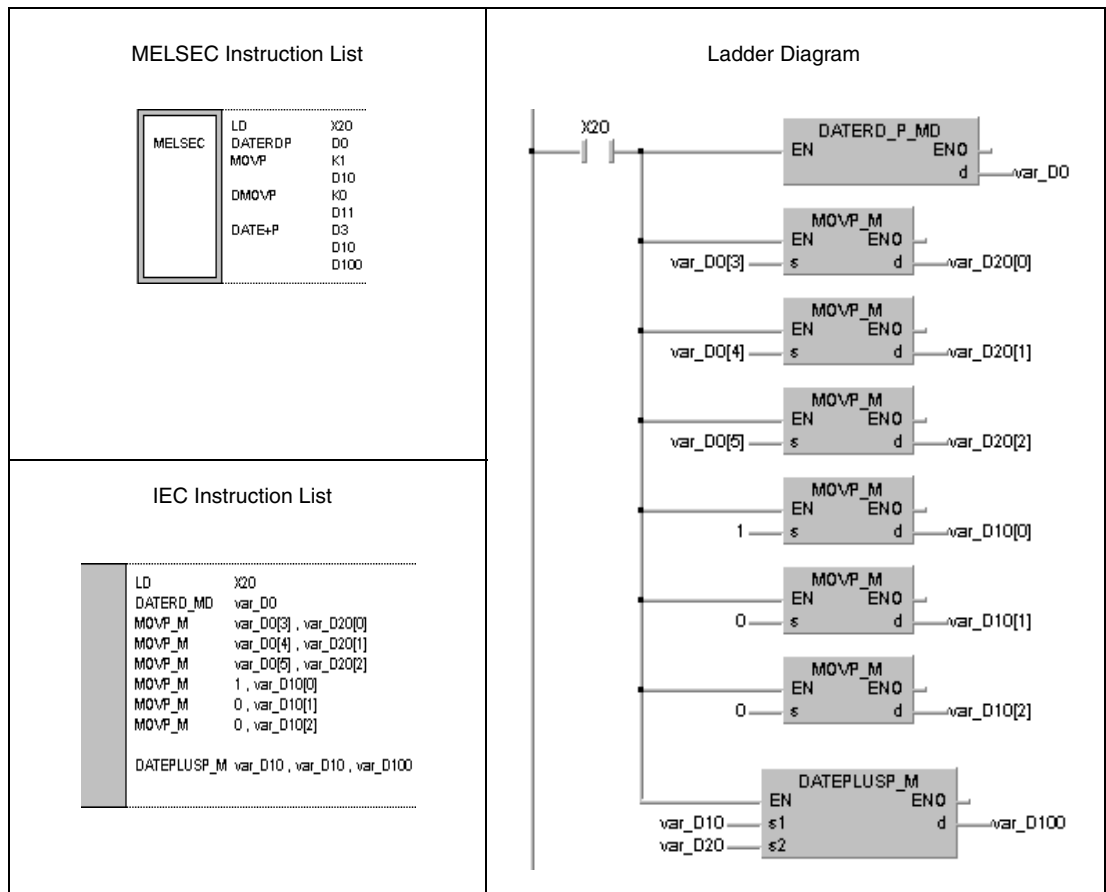
- The clock data in (s1)+0 through (s1)+2 and (s2)+0 through (s2)+2 exceed the input range.

**Program Example**

DATE+P

With leading edge from X20, the following program reads the clock data from the internal CPU clock via the DATERDP instruction and stores it in the registers D0 through D6 (see diagrams below). The DATE+P instruction adds one hour (D10, D11, D12) to the read data. The result is stored in D100 through D102 (see diagrams below).

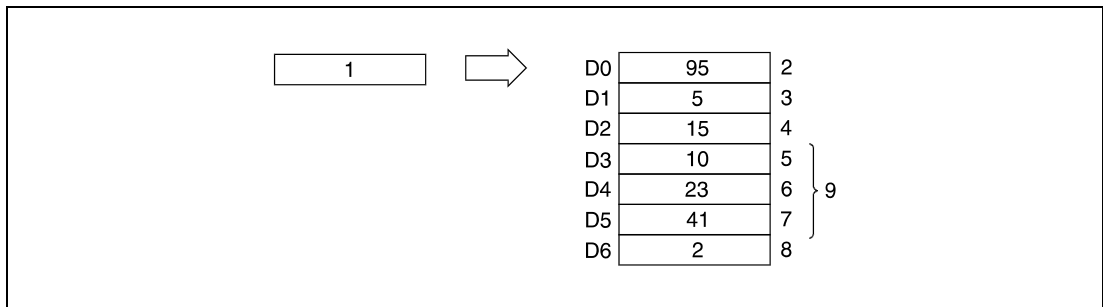
Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	D0 (var_D0[0])	D1 (var_D0[1])	D2 (var_D0[2])	D3 (var_D0[3])	D4 (var_D0[4])	D5 (var_D0[5])	D6 (var_D0[6])
Devices	—	—	—	D20 (var_D20[0])	D21 (var_D20[1])	D22 (var_D20[2])	—
Devices	—	—	—	D10 (var_D10[0])	D11 (var_D10[1])	D12 (var_D10[2])	—
Devices	—	—	—	D100 (var_D100[0])	D101 (var_D100[1])	D102 (var_D100[2])	—



**NOTE**

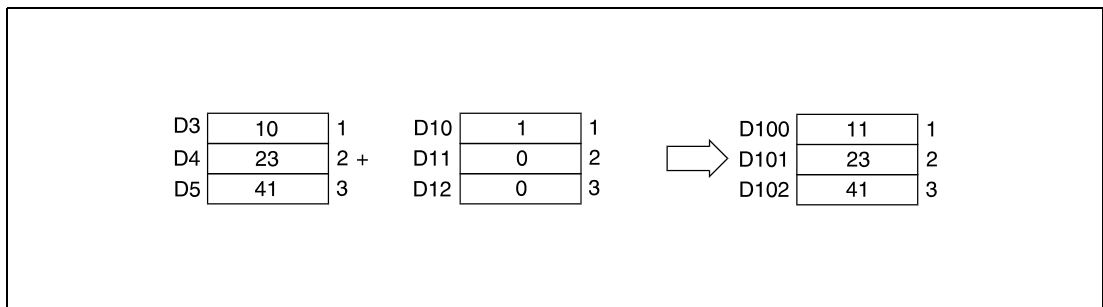
*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

The diagram below illustrates reading clock data via the DATERDP instruction.



- <sup>1</sup> QnA CPU clock
- <sup>2</sup> Year
- <sup>3</sup> May (January = 1, December = 12)
- <sup>4</sup> Day
- <sup>5</sup> Hour (24-hour format)
- <sup>6</sup> Minute
- <sup>7</sup> Second
- <sup>8</sup> Day of the week
- <sup>9</sup> Clock data

The diagram below illustrates the addition via the DATE+P instruction.



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

7.15.4 DATE-, DATE-P

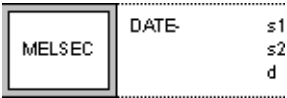
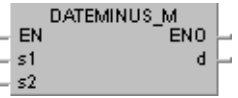
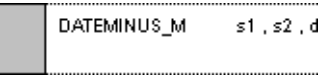
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

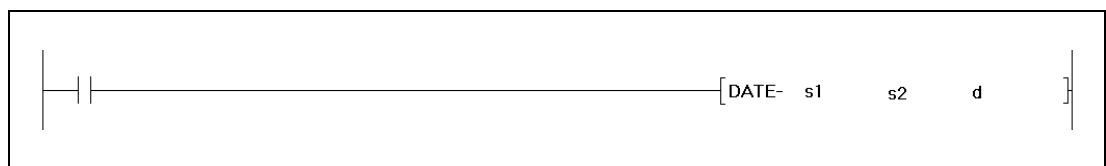
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other DY
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	SM0	4	
s2	—	●	●	—	—	—	—	—			
d	—	●	●	—	—	—	—	—			

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	--	---

GX Developer



Variables

Set Data	Meaning	Data Type
s1	First number of device storing clock data to be subtracted from.	BIN 16-bit
s2	First number of device storing clock data to be subtracted.	
d	First number of device storing the clock data of the subtraction result.	



**Functions Subtracting clock data**

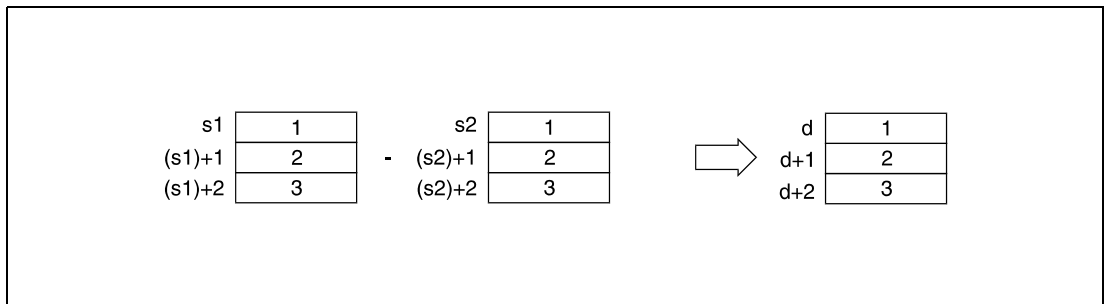
**DATE- Subtraction instruction**

The DATE instruction subtracts clock data stored in the device specified from s2 on from the clock data in the device specified from s1 on. The clock data of the operation result is stored in the device specified from d on.

The following table shows the input ranges of clock data stored in (s1)+0 through (s1)+2, (s2)+0 through (s2)+2 and d+0 through d+2.

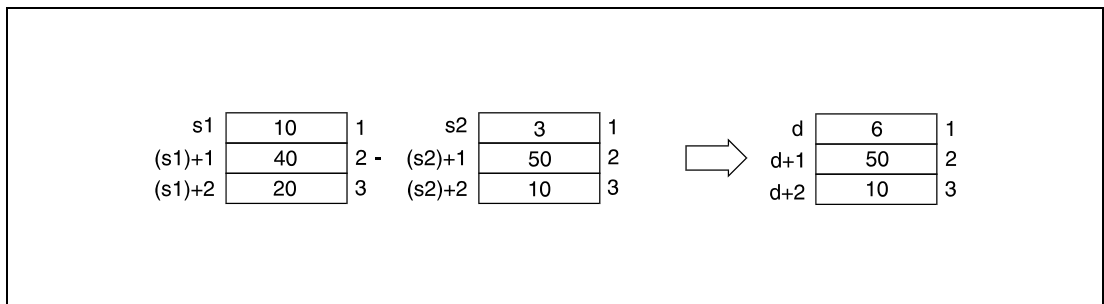
Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Input range	—	—	—	0 - 23	0 - 59	0 - 59	—

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	—	—	—	s1+0 (array_s1[0])	s1+1 (array_s1[1])	s1+2 (array_s1[2])	—
Devices	—	—	—	s2+0 (array_s2[0])	s2+1 (array_s2[1])	s2+2 (array_s2[2])	—
Devices	—	—	—	d+0 (array_d[0])	d+1 (array_d[1])	d+2 (array_d[2])	—



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

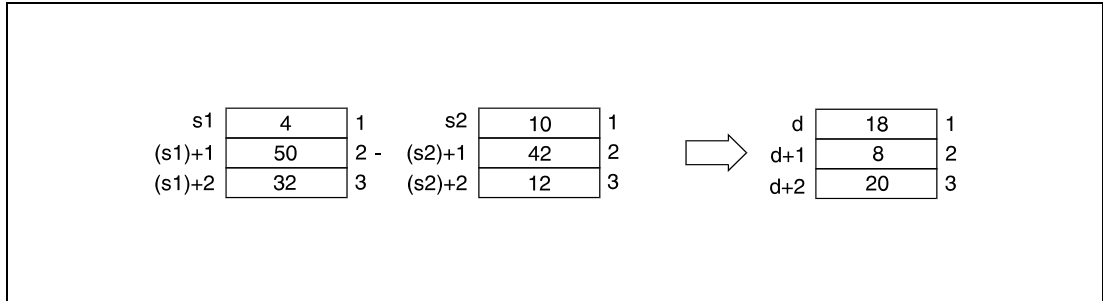
The following diagram illustrates the subtraction of 3 hours, 50 minutes, and 10 seconds ((s2)+0 - (s2)+2) from 10 hours, 40 minutes, and 20 ((s1)+0 - (s1)+2). The result, 6 hours, 50 minutes, and 10 seconds is stored in d+0 through d+2.



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

If the subtraction result of clock data becomes negative, 24 hours are added automatically to achieve a correct time value.

The following diagram illustrates the subtraction of 10 hours, 42 minutes, and 12 seconds from 4 hours, 50 minutes, and 32 seconds. The result would be -6 hours, 8 minutes, and 20 seconds. Since this result is not a correct time format, after the addition of 24 hours, the correct result is 18 hours, 8 minutes, and 20 seconds (18:08:20 the day before).



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

**NOTE** Refer to section "Writing clock data" for further information on that topic.

**Operation Errors** In the following cases an operation error occurs and the error flag is set:

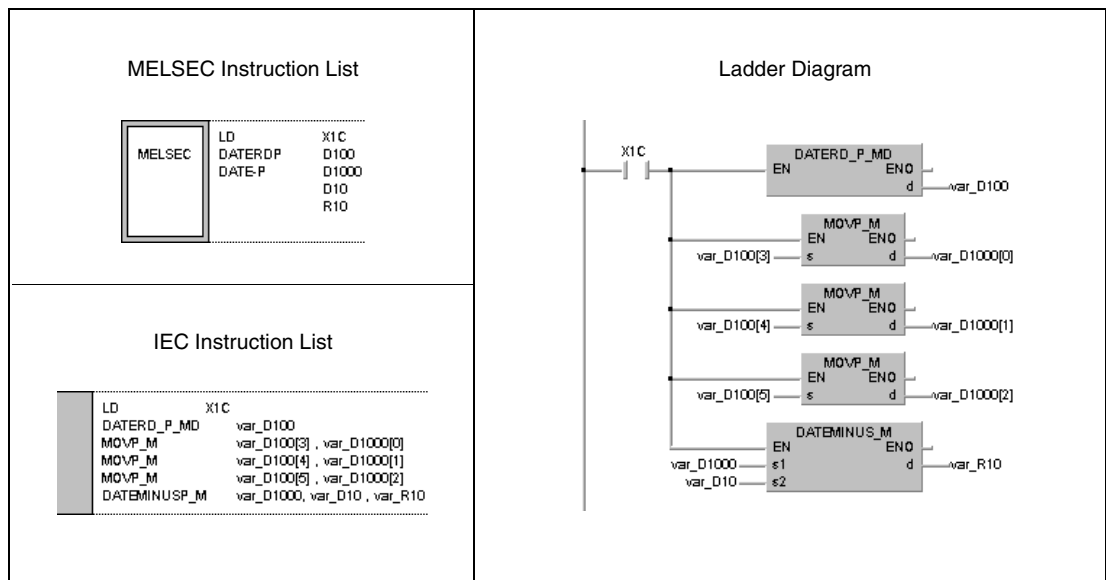
- The clock data in (s1)+0 through (s1)+2 and (s2)+0 through (s2)+2 exceed the input range.

**Program Example**

**DATE-P**

With leading edge from X1C, the following program reads the clock data from the internal CPU clock via the DATERDP instruction and stores it in the registers D100 through D106 (see diagrams below). The DATE-P instruction subtracts 10 hours (D10), 40 minutes (D11) and 10 seconds (D12) from the read data. The negative subtraction result, -8 hours, 41 minutes and 10 seconds is added 24 hours. The correct result, 16 hours, 41 minutes and 10 seconds (16:41:10 the day before) is stored in R10 through R12 (see diagrams below).

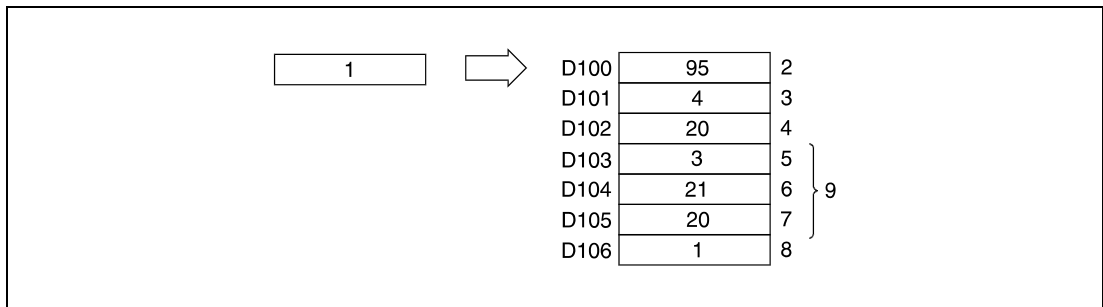
Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	D100 (var_D100[0])	D101 (var_D100[1])	D102 (var_D100[2])	D103 (var_D100[3])	D104 (var_D100[4])	D105 (var_D100[5])	D106 (var_D100[6])
Devices	—	—	—	D1000 (var_D1000[0])	D1001 (var_D1000[1])	D1002 (var_D1000[2])	—
Devices	—	—	—	D10 (var_D10[0])	D11 (var_D10[1])	D12 (var_D10[2])	—
Devices	—	—	—	R10 (var_R10[0])	R11 (var_R10[1])	R12 (var_R10[2])	—



**NOTE**

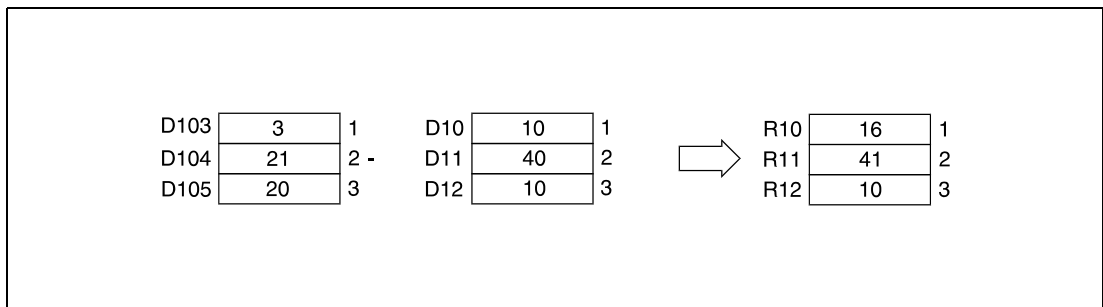
*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

The diagram below illustrates reading clock data via the DATERDP instruction.



- <sup>1</sup> QnA CPU clock
- <sup>2</sup> Year
- <sup>3</sup> May (January = 1, December = 12)
- <sup>4</sup> Day
- <sup>5</sup> Hour (24-hour format)
- <sup>6</sup> Minute
- <sup>7</sup> Second
- <sup>8</sup> Day of the week
- <sup>9</sup> Clock data

The diagram below illustrates the subtraction via the DATE-P instruction.



- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second

**7.15.5 SECOND, SECONDP, HOUR, HOURP**

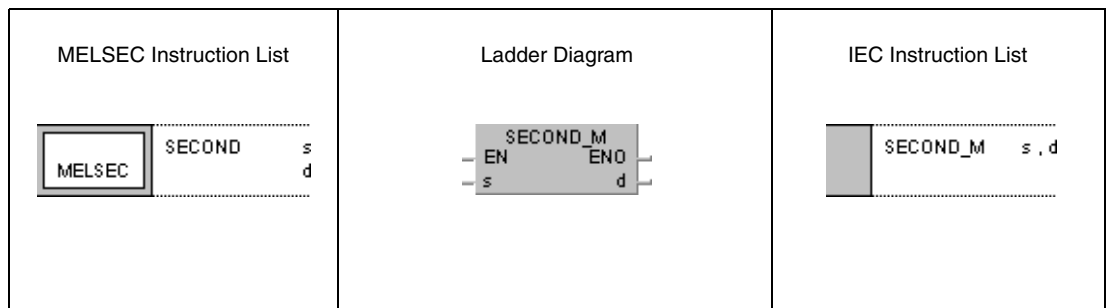
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

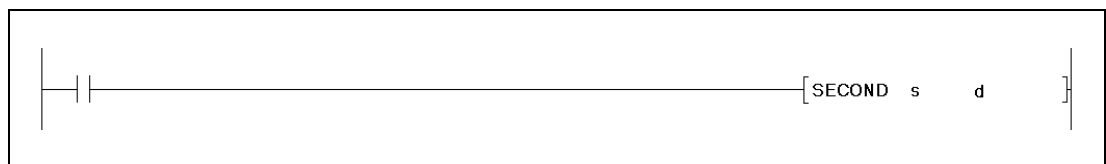
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
SECOND											
s	—	●	●	—	—	—	—	—	—	SM0	3
d	●	●	●	●	●	●	●	—	—		
HOUR											
s	●	●	●	●	●	●	●	●	—	SM0	3
d	—	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
SECOND			
s	Hours, minutes, seconds	BIN 16-/32-bit	Array [0..2] of ANY16
d	Seconds		ANY32
HOUR			
s	Seconds	BIN 16-/32-bit	ANY32
d	Hours, minutes, seconds		Array [0..2] of ANY16

**Functions Changing the clock data format**

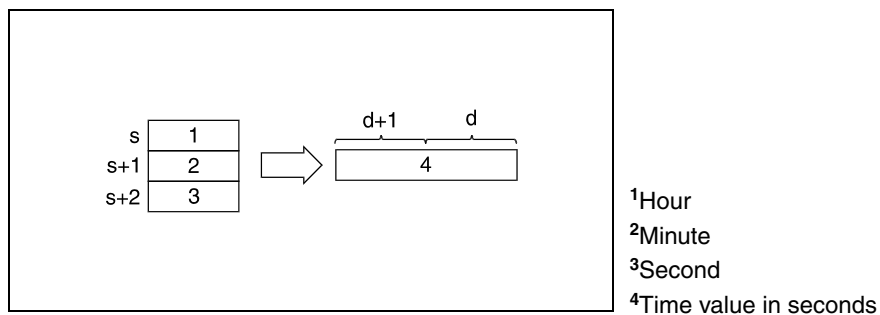
**SECOND Changing time format from hh:mm:ss to seconds**

The SECOND instruction changes the clock data in the devices s+0 through s+2 from the time format hh:mm:ss to the format seconds only. The result is stored in the devices specified by d and d+1.

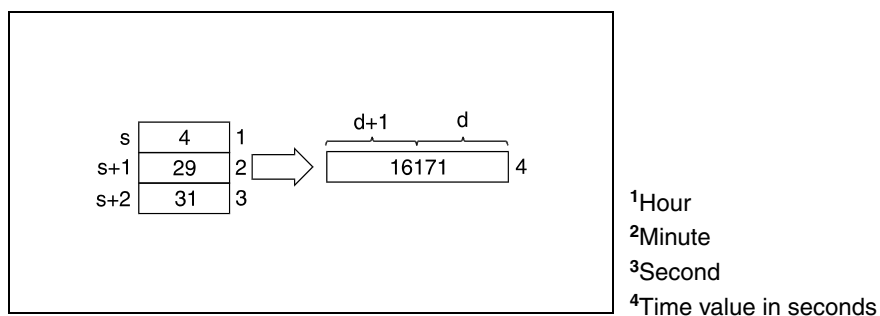
The following table shows the input ranges of clock data stored in s+0 through s+2:

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Input range	—	—	—	0 - 23	0 - 59	0 - 59	—

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	—	—	—	s+0 (array_s[0])	s+1 (array_s[1])	s+2 (array_s[2])	—
Devices	—	—	—	—	—	d+0 (array_d[0]) through d+1 (array_d[1])	—



The following diagram shows the conversion of 4 hours, 29 minutes, and 31 seconds into 16171 seconds.



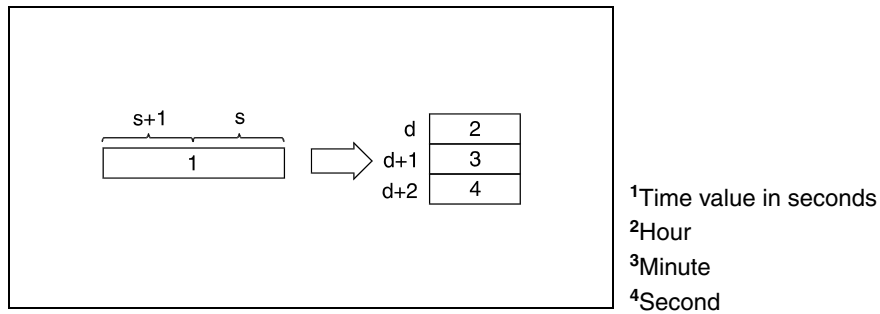
**HOUR Changing time format from seconds to hh:mm:ss**

The HOUR instruction changes the clock data in the devices s+0 through s+1 from the time format seconds only to the format hh:mm:ss.

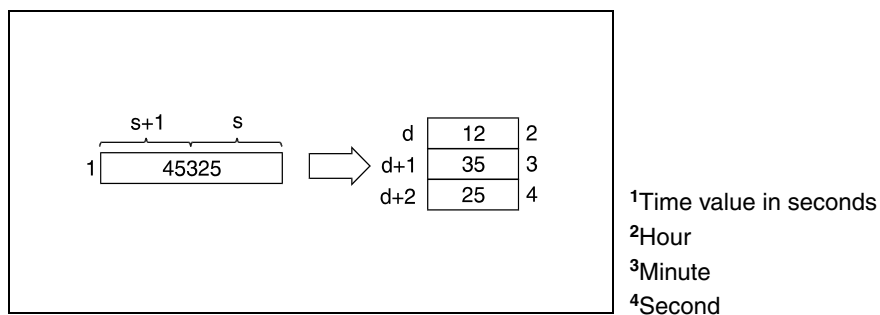
The following table shows the input ranges of clock data to be stored in d+0 through d+2:

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Input range	—	—	—	0 - 23	0 - 59	0 - 59	—

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	—	—	—	d+0 (array_d[0])	d+1 (array_d[1])	d+2 (array_d[2])	—
Devices	—	—	—	—	—	s+0 (array_s[0]) through s+1 (array_s[1])	—



The following diagram shows the conversion of 45325 seconds into 12 hours, 35 minutes, and 25 seconds.



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

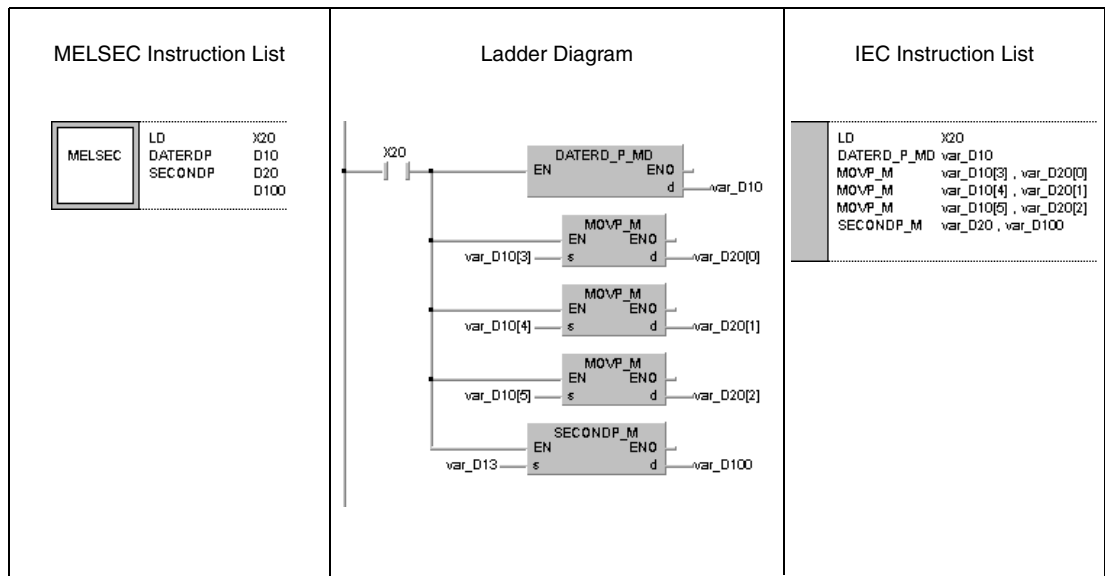
- The clock data in s+0 (array\_s[0]) through s+2 (array\_s[2]) for the SECOND instruction or in s+0 and s+1 for the HOUR instruction exceed the input range (error code 4100).

**Program Example 1**

**SECONDP**

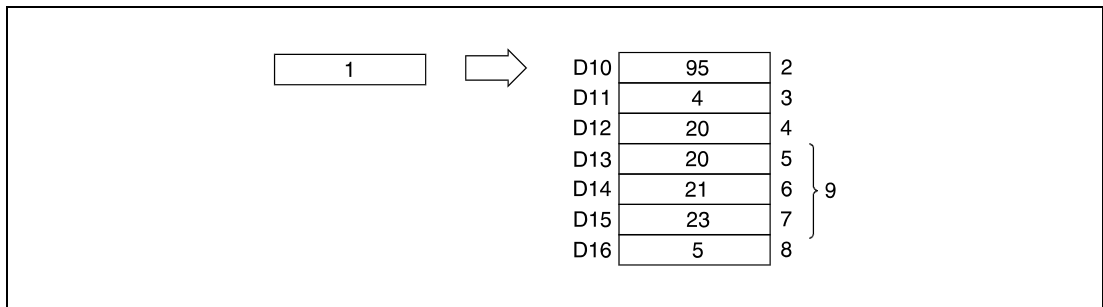
With leading edge from X20, the following program reads clock data from the internal CPU clock via the DATERDP instruction and stores it in the registers D10 through D16 (see diagrams below). The hours D20, minutes D21, and seconds D22 of clock data are converted into seconds only via the SECONDP instruction. The result is stored in D100 and D101 (see diagrams below).

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	D10 (var_D10[0])	D11 (var_D10[1])	D12 (var_D10[2])	D13 (var_D10[3])	D14 (var_D10[4])	D15 (var_D10[5])	D16 (var_D10[6])
Devices	—	—	—	D20 (var_D20[0])	D21 (var_D20[1])	D22 (var_D20[2])	—
Devices	—	—	—	—	—	D100 (var_D10[0]) bis D101 (var_D10[1])	—



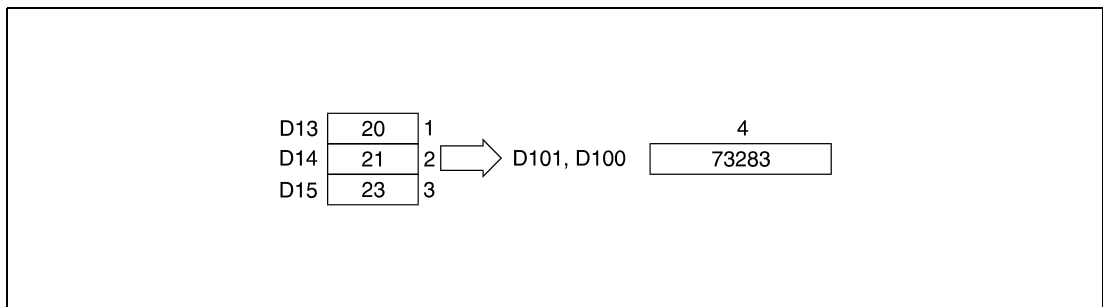


The diagram below illustrates reading clock data via the DATERDP instruction.



- <sup>1</sup> QnA CPU clock
- <sup>2</sup> Year
- <sup>3</sup> May (January = 1, December = 12)
- <sup>4</sup> Day
- <sup>5</sup> Hour (24-hour format)
- <sup>6</sup> Minute
- <sup>7</sup> Second
- <sup>8</sup> Day of the week
- <sup>9</sup> Clock data

The diagram below illustrates the conversion into seconds via the SECONDP instruction.

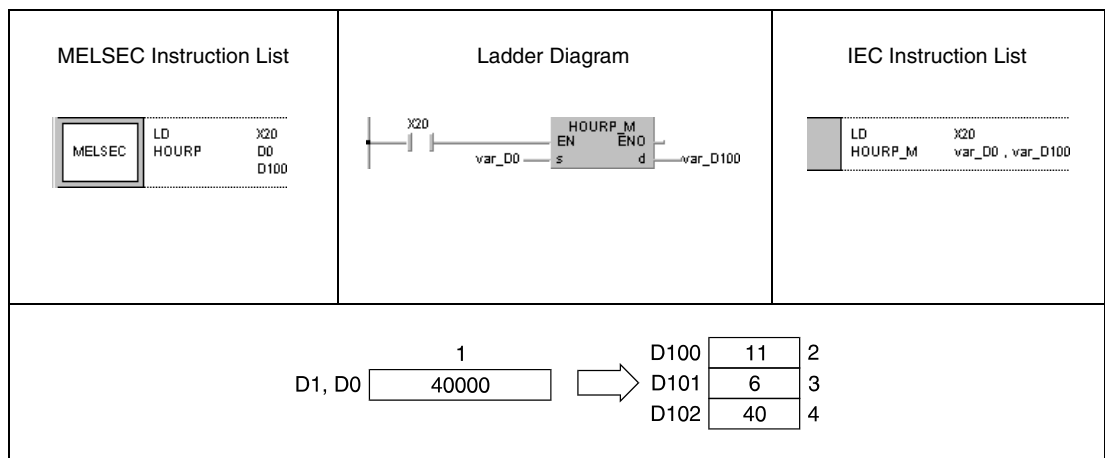


- <sup>1</sup> Hour
- <sup>2</sup> Minute
- <sup>3</sup> Second
- <sup>4</sup> Converted seconds

**Program Example 2** HOURP

With leading edge from X20, the following program converts the seconds stored in D0 and D1 into hours, minutes, and seconds. The result is stored in the devices in brackets.

Clock Data	Year	Month	Day	Hour	Minute	Second	Day of the week
Devices	—	—	—	D0 (var_D0[1])	D1 (var_D0[2])	D2 (var_D0[3])	—
Devices	—	—	—	—	—	D100 (var_D100[0]) bis D101 (var_D100[1])	—



- <sup>1</sup> Value to be converted into seconds
- <sup>2</sup> Hour
- <sup>3</sup> Minute
- <sup>4</sup> Second

**NOTE** *These program examples will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 7.16 Peripheral device instructions

The peripheral device instructions support the output of messages to peripheral devices and the input of data through keyboards at peripheral devices.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Output of messages to peripheral devices	MSG	MSG_M
Key input of data from peripheral devices	PKEY	PKEY_M

7.16.1 MSG

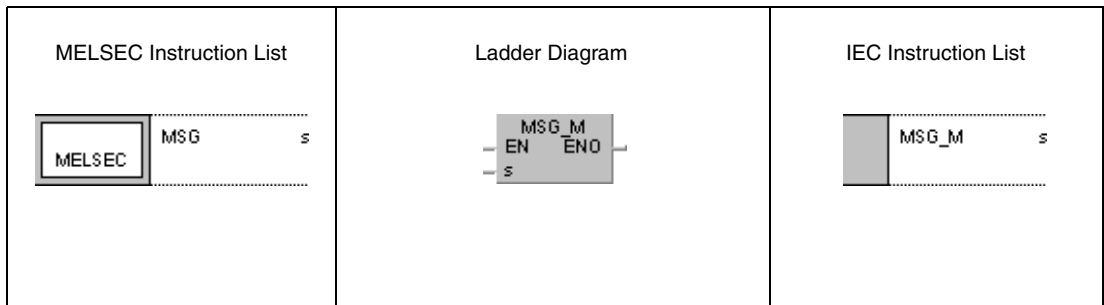
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	2	

GX IEC Developer



GX Developer

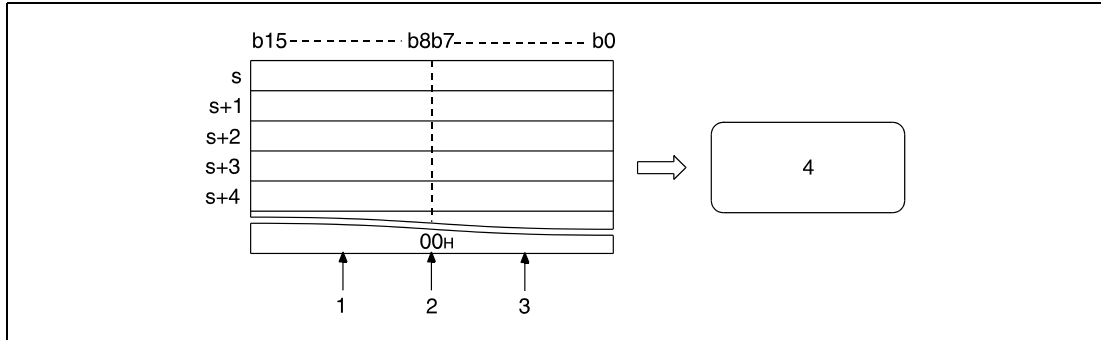


Variables

Set Data	Meaning	Data Type
s	Character string data to be displayed at the peripheral device or first number of device storing such data.	Character string

**Functions**    **Output of messages to peripheral devices****MSG**    **Output instruction**

The MSG instruction outputs a character string stored in a device specified from *s* to a peripheral device specified in terminal mode. The end of the character string is indicated by the character code "00H".



<sup>1</sup> 2., 4., ..., (n+1)-th character

<sup>2</sup> The character code "00H" indicates the end of the character string

<sup>3</sup> 1., 3., ..., nth character

<sup>4</sup> Display of the character strings (messages) at a peripheral device

Up to 64 characters can be displayed at the peripheral device.

The character string data in *s* is stored in the special registers SD738 through SD773 (storage area for messages).

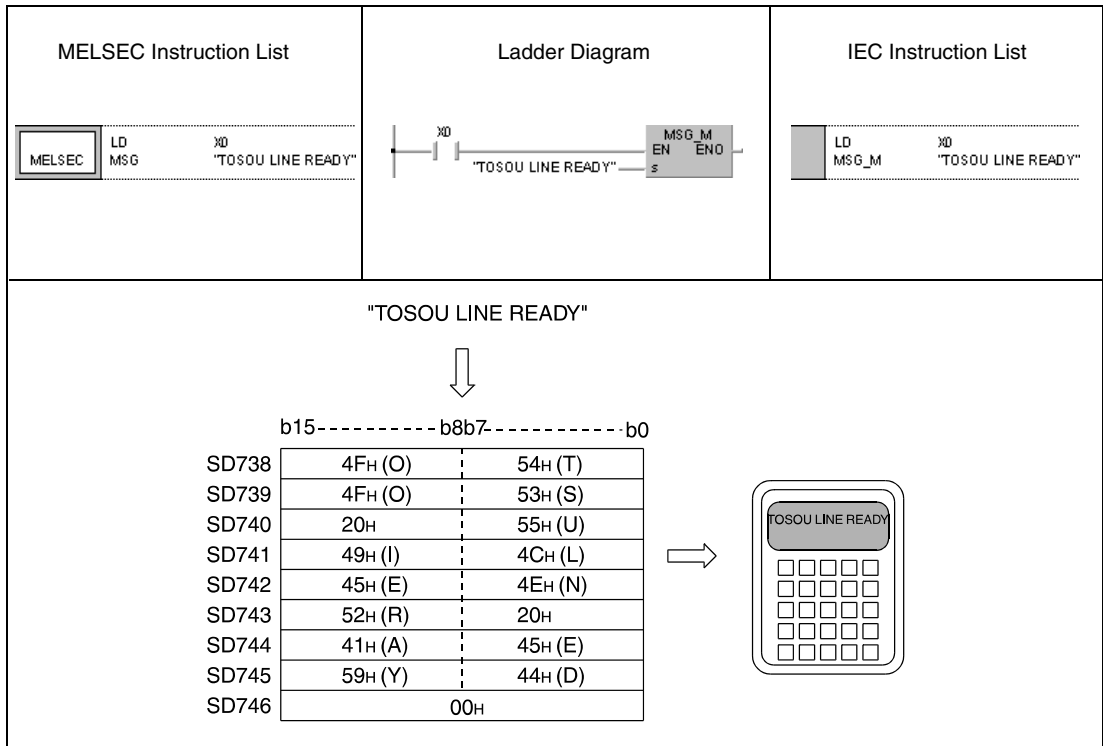
During execution of the MSG instruction the special relay SM738 (execution signal for the MSG instruction) is set. If the special relay SM738 is set, no other MSG instruction will be executed.

After completion of the MSG instruction, ie. after display of all characters, the special register SM738 is reset and the contents (character string) of the special registers SD738 through SD773 are cleared (overwritten by the character code "00H").

**Program Example**

MSG

If X0 is set, the following program outputs and displays the character string "TOSOU LINE READY" as message to the display of a peripheral device.



**7.16.2 PKEY**

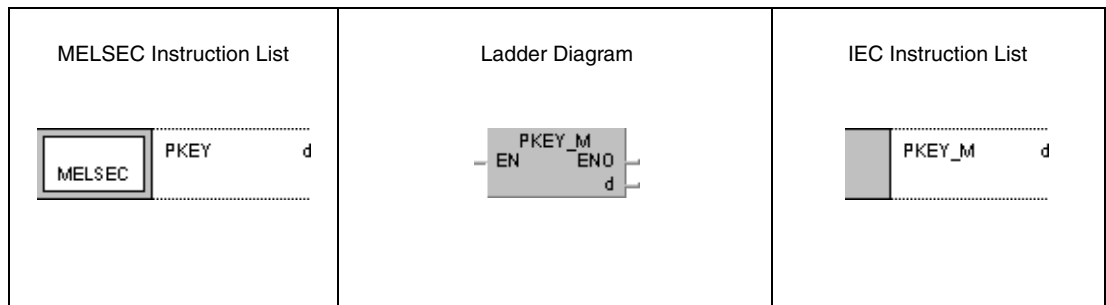
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

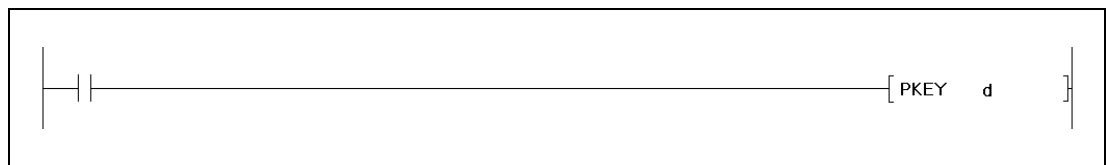
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other U
	Bit	Word		Bit	Word						
d	—	●	●	—	—	—	—	●	—	SM0	2

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
d	First number of device storing the input character string.	BIN 16-bit

**Functions**     **Key input of data at peripheral devices**

**PKEY    Input instruction**

The PKEY instruction clears the data words in the devices specified in d+0 through d+17 and sets the special relay SM736 (execution signal for the PKEY instruction). In addition, the special relay SM737 (key input reception flag) is set. After completion of the PKEY instruction, the key input data (characters) are read from the peripheral device specified in terminal mode and written in ASCII format to the devices specified in d+0 through d+17.

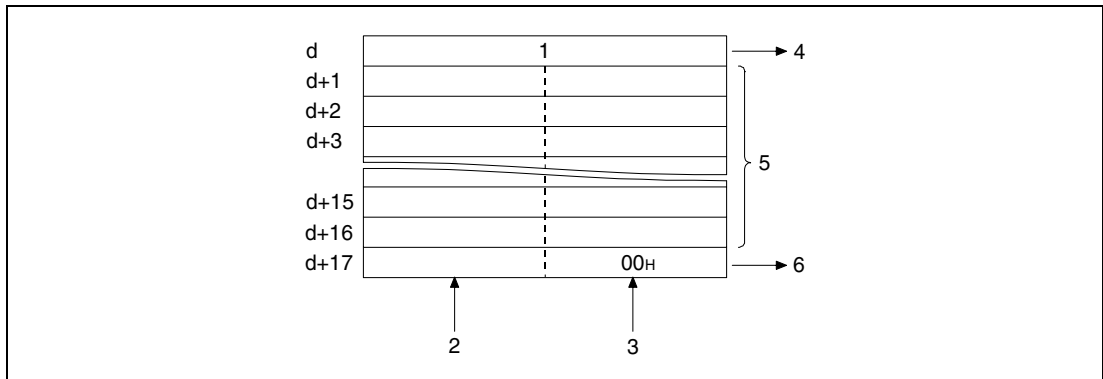
Resetting the execution condition for the PKEY instruction also resets the special relays SM736 and SM737.

The special relay SM737 is set, if a character entered via the keyboard is received by the peripheral device, and reset, if the CPU stores the key input. While the special relay SM737 is set, key input data cannot be received by the peripheral device.

The key input at the peripheral device is completed, when it receives the character string "CR".

In total, 32 characters can be entered. After the input of 32 characters, the reception of key input data is terminated by the peripheral device, without having received the character string "CR".

The storage of key input data (characters) in the devices specified in d+1 through d+17 is illustrated below:



- <sup>1</sup> Counter
- <sup>2</sup> 2nd to 32nd character
- <sup>3</sup> 1st to 31st character
- <sup>4</sup> Number of characters entered (binary data value)
- <sup>5</sup> Maximum 16 characters
- <sup>6</sup> The character code "00H" indicates the end of the entered character string (number of entered characters: odd = upper byte, even = lower byte)

The PKEY instruction cannot be executed from more than one location at the same time. If key input is intended from more than one location, an interlock has to be established via the special relay SM736 (execution signal of the PKEY instruction) to prevent simultaneous input.

**Operation Errors**

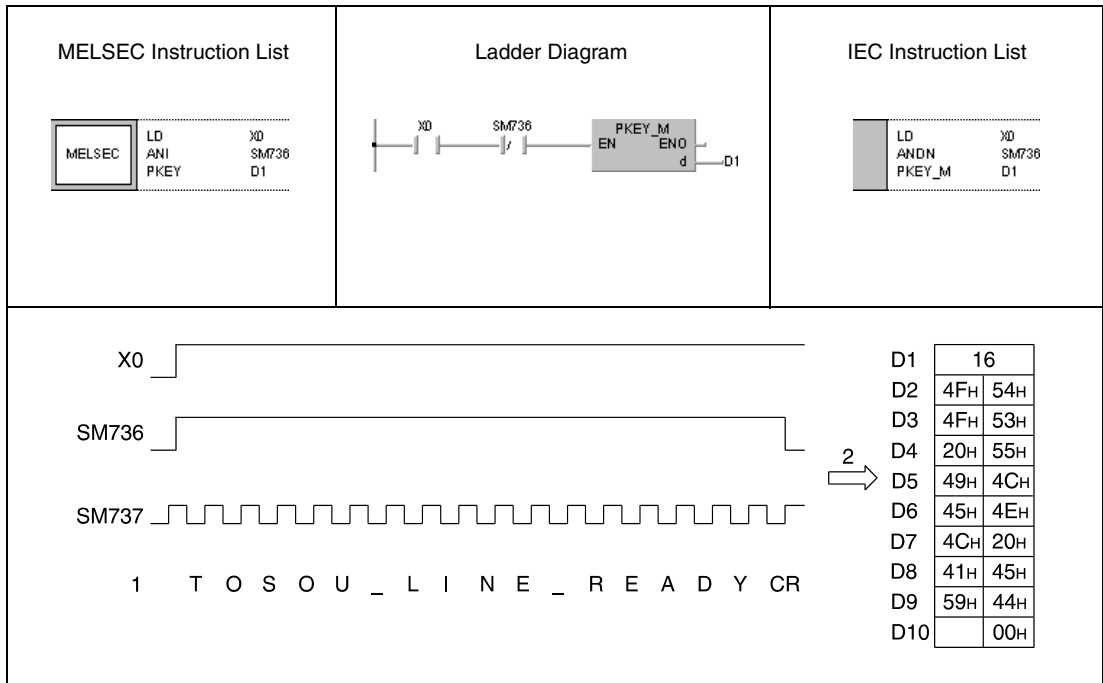
In the following cases an operation error occurs and the error flag is set:

- The entered key input data exceeds the relevant storage device range of the devices specified in d+0 through d+17 (error code 4101).



**Program Example** PKEY

If X0 is set, the following program stores the character string "TOSOU LINE READY" entered into the peripheral device via keyboard in the registers D1 through D10.



<sup>1</sup> Key input data

<sup>2</sup> Storage of entered data

## 7.17 Program control instructions

The program control instructions toggle different program operation modes. The table below gives an overview of the instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Switching programs into stand-by mode	PSTOP	PSTOP_M
	PSTOPP	PSTOPP_M
Switching programs into stand-by mode and reset of outputs	POFF	POFF_M
	POFFP	POFFP_M
Switching programs into scan execution mode	PSCAN	PSCAN_M
	PSCANP	PSCANP_M
Switching programs into low-speed execution mode	PLOW	PLOW_M
	PLOWP	PLOWP_M

### NOTE

*Please check, whether these functions are available and supported by your version of the GX IEC Developer.*

**7.17.1 PSTOP, PSTOPP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

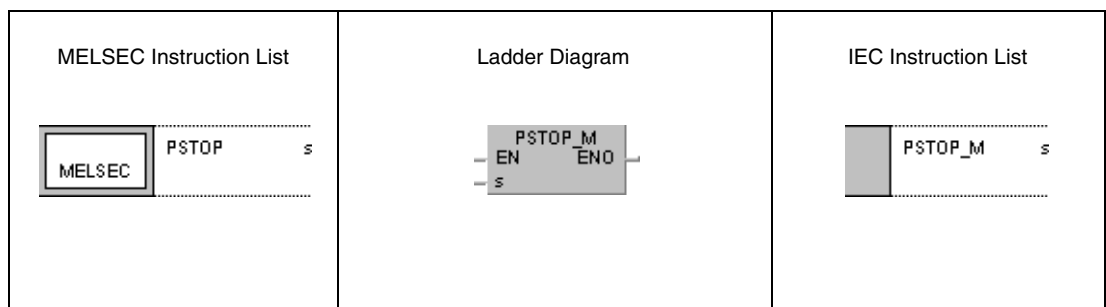
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

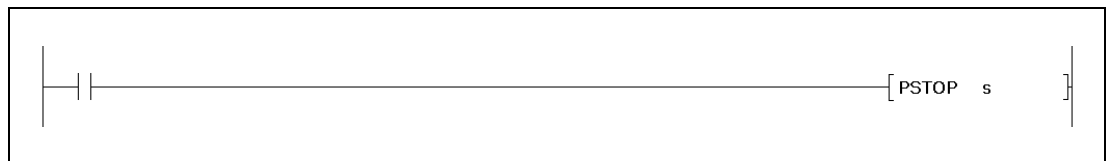
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	2 + n <sup>1)</sup>

<sup>1</sup> n = (number of program name characters)/2 = Number of additional steps (Decimal fractions are rounded up)

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
s	File name of program file to be set into stand-by mode or first number of device storing such data.	Character string

**Functions**      **Setting a program into the stand-by mode**

**PSTOP Switch instruction for the stand-by mode**

The PSTOP instruction sets the program specified by the device in s into the stand-by mode. In this mode the program is only executed if requested.

Only program files stored in the internal memory (drive 0) can be set into the stand-by mode.

The stand-by mode is only entered after END processing.

The PSTOP instruction is even given priority if the execution mode is specified via parameters.

The file extension .QPG is not needed to be entered for file specification since the type of file is recognized automatically.

**Operation Errors**

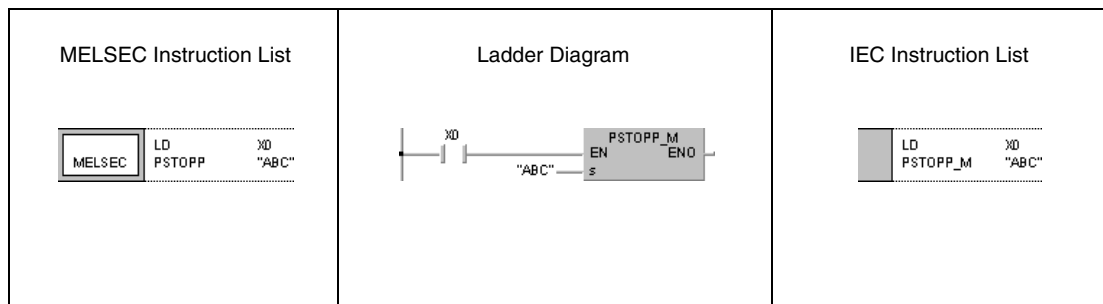
In the following cases an operation error occurs and the error flag is set:

- The specified program file does not exist (error code 2410).

**Program Example**

**PSTOPP**

With leading edge from X0, the following program sets a program named "ABC" into the stand-by mode.



**7.17.2 POFF, POFFP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

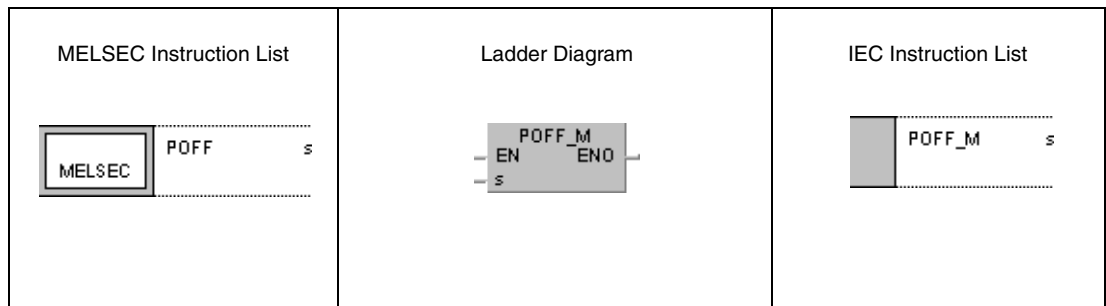
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	2 + n <sup>1)</sup>

<sup>1</sup> n = (number of program name characters)/2 = Number of additional steps (Decimal fractions are rounded up)

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	First number of device storing file name of program file to be set into stand-by mode including reset of outputs.	Character string

**Functions**      **Setting a program into the stand-by mode including reset of the outputs**

**POFF**      **Switch instruction for the stand-by mode with reset outputs**

The POFF instruction sets the program specified by the device in s into the stand-by mode and resets the outputs addressed by the program. First in this mode all outputs, addressed by the program are reset to the same status as if the execution conditions for the instructions addressing them were not set. Then the program enters the stand-by mode.

Only program files stored in the internal memory (drive 0) can be set into the stand-by mode.

The stand-by mode is only entered after END processing.

The POFF instruction is even given priority if the execution mode is specified via parameters.

The file extension .QPG is not needed to be entered for file specification since the type of file is recognized automatically.

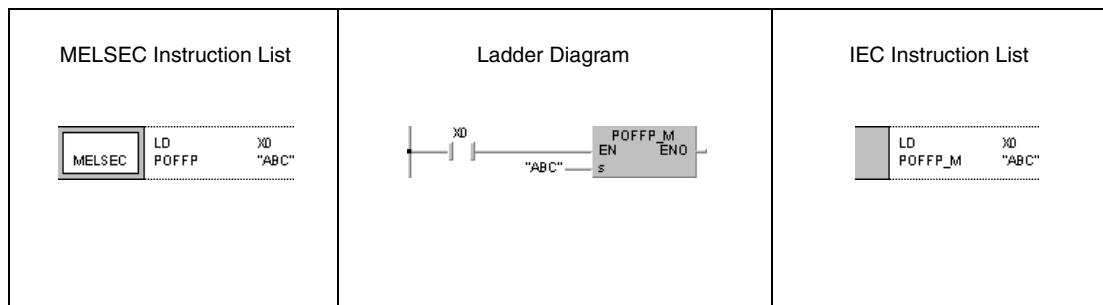
**NOTE**      *On execution of the POFF instruction the coils addressed by an OUT instruction are reset (see functions).*

**Operation Errors**      In the following cases an operation error occurs and the error flag is set:

- The specified program file does not exist (error code 2410).

**Program Example**      **POFFP**

With leading edge from X0, the following program sets a program named "ABC" into the stand-by mode. First in this mode all outputs, addressed by the program "ABC" are reset to the same status as if the execution conditions for the instructions addressing them were not set. Then the program "ABC" enters the stand-by mode.



**7.17.3 PSCAN, PSCANP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

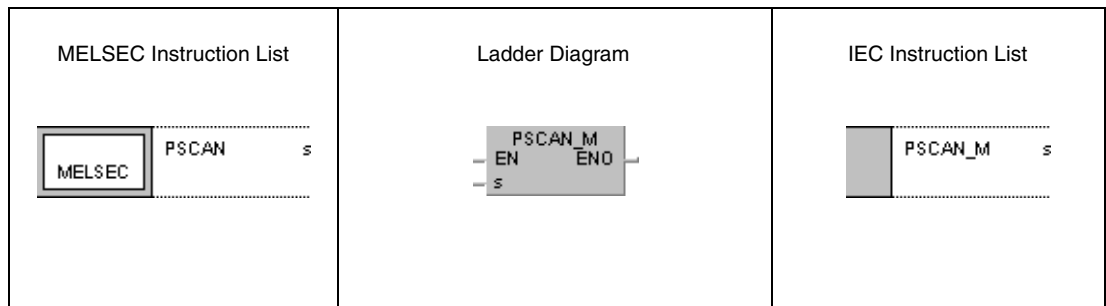
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

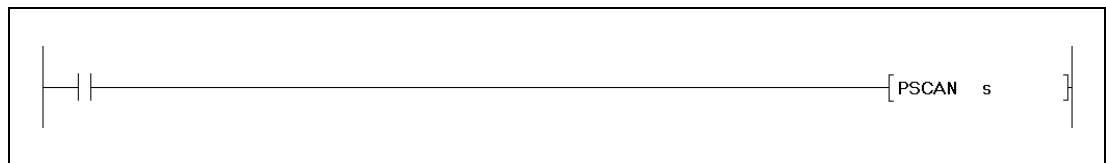
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other DY
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	2 + n <sup>1)</sup>

<sup>1</sup> n = (number of program name characters)/2 = Number of additional steps (Decimal fractions are rounded up)

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	File name of program file to be set into scan execution mode or first number of device storing such data.	Character string

**Functions**      **Setting a program into the scan execution mode**

**PSCAN Switch instruction for the scan execution mode**

The PSCAN instruction sets the program specified by the device in s into the scan execution mode. In this mode the program is only executed once during one program scan.

Only program files stored in the internal memory (drive 0) can be set into the scan execution mode.

The scan execution mode is only entered after END processing.

The PSCAN instruction is even given priority if the execution mode is specified via parameters.

The file extension .QPG is not needed to be entered for file specification since the type of file is recognized automatically.

**Operation Errors**

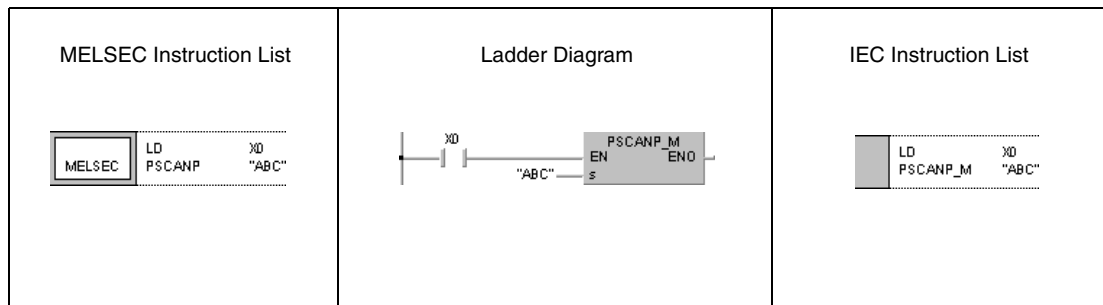
In the following cases an operation error occurs and the error flag is set:

- The specified program file does not exist (error code 2410).

**Program Example**

**PSCANP**

With leading edge from X0, the following program sets a program named "ABC" into the scan execution mode.





**7.17.4 PLOW, PLOWP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

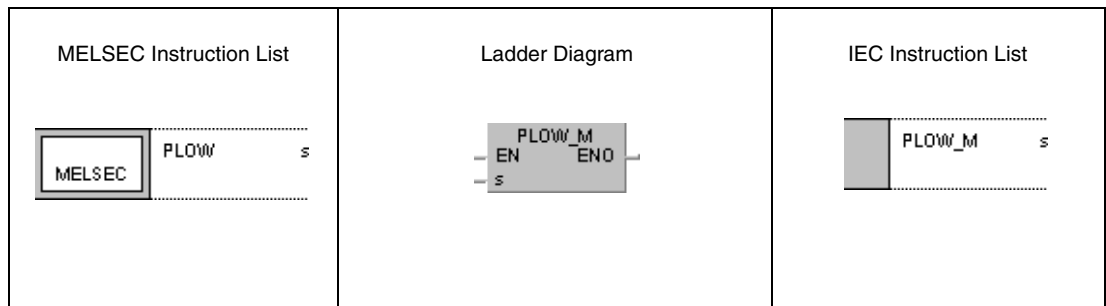
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant \$			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	●	—	SM0	2 + n <sup>1)</sup>

<sup>1</sup> n = (number of program name characters)/2 = Number of additional steps (Decimal fractions are rounded up)

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type
s	File name of program file to be set into low-speed execution mode or first number of device storing such data.	Character string

**Functions**      **Setting a program into the low-speed execution mode**

**PLOW**    **Switch instruction for the low-speed execution mode**

The PLOW instruction sets the program specified by the device in s into the low-speed execution mode. In this mode the program is only executed at low processing speed.

Only program files stored in the internal memory (drive 0) can be set into the scan execution mode.

The low-speed execution mode is only entered after END processing.

The PLOW instruction is even given priority if the execution mode is specified via parameters.

The file extension .QPG is not needed to be entered for file specification since the type of file is recognized automatically.

**Operation Errors**

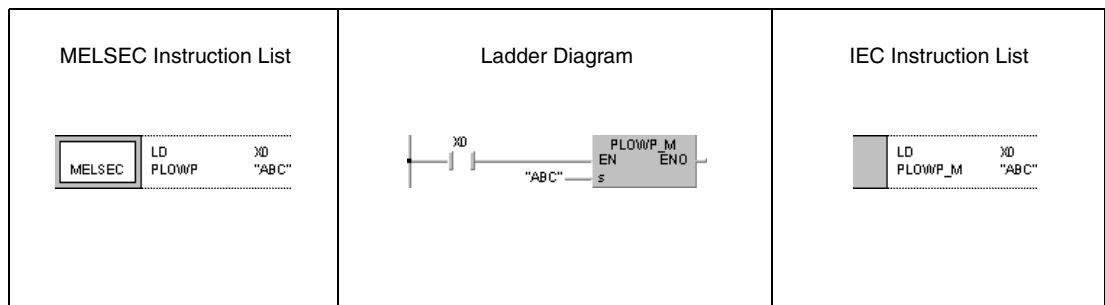
In the following cases an operation error occurs and the error flag is set:

- The specified program file does not exist (error code 2410).
- The program file contains a CHK instruction (error code 4235).

**Program Example**

PLOWP

With leading edge from X0, the following program sets a program named "ABC" into the low-speed execution mode.



## 7.18 Other convenient instructions

This section contains miscellaneous instructions for setting and resetting WDT (watchdog timer) and carry flags, for settings of the number of program scans to be executed, for reading, writing, and entering of data from and to several memories. The table below gives an overview of the instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Reset watchdog timer	WDT	WDT_M
Set and reset carry flag	STC	STC_M
	CLC	CLC_M
Preset number of execution scans	DUTY	DUTY_M
Direct read of one byte	ZRRDB	ZRRDB_M
	ZRRDBP	ZRRDBP_M
Direct write of one byte	ZRWRB	ZRWRB_M
	ZRWRBP	ZRWRBP_M
Store device for indirect designation	ADRSET	ADRSET_M
	ADRSETP	ADRSETP_M
Numerical key input from keyboard	KEY	KEY_MD
Batch save of index register contents	ZPUSH	ZPUSH_M
	ZPUSHP	ZPUSHP_M
Batch recovery of index register contents	ZPOP	ZPOP_M
	ZPOPP	ZPOPP_M
Batch write of data to EEPROM register	EROMWR	EROMWR_M
	EROMWRP	EROMWRP_M

### NOTE

*The instructions ADRSET and ADRSETP are not supported by the GX IEC Developer.*

7.18.1 WDT, WDTP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

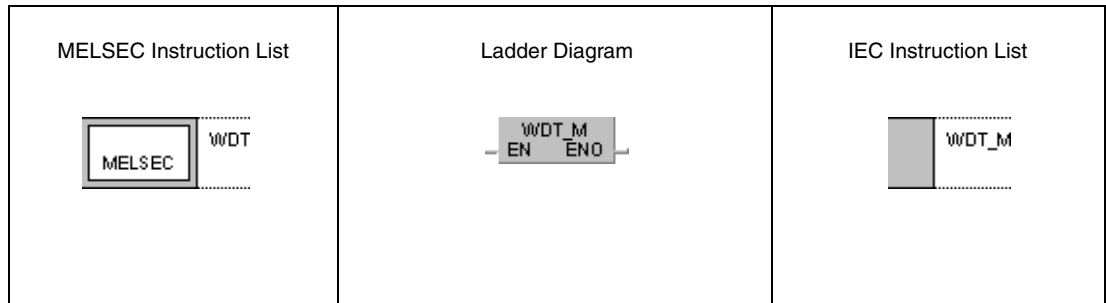
Usable Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag	
Bit Devices						Word Devices (16-bit)						Constant		Pointer		Level								
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N	M9012
																						● <sup>1</sup>		

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

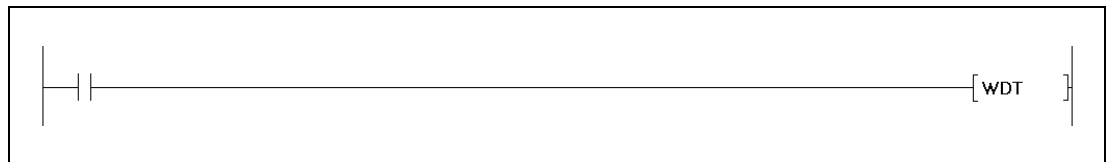
Devices  
MELSEC Q

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

GX IEC  
Developer



GX  
Developer



Variables

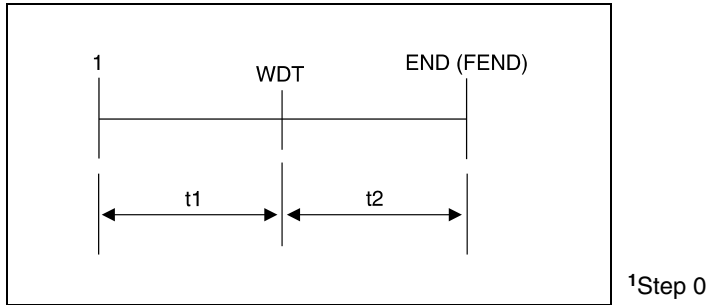
Set Data	Meaning	Data Type
—	—	—

**Functions**      **Resetting the watchdog timer**  
**WDT      Reset**

The WDT instruction resets the watchdog timer (WDT) during execution of a sequence program.

The WDT instruction is only needed, if the program scan time of a sequence program from program step 0 up to the END/FEND instruction exceeds the default time setting of the WDT under certain conditions. If the default time setting of the WDT is exceeded any program scan the parameter setting of the WDT has to be adjusted accordingly.

The setting value of the WDT has to be adjusted so that neither the time period t1 (step 0 to WDT instruction) nor t2 (WDT and END/FEND instructions) exceed the WDT setting value.



The WDT instruction can be set any number of times within one program scan. Nevertheless, for programming remind that the outputs are not reset (0) at once.

The values of the program scan time stored in the registers are not cleared via the WDT instruction. Therefore, the stored values may be greater than the WDT values set through parameters.

**NOTE**      *The following A series CPUs only supply read-only (fixed) values for watchdog timers: A3H, A3M, AnA, AnAS, and AnU*

7.18.2 STC, CLC

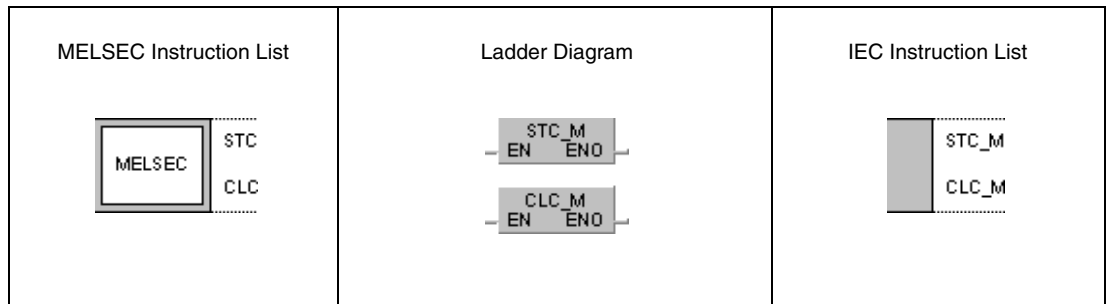
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

Devices  
MELSEC A

Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag				
Bit Devices							Word Devices (16-bit)							Constant		Pointer						Level			
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I	N	M9012	M9010 M9011
																							1		

GX IEC  
Developer



GX  
Developer



Variables

Set Data	Meaning	Data Type
—	—	—

**Functions**     **Setting and resetting the carry flag**

**STC     Set carry flag**

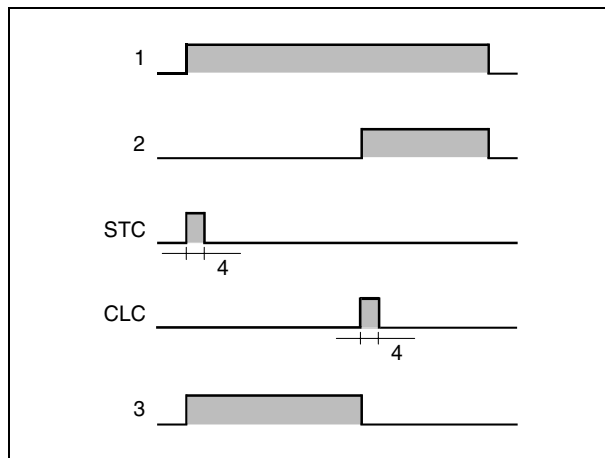
The carry flag stores the carry (0 or 1) of rotation and shift operations. The carry is represented in the program as a contact by the special relay M9012. M9012 is set, if the carry flag is 1, and not set, if the carry flag is 0.

On execution of the STC instruction the carry flag (M9012) is forced ON.

**CLC     Reset carry flag**

The carry flag is reset after the execution of the CLC instruction. At the same time the special relay M9012 is reset.

The STC/CLC instruction is executed once at leading edge from the input condition.

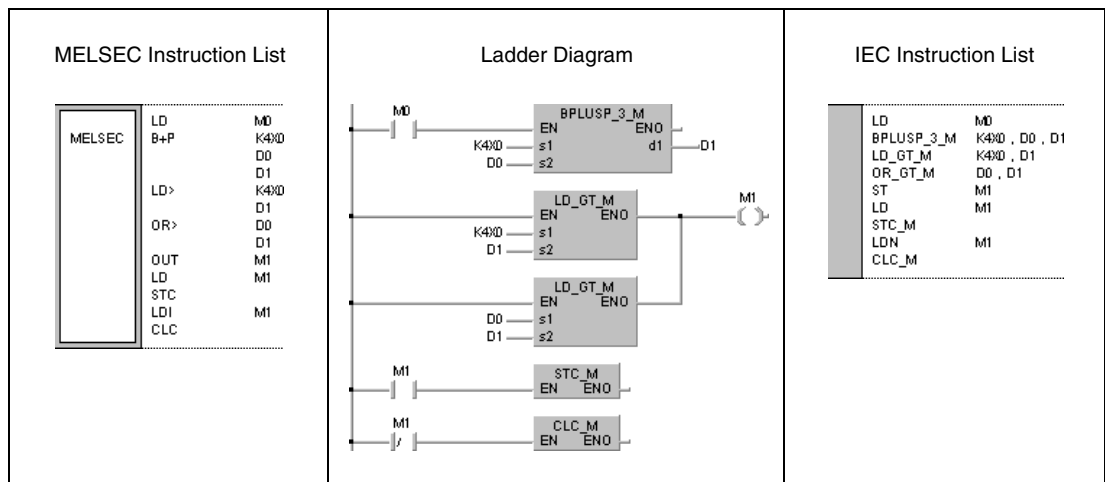


- <sup>1</sup>Execution condition of the STC instruction
- <sup>2</sup>Execution condition of the CLC instruction
- <sup>3</sup>Carry flag (M9012)
- <sup>4</sup>One execution

**Program Example**

**STC, CLC**

With leading edge from M0, the following program adds the BCD data at X0 through XF to the BCD data in D0. The result is stored in D1. If the result of the addition is greater than 9999, M1 is set and the STC instruction is executed (M9012 is set). If the result is less than or equal to 9999, the carry flag is not set.



7.18.3 DUTY

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	●

Devices  
MELSEC A

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
n1																●	●						
n2																●	●						●
d		●																					

<sup>1</sup> Refer to chapter "Programming an AnA, AnAS, and AnU CPU" in the Programming Manual for the according number of steps.

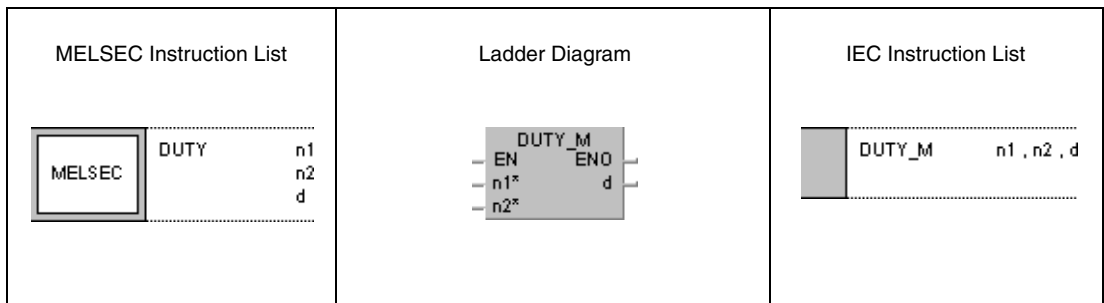
<sup>2</sup> Index qualification supported by A3H, A3M, AnA, AnAS and AnU CPU only.

Devices  
MELSEC Q

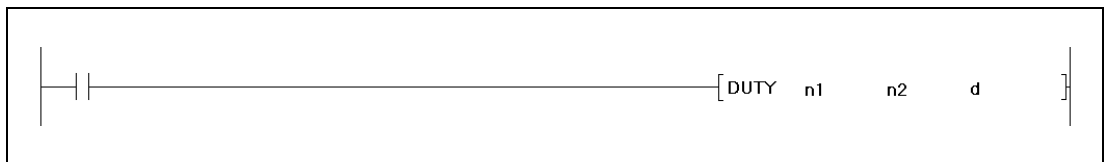
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
n1	●	●	●	●	●	●	●	●	●	SM0	4
n2	●	●	●	●	●	●	●	●	●		
d	● <sup>1</sup>	—	—	—	—	—	—	—	—		

<sup>1</sup> SM420 through SM424 and SM430 through SM434

GX IEC Developer



GX Developer



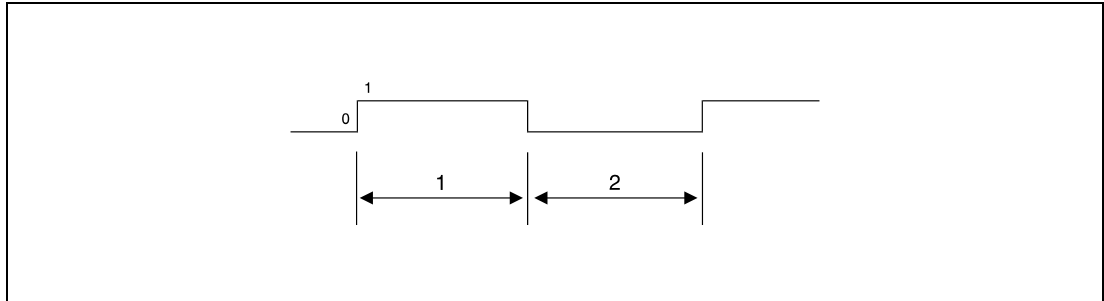
Variables

Set Data	Meaning	Data Type
n1	Number of scans the special relays are set.	BIN 16-bit
n2	Number of scans the special relays are reset.	
d	Address of special relay (A series = M9020 – M9024, Q series and System Q = SM420 – SM424 and SM430 – SM434).	Bit



**Functions Presetting the number of execution scans of a device****DUTY Preset execution scans**

The DUTY instruction turns the devices specified by d (A series = M9020 through M9024, Q series and System Q = SM420 through SM424 and SM430 through SM434) ON for the number of program scans specified by n1 and OFF for the number of program scans specified by n2. The according special relay serves as input condition for following operations.



<sup>1</sup> Number of program scans with execution

<sup>2</sup> Number of program scans without execution

Programs being executed once per program scan apply the relays SM420 through SM424 (Q series and System Q).

Low-speed execution programs apply the relays SM430 through SM434 (Q series and System Q).

At the beginning of the execution (initializing) the relays (A series = M9020 through M9024, Q series/System Q = SM420 through SM424 and SM430 through SM434) are reset.

If the value in n1 = 0, the relays remain reset.

If the value in n2 = 0 and the value in n1 is greater than 0, the relays will be and remain set.

The values in n1, n2, and d are set when the DUTY instruction is invoked. The scan pulse (relay) is set ON or OFF when the END instruction is reached.

**Operation Errors**

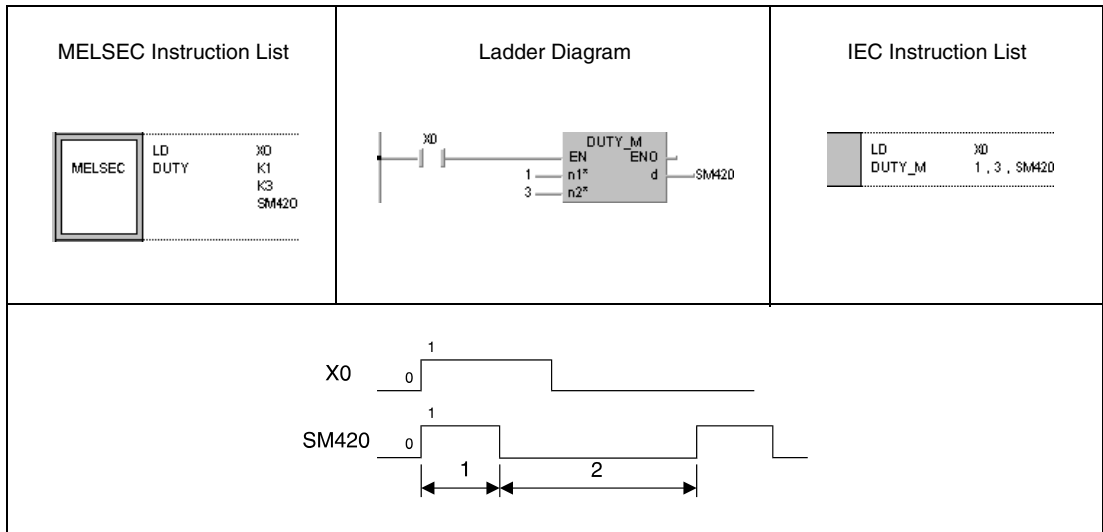
In the following cases an operation error occurs and the error flag is set:

- The device specified by d is no relevant relay of the A or Q series (error code 4101).
- The values in n1 and n2 are less than 0 (error code 4100).

**Program Example**

DUTY (Q series and System Q)

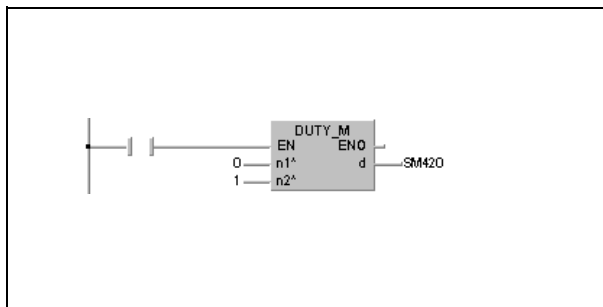
With leading edge from X0, the following program sets the special relay for one program scan and resets it for 3 program scans. This operations are repeated as long as the program is executed (see NOTE below).



- <sup>1</sup> One program scan with execution
- <sup>2</sup> Three program scans without execution

**NOTE**

After the execution condition is reset (X0 = OFF) the output of scan pulse of the DUTY instruction and the cyclic setting / resetting of the specified relay are proceeded. In order to stop the continued output of scan pulses the following program part has to be inserted.



**7.18.4 ZRRDB, ZRRDBP**


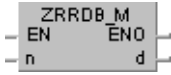
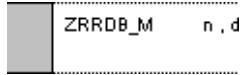
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

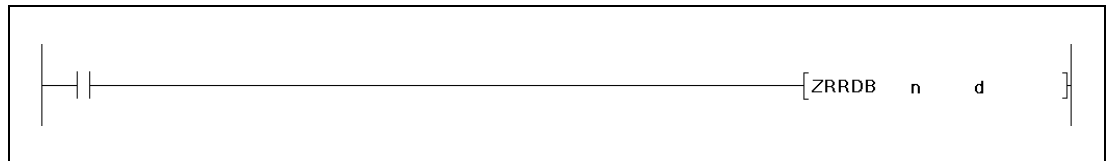
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
n	●	●	●	●	●	●	●	—	SM0	3	
d	●	●	●	●	●	●	—	—			

**GX IEC Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

**GX Developer**



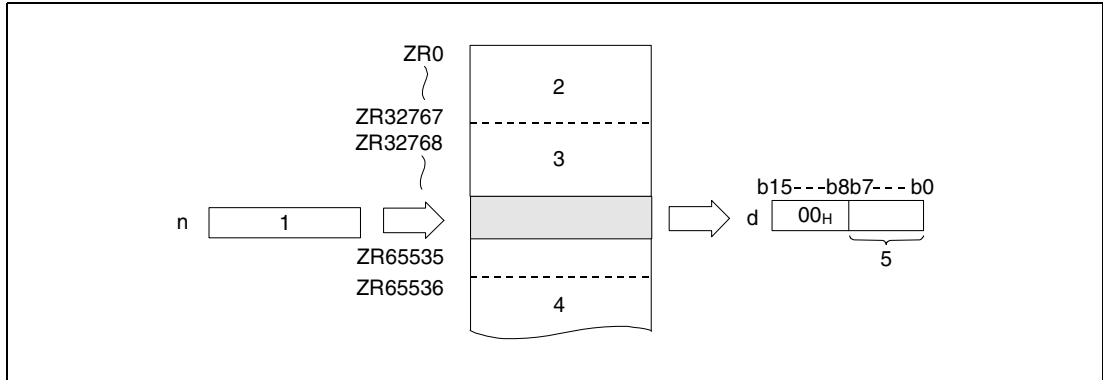
**Variables**

Set Data	Meaning	Data Type
n	Serial byte number for file register to be read.	BIN 32-bit
d	Number of device storing the read byte.	BIN 16-bit

**Functions Direct read of one byte from a file register**

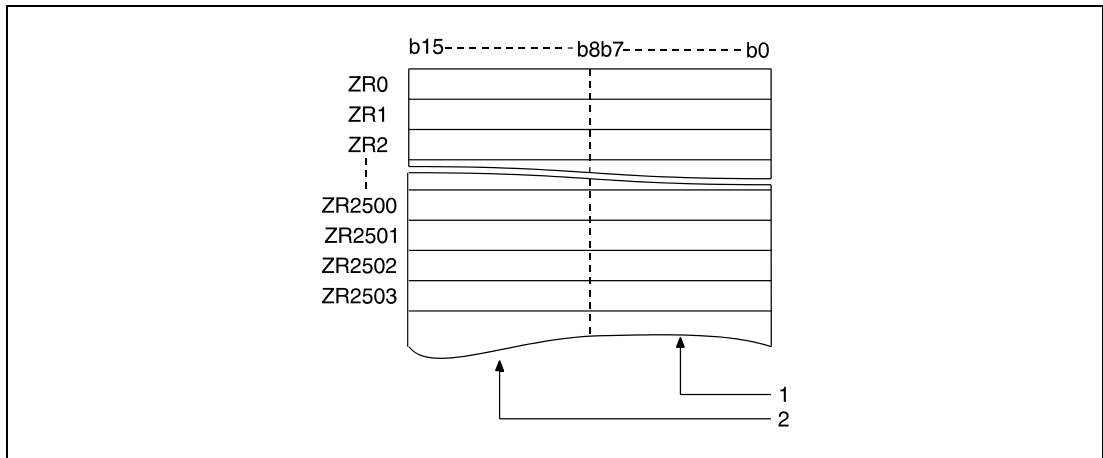
**ZRRDB Read one byte**

The ZRRDB instruction reads one byte specified by n via the serial byte number from a file register. The byte number does not specify a block address. The byte is stored in the lower byte of the device specified by d. The upper byte in the device specified by d stores the value "00H".



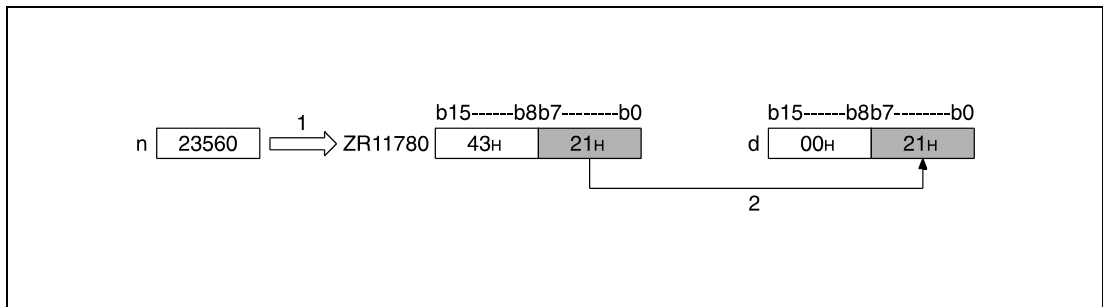
- <sup>1</sup> Serial byte number
- <sup>2</sup> File register area for block 0
- <sup>3</sup> File register area for block 1
- <sup>4</sup> File register area for block 2
- <sup>5</sup> Read byte

The assignment of file register numbers to the according serial byte numbers is shown below:



- <sup>1</sup> Storage area for even byte numbers (here: address 0 through address 5006)
- <sup>2</sup> Storage area for odd byte numbers (here: address 1 through address 5007)

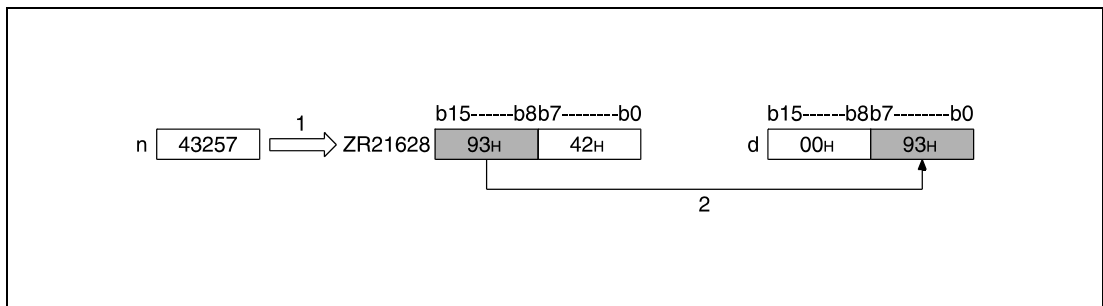
If the byte number 23560 is specified, the lower byte of the file register ZR11780 is read.



<sup>1</sup> Address

<sup>2</sup> Storage

If the byte number 43257 is specified, the lower byte of the file register ZR21628 is read.



<sup>1</sup> Address

<sup>2</sup> Storage

**Operation Errors**

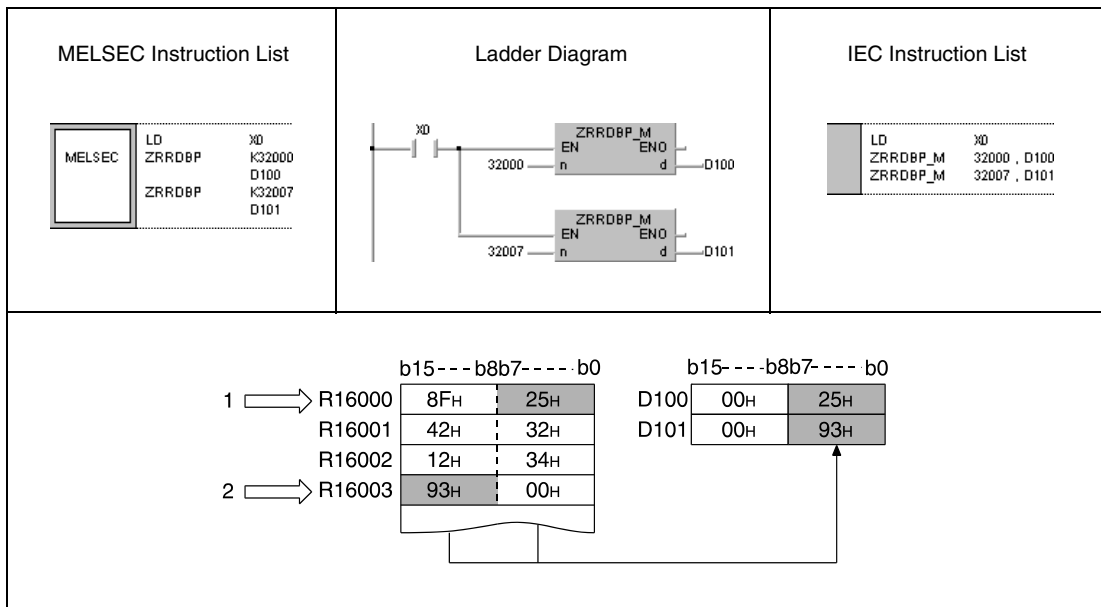
In the following cases an operation error occurs and the error flag is set:

- The number of device (serial byte address) exceeds the relevant storage device range (error code 4101).

**Program Example**

**ZRRDBP**

With leading edge from X0, the following program reads the lower byte of file registers R16000 (byte number 32000) and the upper byte of the file register R16003 (byte number 32007). The bytes are stored in D100 and D101.



<sup>1</sup> Serial byte number 32000 (lower byte in file register R16000)

<sup>2</sup> Serial byte number 32007 (upper byte in file register R16003)

7.18.5 ZRWRB, ZRWRBP

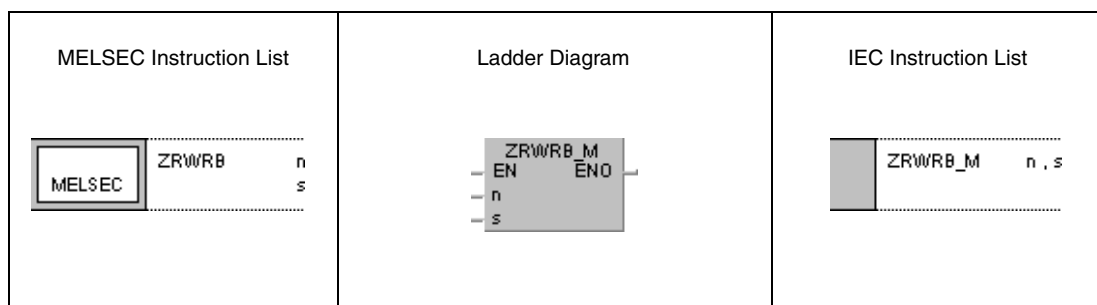
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

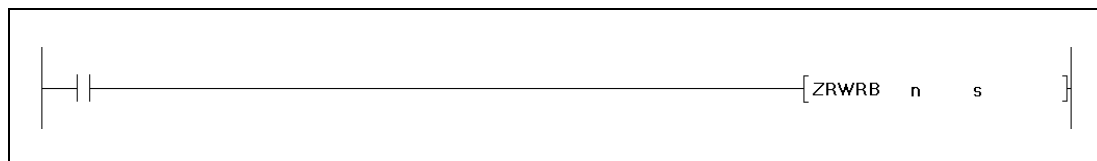
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
n	●	●	●	●	●	●	●	—	SM0	3	
s	●	●	●	●	●	●	●	—			

GX IEC Developer



GX Developer



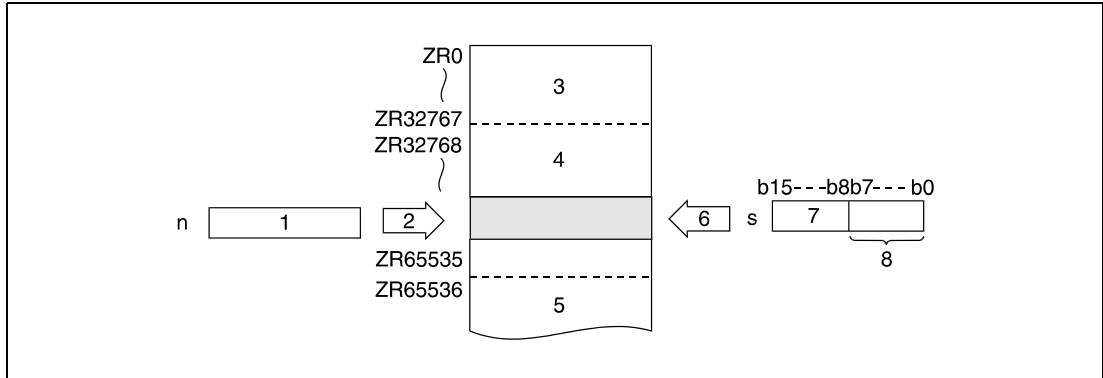
Variables

Set Data	Meaning	Data Type
n	Serial byte number in file register to be written to.	BIN 32-bit
s	Device storing data to be written.	BIN 16-bit

**Functions Direct write of one byte to a file register**

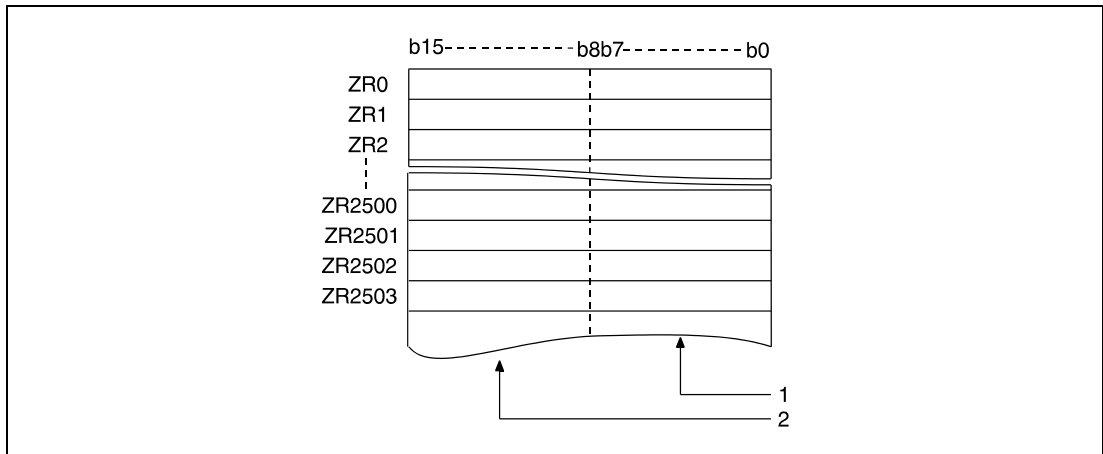
**ZRWRB Write one byte**

The ZRRDB instruction writes the contents of the lower byte in the device specified by s to the file register specified by n via serial byte number. The byte number in s does not specify a block address. The upper byte of the device in s is ignored.



- 1 Serial byte number
- 2 Address
- 3 File register area for block 0
- 4 File register area for block 1
- 5 File register area for block 2
- 6 Write data
- 7 This byte is ignored
- 8 Byte to be written

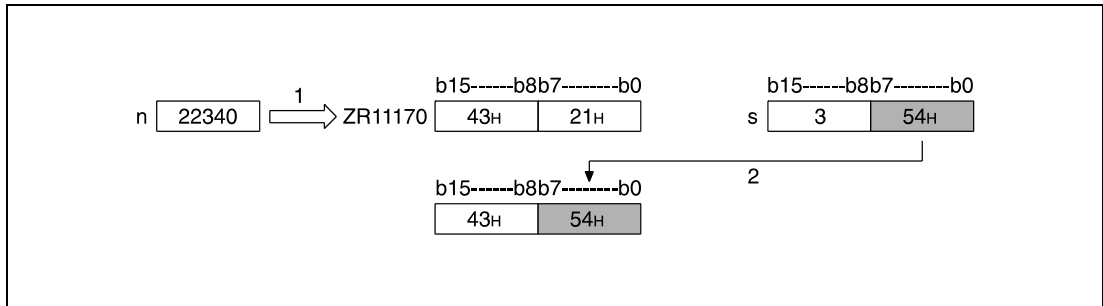
The assignment of file register numbers to the according serial byte numbers is shown below:



- 1 Storage area for even byte numbers (here: address 0 through address 5006)
- 2 Storage area for odd byte numbers (here: address 1 through address 5007)

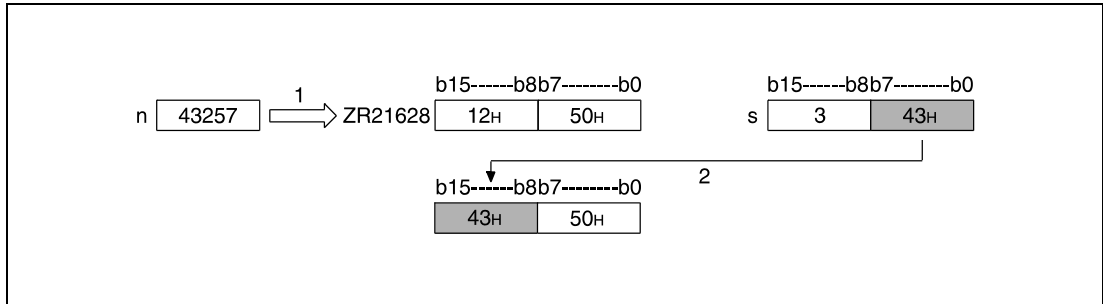


If the byte number 22340 is specified, the lower byte of the device specified by s is written to the lower byte of the file register ZR11170.



- <sup>1</sup> Address
- <sup>2</sup> Write byte
- <sup>3</sup> This byte is ignored

If the byte number 43257 is specified, the lower byte of the device specified by s is written to the upper byte of the file register ZR21628.



- <sup>1</sup> Address
- <sup>2</sup> Write byte
- <sup>3</sup> This byte is ignored

**Operation Errors**

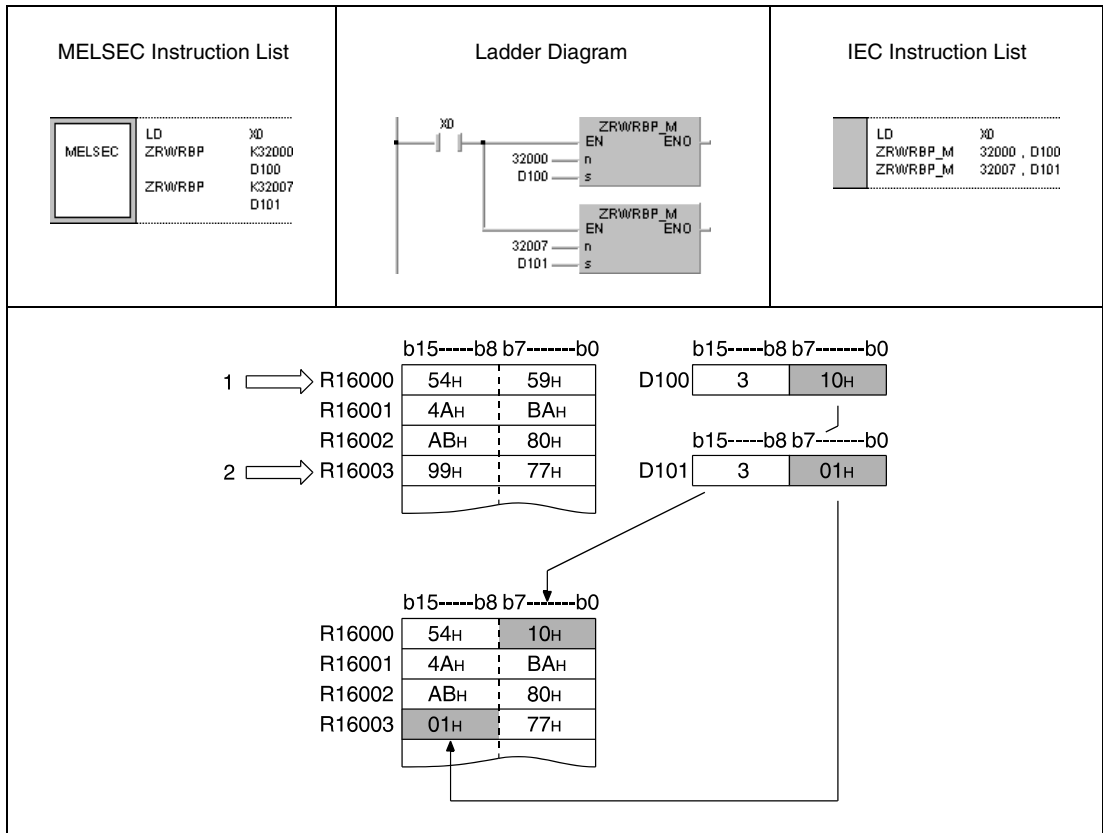
In the following cases an operation error occurs and the error flag is set:

- The number of device (serial byte number) specified by n exceeds the relevant storage device range (error code 4101).

**Program Example**

ZRWRBP

With leading edge from X0, the following program writes the contents of the lower bytes of the registers D100 and D101 to the lower byte of the file register R16000 (byte number 32000) and to the upper byte of the file register R16003 (byte number 32007).



- <sup>1</sup> Serial byte number 32000 (lower byte of file register R16000)
- <sup>2</sup> Serial byte number 32007 (upper byte of file register R16003)
- <sup>3</sup> These bytes are ignored

**7.18.6 ADRESET, ADRSETP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

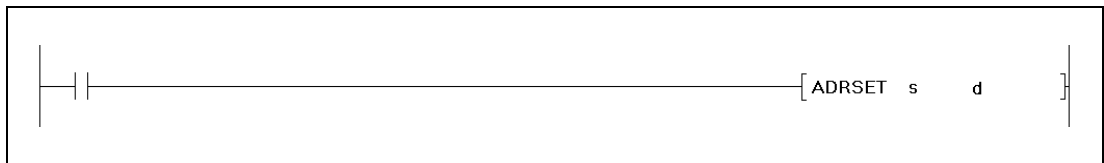
**Devices MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	●	●	●	—	—	—	—	—	SM0	3	
d	●	●	●	—	—	—	—	—			

**NOTE**

The instructions *ADRSET* and *ADRSETP* are not supported by the GX IEC Developer.

**GX Developer**



**Variables**

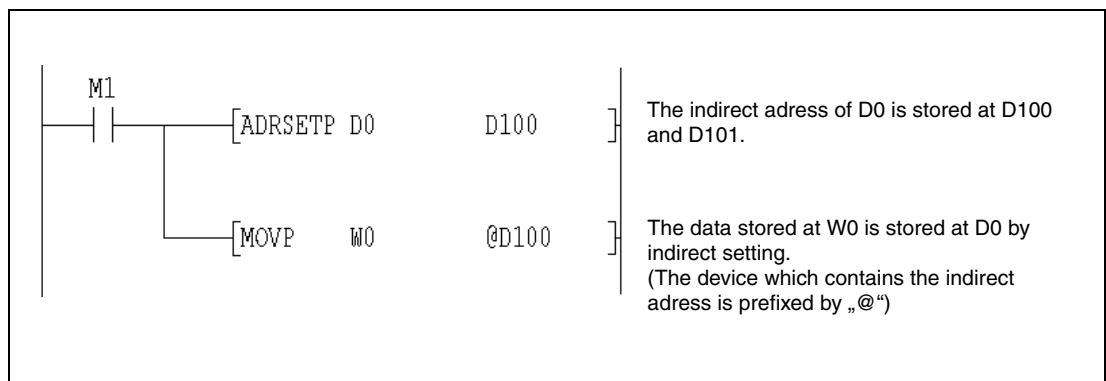
Set Data	Meaning	Data Type
s	Number of device for indirect address read	Device name
d	Number of device that will store the indirect address of the device designated by s	BIN 32-bit

**Functions**

**Indirect address read operations**

**ADRSET Stores the indirect address**

Stores the indirect address of the device designated by s at d and d + 1. The address stored at the device designated by d is used when reading of an indirect device address is performed by the sequence program. A bit device designation cannot be made at s.



**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- A device for which designation is not allowed has been designated (error code 4101).

7.18.7 KEY

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
		● <sup>1</sup>	● <sup>1</sup>	●	● <sup>2</sup>

<sup>1</sup> Using an AnA and AnU CPU this dedicated instruction in the IEC editor can be programmed as function, and in the MELSEC editor can be programmed in combination with the LEDA, LEDC, and LEDR instructions.

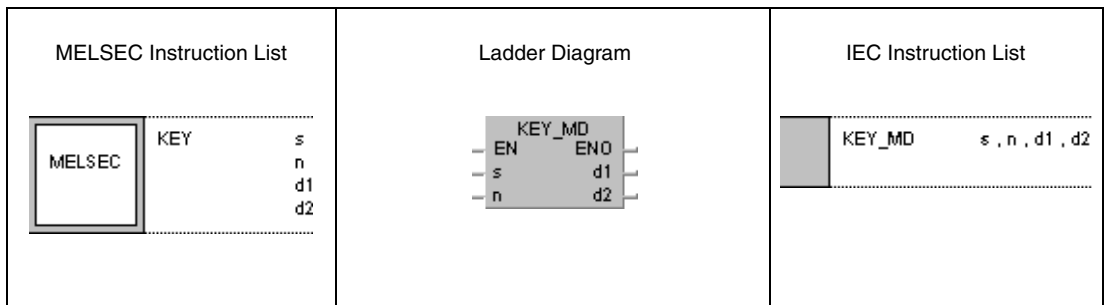
<sup>2</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices MELSEC Q

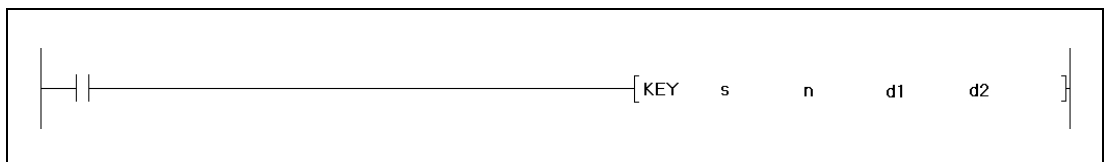
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	● <sup>1</sup>	—	—	—	—	—	—	—	—	SM0	5
n	●	●	●	●	●	●	●	●	—		
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	●	●	●	—	—	—		

<sup>1</sup> X only

GX IEC Developer



GX Developer



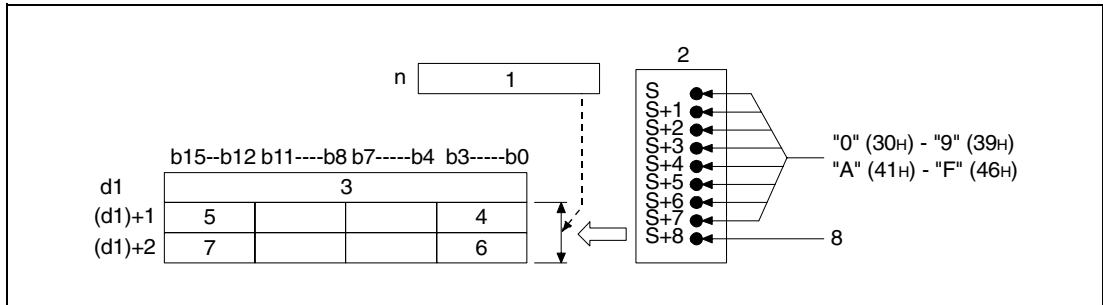
Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
s	First number of devices (X), receiving numerical key input.	Bit	Array [1..9] of BOOL
n	Number of digits to be input.	BIN 16-bit	ANY16
d1	First number of device storing numerical key input.	BIN 16-bit	Array [1..3] of ANY16
d2	Number of bit device to be set after completion of key input.	Bit	BOOL

**Functions Numerical key input**

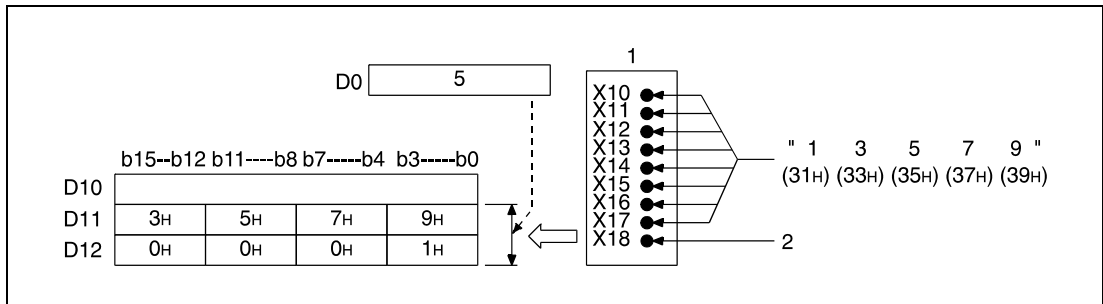
**KEY Input instruction**

The KEY instruction supports the key input of the ASCII characters 0 (30H) through 9 (39H) and A (41H) through F (46H) at the inputs specified by s+0 (array\_s[0]) through s+7 (array\_s[7]). The values entered at the inputs are encoded in hexadecimal format and stored in the devices specified by (d1)+0 (array\_d1[0]) through (d1)+2 (array\_d1[2]). The number of characters to be input is specified by n.



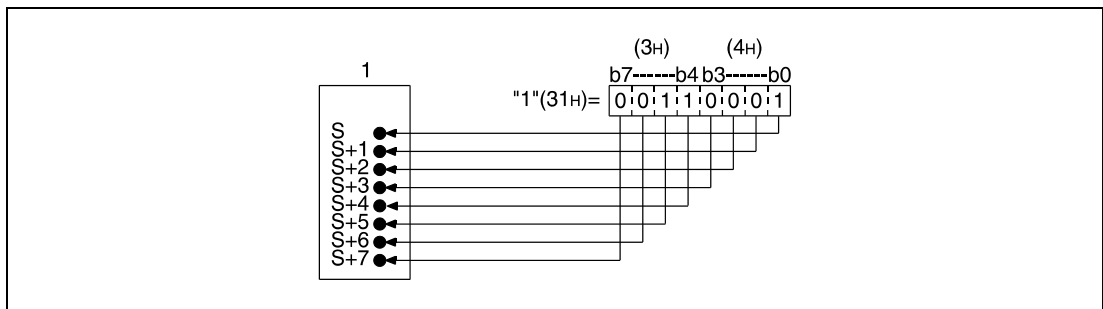
- <sup>1</sup> Number of values to be entered
- <sup>2</sup> Input module
- <sup>3</sup> Number of entered values
- <sup>4</sup> 8th entered character
- <sup>5</sup> 5th entered character
- <sup>6</sup> 4th entered character
- <sup>7</sup> 1st entered character
- <sup>8</sup> Strobe signal

In the following diagram n is specified 5 and the values 1 (31H) through 5 (35H) are entered at the inputs X10 through X18 of the input module.



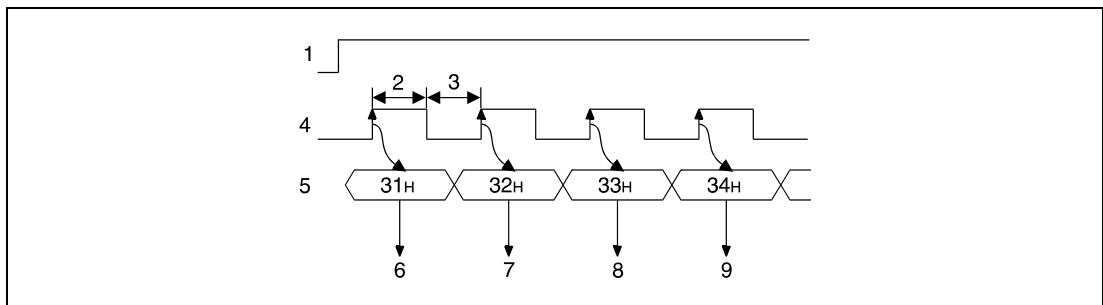
- <sup>1</sup> Input module
- <sup>2</sup> Strobe signal

The ASCII characters entered at the inputs (X) specified in s+0 (array\_s[0]) through s+7 (array\_s[7]) are encoded in 8-bit binary format as illustrated below:



<sup>1</sup> Input module

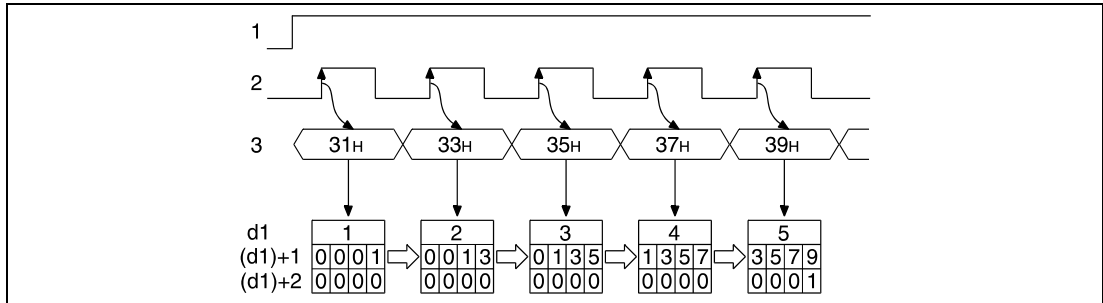
After the input of an ASCII character at s+0 (array\_s[0]) through s+7 (array\_s[7]) the strobe signal (s+8, array\_s[8]) is set, to link the input data internally. The time period the strobe signal remains set or reset must exceed one program scan time to ensure accurate linking of input data.



- <sup>1</sup> Execution condition for the KEY instruction
- <sup>2</sup> Set for more than one program scan
- <sup>3</sup> Reset for more than one program scan
- <sup>4</sup> Strobe signal (s+8, array\_s[8])
- <sup>5</sup> ASCII input data (s+0 through s+7, array\_s[0] through array\_s[7])
- <sup>6</sup> Reading "1"
- <sup>7</sup> Reading "2"
- <sup>8</sup> Reading "3"
- <sup>9</sup> Reading "4"

The KEY instruction can only be executed with the execution condition set. The execution condition must remain set until the input of the number of characters specified by n is completed.

The number of entered values is stored in (d1)+0 (array\_d[0]). The entered ASCII characters are actually stored in the devices specified in (d1)+1 (array\_d[1]) and (d1)+2 (array\_d[2]) and (d1)+2 (array\_d[2]) as hexadecimal binary values; i.e. there are 4 bits per character supplied. The hexadecimal binary values of the characters 0H through FH range from "0000" through "1111".

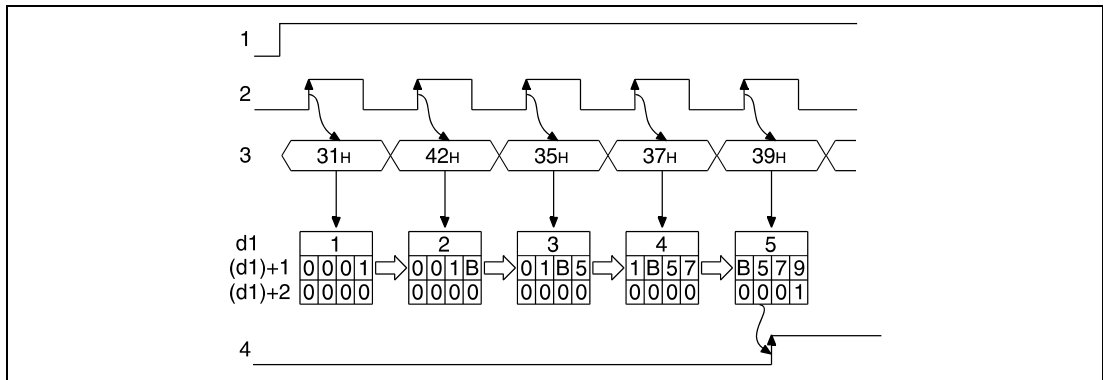


- <sup>1</sup> Execution condition for the KEY instruction
- <sup>2</sup> Strobe signal (s+8, array\_s[8])
- <sup>3</sup> ASCII input data (s+0 through s+7, array\_s[0] through array\_s[7])

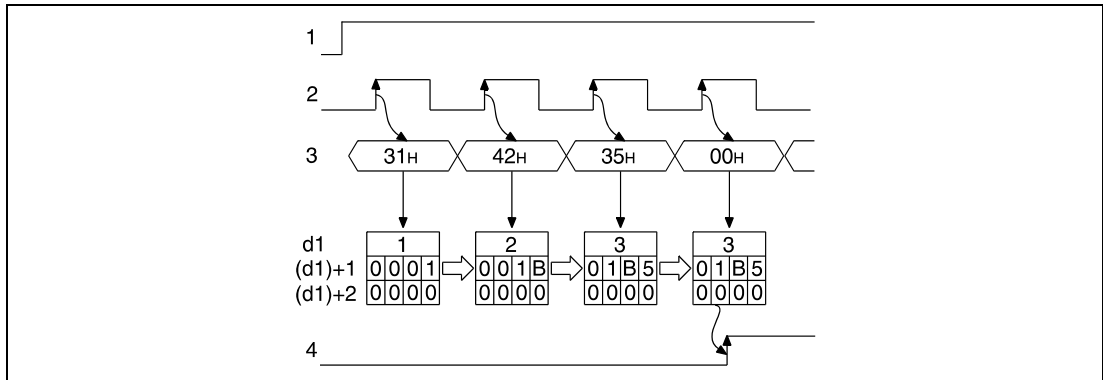
The number of characters to be entered specified by n must range within 1 and 8.

If the specified number of characters or the character code "00H" are entered, the linking of the input data is completed and the device specified by d2 is set. The following diagrams illustrate these operations. For n 5 is specified.

In the following diagram the input is completed after 5 characters. In the next but one diagram the input is completed after the character code "00H".



- <sup>1</sup> Execution condition for the KEY instruction
- <sup>2</sup> Strobe signal (s+8, array\_s[8])
- <sup>3</sup> ASCII input data (s+0 through s+7, array\_s[0] through array\_s[7])
- <sup>4</sup> Input of characters completed (the device specified by d2 is set)



- <sup>1</sup> Execution condition for the KEY instruction
- <sup>2</sup> Strobe signal (s+8, array\_s[8])
- <sup>3</sup> ASCII input data (s+0 through s+7, array\_s[0] through array\_s[7])
- <sup>4</sup> Input of characters completed (the device specified by d2 is set)

Prior to a new input of characters the contents of the devices specified in (d1)+0 (array\_d1[0]) through (d1)+2 (array\_d[2]) have to be cleared and the device specified by d2 has to be reset; otherwise a new input of characters is not possible.

**Operation Errors**

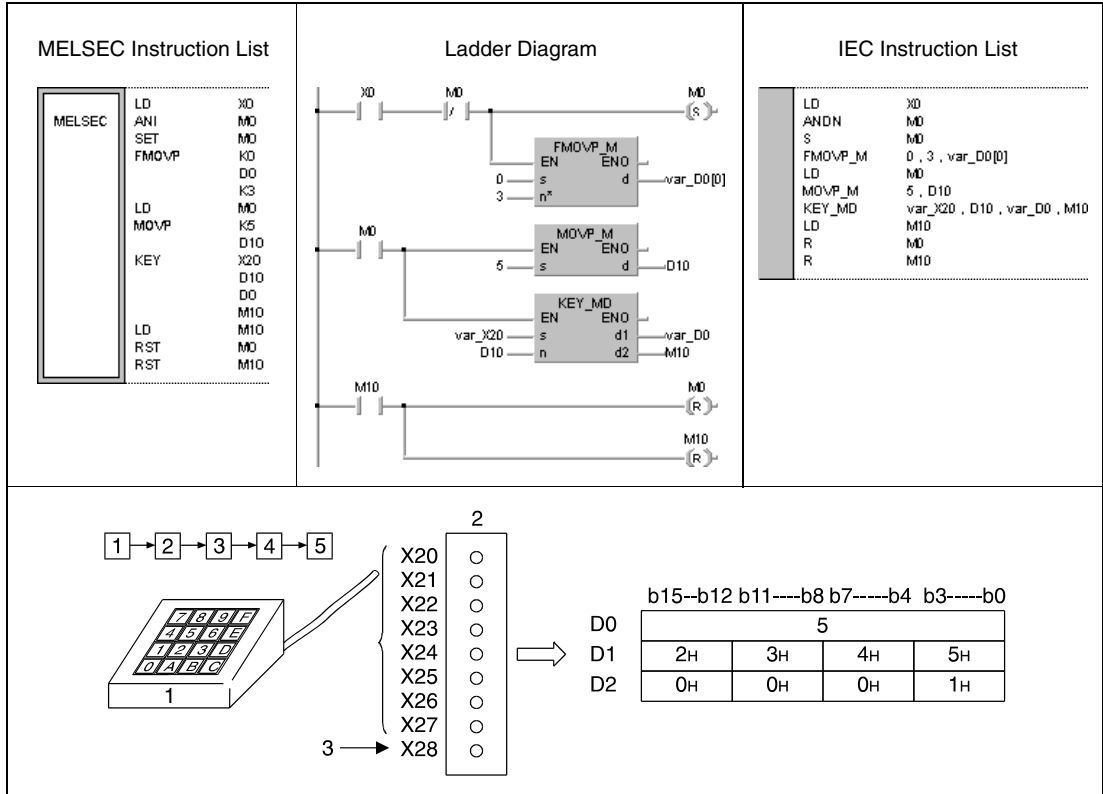
In the following cases an operation error occurs and the error flag is set:

- The device specified by s is not an input (X) (error code 4100).
- The number of characters specified by n does not range within 1 and 8.



**Program Example**

The following program enables key input of up to 5 numerical values via the inputs X20 (var\_X20[0]) through X27 (var\_X20[7]). The values are stored in the registers D1 (var\_D0[1]) and D2 (var\_D0[2]) binary coded in hexadecimal format. The number of values already entered is stored in D0 (var\_D0[0]). Prior to the execution of the KEY instruction the registers D0 (var\_D0[0]) through D2 (var\_D0[2]) are cleared and the number of input values (5) is stored. After execution of the KEY instruction the relay M10 (input completed) is reset. The strobe signal is supplied at the inputs X28 (var\_X20[8]).



- <sup>1</sup> Numerical key pad
- <sup>2</sup> Input module
- <sup>3</sup> Strobe signal

7.18.8 ZPUSH, ZPUSHP, ZPOP, ZPOPP


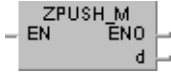
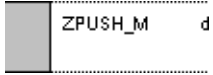
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

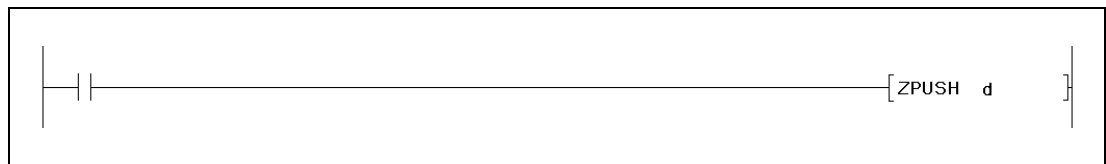
Devices  
MELSEC Q

d	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
	—	●	●	—	—	—	—	—	SM0	3	

GX IEC  
Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

GX  
Developer



Variables

Set Data	Meaning	Data Type
d	First number of device storing index register contents.	BIN 16-bit

**Functions Batch save and batch recovery of index register contents**

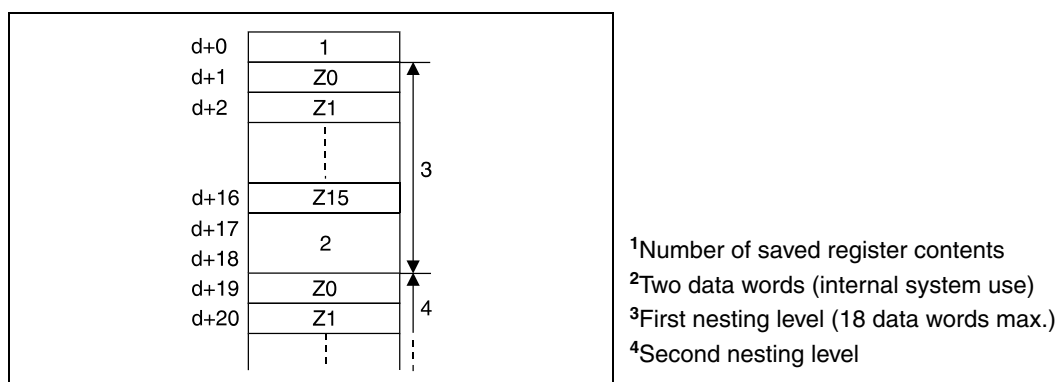
**ZPUSH Batch save of index register contents**

The ZPUSH instruction saves the contents of the index registers Z0 through Z15 in the devices specified from d on.

These data can be recovered via the ZPOP instruction. The instruction can be applied to different nestings that are included in ZPUSH / ZPOP loop.

On execution of the instructions in different nestings each execution of the ZPUSH instruction requires an area of 18 registers with 16 bits in the devices specified from d on. Therefore, for the execution of the ZPUSH instruction the according amount of storage area has to be available.

The following diagram illustrates the organization of the storage area from d on:



**ZPOP Batch recovery of index register contents**

The ZPOP instruction recovers index register contents saved via the ZPUSH instruction. The contents of the storage area specified from d on are read and re-written to the according index registers.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The storage area specified from d on exceeds the relevant storage device range (error code 4101).
- The content of the device specified in d+0 is 0 (number of saved registers) (error code 4100).

**Program Example**

**ZPUSH/ZPOP**

If X20 is set, the following program saves the contents of the index registers in the storage area from register D0 on. Then the sub-routine at the jump destination label\_0 is called that uses the index registers.

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 10%;">MELSEC</td> <td style="width: 10%;">LD</td> <td style="width: 10%;">X0</td> </tr> <tr> <td></td> <td>ZPUSH</td> <td>D0</td> </tr> <tr> <td></td> <td>LD</td> <td>X1</td> </tr> <tr> <td></td> <td>ZPOP</td> <td>D0</td> </tr> </table>	MELSEC	LD	X0		ZPUSH	D0		LD	X1		ZPOP	D0	<p>Ladder Diagram</p>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">LD</td> <td style="width: 10%;">X0</td> </tr> <tr> <td></td> <td>ZPUSH_M</td> <td>D0</td> </tr> <tr> <td></td> <td>LD</td> <td>X1</td> </tr> <tr> <td></td> <td>ZPOP_M</td> <td>D0</td> </tr> </table>		LD	X0		ZPUSH_M	D0		LD	X1		ZPOP_M	D0
MELSEC	LD	X0																								
	ZPUSH	D0																								
	LD	X1																								
	ZPOP	D0																								
	LD	X0																								
	ZPUSH_M	D0																								
	LD	X1																								
	ZPOP_M	D0																								

7.18.9 EROMWR, EROMWRP

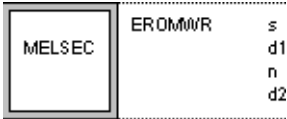
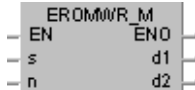
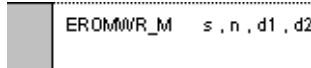
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

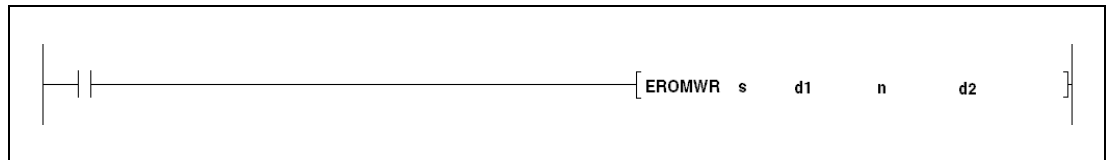
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other U		
	Bit	Word		Bit	Word						
s	—	●	—	—	—	—	—	—	—	SM0	6
d1	—	—	●	—	—	—	—	—	—		
n	●	●	●	●	●	●	●	●	—		
d2	●	—	—	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	---	---

GX Developer



Variables

Set Data	Meaning	Data Type
s	First number of device storing data to be written.	BIN 16-bit
d1	First number of EEPROM file register to be written to.	
n	Number of data words to be written.	
d2	Device to be set after operation is completed.	Bit

**Functions**      **Batch write of data to an EEPROM file register****EROMWR**      **Write instruction**

The EROMWR instruction writes the number specified by n of data words stored in the device specified by s to an EEPROM file register specified by d1.

After completion of the write operation the device specified by d2 is set and after one program scan reset again automatically.

The EROMWR instruction is executed until END processing. Before END processing 64 data words can be written each program scan. The number of program scans results from the rounded up quotient of the number of data words specified by n divided by 64. The processing time can be calculated on the basis of a scan time of approx. 10 ms.

The data specified by s must not be refreshed during the write operation, otherwise data can be lost.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The storage area for the number of data words specified by n exceeds the relevant storage device range specified by s and d1 (error code 4101).
- The file register specified by d1 does not exist or is not an EEPROM file register (error code 4101).

## 8 Data Link Instructions

### 8.1 Fundamentals

A QnA(S) CPU can be used within the network systems MELSECNET(II)/B/10. A CPU of the System Q supports the network systems MELSECNET/10 and MELSECNET/H.

**NOTE**

*The terms MELSECNET/10 and MELSECNET/H used here refer to the network systems MELSECNET/10 and MELSECNET/H.*

*The term MELSECNET used here refers to the network systems MELSECNET(I), MELSECNET(II), and MELSECNET/B.*

Via the data link instructions the CPU is able to exchange data with other stations connected to the MELSECNET and MELSECNET/10.

### 8.2 Categories of instructions

The data link instructions are subdivided into the following four categories:

1. Data refresh instructions

These instructions refresh data in the designated network modules.

2. Dedicated data link instructions

These data link instructions are applied in combination with a QnA CPU or System Q CPU. For the data communication multiple channels of the network module are used.

3. A series compatible link instructions

These instructions are identical to the dedicated ACPU instructions.

4. Read/Write routing information

These instructions read and write routing parameters from and to relay and routing stations.

For the MELSECNET and MELSECNET/10 systems only specific data link instructions can be applied. Which instructions can be applied within MELSECNET/10 furthermore depends on whether the object station is a System Q CPU, an A CPU, an QnA CPU, or a remote I/O station.

The following table gives an overview of the data link instructions:

<b>Category</b>	<b>Meaning</b>
Network refresh instructions	Instructions for data refresh operations in network modules.
Dedicated data link instructions	Read and write CPU data from and to object stations in object networks. Send data to network modules in object stations in object networks. Read CPU data sent via SEND instruction. Data requests to different stations (write/read operations with clock data, RUN/STOP operations). Read and write data from and to special function modules in remote I/O stations.
A series compatible data link instructions	Read and write CPU data from and to object stations in different networks. Read and write CPU data from and to local stations (at master stations only). Read and write data from and to special function modules in remote I/O stations.
Read/Write routing information	Read and write routing parameters (network number and station number of relay station, station number of routing station).



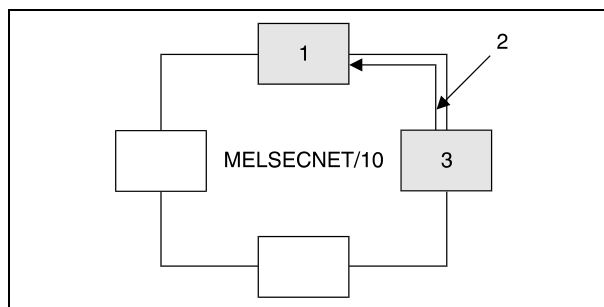
## 8.3 Data read/write ranges

### 8.3.1 MELSECNET/10

With MELSECNET/10 a host station performs read/write operations with stations within one network or via respective addressing (routing parameters) with stations in other networks.

#### Read/write operations with stations within one network

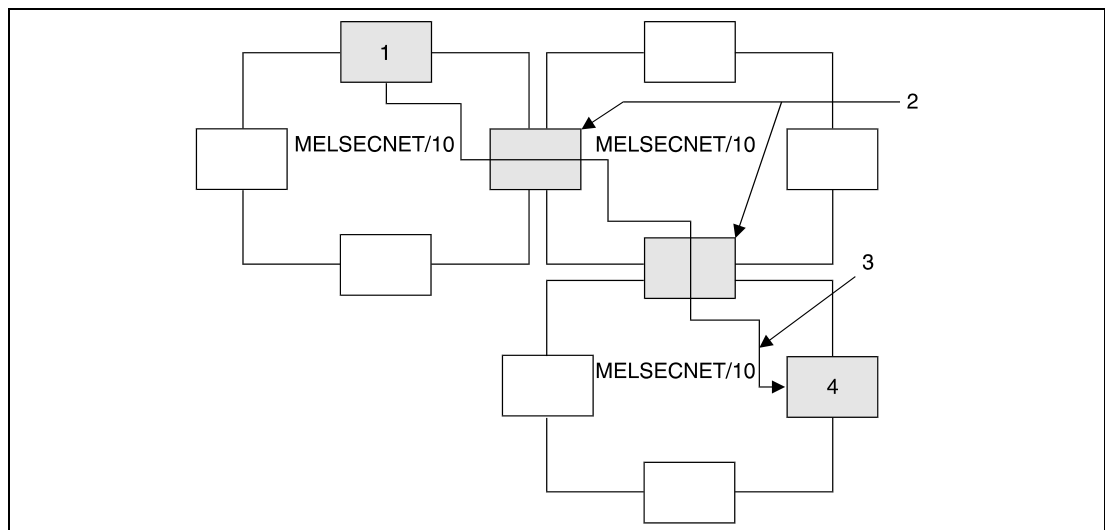
The network number of the object station and that of the network module the host station is connected to must be the same. This function reads and writes data from and to any station within one network.



- <sup>1</sup>Station executing the instruction
- <sup>2</sup>Read operation
- <sup>3</sup>Object station

#### Read/write operations with stations within different networks

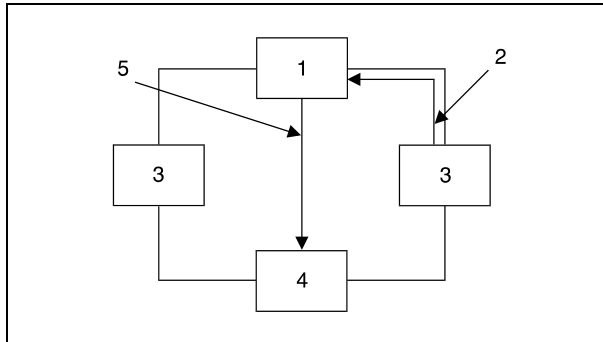
The network number of the object station and that of the network module the host station is connected to must be different. One station in the network of the host station serves as relay station forwarding the read/write operations to the object station in another network.



- <sup>1</sup> Station executing the instruction
- <sup>2</sup> Relay stations (routing parameters must be set)
- <sup>3</sup> Read operation
- <sup>4</sup> Object station

**8.3.2 MELSECNET**

With MELSECNET(I/II/B) a master station performs read/write operations with local stations and remote I/O stations.



- 1 Master station
- 2 Read/write operation
- 3 Local station
- 4 Remote I/O station
- 5 Read/write operation with special function modules

**8.4 Dedicated data link instructions**

In the following, several considerations for the use of the dedicated data link instructions for Q series CPUs and System Q CPUs are described.

**8.4.1 Simultaneous execution**

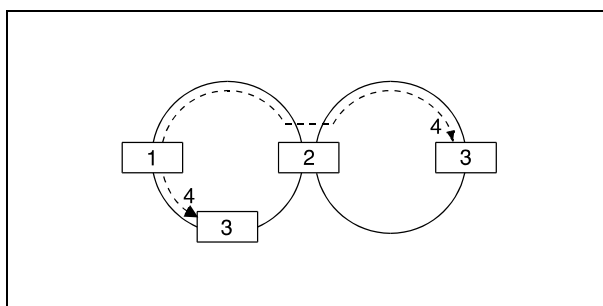
Network modules for the MELSECNET/10 system provide 8 areas for communication used by data link instructions. These network modules do not support the simultaneous execution of multiple data link instructions within one communication area. If within one communication area of the CPU more than one data link instruction is to be executed, a successive execution of the individual instructions must be ensured via the completion devices set after each completed read/write instruction.

**8.4.2 Transmission completion**

Applying the dedicated data link instructions for the Q series and System Q it can be specified whether the completed transmission of data is confirmed or not.

**Confirmation of transmission completion**

The following figure shows the mode in which the completed execution or data transmission is confirmed when the data was written to the designated channel of the designated object station (for read operations only this mode can be selected).



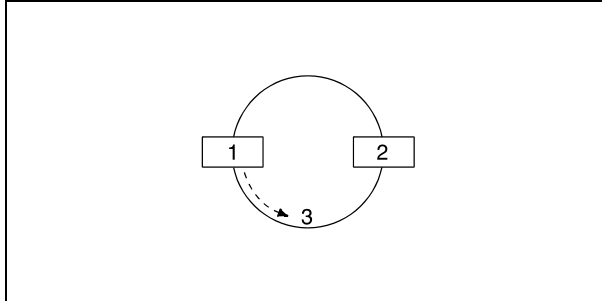
- 1 Execution source
- 2 Relay station
- 3 Object station
- 4 Execution/transfer completed

### No confirmation of transmission completion

The following figures show the mode in which the completed execution or data transmission is not confirmed.

Within one network:

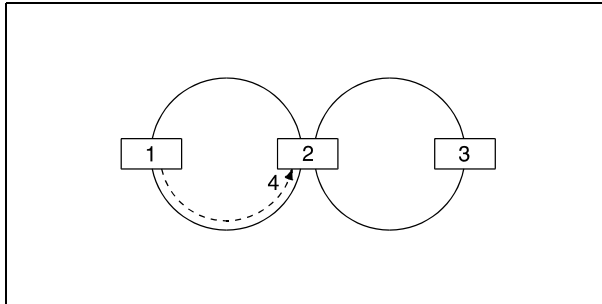
The execution or data transfer is completed when the host station has sent all data.



- <sup>1</sup>Execution source
- <sup>2</sup>Object station
- <sup>3</sup>Execution/transfer completed

Among different networks:

The execution or data transmission is completed when the sent data has arrived at a relay station in the network of the host.



- <sup>1</sup>Execution source
- <sup>2</sup>Relay station
- <sup>3</sup>Object station
- <sup>4</sup>Execution/transfer completed

### NOTE

*In order to improve the data integrity, it is recommended to select the mode with the confirmed transmission completion.*

*If the mode without confirmation of the transmission completion is specified, the transmission is completed after the data has been sent, regardless of occurring errors during transmission. Furthermore, the object station returns a "reception buffer full" error in case several stations execute data link instructions at the same time, even if the data was transmitted correctly. Nevertheless, the transmitting station completes the operation in this case.*

## 8.5 Data refresh instructions

The following instructions refresh data in network modules. The following table gives an overview of the instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor	Designated Station in MELSECNET/10			MELSECNET
			QnA CPU System Q	ACPU	Remote I/O Station	
Data refresh instructions	ZCOM	ZCOM_J_M	—	—	—	—
		ZCOM_JP_M				
		ZCOM_U_M				
		ZCOM_UP_M				

### 8.5.1 ZCOM

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	SM0	6

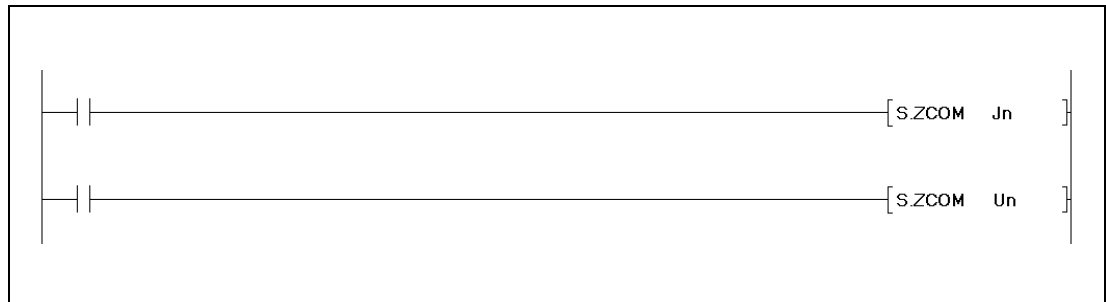
**GX IEC Developer**

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="border: 2px solid black; text-align: center;">MELSEC</td> <td>J.ZCOM</td> <td>Jn</td> </tr> <tr> <td></td> <td>G.ZCOM</td> <td>Un</td> </tr> </table>	MELSEC	J.ZCOM	Jn		G.ZCOM	Un	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td>ZCOM_J_M</td> <td>Jn</td> </tr> <tr> <td>ZCOM_U_M</td> <td>Un</td> </tr> </table>	ZCOM_J_M	Jn	ZCOM_U_M	Un
MELSEC	J.ZCOM	Jn										
	G.ZCOM	Un										
ZCOM_J_M	Jn											
ZCOM_U_M	Un											

**GX Developer (QnA-CPU)**



**GX Developer (System Q CPU)**



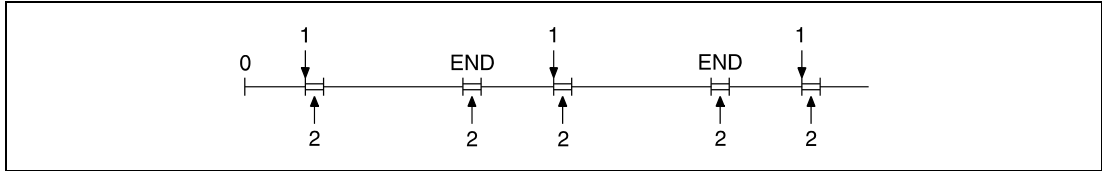
**Variables**

Set Data	Meaning	Data Type
Jn	Network number for host station.	BIN 16-bit
Un	Head I/O number of host station network.	

**Functions Network data refresh**

**ZCOM Data refresh in network modules**

On execution of the ZCOM instruction the CPU suspends processing the sequence program and refreshes the data in the network modules specified by Jn and Un.

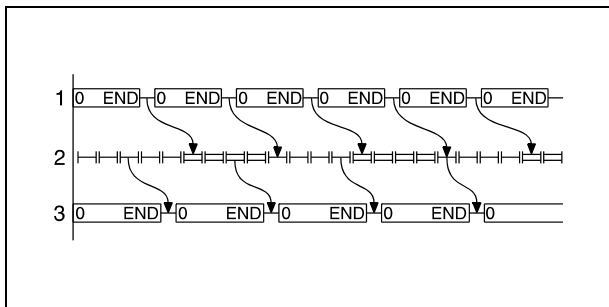


<sup>1</sup>Execution of the ZCOM instruction

<sup>2</sup>Data refresh

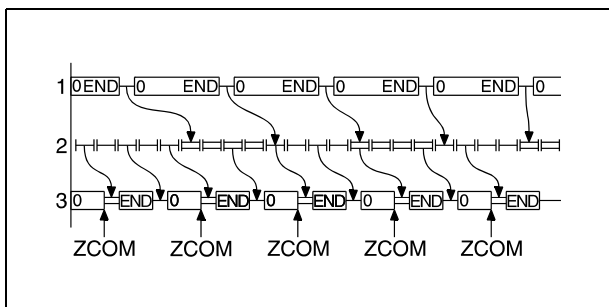
In cases where the scan time of the sequence program of the host station exceeds the scan time of the other stations, the ZCOM instruction ensures that the data from the other station is incorporated properly.

The following figure shows an example for data communication without applying the ZCOM instruction:



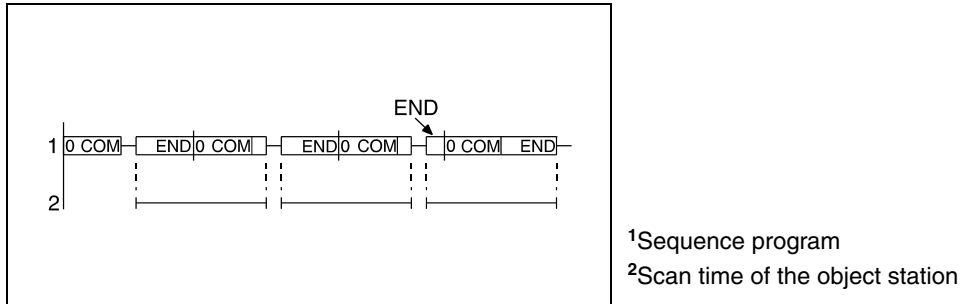
<sup>1</sup>Program of the control station  
<sup>2</sup>Program scan of the linked station  
<sup>3</sup>Program of the normal station

The following figure shows an example for data communication applying the ZCOM instruction:



<sup>1</sup>Program of the control station  
<sup>2</sup>Program scan of the linked station  
<sup>3</sup>Program of the normal station

In cases where the scan time of the object station exceeds the scan time of the sequence program, the ZCOM instruction does not improve data communication.



The ZCOM instruction may be executed any times within a sequence program. However, note that each execution increases the scan time of the sequence program by the execution time of the data refresh.

The ZCOM instruction cannot be applied with the following operations:

- Communication between the CPU and peripheral units.
- Monitoring other stations.
- Reading the buffer memory of other special function modules via a computer link module.

#### NOTE

*With a Q series or System Q CPU, designating "Un" in the argument enables the access not only to network modules but also to intelligent function modules. In this case, the automatic refresh is performed for the buffer memory of the intelligent function module. (replaces the FROM/TO instructions).*

#### Operation Errors

In the following cases an operation error occurs and the error flag is set:

- The specified network number is not connected to the host station (error code 4102).
- The module for the specified I/O number is not a network unit or link unit (error code 2111).

#### NOTE

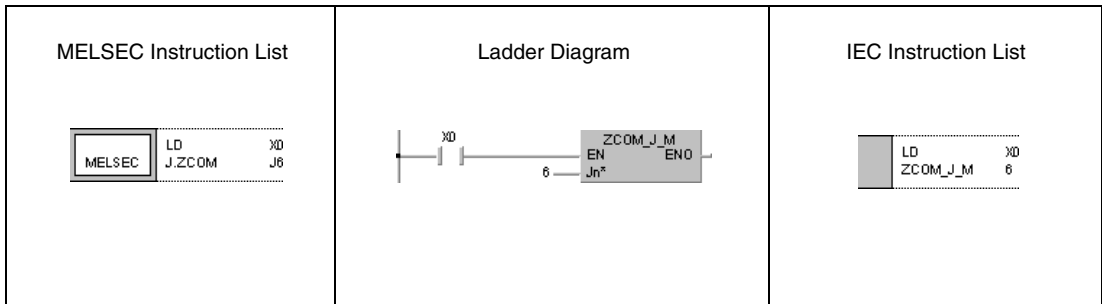
*For exclusive common data processing apply the COM instruction.*

*Note that non-consistent data might occur, i.e., a device might change during a program scan.*

**Program Example 1**

**J.ZCOM**

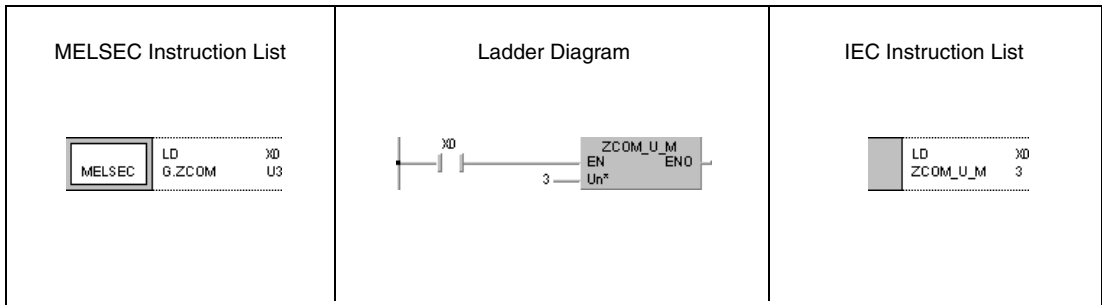
While X0 is set, the following program refreshes data in the network module with the network number 6.



**Program Example 2**

**G.ZCOM**

While X0 is set, the following program refreshes data in the network module at the I/O numbers X/Y30 through X/Y4F.





## 8.6 Dedicated data link instructions for the QnA series

These instructions support the data communication among stations with QCPUs as well as between QnA CPUs and remote I/O stations within MELSECNET/10. The following table gives an overview of these instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor	Designated Station in MELSECNET/10			MELSECNET
			QnA CPU	ACPU	Remote I/O Station	
Read QnA CPU data from object stations in object networks	READ	READ_M	●	—	—	—
		READP_M				
		READ_JP_M				
		READ_UP_M				
	SREAD	SREAD_JP_M	●	—	—	—
		SREAD_UP_M				
Write QnA CPU data to object stations in object networks	WRITE	WRITE_JP_M	●	—	—	—
		WRITE_UP_M				
	SWRITE	SWRITE_M	●	—	—	—
		SWRITE_JP_M				
		SWRITE_UP_M				
	Send data to network modules in object stations in object networks	SEND	SEND_M	●	—	—
SEND_4_M						
SEND_4_P_M						
SEND_JP_M						
SEND_UP_M						
Read QnA CPU data sent via SEND instruction	RECV	RECV_M	●	—	—	—
		RECV_P_M				
		RECV_JP_M				
		RECV_UP_M				
Data request from other stations (write/read operations with clock data, remote RUN/STOP)	REQ	REQ_M	●	—	—	—
		REQP_M				
		REQ_JP_M				
		REQ_UP_M				
Read data from special function modules in remote I/O stations	ZNFR	ZNFR_JP_M	—	—	●	—
		ZNFR_UP_M				
Write data to special function modules in remote I/O stations.	ZNT0	ZNT0_J_M	—	—	●	—
		ZNT0_U_M				
		ZNT0_JP_M				
		ZNT0_UP_M				

8.6.1 READ

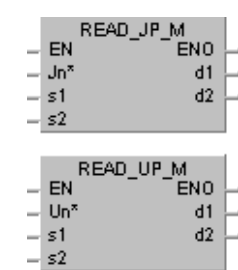
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

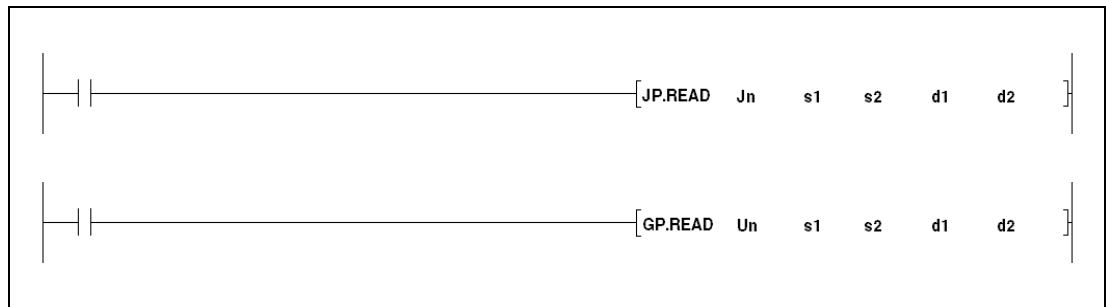
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—	SM0	11
s2	—	●	●	—	—	—	—	●	—		
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">MELSEC</td> <td>JP.READ</td> <td>Jn</td> <td>s1</td> <td>s2</td> <td>d1</td> <td>d2</td> </tr> <tr> <td></td> <td>GP.READ</td> <td>Un</td> <td>s1</td> <td>s2</td> <td>d1</td> <td>d2</td> </tr> </table>	MELSEC	JP.READ	Jn	s1	s2	d1	d2		GP.READ	Un	s1	s2	d1	d2	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td>READ_JP_M</td> <td>Jn , s1 , s2 , d1 , d2</td> </tr> <tr> <td>READ_UP_M</td> <td>Un , s1 , s2 , d1 , d2</td> </tr> </table>	READ_JP_M	Jn , s1 , s2 , d1 , d2	READ_UP_M	Un , s1 , s2 , d1 , d2
MELSEC	JP.READ	Jn	s1	s2	d1	d2														
	GP.READ	Un	s1	s2	d1	d2														
READ_JP_M	Jn , s1 , s2 , d1 , d2																			
READ_UP_M	Un , s1 , s2 , d1 , d2																			

GX Developer



Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First device of host station storing control data.	Device number	Array [1..18] of ANY16
s2	First device of station storing data to be read.		ANY16
d1	First device of host station storing read data.		
d2	Device set ON for 1 scan after completion of instruction.	Bit	BOOL

**NOTE**

- <sup>1</sup> *The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.*
- <sup>2</sup> *The head I/O number of the network unit for the host station must range within 0 and FEH. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.*

*The READ instruction can only be executed, if the object station is a QnA CPU.*

*With an ACPU in MELSECNET/10 the READ instruction cannot be applied.*

*Only station numbers for QnA CPUs are valid numbers for the object station.*

Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	The transmission confirmation is set: (Bit 0 (b0) = 1, fixed)	0001H 0081H	User
	Error completion mode	Storage of clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+11 (Array_s1[12]) onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Channel used by host station	Host channel designation.	1 to 8	User
(s1)+3 Array_s1[4]	Dummy	Not used	0	—
(s1)+4 Array_s1[5]	Network number of object station	Sets network number for station to be read from.	1 to 239 254 ● <sup>4</sup>	User
(s1)+5 Array_s1[6]	Number of object station	Sets station number for object station.	1 to 64	User
(s1)+6 Array_s1[7]	Dummy	Not used	—	—
(s1)+7 Array_s1[8]	Number of transmission retries	Sets the number of retries to gain a completion of the READ instruction within the WDT time setting stored in (s1)+8 (Array_s1[9]).	1 to 15	User
	Number of executed transmission retries	Stores the number of executed transmission retries.		System
(s1)+8 Array_s1[9]	Transmission time setting of WDT	Sets the monitoring time for READ operations in seconds. If the operation is not completed within the set time, the transmission is repeated for the number of times set in (s1)+7 (Array_s1[8]).	1 to 32767 0 = 10 s (fixed)	User
(s1)+9 Array_s1[10]	Receive data length	Sets the number of data blocks to be read.	1 to 480	User
(s1)+10 Array_s1[11]	Dummy	Not used	—	—
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		
(s1)+16 Array_s1[17]	Number of network where error occurred	Stores the network number of the station in which the error occurred. The network number ranges between 1 and 239.	—	System
(s1)+17 Array_s1[18]	Number of station where error occurred	Stores the number of the station in which the error occurred. The number is not stored if the completion status of the instruction execution is "Channel in Use". The station number ranges between 1 and 64.	—	System

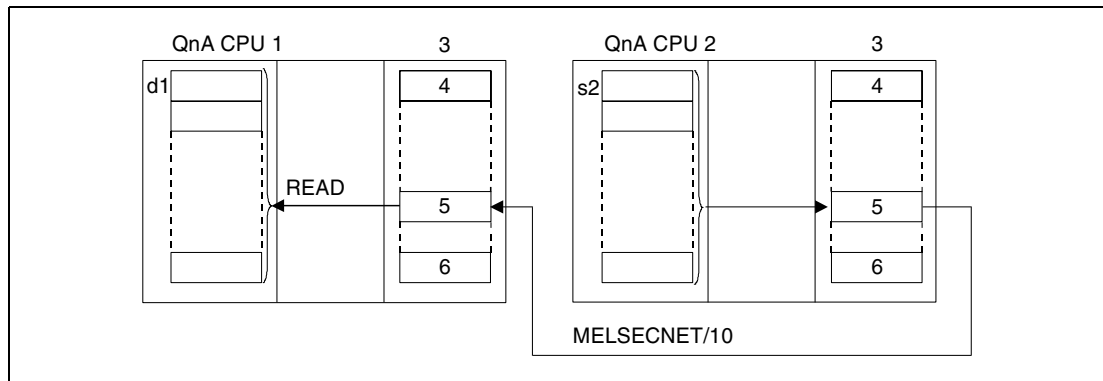
●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

●<sup>4</sup> The network number 254 is designated if set by Jn.

**Functions    Reading word device data from another station****READ    Read instructions**

The READ instruction reads the data stored from s2 onwards from a station connected to the MELSECNET/10. The station and network number are specified in the control data. The data read from the station are stored from d1 onwards in the host station.

After the completion of the read operation the device d2 in the object station is set.



- 1 Host station
- 2 Object station
- 3 Network module
- 4 Channel 1
- 5 Channel n
- 6 Channel 8

Through a relay station and set routing parameters also stations in different networks can be accessed.

Data link instructions cannot be executed in more than one location with common access to the same channel. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further data link instructions.

The execution status and the completion status (normal, not normal) of the READ instruction can be checked via

- the communications directive flag (●<sup>5</sup>) of the used channel,
- the host station completion device (d2) being set after completion of the operation,
- the status display of the operation completion (completion of an errorfree or faulty transmission) ((d2)+1)

as follows:

**Communications directive flag**

This flag is set during the execution of the READ instruction. The flag is reset with the execution of the END instruction during the program scan the read operation was completed in.

**Host station completion device**

This device is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing.

Status display of the operation completion

This device is set depending on the completion result of the read instruction.

Remains reset for a normal (errorfree) transmission.

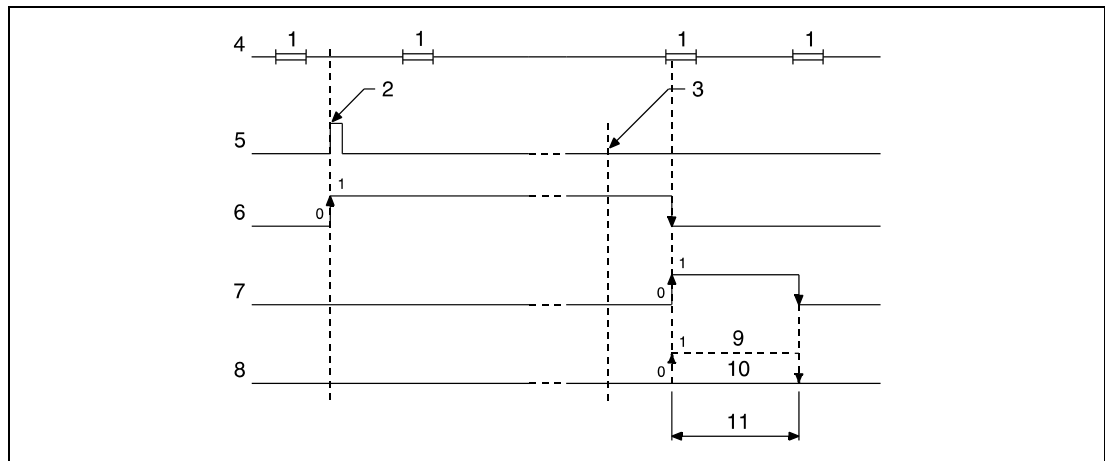
For the completion of a faulty transmission this device is set with the END instruction within the program scan the READ instruction was completed in. The device is reset with the next END processing.

**NOTE**

●<sup>5</sup> The following table assigns the channel numbers to the according communications channel flags:

Channel number	1	2	3	4	5	6	7	8
Communications channel flag	SB30	SB32	SB34	SB36	SB38	SB3A	SB3C	SB3E

The following figure shows the operations of the host station during the execution of a READ instruction:



- 1 END processing
- 2 Execution of the READ instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 READ instruction
- 6 Communications channel flag
- 7 Host station completion device set after completion of the operation (d2)
- 8 Status display of the operation completion ((d2)+1)
- 9 Completion of a faulty transmission
- 10 Completion of an errorfree transmission
- 11 One scan

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in Jn is not connected to the station (error code 4102).
- The module with the I/O address specified in Un is not a network module (error code 2111).

**8.6.2 SREAD**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

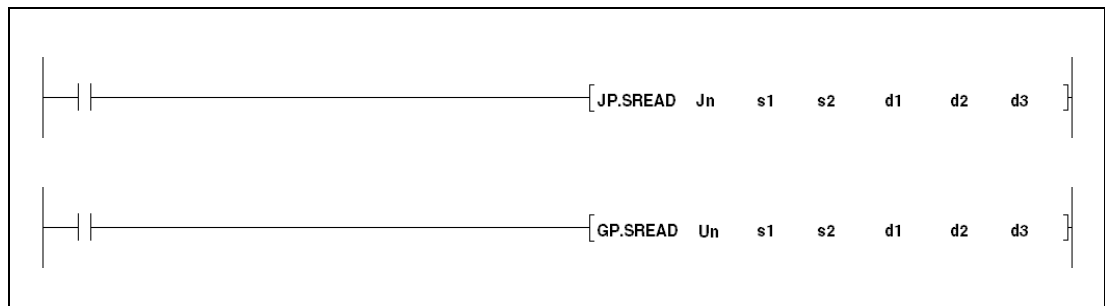
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—	SM0	13
s2	—	●	●	—	—	—	—	●	—		
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		
d3	●	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**

MELSEC Instruction List	Ladder Diagram	IEC Instruction List
<div style="border: 1px solid black; padding: 5px; width: fit-content;">MELSEC</div> <p>JP.SREAD Jn s1 s2 d1 d2 d3</p> <p>GP.SREAD Un s1 s2 d1 d2 d3</p>		<div style="border: 1px solid black; padding: 5px; width: fit-content;">SREAD</div> <p>SREAD_JP_M Jn , s1 , s2 , d1 , d2 , d3</p> <p>SREAD_UP_M Un , s1 , s2 , d1 , d2 , d3</p>

**GX  
Developer**



**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First device of host station storing control data.	Device number	Array [1..18] of ANY16
s2	First device of station storing data to be read.		ANY16
d1	First device of host station storing read data.		
d2	Device of host station set ON for 1 scan after completion of instruction.	Bit	BOOL
d3	Device of object station set ON for 1 scan after completion of instruction.		

**NOTE**

- <sup>1</sup> The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.
- <sup>2</sup> The head I/O number of the network unit for the host station must range within 0 and FEH. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.

The SREAD instruction can only be executed, if the object station is a QnA CPU.

With an ACPU in MELSECNET/10 the SREAD instruction cannot be applied.

Only station numbers for QnA CPUs are valid numbers for the object station.



## Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	The transmission confirmation is set: (Bit 0 (b0) = 1, fixed)	0001H 0081H	User
	Error completion mode	Storage of clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+11 (Array_s1[12]) onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Channel used by host station	Host channel designation.	1 to 8	User
(s1)+3 Array_s1[4]	Dummy	Not used	0	—
(s1)+4 Array_s1[5]	Network number of object station	Sets network number for station to be read from.	1 to 239 254 ● <sup>4</sup>	User
(s1)+5 Array_s1[6]	Number of object station	Sets station number for object station.	1 to 64	User
(s1)+6 Array_s1[7]	Dummy	Not used	—	—
(s1)+7 Array_s1[8]	Number of transmission retries	Sets the number of retries to gain a completion of the READ instruction within the WDT time setting stored in (s1)+8 (Array_s1[9]).	1 to 15	User
	Number of executed transmission retries	Stores the number of executed transmission retries.		System
(s1)+8 Array_s1[9]	Transmission time setting of WDT	Sets the monitoring time for READ operations in seconds. If the operation is not completed within the set time, the transmission is repeated for the number of times set in (s1)+7 (Array_s1[8]).	1 to 32767 0 = 10 s (fixed)	User
(s1)+9 Array_s1[10]	Receive data length	Sets the number of data blocks to be read.	1 to 480	User
(s1)+10 Array_s1[11]	Dummy	Not used	—	—
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		
(s1)+16 Array_s1[17]	Number of network where error occurred	Stores the network number of the station in which the error occurred. The network number ranges between 1 and 239.	—	System
(s1)+17 Array_s1[18]	Number of station where error occurred	Stores the number of the station in which the error occurred. The number is not stored if the completion status of the instruction execution is "Channel in Use". The station number ranges between 1 and 64.	—	System

●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

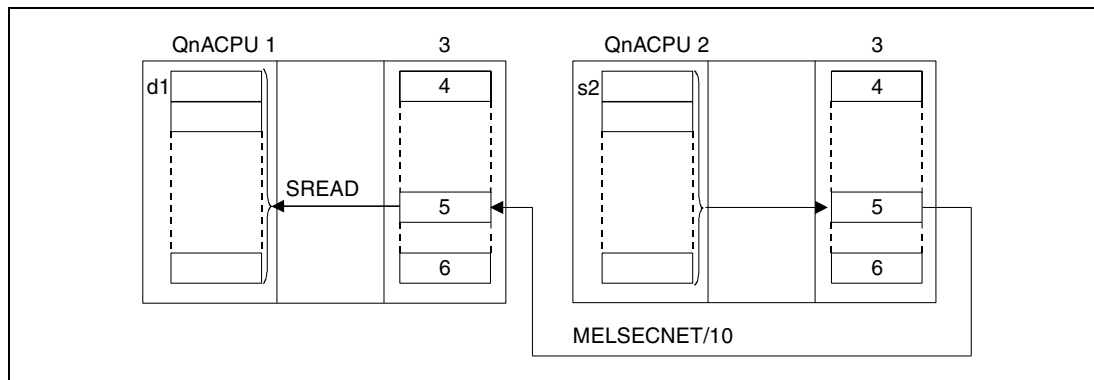
●<sup>4</sup> The network number 254 is designated if set by Jn.

## Functions Reading word device data from another station

### SREAD Read instructions

The SREAD instruction reads the data stored from s2 onwards from a station connected to the MELSECNET/10. The station and network number are specified in the control data. The data read from the station are stored from d1 onwards in the host station.

After the completion of the read operation the device d2 in the host station and the device d3 in the object station are set.



- <sup>1</sup> Host station
- <sup>2</sup> Object station
- <sup>3</sup> Network module
- <sup>4</sup> Channel 1
- <sup>5</sup> Channel n
- <sup>6</sup> Channel 8

Through a relay station and set routing parameters also stations in different networks can be accessed.

Data link instructions cannot be executed in more than one location with common access to the same channel. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further data link instructions.

The execution status and the completion status (normal, not normal) of the SREAD instruction can be checked via

- the communications directive flag (●<sup>5</sup>) of the used channel,
- the host station completion device (d2) and the object station completion device (d3) being set after completion of the operation,
- the status display of the operation completion (completion of an errorfree or faulty transmission) ((d2)+1)

as follows:

#### Communications directive flag

This flag is set during the execution of the SREAD instruction. The flag is reset with the execution of the END instruction during the program scan the read operation was completed in.

#### Host station completion device

This device is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing.

Status display of the operation completion

This device is set depending on the completion result of the read instruction.

Remains reset for a normal (errorfree) transmission.

For the completion of a faulty transmission this device is set with the END instruction within the program scan the SREAD instruction was completed in. The device is reset with the next END processing.

Object station completion device

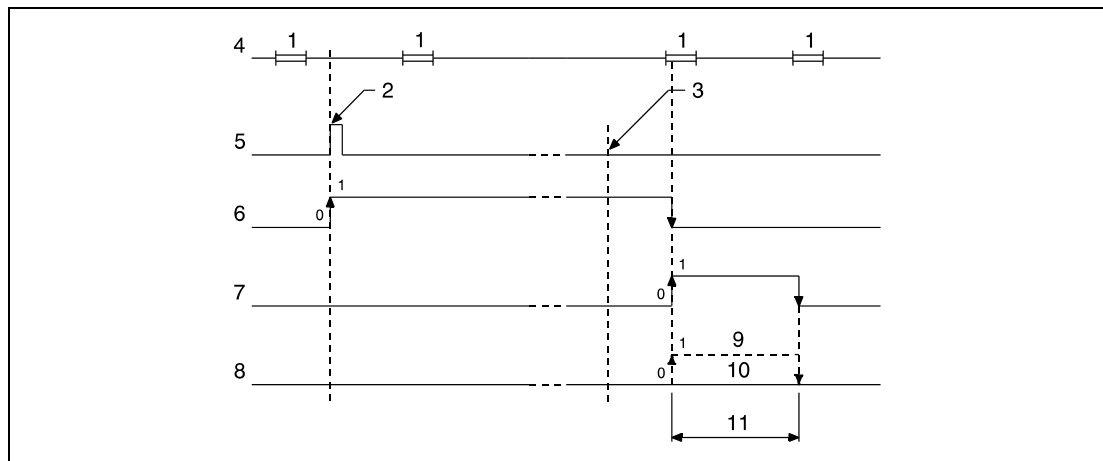
This device is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing.

**NOTE**

●<sup>5</sup> The following table assigns the channel numbers to the according communications channel flags:

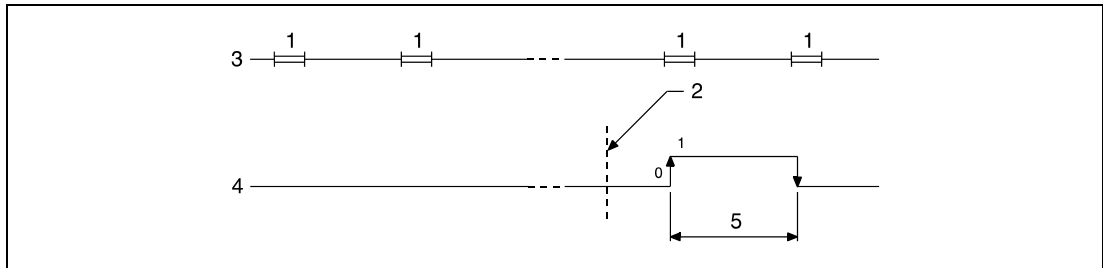
Channel number	1	2	3	4	5	6	7	8
Communications channel flag	SB30	SB32	SB34	SB36	SB38	SB3A	SB3C	SB3E

The following figure shows the operations of the host station during the execution of an SREAD instruction:



- 1 END processing
- 2 Execution of the SREAD instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 SREAD instruction
- 6 Communications channel flag
- 7 Host station completion device set after completion of the operation (d2)
- 8 Status display of the operation completion ((d2)+1)
- 9 Completion of a faulty transmission
- 10 Completion of an errorfree transmission
- 11 One scan

The following figure shows the operations of the object station during the execution of an SREAD instruction:



- 1 END processing
- 2 Completion of the operation
- 3 Program of the object station
- 4 Object station completion device set after completion of the operation (d3)
- 5 One scan

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in Jn is not connected to the station (error code 4102).
- The module with the I/O address specified in Un is not a network module (error code 2111).

**8.6.3 WRITE**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	SM0	12	
s2	—	●	●	—	—	—	—	—			
d1	●	●	●	—	—	—	—	—			
d2	●	●	●	—	—	—	—	—			

**GX IEC  
Developer**

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 10%;">MELSEC</td> <td style="width: 10%;">JP.WRITE</td> <td style="width: 10%;">Jn</td> <td style="width: 10%;">s1</td> <td style="width: 10%;">s2</td> <td style="width: 10%;">d1</td> <td style="width: 10%;">d2</td> </tr> <tr> <td></td> <td>GP.WRITE</td> <td>Un</td> <td>s1</td> <td>s2</td> <td>d1</td> <td>d2</td> </tr> </table>	MELSEC	JP.WRITE	Jn	s1	s2	d1	d2		GP.WRITE	Un	s1	s2	d1	d2	<p>Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">WRITE_JP_M</td> <td style="width: 10%;">Jn</td> <td style="width: 10%;">s1</td> <td style="width: 10%;">s2</td> <td style="width: 10%;">d1</td> <td style="width: 10%;">d2</td> </tr> <tr> <td></td> <td>WRITE_UP_M</td> <td>Un</td> <td>s1</td> <td>s2</td> <td>d1</td> <td>d2</td> </tr> </table>		WRITE_JP_M	Jn	s1	s2	d1	d2		WRITE_UP_M	Un	s1	s2	d1	d2
MELSEC	JP.WRITE	Jn	s1	s2	d1	d2																								
	GP.WRITE	Un	s1	s2	d1	d2																								
	WRITE_JP_M	Jn	s1	s2	d1	d2																								
	WRITE_UP_M	Un	s1	s2	d1	d2																								

**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First device of host station storing control data.	Device number	Array [1..18] of ANY16
s2	First device of station storing data to be written.		ANY16
d1	First device of object station storing written data.		
d2	Device set ON for 1 scan after completion of instruction.	Bit	BOOL

**NOTE**

- <sup>1</sup> *The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.*
- <sup>2</sup> *The head I/O number of the network unit for the host station must range within 0 and FEH. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.*

*The WRITE instruction can only be executed, if the object station is a QnA CPU.*

*With an ACPU in MELSECNET/10 the WRITE instruction cannot be applied.*

*The WRITE instruction can only address the number "FFH" (all stations in the object network) for networks with connected QnA CPUs exclusively. The number "FFH" cannot be designated in networks with mixed QnA and A CPUs.*

## Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	Confirmation of transmission completion = reset bit 0 to 0 No confirmation of transmission completion = set bit 0 to 1	0000H 0001H 0080H 0081H	User
	Error completion mode	Storage of clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+10 (Array_s1[11] onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Channel used by host station	Host channel designation.	1 to 8	User
(s1)+3 Array_s1[4]	Dummy	Not used	0	—
(s1)+4 Array_s1[5]	Network number of object station	Sets network number for object station.	1 to 239 254 ● <sup>4</sup>	User
(s1)+5 Array_s1[6]	Number of object station	Sets station number for object station.	Station number: 1 to 64 Group designation: 81H to 89H All stations in object network: FFH	User
(s1)+6 Array_s1[7]	Dummy	Not used	—	—
(s1)+7 Array_s1[8]	Number of transmission retries	Sets the number of retries to gain a completion of the WRITE instruction within the WDT time setting stored in (s1)+8 (Array_s1[9]). Active only if the execution mode (s1)+0 Array_s1[1] is set (1).	1 to 15	User
	Number of executed transmission retries	Stores the number of executed transmission retries.		System
(s1)+8 Array_s1[9]	Transmission time setting of WDT	Sets the monitoring time for WRITE operations in seconds. If the operation is not completed within the set time, the transmission is repeated for the number of times set in (s1)+7 (Array_s1[8]).	1 to 32767 0 = 10 s (fixed) Active only if the execution mode (s1)+0 Array_s1[1] is set (1).	User
(s1)+9 Array_s1[10]	Send data length	Sets the number of data blocks to be written.	1 to 480	User
(s1)+10 Array_s1[11]	Dummy	Not used	—	—
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System

Device	Meaning	Function	Value Range	Set by
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		
(s1)+16 Array_s1[17]	Number of network where error occurred	Stores the network number of the station in which the error occurred. The network number ranges between 1 and 239.	—	System
(s1)+17 Array_s1[18]	Number of station where error occurred	Stores the number of the station in which the error occurred. The number is not stored if the completion status of the instruction execution is "Channel in Use". The station number ranges between 1 and 64.	—	System

●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

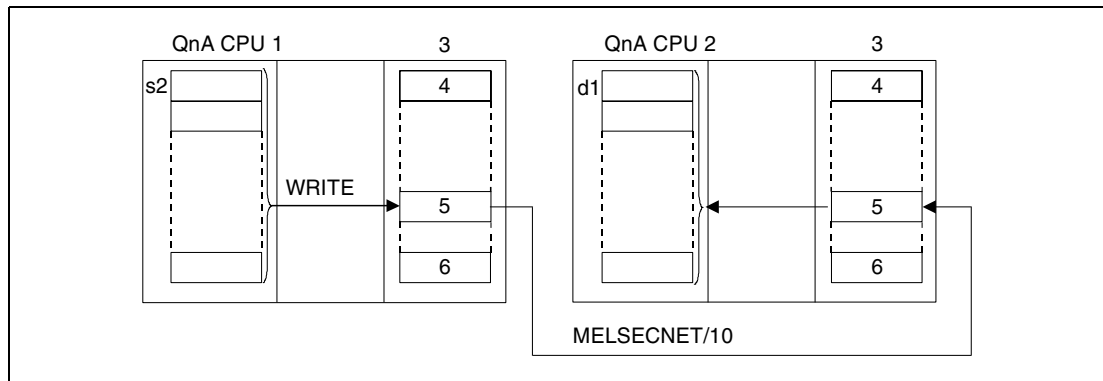
●<sup>4</sup> The network number 254 is designated if set by Jn.



**Functions Writing word device data to another station****WRITE Write instruction**

The WRITE instruction writes the data stored from s2 onwards from the host station to a station connected to the MELSECNET/10. The station and network number are specified in the control data. The data is stored from d1 onwards in the object station.

After the completion of the write operation the device d2 in the object station is set.



- <sup>1</sup> Host station
- <sup>2</sup> Object station
- <sup>3</sup> Network module
- <sup>4</sup> Channel 1
- <sup>5</sup> Channel n
- <sup>6</sup> Channel 8

Through a relay station and set routing parameters also stations in different networks can be accessed.

Data link instructions cannot be executed from more than one location with common access to the same channel. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further data link instructions.

The execution status and the completion status (normal, not normal) of the WRITE instruction can be checked via

- the communications directive flag (●<sup>5</sup>) of the used channel,
- the host station completion device (d2) being set after completion of the operation,
- the status display of the operation completion (completion of an errorfree or faulty transmission) ((d2)+1)

as follows:

**Communications directive flag**

This flag is set during the execution of the WRITE instruction. The flag is reset with the execution of the END instruction during the program scan the write operation was completed in.

**Host station completion device**

This device is set with the execution of the END instruction within the program scan the write operation was completed in. The device is reset with the next END processing.

Status display of the operation completion

This device is set depending on the completion result of the write instruction.

Remains reset for a normal (errorfree) transmission.

For the completion of a faulty transmission this device is set with the END instruction within the program scan the WRITE instruction was completed in. The device is reset with the next END processing.

Object station completion device

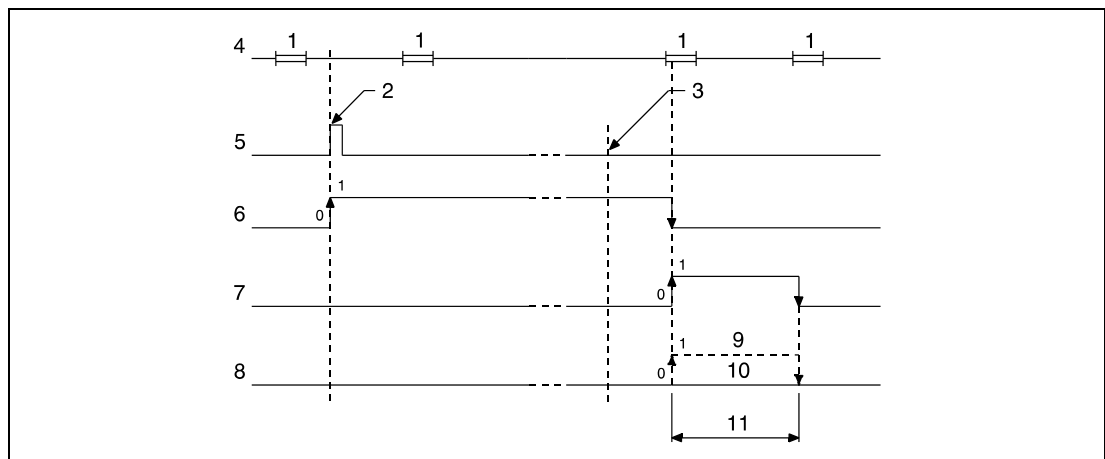
This device is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing.

**NOTE**

●<sup>5</sup> The following table assigns the channel numbers to the according communications channel flags:

Channel number	1	2	3	4	5	6	7	8
Communications channel flag	SB30	SB32	SB34	SB36	SB38	SB3A	SB3C	SB3E

The following figure shows the operations of the host station during the execution of a WRITE instruction:



- 1 END processing
- 2 Execution of the WRITE instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 WRITE instruction
- 6 Communications channel flag
- 7 Host station completion device set after completion of the operation (d2)
- 8 Status display of the operation completion ((d2)+1)
- 9 Completion of a faulty transmission
- 10 Completion of an errorfree transmission
- 11 One scan

**Operation  
Errors**

In the following cases an operation error occurs and the error flag is set:

- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in Jn is not connected to the station (error code 4102).
- The module with the I/O address specified in Un is not a network module (error code 2111).

**8.6.4 SWRITE**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	13
s2	—	●	●	—	—	—	—	—	—		
d1	●	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		
d3	●	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**

MELSEC Instruction List	Ladder Diagram	IEC Instruction List
<div style="border: 1px solid black; padding: 5px; width: fit-content;">MELSEC</div> <pre> JP.SWRITE  Jn             s1             s2             d1             d2             d3  GP.SWRITE  Un             s1             s2             d1             d2             d3                     </pre>		<pre> SWRITE_JP_M Jn , s1 , s2 , d1 , d2 , d3  SWRITE_UP_M Un , s1 , s2 , d1 , d2 , d3                     </pre>

**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First device of host station storing control data.	Device number	Array [1..18] of ANY16
s2	First device of station storing data to be written.		ANY16
d1	First device of object station storing written data.		
d2	Device set ON for 1 scan after completion of instruction.	Bit	BOOL
d3	Device of object station set ON for 1 scan after completion of instruction.		

**NOTE**

- <sup>1</sup> *The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.*
- <sup>2</sup> *The head I/O number of the network unit for the host station must range within 0 and FF<sub>H</sub>. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.*

*The SWRITE instruction can only be executed, if the object station is a QnA CPU.*

*With an ACPU in MELSECNET/10 the SWRITE instruction cannot be applied.*

*The WRITE instruction can only address the number "FF<sub>H</sub>" (all stations in the object network) for networks with connected QnA CPUs exclusively. The number "FF<sub>H</sub>" cannot be designated in networks with mixed QnA and A CPUs.*

Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	Confirmation of transmission completion = reset bit 0 to 0 No confirmation of transmission completion = set bit 0 to 1	0000H 0001H 0080H 0081H	User
	Error completion mode	Storage of clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+10 (Array_s1[11] onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Channel used by host station	Host channel designation.	1 to 8	User
(s1)+3 Array_s1[4]	Dummy	Not used	0	—
(s1)+4 Array_s1[5]	Network number of object station	Sets network number for object station.	1 to 239 254 ● <sup>4</sup>	User
(s1)+5 Array_s1[6]	Number of object station	Sets station number for object station.	Station number: 1 to 64 Group designation: 81H to 89H All stations in object network: FFH	User
(s1)+6 Array_s1[7]	Dummy	Not used	—	—
(s1)+7 Array_s1[8]	Number of transmission retries	Sets the number of retries to gain a completion of the WRITE instruction within the WDT time setting stored in (s1)+8 (Array_s1[9]). Active only if the execution mode (s1)+0 Array_s1[1] is set (1).	1 to 15	User
	Number of executed transmission retries	Stores the number of executed transmission retries.		System
(s1)+8 Array_s1[9]	Transmission time setting of WDT	Sets the monitoring time for WRITE operations in seconds. If the operation is not completed within the set time, the transmission is repeated for the number of times set in (s1)+7 (Array_s1[8]).	1 to 32767 0 = 10 s (fixed) Active only if the execution mode (s1)+0 Array_s1[1] is set (1).	User
(s1)+9 Array_s1[10]	Send data length	Sets the number of data blocks to be written.	1 to 480	User
(s1)+10 Array_s1[11]	Dummy	Not used	—	—
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System

Device	Meaning	Function	Value Range	Set by
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		
(s1)+16 Array_s1[17]	Number of network where error occurred	Stores the network number of the station in which the error occurred. The network number ranges between 1 and 239.	—	System
(s1)+17 Array_s1[18]	Number of station where error occurred	Stores the number of the station in which the error occurred. The number is not stored if the completion status of the instruction execution is "Channel in Use". The station number ranges between 1 and 64.	—	System

●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

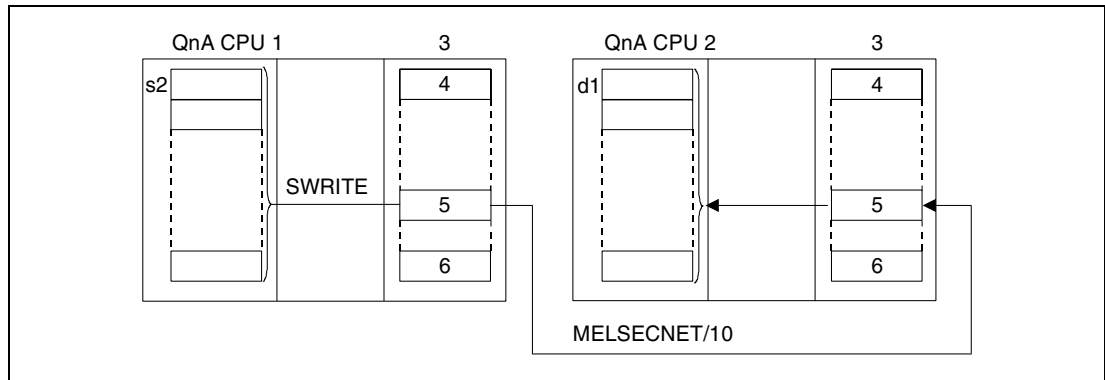
●<sup>4</sup> The network number 254 is designated if set by Jn.

**Functions Writing word device data to another station**

**SWRITE Write instruction**

The SWRITE instruction writes the data stored from s2 onwards from the host station to a station connected to the MELSECNET/10. The station and network number are specified in the control data. The data is stored from d1 onwards in the object station.

After the completion of the write operation the device d2 in the host station and the device d3 in the object station are set.



- <sup>1</sup> Host station
- <sup>2</sup> Object station
- <sup>3</sup> Network module
- <sup>4</sup> Channel 1
- <sup>5</sup> Channel n
- <sup>6</sup> Channel 8

Through a relay station and set routing parameters also stations in different networks can be accessed.

Data link instructions cannot be executed from more than one location with common access to the same channel. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further data link instructions.

The execution status and the completion status (normal, not normal) of the SWRITE instruction can be checked via

- the communications directive flag (●<sup>5</sup>) of the used channel,
- the host station completion devices in the host station (d2) and in the object station (d3) being set after completion of the operation,
- the status display of the operation completion (completion of an errorfree or faulty transmission) ((d2)+1)

as follows:

**Communications directive flag**

This flag is set during the execution of the SWRITE instruction. The flag is reset with the execution of the END instruction during the program scan the write operation was completed in.

**Host station completion device**

This device is set with the execution of the END instruction within the program scan the write operation was completed in. The device is reset with the next END processing.



Status display of the operation completion

This device is set depending on the completion result of the write instruction.

Remains reset for a normal (errorfree) transmission.

For the completion of a faulty transmission this device is set with the END instruction within the program scan the SWRITE instruction was completed in. The device is reset with the next END processing.

Object station completion device

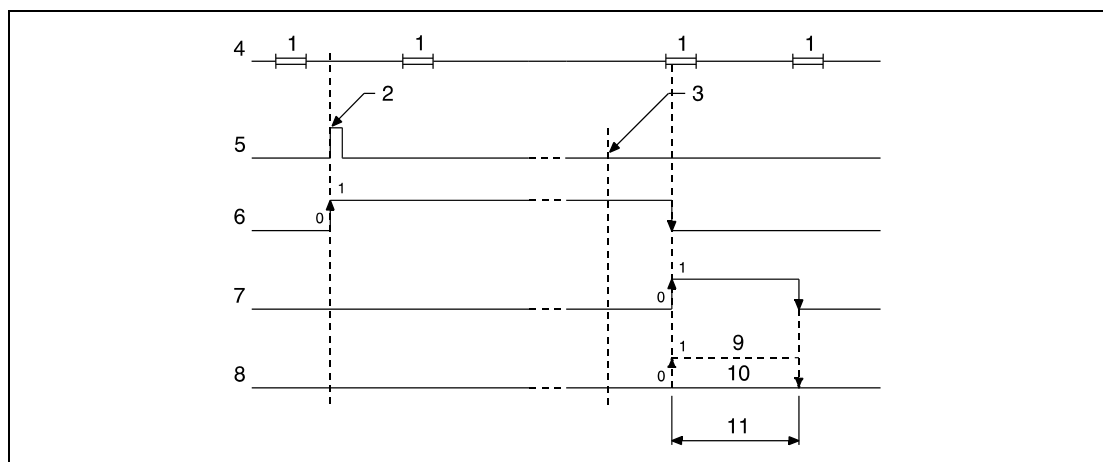
This device is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing.

**NOTE**

●<sup>5</sup> The following table assigns the channel numbers to the according communications channel flags:

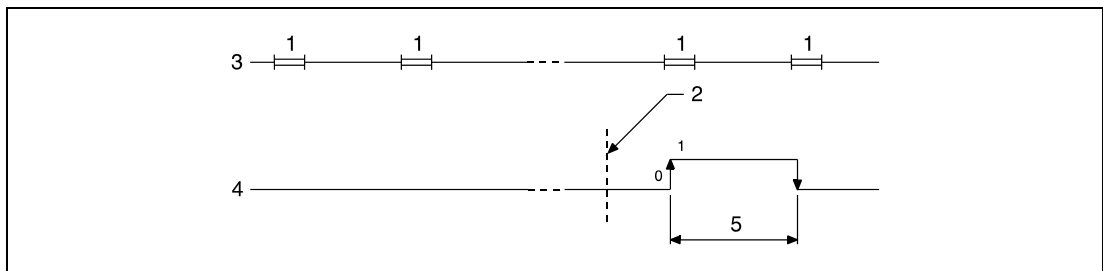
Channel number	1	2	3	4	5	6	7	8
Communications channel flag	SB30	SB32	SB34	SB36	SB38	SB40	SB42	SB44

The following figure shows the operations of the host station during the execution of an SWRITE instruction:



- 1 END processing
- 2 Execution of the SWRITE instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 SWRITE instruction
- 6 Communications channel flag
- 7 Host station completion device set after completion of the operation (d2)
- 8 Status display of the operation completion ((d2)+1)
- 9 Completion of a faulty transmission
- 10 Completion of an errorfree transmission
- 11 One scan

The following figure shows the operations of the object station during the execution of an SWRITE instruction:



<sup>1</sup> END processing

<sup>2</sup> Completion of the operation

<sup>3</sup> Program of the object station

<sup>4</sup> Object station completion device set after completion of the operation (d3)

<sup>5</sup> One scan

## Operation Errors

In the following cases an operation error occurs and the error flag is set:

- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in Jn is not connected to the station (error code 4102).
- The module with the I/O address specified in Un is not a network module (error code 2111).

**8.6.5 SEND**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	10
s2	—	●	●	—	—	—	—	—	—		
d	●	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**

<p style="text-align: center;">MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">MELSEC</p> </div> <p>JP.SEND    Jn                   s1                   s2                   d</p> <p>GP.SEND    Un                   s1                   s2                   d</p>	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">IEC</p> </div> <p>SEND_JP_M    Jn , s1 , s2 , d</p> <p>SEND_UP_M    Un , s1 , s2 , d</p>
---	---	---

**Variables**

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First number of device storing control data.	Device number	Array [1..18] of ANY16
s2	First number of device storing data to be sent.		ANY16
d	Device set ON for 1 scan after completion of instruction.	Bit	BOOL

**NOTE**

- <sup>1</sup> The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.
- <sup>2</sup> The head I/O number of the network unit for the host station must range within 0 and FEH. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.

## Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	Confirmation of transmission completion = reset bit 0 to 0 No confirmation of transmission completion = set bit 0 to 1	0000H 0001H 0080H 0081H	User
	Error completion mode	Stores clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+10 (Array_s1[11] onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Channel used by host station	Host channel designation.	1 to 8	User
(s1)+3 Array_s1[4]	Channel used by object station	Object station designation.	1 to 8	User
(s1)+4 Array_s1[5]	Network number of object station	Sets network number for object station.	1 to 239 254 ● <sup>4</sup>	User
(s1)+5 Array_s1[6]	Number of object station	Sets station number for object station.	Station number: 1 to 64 Group designation: 81H to 89H All stations in object network: FFH	User
(s1)+6 Array_s1[7]	Dummy	Not used	—	—
(s1)+7 Array_s1[8]	Number of transmission retries	Sets the number of retries to gain a completion of the WRITE instruction within the WDT time setting stored in (s1)+8 (Array_s1[9]). Active only if the execution mode (s1)+0 Array_s1[1] is set (1).	1 to 15	User
	Number of executed transmission retries	Stores the number of executed transmission retries.		System
(s1)+8 Array_s1[9]	Transmission time setting of WDT	Sets the monitoring time for WRITE operations in seconds. If the operation is not completed within the set time, the transmission is repeated for the number of times set in (s1)+7 (Array_s1[8]).	1 to 32767 0 = 10 s (fixed) Active only if the execution mode (s1)+0 Array_s1[1] is set (1).	User
(s1)+9 Array_s1[10]	Send data length	Sets the number of data blocks to be written.	1 to 480	User
(s1)+10 Array_s1[11]	Dummy	Not used	—	—
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System

Device	Meaning	Function	Value Range	Set by
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		
(s1)+16 Array_s1[17]	Number of network where error occurred	Stores the network number of the station in which the error occurred. The network number ranges between 1 and 239.	—	System
(s1)+17 Array_s1[18]	Number of station where error occurred	Stores the number of the station in which the error occurred. The number is not stored if the completion status of the instruction execution is "Channel in Use". The station number ranges between 1 and 64.	—	System

●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

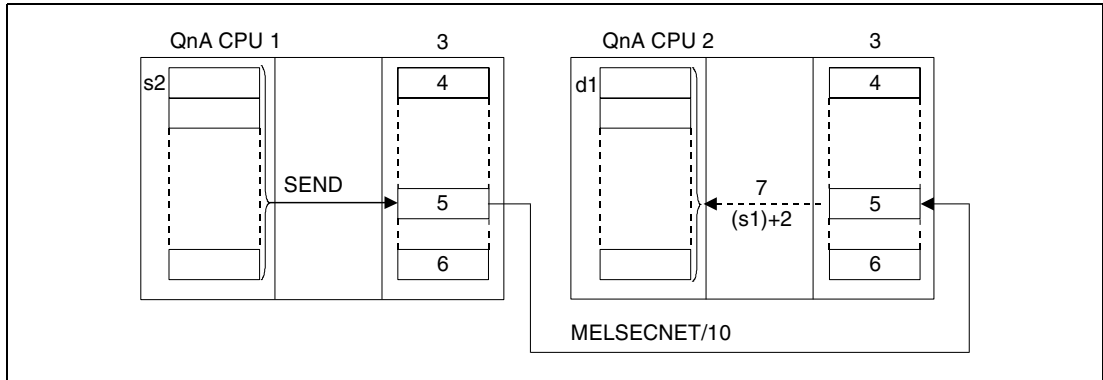
●<sup>4</sup> The network number 254 is designated if set by Jn.

**Functions**    **Sending data to other stations**

**SEND    Send instruction**

The SEND instruction sends the data stored from s2 onwards from the host station to a station connected to the MELSECNET/10. The transfer channel is specified in (s1)+2. The station and network number are specified in the control data.

After the completion of the write operation in the object station the device specified in d is set.



- 1 Host station
- 2 Object station
- 3 Network module
- 4 Channel 1
- 5 Channel n
- 6 Channel 8
- 7 The read operation is triggered by the RECV instruction

Through a relay station and set routing parameters also stations in different networks can be accessed.

Data link instructions cannot be executed from more than one location with common access to the same channel. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further data link instructions.

The execution status and the completion status (normal, not normal) of the SEND instruction can be checked via

- the communications directive flag (●<sup>5</sup>) of the used channel,
- the completion device (d) being set after completion of the operation,
- the status display of the operation completion (completion of an errorfree or faulty transmission) (d+1)

as follows:

**Communications directive flag**

This flag is set during the execution of the SEND instruction. The flag is reset with the execution of the END instruction during the program scan the operation was completed in.

**Host station completion device**

This device is set with the execution of the END instruction within the program scan the write operation was completed in. The device is reset with the next END processing.

Status display of the operation completion

This device is set depending on the completion result of the write instruction.

Remains reset for a normal (errorfree) transmission.

For the completion of a faulty transmission this device is set with the END instruction within the program scan the SEND instruction was completed in. The device is reset with the next END processing.

Object station completion device

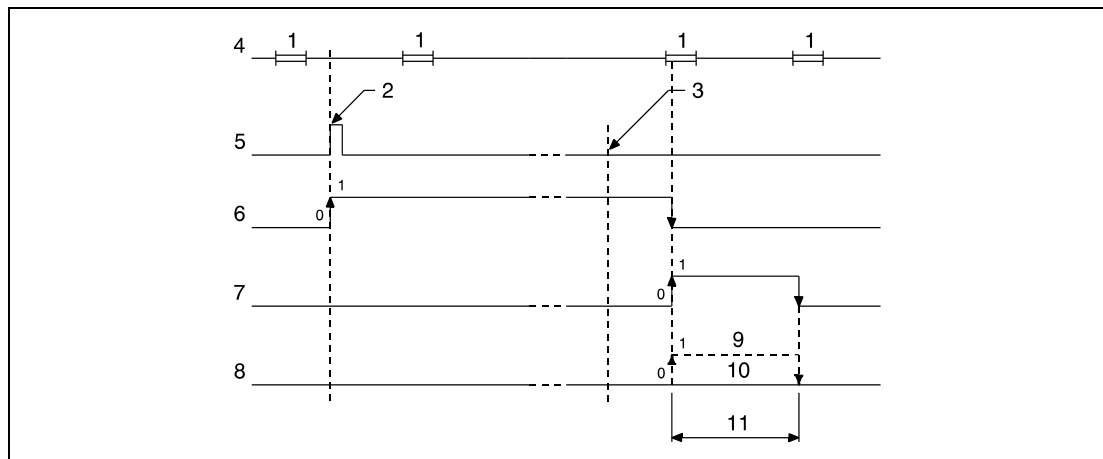
This device is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing.

**NOTE**

●<sup>5</sup> The following table assigns the channel numbers to the according communications channel flags:

Channel number	1	2	3	4	5	6	7	8
Communications channel flag	SB30	SB32	SB34	SB36	SB38	SB3A	SB3C	SB3E

The following figure shows the operations of the host station during the execution of a SEND instruction:



- 1 END processing
- 2 Execution of the SEND instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 SEND instruction
- 6 Communications channel flag
- 7 Host station completion device set after completion of the operation (d)
- 8 Status display of the operation completion (d+1)
- 9 Completion of a faulty transmission
- 10 Completion of an errorfree transmission
- 11 One scan

**Operation  
Errors**

In the following cases an operation error occurs and the error flag is set:

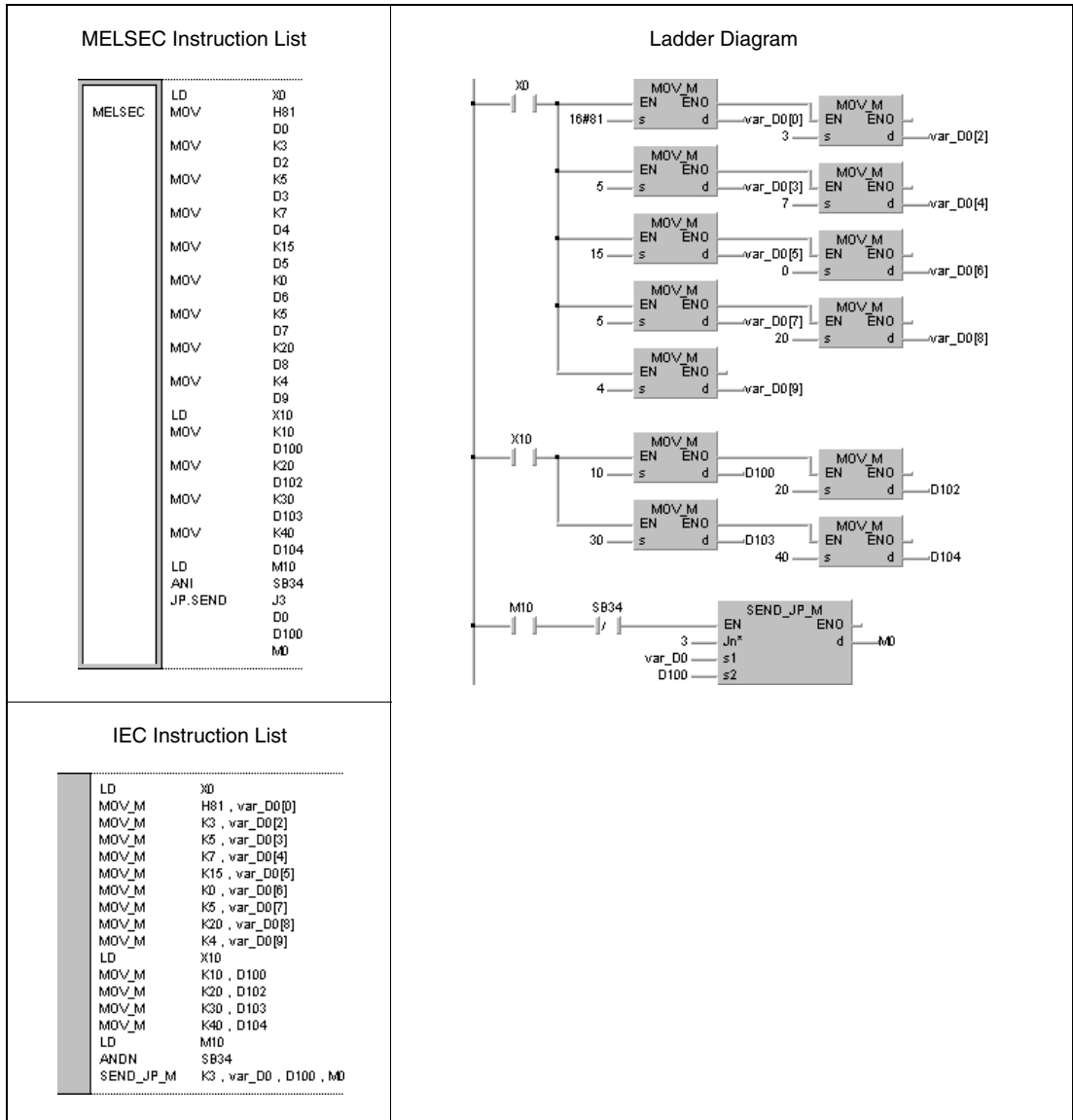
- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in Jn is not connected to the station (error code 4102).
- The module with the I/O address specified in Un is not a network module (error code 2111).



**Program** JP.SEND**Example**

With leading edge from M10, the following program sends data from the host station to an object station. The execution of the SEND instruction is interlocked via the normally open contact of the flag SB34. The following table contains further information on the host station, the object station, and the applied MOV instructions.

Device/Instruction	Meaning/Function
Host station	–
Host network	7
Host channel	3
Communications channel flag	SB34
Object station	15
Object network	5
Object channel	5
1. MOV instruction	Sets the input condition and the clock data
2. MOV instruction	Sets the channel for the host station
3. MOV instruction	Sets the channel for object station
4. MOV instruction	Sets the network number for the object station
5. MOV instruction	Sets the number for the object station
6. MOV instruction	–
7. MOV instruction	Sets the number of transmission retries
8. MOV instruction	Sets the WDT time setting (20 s)
9. MOV instruction	Sets the number of blocks to be sent (4)
10. MOV instruction	Sets the data to be sent
11. MOV instruction	
12. MOV instruction	
13. MOV instruction	



**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

### 8.6.6 RECV

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	SM0	9	
d1	—	●	●	—	—	—	—	—			
d2	●	●	●	—	—	—	—	—			

GX IEC Developer

<p style="text-align: center;">MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">MELSEC</p> </div> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">JP.RECV</td> <td style="width: 10%; text-align: center;">Jn</td> <td style="width: 10%; text-align: center;">s</td> <td style="width: 10%; text-align: center;">d1</td> <td style="width: 10%; text-align: center;">d2</td> </tr> <tr> <td>GP.RECV</td> <td style="text-align: center;">Un</td> <td style="text-align: center;">s</td> <td style="text-align: center;">d1</td> <td style="text-align: center;">d2</td> </tr> </table>	JP.RECV	Jn	s	d1	d2	GP.RECV	Un	s	d1	d2	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">RECV_JP_M</td> <td style="width: 10%; text-align: center;">Jn</td> <td style="width: 10%; text-align: center;">s</td> <td style="width: 10%; text-align: center;">d1</td> <td style="width: 10%; text-align: center;">d2</td> </tr> <tr> <td>RECV_UP_M</td> <td style="text-align: center;">Un</td> <td style="text-align: center;">s</td> <td style="text-align: center;">d1</td> <td style="text-align: center;">d2</td> </tr> </table>	RECV_JP_M	Jn	s	d1	d2	RECV_UP_M	Un	s	d1	d2
JP.RECV	Jn	s	d1	d2																		
GP.RECV	Un	s	d1	d2																		
RECV_JP_M	Jn	s	d1	d2																		
RECV_UP_M	Un	s	d1	d2																		

Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s	First number of device storing control data.	Device number	Array [1..16] of ANY16
d1	First number of device storing data to be sent.		ANY16
d2	Device set ON for 1 scan after completion of instruction.	Bit	BOOL

NOTE

- <sup>1</sup> The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.
- <sup>2</sup> The head I/O number of the network unit for the host station must range within 0 and FE<sub>H</sub>. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.

Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	Waiting for data (this mode waits for data repeatedly with a fixed time setting) (bit 0 (b0) = 0, fixed)	0000H 0080H	User
	Error completion mode	Stores clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+10 (Array_s1[11] onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Channel used by host station	Sets channel storing data to be received.	1 to 8	User
(s1)+3 Array_s1[4]	Channel used by object station	Stores channel for the sending station.	1 to 8	System
(s1)+4 Array_s1[5]	Network number of object station	Stores network number for the sending station.	1 to 239	System
(s1)+5 Array_s1[6]	Number of object station	Stores station number for the sending station.	1 to 64	System
(s1)+6 Array_s1[7]	Dummy	Not used	—	—
(s1)+7 Array_s1[8]	Dummy	Not used	—	—
(s1)+8 Array_s1[9]	Transmission time setting of WDT	Sets the monitoring time for the operation in seconds.	1 to 32767 0 = 10 s (fixed) Active only if the execution mode is set (1).	User
(s1)+9 Array_s1[10]	Send data length	Stores number of received data blocks.	1 to 480	System
(s1)+10 Array_s1[11]	Dummy	Not used	—	—
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		

●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

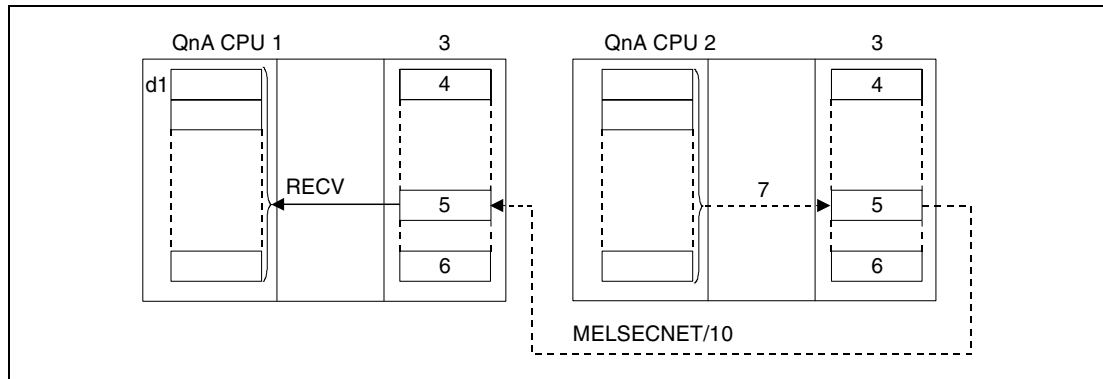
## Functions

### Receiving sent data from other stations

#### RECV Receive instruction

The RECV instruction receives the data sent via the SEND instruction from a station connected to the MELSECNET/10. The station and network numbers are specified in the control data. The data is stored from d1 onwards.

After the completion of the operation the device specified in d2 is set.



- 1 Host station
- 2 Object station
- 3 Network module
- 4 Channel 1
- 5 Channel n
- 6 Channel 8
- 7 The write operation is triggered via the SEND instruction

Through a relay station and set routing parameters also stations in different networks can be accessed.

Data link instructions cannot be executed from more than one location with common access to the same channel. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further data link instructions.

The execution status and the completion status (normal, not normal) of the RECV instruction can be checked via

- the communications directive flag (●<sup>4</sup>) of the used channel,
- the completion device (d2) being set after completion of the operation,
- the status display of the operation completion (completion of an errorfree or faulty transmission) ((d2)+1)

as follows:

#### Communications directive flag

This flag is set during the execution of the RECV instruction. The flag is reset with the execution of the END instruction during the program scan the operation was completed in.

#### Host station completion device

This device is set with the execution of the END instruction within the program scan the write operation was completed in. The device is reset with the next END processing.

Status display of the operation completion

This device is set depending on the completion result of the instruction.

Remains reset for a normal (errorfree) transmission.

For the completion of a faulty transmission this device is set with the END instruction within the program scan the RECV instruction was completed in. The device is reset with the next END processing.

Object station completion device

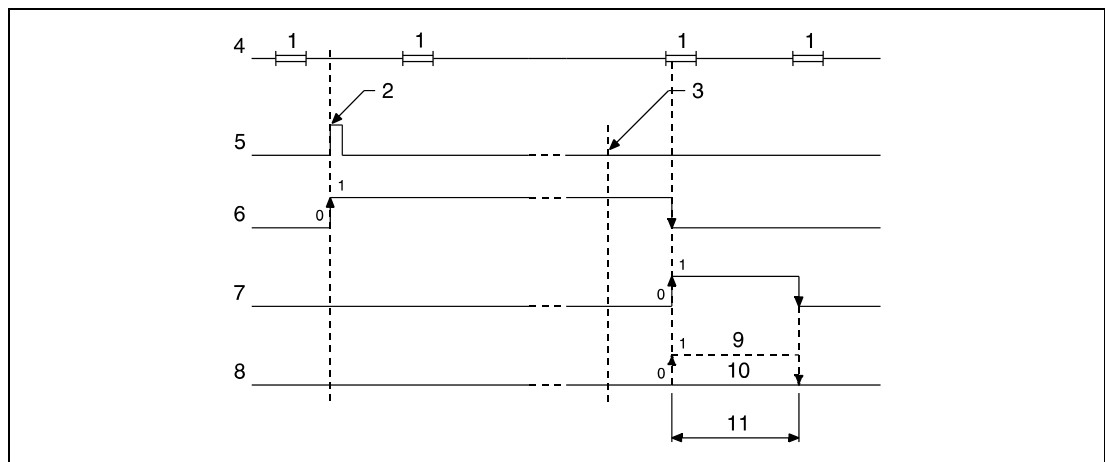
This device is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing.

**NOTE**

●<sup>4</sup> The following table assigns the channel numbers to the according communications channel flags:

Channel number	1	2	3	4	5	6	7	8
Communications channel flag	SB30	SB32	SB34	SB36	SB38	SB3A	SB3C	SB3E

The following figure shows the operations of the host station during the execution of a RECV instruction:



- <sup>1</sup> END processing
- <sup>2</sup> Execution of the RECV instruction
- <sup>3</sup> Completion of the operation
- <sup>4</sup> Program of the host station
- <sup>5</sup> RECV instruction
- <sup>6</sup> Communications channel flag
- <sup>7</sup> Host station completion device set after completion of the operation (d2)
- <sup>8</sup> Status display of the operation completion ((d2)+1)
- <sup>9</sup> Completion of a faulty transmission
- <sup>10</sup> Completion of an errorfree transmission
- <sup>11</sup> One scan

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

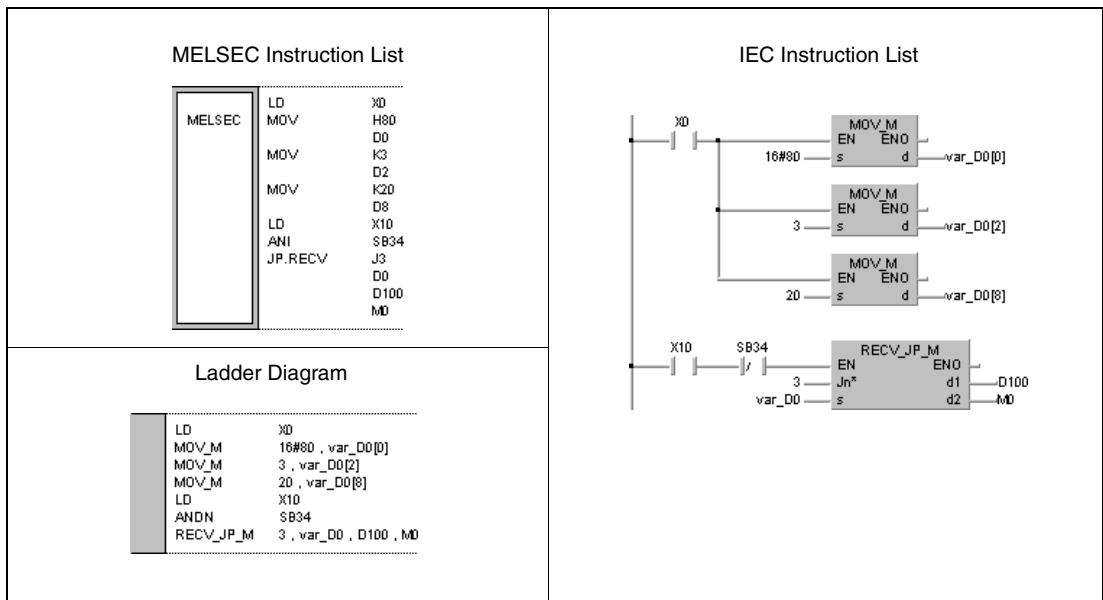
- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in Jn is not connected to the station (error code 4102).
- The module with the I/O address specified in Un is not a network module (error code 2111).

**Program Example**

JP.RECV

With leading edge from X10, the following program reads data sent from a station via the SEND instruction. The execution of the RECV instruction is interlocked via the normally open contact of the flag SB34. The following table contains further information on the host station, the sending station, and the applied MOV instructions.

Device/Instruction	Meaning/Function
Host station	–
Host network	–
Host channel	3
Communications channel flag	SB34
Sending station	–
Network for the sending station	3
Channel for the sending station	3
1. MOV instruction	Sets the clock data
2. MOV instruction	Sets the channel for the host station
3. MOV instruction	Sets the WDT time setting (20 s)



**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

8.6.7 REQ

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	10
s2	—	●	●	—	—	—	—	—	—		
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		

GX IEC Developer

MELSEC Instruction List	Ladder Diagram	IEC Instruction List
<div style="border: 1px solid black; padding: 5px; width: fit-content;">MELSEC</div> <pre> JP.REQ      Jn               s1               s2               d1               d2  GP.REQ      Un               s1               s2               d1               d2                     </pre>		<pre> REQ_JP_M     Jn , s1 , s2 , d1 , d2 REQ_UP_M     Un , s1 , s2 , d1 , d2                     </pre>

Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First number of device storing control data.	Device number	Array [1..18] of ANY16
s2	First number of device storing requested data.		Array [1..7] of ANY16
d1	First number of device storing response data.		Array [1..4] of ANY16
d2	Device set ON for 1 scan after completion of instruction.	Bit	BOOL



**NOTE**

- <sup>1</sup> *The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.*
- <sup>2</sup> *The head I/O number of the network unit for the host station must range within 0 and FEH. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.*

*The REQ instruction can only be executed, if the object station is a QnA CPU.*

*With an ACPUs in MELSECNET/10 the REQ instruction cannot be applied.*

*Only station numbers for QnA CPUs are valid numbers for the object station.*

## Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	Confirmation of transmission completion is set (Bit 0 (b0) = 1, fixed)	0001H 0081H	User
	Error completion mode	Stores clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+10 (Array_s1[11] onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>1</sup>	—	System
(s1)+2 Array_s1[3]	Channel used by host station	Sets channel used by host station.	1 to 8	User
(s1)+3 Array_s1[4]	Dummy	Not used	—	—
(s1)+4 Array_s1[5]	Network number for object station.	Sets number of network for station to read from	1 to 239 254 ● <sup>2</sup>	User
(s1)+5 Array_s1[6]	Number for object station.	Sets number for object station.	1 to 64	User
(s1)+6 Array_s1[7]	Dummy	Not used	—	—
(s1)+7 Array_s1[8]	Number of transmission retries	Sets the number of retries to gain a completion of the REQ instruction within the WDT time setting stored in (s1)+8 (Array_s1[9]).	1 bis 15	User
	Number of executed transmission retries	Stores the number of executed transmission retries.	—	System
(s1)+8 Array_s1[9]	Transmission time setting of WDT	Sets the monitoring time for the operation in seconds. If the operation is not completed within the set time, the transmission is repeated for the number of times set in (s1)+7 (Array_s1[8]).	1 to 32767 0 = 10 seconds (fixed)	User
(s1)+9 Array_s1[10]	Length of request data.	Sets the length of requested data. If clock data is read = 2 If clock data is written = 7 During remote RUN/STOP = 4	2, 7, 4	User
(s1)+10 Array_s1[11]	Length of response data	Stores the length of response data. If clock data is read = 2	0, 4	User
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		
(s1)+16 Array_s1[17]	Number of network where error occurred	Stores the network number of the station in which the error occurred. The network number ranges from 1 to 239. The number is not stored if the completion status of the instruction execution is "Channel in Use (F7C1H)".	—	System

Device	Meaning	Function	Value Range	Set by
(s1)+17 Array_s1[18]	Number of station where error occurred	Stores the number of the station in which the error occurred. The station number ranges from 1 to 64. The number is not stored if the completion status of the instruction execution is "Channel in Use(F7C1H)".	—	System

●<sup>1</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

●<sup>2</sup> The network number 254 is designated if set by Jn.

**Request/response data during write/read operation of clock data**

**Request data**

Device	Meaning	Function	Read Clock Data	Write Clock Data
(s2)+0 Array_s2[1]	Request type	0001H = Read clock data 0011H = Write clock data	●	●
(s2)+1 Array_s2[2]	Request type of subroutine	0002H = Read clock data 0001H = Write clock data	●	●
(s2)+2 Array_s2[3]	Update pattern	Specify clock data item in (s2)+3 (Array_s2[4]) through (s2)+6 (Array_s2[7]) to be updated. If the device is set (1) the clock/ data item is updated.  b15 ----- b7 b6 b5 b4 b3 b2 b1 b0 0 [ ] 0 W Sec Min H D M Y		●
(s2)+3 Array_s2[4]	Month and year to be updated	Month and year stored in BCD code (last two digits).  b15 ----- b8 b7 ----- b0 M (01H - 12H) Y (00H - 99H)		●
(s2)+4 Array_s2[5]	Hour and day to be updated	Hour and day stored in BCD code.  b15 ----- b8 b7 ----- b0 H (00H - 23H) D (01H - 31H)		●
(s2)+5 Array_s2[6]	Minute and second to be updated	Second and minute stored in BCD code.  b15 ----- b8 b7 ----- b0 Sec (00H - 59H) Min (00H - 59H)		●
(s2)+6 Array_s2[7]	Day of week to be updated	Day of week stored in BCD code (00H = Sunday, 06H = Saturday).  b15 ----- b8 b7 ----- b0 00H W (00H - 06H)		●

M = Month  
 Y = Year  
 H = Hour  
 D = Day  
 Sec = Second  
 Min = Minute  
 W = Day of week

## Response data

Device	Meaning	Function	Read Clock Data	Write Clock Data
(d1)+0 Array_d1[1]	Month and year being read	Month and year stored in BCD code (last two digits).   b15 ----- b8 b7 ----- b0 M (01H - 12H)   Y (00H - 99H)	●	
(d1)+1 Array_d1[2]	Hour and day being read	Hour and day stored in BCD code.  b15 ----- b8 b7 ----- b0 H (00H - 23H)   D (01H - 31H)	●	
(d1)+2 Array_d1[3]	Minute and second being read	Second and minute stored in BCD code.  b15 ----- b8 b7 ----- b0 Sec (00H - 59H)   Min (00H - 59H)	●	
(d1)+3 Array_d1[4]	Day of week being read	Day of week stored in BCD code (00H = Sunday, 06H = Saturday).  b15 ----- b8 b7 ----- b0 00H   W (00H - 06H)	●	

M = Month  
Y = Year  
H = Hour  
D = Day  
Sec = Second  
Min = Minute  
W = Day of week

**NOTE**

*Write/ read operations are disabled if the "Memory Protect" function is engaged on the CPU of the object station (system switch 1, SW5 (QnA, Q4AR), SW1 (QnAS) set ON).*

**Request data during RUN/STOP operation at a remote station****Request data**

Device	Meaning	Function	RUN Operation	STOP Operation
(s2)+0 Array-s2[1]	Request type	0010H	●	●
(s2)+1 Array_s2[2]	Request type of subroutine	0001H = RUN operation at a remote station 0002H = STOP operation at a remote station	●	●
(s2)+2 Array_s2[3]	Mode	Set forced RUN operation at a remote station: 0001H = Do not force RUN 0003H = Force RUN (set during remote STOP) If the station performing a STOP operation at a remote station cannot execute a RUN operation, the remote RUN operation can be forced from a different station.	●	●
(s2)+3 Array_s2[4]	Clear mode	Set memory status of the CPU during execution of the RUN operation at a remote station: 0000H = Do not clear (set during remote STOP) 0001H = Clear (exclusive latch range) 0002H = Clear (including latch range)	●	●

**NOTE**

*The RUN/ STOP function can only be executed, if the RUN/ STOP key switch of the CPU on the object station is set to RUN.*

*Write/ read operations are disabled if the "Memory Protect" function is engaged on the CPU of the object station (system switch 1, SW5 (QnA, Q4AR), SW1 (QnAS) set ON).*

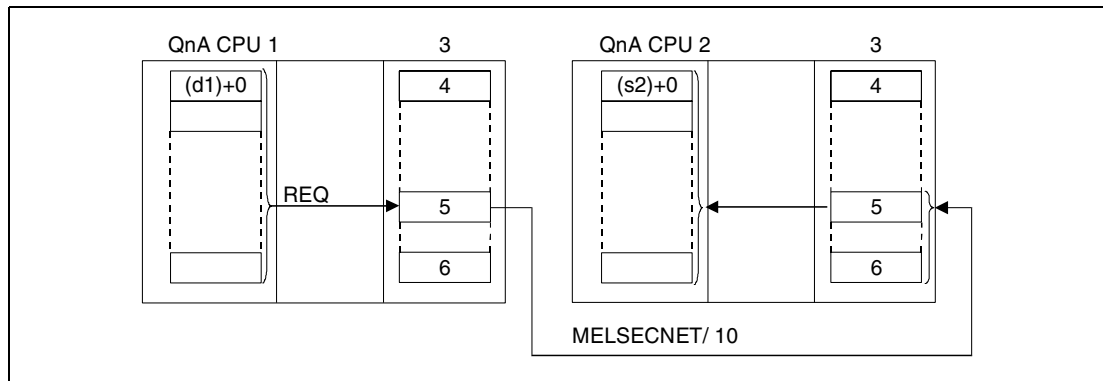
*If the object station is already set into the remote STOP/ PAUSE mode by a different station, the RUN operation can only be forced if the mode in (s2)+2 is set to "do not force RUN (0001H)".*

*If the QnA CPU of the object station executing the RUN/ STOP operation is reset the information of the remote RUN/ STOP operation in the object station will be lost.*

**Functions Request data from other stations****REQ Request instruction**

The REQ instruction transfers requested data stored from (d1)+0 (Array \_d1[1]) onwards from a station connected to the MELSECNET/10. The station number and network number are specified in the control data. The data is stored from (s2)+0 (Array \_s2[1]) onwards.

After the completion of the operation the device specified in d2 is set.



- <sup>1</sup> Host station
- <sup>2</sup> Object station
- <sup>3</sup> Network module
- <sup>4</sup> Channel 1
- <sup>5</sup> Channel n
- <sup>6</sup> Channel 8

Through a relay station and set routing parameters also stations in different networks can be accessed.

Data link instructions cannot be executed from more than one location with common access to the same channel. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further data link instructions.

The execution status and the completion status (normal, not normal) of the REQ instruction can be checked via

- the communications directive flag (●<sup>3</sup>) of the used channel,
- the completion device (d2) being set after completion of the operation,
- the status display of the operation completion (completion of an errorfree or faulty transmission) ((d2)+1)

as follows:

**Communications directive flag**

This flag is set during the execution of the REQ instruction. The flag is reset with the execution of the END instruction during the program scan the operation was completed in.

**Host station completion device**

This device is set with the execution of the END instruction within the program scan the write operation was completed in. The device is reset with the next END processing.

Status display of the operation completion

This device is set depending on the completion result of the instruction.

Remains reset for a normal (errorfree) transmission.

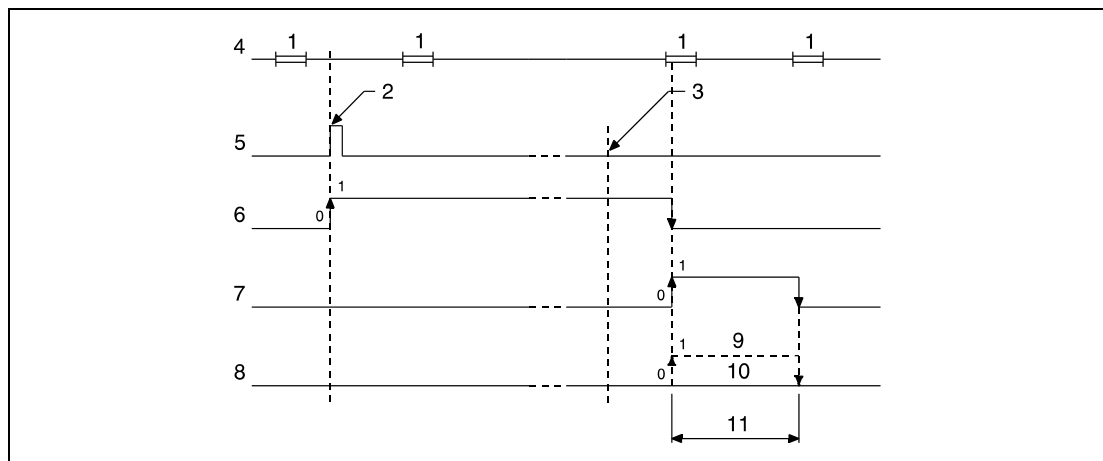
For the completion of a faulty transmission this device is set with the END instruction within the program scan the REQ instruction was completed in. The device is reset with the next END processing.

**NOTE**

●<sup>3</sup> The following table assigns the channel numbers to the according communications channel flags:

Channel number	1	2	3	4	5	6	7	8
Communications channel flag	SB30	SB32	SB34	SB36	SB38	SB3A	SB3C	SB3E

The following figure shows the operations of the host station during the execution of a REQ instruction:



- 1 END processing
- 2 Execution of the REQ instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 REQ instruction
- 6 Communications channel flag
- 7 Host station completion device set after completion of the operation (d2)
- 8 Status display of the operation completion ((d2)+1)
- 9 Completion of a faulty transmission
- 10 Completion of an errorfree transmission
- 11 One scan

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in Jn is not connected to the station (error code 4102).
- The module with the I/O address specified in Un is not a network module (error code 2111).

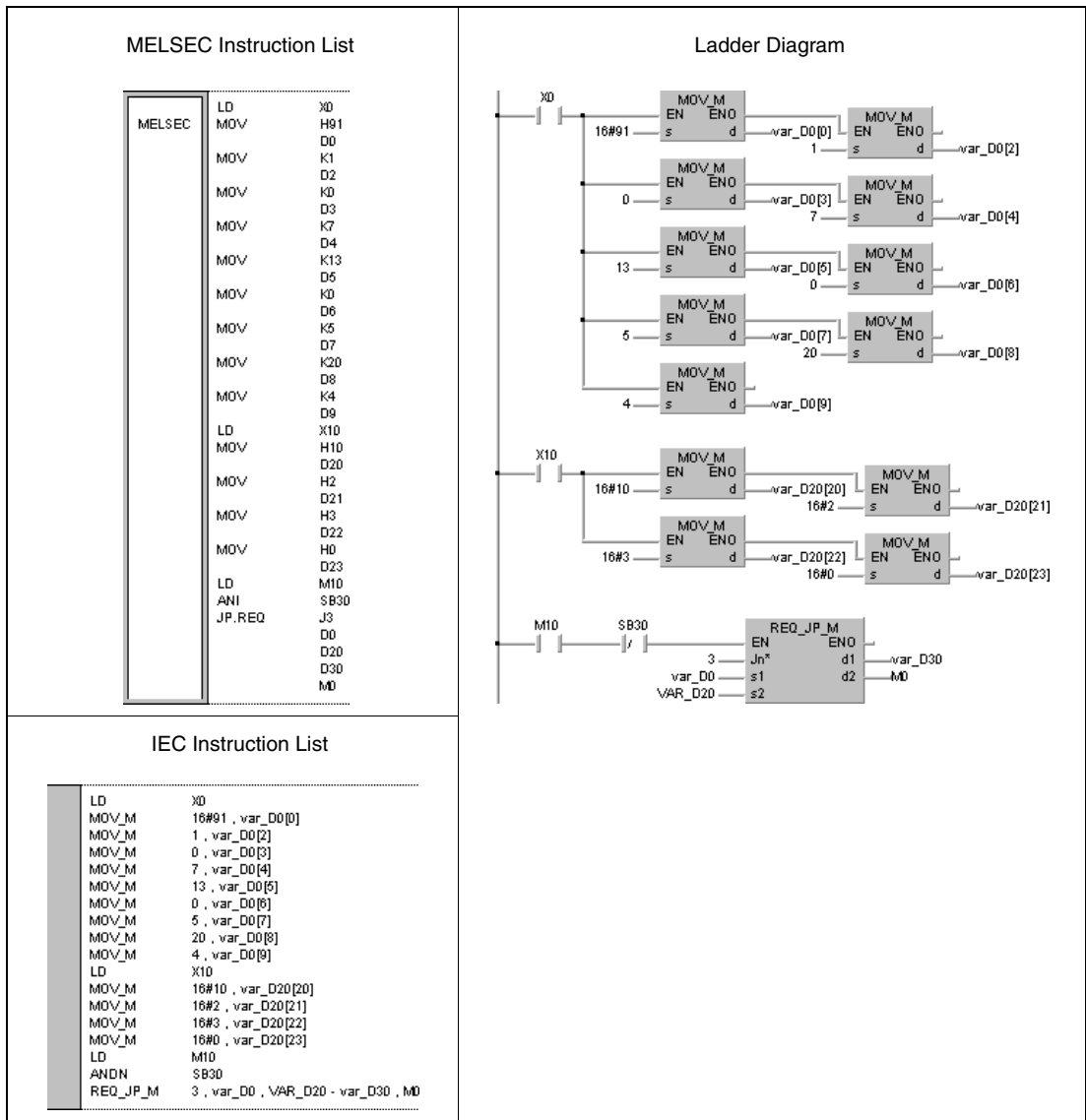
**Program Example**

JP.REQ

With leading edge from X10, the following program performs a STOP operation on an object station. The execution of the REQ instruction is interlocked via the normally open contact of the flag SB30. The following table contains further information on the host station, the sending station, and the applied MOV instructions.

Device/Instruction	Meaning/Function
Host station	–
Host network	–
Host channel	1
Communications channel flag	SB30
Object station	13
Object network	7
Object channel	–
1. MOV instruction	Sets the clock data
2. MOV instruction	Sets the channel for the host station
3. MOV instruction	–
4. MOV instruction	Sets the network number for the object station
5. MOV instruction	Sets the number for the object station
6. MOV instruction	–
7. MOV instruction	Sets the number of transmission retries
8. MOV instruction	Sets the WDT time setting (20 s)
9. MOV instruction	Sets the number of blocks to be sent (4)
10. MOV instruction	Sets the request type
11. MOV instruction	Sets the request type of subroutine
12. MOV instruction	Sets the mode
13. MOV instruction	Sets the clear mode





**NOTE** This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

8.6.8 ZNFR

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	9	
s2	—	● <sup>1</sup>	—	—	—	—	—	—			
d	●	—	—	—	—	—	—	—			

<sup>1</sup> Link registers only

GX IEC  
Developer

MELSEC Instruction List	Ladder Diagram	IEC Instruction List																				
<div style="border: 1px solid black; padding: 5px; width: fit-content;">MELSEC</div> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border-bottom: 1px dotted black;">JP.ZNFR</td> <td style="border-bottom: 1px dotted black;">Jn</td> </tr> <tr> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;">s1</td> </tr> <tr> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;">s2</td> </tr> <tr> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;">d</td> </tr> <tr> <td style="border-bottom: 1px dotted black;">GP.ZNFR</td> <td style="border-bottom: 1px dotted black;">Un</td> </tr> <tr> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;">s1</td> </tr> <tr> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;">s2</td> </tr> <tr> <td style="border-bottom: 1px dotted black;"></td> <td style="border-bottom: 1px dotted black;">d</td> </tr> </table>	JP.ZNFR	Jn		s1		s2		d	GP.ZNFR	Un		s1		s2		d		<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border-bottom: 1px dotted black;">ZNFR_JP_M</td> <td style="border-bottom: 1px dotted black;">Jn , s1 , s2 , d</td> </tr> <tr> <td style="border-bottom: 1px dotted black;">ZNFR_UP_M</td> <td style="border-bottom: 1px dotted black;">Un , s1 , s2 , d</td> </tr> </table>	ZNFR_JP_M	Jn , s1 , s2 , d	ZNFR_UP_M	Un , s1 , s2 , d
JP.ZNFR	Jn																					
	s1																					
	s2																					
	d																					
GP.ZNFR	Un																					
	s1																					
	s2																					
	d																					
ZNFR_JP_M	Jn , s1 , s2 , d																					
ZNFR_UP_M	Un , s1 , s2 , d																					

Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First number of device storing control data.	Device number	Array [1..15] of ANY16
s2	First number of link register (W) in the host station storing the data being read.		ANY16
d	Device set ON for 1 scan after completion of instruction.	Bit	BOOL

NOTE

- <sup>1</sup> The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.
- <sup>2</sup> The head I/O number of the network unit for the host station must range within 0 and FE<sub>H</sub>. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.

## Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	Confirmation of transmission completion is set (Bit 0 (b0) = 1, fixed)	0001H 0081H	User
	Error completion mode	Stores clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+11 (Array_s1[12] onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Dummy	Not used	—	—
(s1)+3 Array_s1[4]	Buffer memory address	Sets the first number in buffer memory.	● <sup>4</sup>	User
(s1)+4 Array_s1[5]	Dummy	Not used	—	—
(s1)+5 Array_s1[6]	Number for object station.	Sets number for remote I/O station reading the data.	1 to 64	User
(s1)+6 Array_s1[7]	Position of special function module	Sets the position of the special function module within the series of special function modules installed at the object station.	—	User
(s1)+7 Array_s1[8]	Dummy	Not used	—	—
(s1)+8 Array_s1[9]	Dummy	Not used	—	—
(s1)+9 Array_s1[10]	Length of data	Sets the number of data to be read.	1 to 256	User
(s1)+10 Array_s1[11]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System
(s1)+11 Array_s1[12]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+12 Array_s1[13]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+13 Array_s1[14]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+14 Array_s1[15]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		

●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

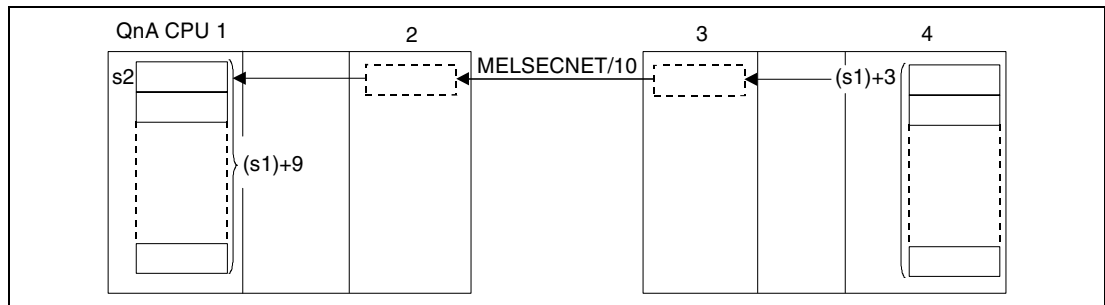
●<sup>4</sup> Refer to the manual of the according special function module reading data for further details.

**Functions Reading data from special function modules in remote I/O stations**

**ZNFR Read instruction**

The ZNFR instruction reads data stored in the buffer memory of a special function module in a remote I/O station connected to the MELSECNET/10. The remote I/O station is specified in the control data. The data read from the module is stored from s2 onwards in the host station.

After the completion of the operation the device specified in d is set.



- 1 Host station/ master station
- 2 Network module (host station/ master station)
- 3 Remote I/O station (object station)
- 4 Special function module (object station/ remote I/O station)

The read operation of a remote I/O station can only be executed via a network module connected to the same network as the remote I/O station connected to the MELSECNET/10.

The ZNFR instruction cannot be executed from more than one location simultaneously by one special function module. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further ZNFR instructions.

The interlock signal sent during the execution of the ZNFR instruction contains

- Read/ write request signals
- Read/ write completion signals
- Host station completion device (d)
- Status display of the operation completion (completion of an errorfree or faulty transmission) (d+1).

The signals and devices are described below:

**Read/ write request signals**

This signal is set with the execution of the dedicated data link instructions of the QnA series. The signal is reset with the next END processing within the scan the read/ write operations are completed in.

**Read/ write completion signals**

This signal is set with the execution of the dedicated data link instructions of the QnA series. The signal is reset with the next END processing within the scan the read/ write operations are completed in.

**Host station completion device**

This device is set with the processing of the END instruction within the scan the ZNFR instruction is completed in. The device is reset with the next END processing.

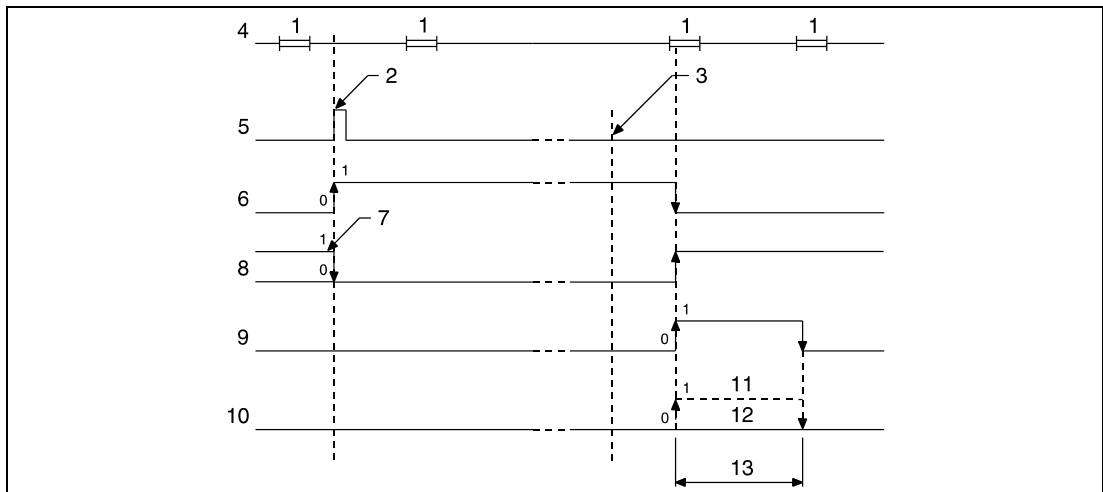
Status display of the operation completion

This device is set depending on the completion result of the instruction.

Remains reset for a normal (errorfree) transmission.

For the completion of a faulty transmission this device is set with the END instruction within the program scan the ZNFR instruction was completed in. The device is reset with the next END processing.

The following figure shows the operations of the host station during the execution of a ZNFR instruction:



- 1 END processing
- 2 Execution of the ZNFR instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 ZNFR instruction
- 6 Read/ write request signal
- 7 After execution of the dedicated data link instruction of the Q series
- 8 Completion of read/ write operation
- 9 Host station completion device set after completion of the operation (d)
- 10 Status display of the operation completion (d+1)
- 11 Completion of a faulty transmission
- 12 Completion of an errorfree transmission
- 13 One scan

The link registers in s2 are set via network parameters "M ← R (to master station from remote I/O station)" and are allocated within the range specified via the link refresh parameters.

The execution of the ZNFR instruction requires link relays and link registers to be used by the operating system. The number of link relays and link registers used by the operating system for the according special function module is as follows:

For  $M \rightarrow R$  (from master station to remote I/O station):

Link relays = 4, link registers = 4

For  $M \leftarrow R$  (to master station from remote I/O station):

Link relays = 4, link registers = 4

**Operation  
Errors**

In the following cases an operation error occurs and the error flag is set:

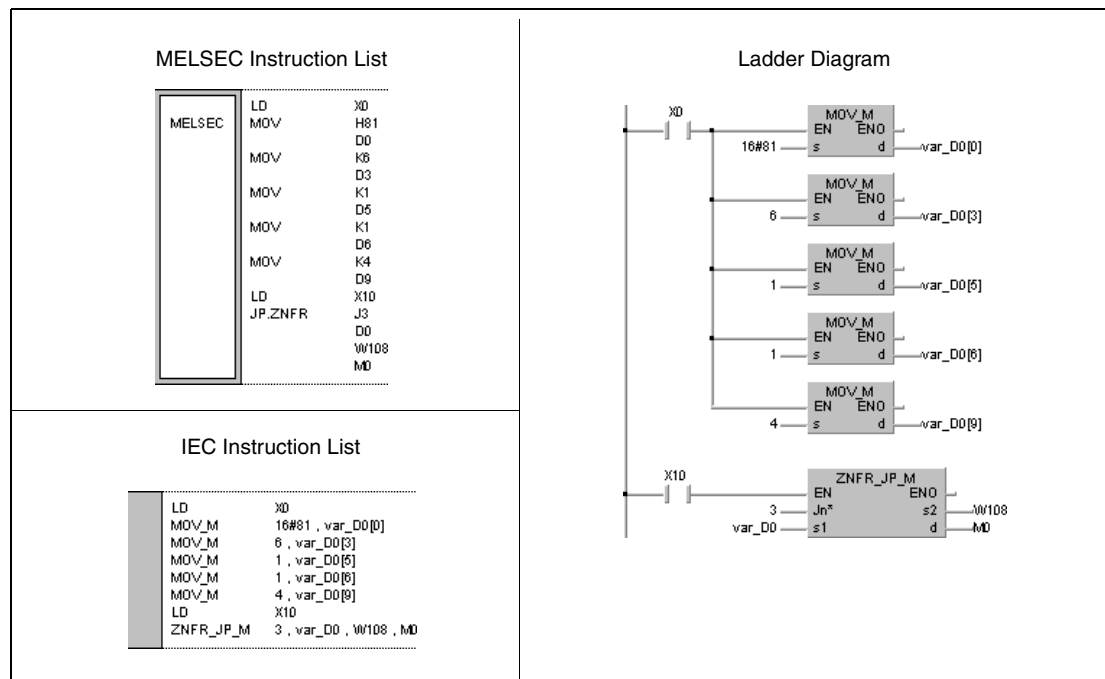
- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in  $J_n$  is not connected to the station (error code 4102).
- The module with the I/O address specified in  $U_n$  is not a network module (error code 2111).

**Program Example**

JP.ZNFR

With leading edge from X10, the following program reads the addresses 6 through 9 of the buffer memory in a special function module of an I/O station. The read data is stored in the link registers W108 through W10B. Further details on the I/O station and applied MOV instructions are given in the table below:

Device/Instruction	Meaning/Function
I/O station	1R1
Network of I/O station	3
Special function module	1
1. MOV instruction	Sets the clock data
2. MOV instruction	Sets the first address in the buffer memory (6)
3. MOV instruction	Sets the number of I/O station
4. MOV instruction	Sets the position of the special function module in sequence
5. MOV instruction	Sets the length of data to be read



**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

8.6.9 ZNTO

CPU

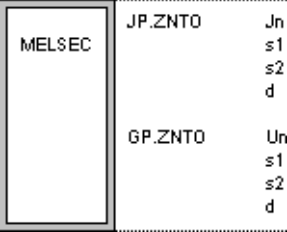
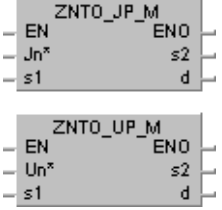
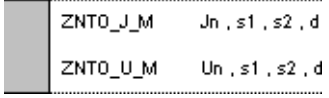
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	9	
s2	—	● <sup>1</sup>	—	—	—	—	—	—			
d	●	—	—	—	—	—	—	—			

<sup>1</sup> Link registers only

GX IEC Developer

MELSEC Instruction List	Ladder Diagram	IEC Instruction List
		

Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit	ANY16
Un	Head I/O number for network unit of host station. ● <sup>2</sup>		
s1	First number of device storing control data.	Device number	Array [1..16] of ANY16
s2	First number of link register (W) in the host station storing the data to be written.		ANY16
d	Device set ON for 1 scan after completion of instruction.	Bit	BOOL

NOTE

- <sup>1</sup> The network number for the host station must range within 1 and 239. The network with the number 254 is configured via settings for access of other stations to the active station.
- <sup>2</sup> The head I/O number of the network unit for the host station must range within 0 and FE<sub>H</sub>. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.



## Overview of devices for control data

Device	Meaning	Function	Value Range	Set by
(s1)+0 Array_s1[1]	Execution mode	Confirmation of transmission completion is set (Bit 0 (b0) = 1, fixed)	0001H 0081H	User
	Error completion mode	Stores clock data setting when error processing is completed: - No storage of clock data, Bit 7 (b7) = 0 - Storage of clock data, Bit 7 (b7) = 1 (clock data from (s1)+11 (Array_s1[12] onwards)		
(s1)+1 Array_s1[2]	Completion status of instruction execution	Status at completion of instruction is stored: 0 = no errors (normal completion) < > 0 = error code ● <sup>3</sup>	—	System
(s1)+2 Array_s1[3]	Dummy	Not used	—	—
(s1)+3 Array_s1[4]	Buffer memory address	Sets the first number in buffer memory.	● <sup>4</sup>	User
(s1)+4 Array_s1[5]	Dummy	Not used	—	—
(s1)+5 Array_s1[6]	Number for object station.	Sets number for object station.	1 to 64	User
(s1)+6 Array_s1[7]	Position of special function module	Sets the position of the special function module within the series of special function modules installed at the object station.	—	User
(s1)+7 Array_s1[8]	Dummy	Not used	—	—
(s1)+8 Array_s1[9]	Dummy	Not used	—	—
(s1)+9 Array_s1[10]	Length of data	Sets the number of data to be written.	1 to 256	User
(s1)+10 Array_s1[11]	Dummy	Not used	—	—
(s1)+11 Array_s1[12]	Clock set flag (set on error only)	Stores clock data enable/disable status set in (s1)+0 (Array_s1[1]): - Clock data storage disabled = 0 - Clock data storage enabled = 1	—	System
(s1)+12 Array_s1[13]	Clock data (set on error only)	Upper byte = Year (0 to 99) Lower byte = Month (1 to 12)	—	System
(s1)+13 Array_s1[14]		Upper byte = Day (1 to 31) Lower byte = Hour (0 to 23)		
(s1)+14 Array_s1[15]		Upper byte = Minute (0 to 59) Lower byte = Second (0 to 59)		
(s1)+15 Array_s1[16]		Upper byte = 00H Lower byte = Day of week (0 to 6) (Sunday = 0, Saturday = 6)		

●<sup>3</sup> Refer to the MELSECNET/10 manual for QnA network systems for further details.

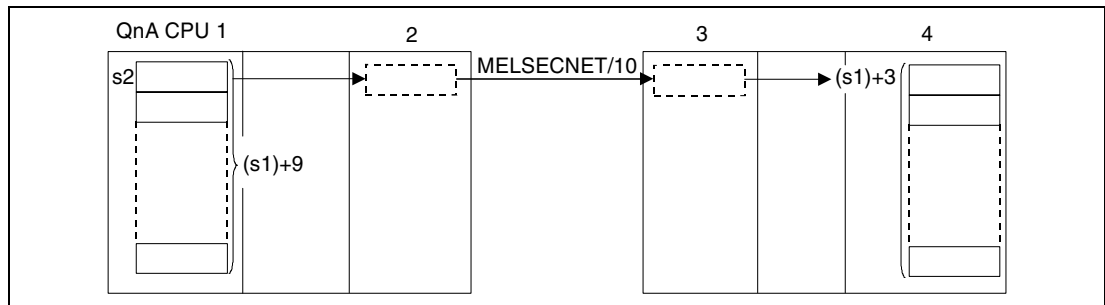
●<sup>4</sup> Refer to the manual of the according special function module reading data for further details.

## Functions Writing data to special function modules in remote I/O stations

### ZNTO Write instruction

The ZNTO instruction writes data stored in the host station from s2 onwards to the buffer memory of a special function module in a remote I/O station connected to the MELSECNET/10. The remote I/O station is specified in the control data.

After the completion of the operation the device specified in d is set.



- 1 Host station/ master station
- 2 Network module (host station/ master station)
- 3 Remote I/O station (object station)
- 4 Special function module (object station/ remote I/O station)

The write operation can only be executed by a master station connected to the MELSECNET/10 to a remote I/O station connected to the same network.

The ZNTO instruction cannot be executed from more than one location simultaneously by one special function module. At simultaneous execution of the instruction from two or more locations a handshake between the two active stations prevents from execution of further ZNTO instructions.

The interlock signal sent during the execution of the ZNTO instruction contains

- Read/ write request signals
- Read/ write completion signals
- Host station completion device (d)
- Status display of the operation completion (completion of an errorfree or faulty transmission) (d+1).

The signals and devices are described below:

#### Read/ write request signals

This signal is set with the execution of the dedicated data link instructions of the QnA series. The signal is reset with the next END processing within the scan the read/ write operations are completed in.

#### Read/ write completion signals

This signal is set with the execution of the dedicated data link instructions of the QnA series. The signal is reset with the next END processing within the scan the read/ write operations are completed in.

#### Host station completion device

This device is set with the processing of the END instruction within the scan the ZNTO instruction is completed in. The device is reset with the next END processing.

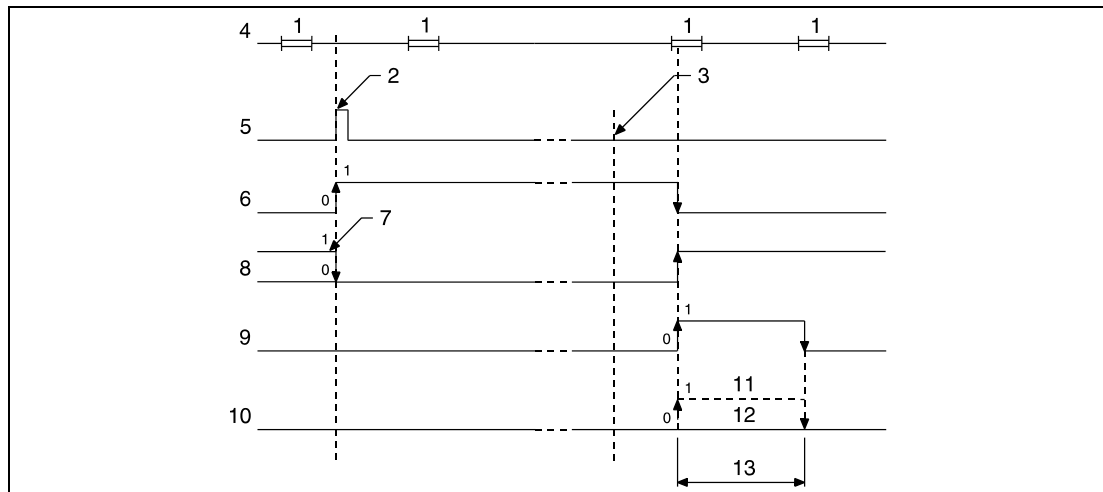
Status display of the operation completion

This device is set depending on the completion result of the instruction.

Remains reset for a normal (errorfree) transmission.

For the completion of a faulty transmission this device is set with the END instruction within the program scan the ZNTO instruction was completed in. The device is reset with the next END processing.

The following figure shows the operations of the host station during the execution of a ZNTO instruction:



- 1 END processing
- 2 Execution of the ZNTO instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 ZNTO instruction
- 6 Read/ write request signal
- 7 After execution of the dedicated data link instruction of the QnA series
- 8 Completion of read/ write operation
- 9 Host station completion device set after completion of the operation (d)
- 10 Status display of the operation completion (d+1)
- 11 Completion of a faulty transmission
- 12 Completion of an errorfree transmission
- 13 One scan

The link registers in s2 are set via network parameters "M ← R (to master station from remote I/O station)" and are allocated within the range specified via the link refresh parameters.

The execution of the ZNTO instruction requires link relays and link registers to be used by the operating system. The number of link relays and link registers used by the operating system for the according special function module is as follows:

For  $M \rightarrow R$  (from master station to remote I/O station):

Link relays = 4, link registers = 4

For  $M \leftarrow R$  (to master station from remote I/O station):

Link relays = 4, link registers = 4

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The control data contents exceed the setting range (error code 4100).
- The network with the number specified in  $J_n$  is not connected to the station (error code 4102).
- The module with the I/O address specified in  $U_n$  is not a network module (error code 2111).

**Program Example**

**JP.ZNTO**

With leading edge from X10, the following program writes data to the addresses 10 through 12 of the buffer memory in a special function module of an I/O station. The data to be written is stored in the host station in the link registers W18 through W1A. Further details on the I/O station and applied MOV instructions are given in the table below:

Device/Instruction	Meaning/Function
I/O station	1R2
Network of I/O station	3
Special function module	2
1. MOV instruction	Sets the clock data
2. MOV instruction	Sets the first address in the buffer memory (10)
3. MOV instruction	Sets the number of I/O station
4. MOV instruction	Sets the position of the special function module in sequence
5. MOV instruction	Sets the length of data to be read
6. MOV instruction	Writes the data to the link registers W18 through W1A
7. MOV instruction	
8. MOV instruction	

MELSEC Instruction List	Ladder Diagram
<pre> MELSEC LD      X0 MOV     H81, D0 MOV     D0, K10 MOV     K2, D6 MOV     K2, D6 MOV     K3, D9 LD      X1 MOV     K100, W18 MOV     K150, W19 MOV     K200, W1A LD      X10 JP.ZNTO J3 D0 W18 M0                     </pre>	
<p style="text-align: center;"><b>IEC Instruction List</b></p> <pre> LD      X0 MOV_M   16#81, var_D0[0] MOV_M   10, var_D0[3] MOV_M   2, var_D0[6] MOV_M   2, var_D0[9] MOV_M   3, var_D0[9] LD      X1 MOV_M   100, W18 MOV_M   150, W19 MOV_M   200, W1A LD      X10 ZNTO_JP_M 3, var_D0, W18, M0                     </pre>	

**NOTE**

*This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 8.7 A series compatible data link instructions

These instructions support the data communication among stations with QnA CPUs, among stations with QnA CPUs and ACPUs, as well as among QnA CPUs or ACPUs and remote I/O stations within MELSECNET and MELSECNET/10. The following table gives an overview of these instructions:

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor	Designated Station in MELSECNET/10			MELSECNET
			QnA CPU	ACPU	Remote I/O Station	
Read QnA data from object stations in object networks	J.ZNRD	ZNRD_J_M	●	●	—	—
	JP.ZNRD	ZNRD_JP_M				
Read data from local stations (master stations only)	J.ZNRD	ZNRD_J_M	—	—	—	●
	JP.ZNRD	ZNRD_JP_M				
Write QnA data to object stations in object networks	J.ZNWR	ZNWR_J_M	●	●	—	—
	JP.ZNWR	ZNWR_JP_M				
Write data to local stations (master stations only)	J.ZNWR	ZNWR_J_M	—	—	—	●
	JP.ZNWR	ZNWR_JP_M				
A series only: Read data from local stations (master stations only)	LRDP	LRDP_M	—	—	—	●
		LRDP_MD				
		LRDP_P_MD				
A series only: Write data to local stations (master stations only)	LWTP	LWTP_M	—	—	—	●
		LWTP_MD				
		LWTP_M_MD				
Read data from special function modules in remote I/O stations	RFRP/ G.RFRP	RFRP_U_M	—	—	—	●
		RFRP_UP_M				
Write data to special function modules in remote I/O stations.	RTOP G.RTOP	RTOP_U_M	—	—	—	●
		RTOP_UP_M				

8.7.1 ZNRD

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
n1	●	●	●	—	—	—	—	●	—		
s	—	● <sup>1</sup>	—	—	—	—	—	—	—		
d1	—	●	●	—	—	—	—	—	—		
n2	●	●	●	—	—	—	—	●	—		
d2	●	●	●	—	—	—	—	—	—		

<sup>1</sup> T, C, D, and W only

GX IEC Developer

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>MELSEC</p> </div> <p>J.ZNRD      Jn                           n1                           s                           d1                           n2                           d2</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>ZNRD_J_M      Jn , n1 , s , n2 , d1 , d2</p> </div>
---	-----------------------	--

Variables

Set Data	Meaning	Data Type
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit
n1	Number of object station.	
s	First device of station storing data to be read.	
d1	First device of host station storing read data.	
n2	Receive data length.	
d2	Device set ON for 1 scan after completion of instruction.	Bit

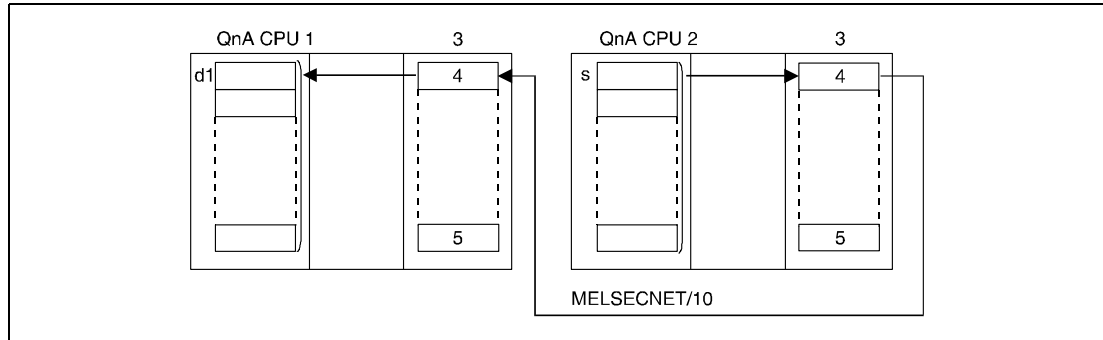
NOTE

●<sup>1</sup> Specify J0 for the instruction applied in MELSECNET(I/II/B).

**Functions    Reading data from other stations****ZNRD    Read instruction**

The ZNRD instruction reads the number of data words specified by n2 and stored in the object station in the MELSECNET/10. The station number is specified in n1. The network number is stored in Jn. The read data is stored from d1 onwards in the host station.

After the completion of the operation the device specified in d2 is set.



- 1 Host station
- 2 Object station
- 3 Network module
- 4 Channel 1
- 5 Channel 8

The read operation can only be executed with an object station connected to the same MELSECNET/10 network as the host station.

The read operation from a local station can only be executed with a master station connected to the MELSECNET network.

The network number Jn can be designated between 1 and 239. The designation of network number 0 (J0) is similar to the designation in the MELSECNET system.

In the MELSECNET system the number of the object network (Jn) is fixed to 0 (J0). The object network numbers (Jn) 1 to 239 are used in the MELSECNET/10.

The station number n1 may range from 1 to 64.

In the MELSECNET/B system the station number may range from 1 to 31.

The receive data length n2 (number of data words) may range from 1 to 230.

Read operations from other stations via ZNRD instruction can be performed by stations with AnU CPUs and QnA CPUs equally.

The data link instructions cannot be executed from several locations simultaneously with common access to the same channel. A simultaneous execution from two or more locations is prevented through a handshake of the two active stations.

Both, host and object station use channel 1 of the network module for the execution of the ZNRD instruction. For the execution of multiple ZNRD instructions channel 1 is accessed several times whereas channel 1 of the network module can only be used once for one instruction. In order to prevent the execution of several simultaneous instructions an interlock should be established through the read/write request signal and the operation completion device.



The execution and the completion of the ZNRD instruction is indicated via the communications directive flag (SB30) and the host station completion device (d2) as follows:

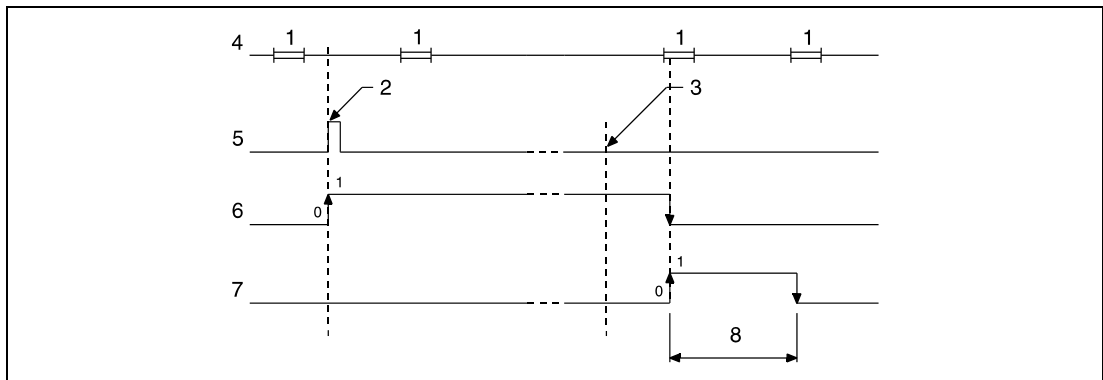
**Communications directive flag**

This flag is set during the execution of the ZNRD instruction. The flag is reset with the execution of the END instruction during the program scan the read operation was completed in.

**Host station completion device**

This device is set with the execution of the END instruction within the program scan the operation was completed in. The device is reset with the next END processing.

The following figure shows the operations of the host station during the execution of a ZNRD instruction:



- 1 END processing
- 2 Execution of the ZNRD instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 ZNRD instruction
- 6 Communications directive flag (SB30)
- 7 Host station completion device set after completion of the operation (d2)
- 8 One scan

The execution status and the completion status (normal, not normal) of the ZNRD instruction is indicated by the operation completion register of the ZNRD instruction (SW31) as follows:

For an errorfree (normal) completion of the operation the contents of register SW31 are 0.

For a faulty (not normal) completion of the operation the corresponding error code is stored in register SW31.

**NOTE**

*Refer to the MELSECNET/10 manual for QnA network systems for further details.*

**Operation Errors**

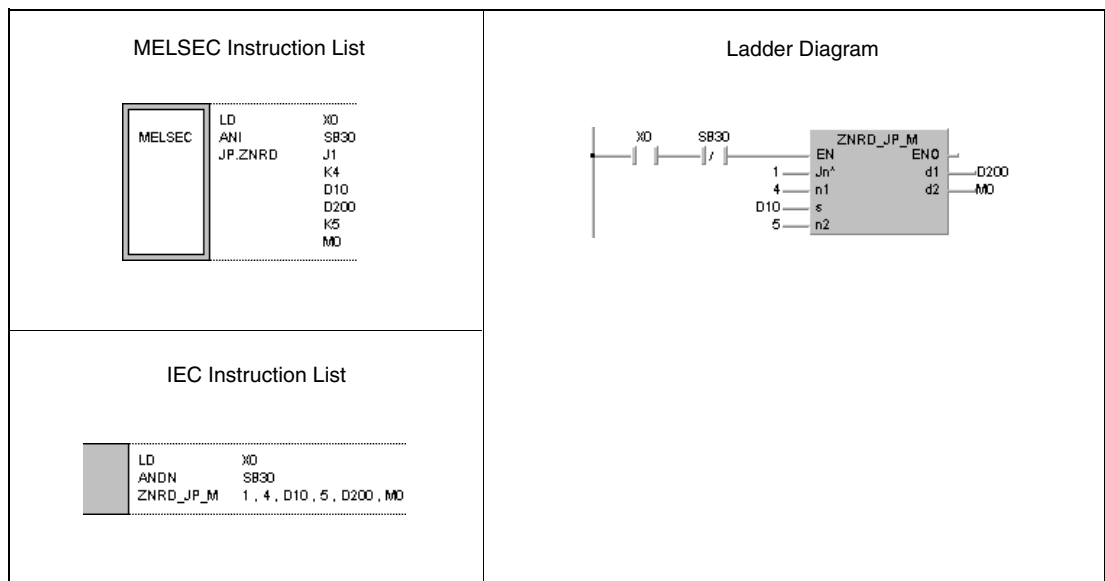
In the following cases an operation error occurs and the error flag is set:

- The receive data length specified by n2 exceeds the relevant storage device range of s1 (error code 4101).
- The network with the number specified by Jn does not exist (error code 4102).
- The station with the number specified by n1 does not exist (error code 4102).
- The receive data length specified by n2 does not range within 1 and 230 (error code 4100).

**Program Example**

JP.ZNRD

With leading edge from X0 the following program reads data from the registers D10 through D14 in station number 4. The read data is stored in the registers D200 through D204 in the host station. The host station and the object station are connected to network number 1.



### 8.7.2 ZNWR

**CPU**

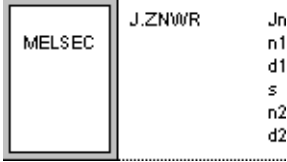
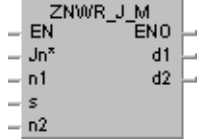

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
n1	●	●	●	—	—	—	—	●	—	SM0	12
d1	—	● <sup>1</sup>	—	—	—	—	—	—	—		
s	—	●	●	—	—	—	—	—	—		
n2	—	●	●	—	—	—	—	●	—		
d2	●	●	●	—	—	—	—	—	—		

<sup>1</sup> T, C, D, and W only

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	--	--

**Variables**

Set Data	Meaning	Data Type
Jn	Network number for host station. ● <sup>1</sup>	BIN 16-bit
n1	Number of object station.	
d1	First device of object station data is written to.	
s	First device of host station storing data to be written.	
n2	Send data length.	
d2	Device set ON for 1 scan after completion of instruction.	Bit

**NOTE**

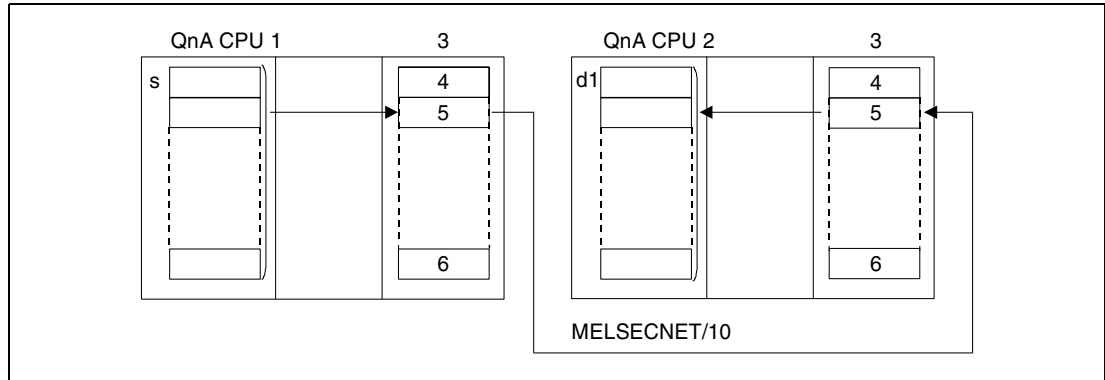
●<sup>1</sup> Specify J0 for the instruction applied in MELSECNET(I/II/B).

**Functions Writing data to other stations**

**ZNWR Write instruction**

The ZNWR instruction writes the number of data words specified by n2 and stored in s in the host station to an object station in the MELSECNET/10. The object station number is specified in n1. The network number is specified in Jn.

After the completion of the operation the device specified in d2 is set.



- <sup>1</sup> Host station
- <sup>2</sup> Object station
- <sup>3</sup> Network module
- <sup>4</sup> Channel 1
- <sup>5</sup> Channel 2
- <sup>6</sup> Channel 8

The write operation can only be executed with an object station connected to the same MELSECNET/10 network as the host station.

The write operation from a local station can only be executed with a master station connected to the MELSECNET network.

The network number Jn can be designated between 1 and 239. The designation of network number 0 (J0) is similar to the designation in the MELSECNET system.

In the MELSECNET system the number of the object network (Jn) is fixed to 0 (J0). The object network numbers (Jn) 1 to 239 are used in the MELSECNET/10.

The station number n1 may range from 1 to 64.

In the MELSECNET/B system the station number may range from 1 to 31.

The send data length n2 (number of data words) may range from 1 to 230.

Write operations from other stations via ZNWR instruction can be performed by stations with AnU CPUs and QnA CPUs equally.

Both, host and object station use channel 2 of the network module for the execution of the ZNWR instruction. For the execution of multiple ZNWR instructions channel 2 is accessed several times whereas channel 2 of the network module can only be used once for one instruction. In order to prevent the execution of several simultaneous instructions an interlock should be established through the read/write request signal and the operation completion device.

The execution and the completion of the ZNWR instruction is indicated via the communications directive flag (SB32) and the host station completion device (d2) as follows:

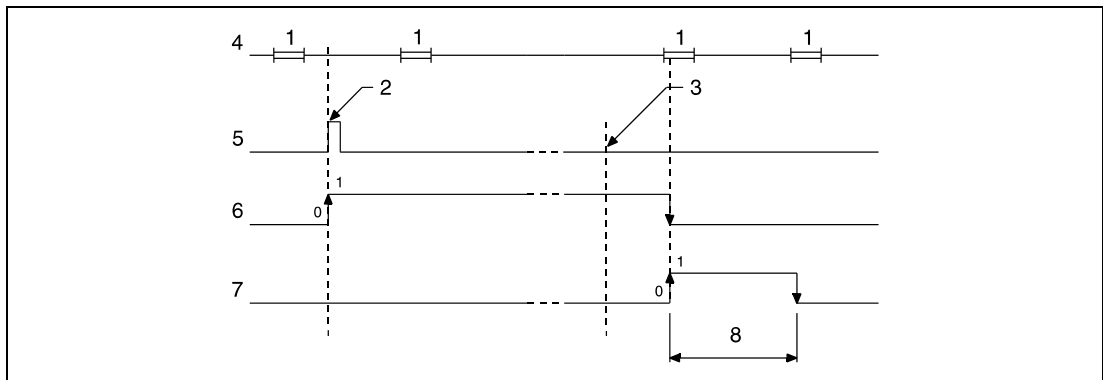
**Communications directive flag**

This flag is set during the execution of the ZNWR instruction. The flag is reset with the execution of the END instruction during the program scan the read operation was completed in.

**Host station completion device**

This device is set with the execution of the END instruction within the program scan the operation was completed in. The device is reset with the next END processing.

The following figure shows the operations of the host station during the execution of a ZNWR instruction:



- 1 END processing
- 2 Execution of the ZNWR instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 ZNWR instruction
- 6 Communications directive flag (SB32)
- 7 Host station completion device set after completion of the operation (d2)
- 8 One scan

The execution status and the completion status (normal, not normal) of the ZNWR instruction is indicated by the operation completion register of the ZNWR instruction (SW33) as follows:

For an errorfree (normal) completion of the operation the contents of register SW33 are 0.

For a faulty (not normal) completion of the operation the corresponding error code is stored in register SW33.

**NOTE**

*Refer to the MELSECNET/10 manual for QnA network systems for further details.*

**Operation Errors**

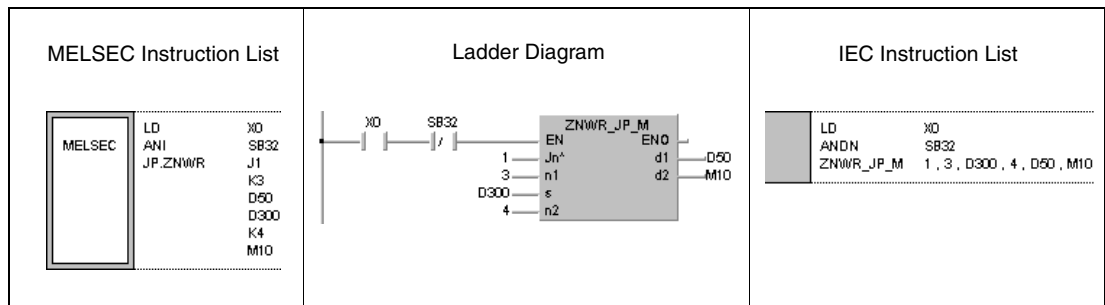
In the following cases an operation error occurs and the error flag is set:

- The send data length specified by n2 exceeds the relevant storage device range of s1 (error code 4101).
- The network with the number specified by Jn does not exist (error code 4102).
- The station with the number specified by n1 does not exist (error code 4102).
- The send data length specified by n2 does not range within 1 and 230 (error code 4100).

**Program Example**

JP.ZNWR

With leading edge from X0 the following program writes data from the registers D300 through D303 from the host station to the registers D50 through D53 in station number 3. The host station and the object station are connected to network number 1.



### 8.7.3 LRDP

**CPU**

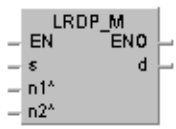
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

**Devices  
MELSEC A**

	Usable Devices																			Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices							Word Devices (16-bit)								Constant		Pointer						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P				I	N
n1																●	●							
s							●	●	●	●														
d							●	●	●	●												●		
n2																●	●							

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">LRDP</td> <td style="padding: 2px;">n1 s d n2</td> </tr> </table> </div>	MELSEC	LRDP	n1 s d n2	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="padding: 2px;">LRDP_M</td> <td style="padding: 2px;">s , n1 , n2 , d</td> </tr> </table> </div>		LRDP_M	s , n1 , n2 , d
MELSEC	LRDP	n1 s d n2						
	LRDP_M	s , n1 , n2 , d						

**Variables**

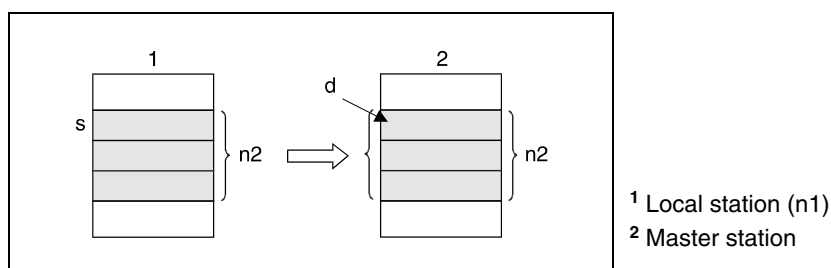
Set Data	Meaning	Data Type
n1	Number of local station.	BIN 16-bit
s	Initial address of the data area in the local station to be read.	
d	Address area of master station storing the read data.	
n2	Receive data length.	

**Functions**    **Reading data from a local station****LRDP    Read instruction**

The LRDP instruction reads data from a local station and stores it in a specified address area of the master station. The initial address of the data area to be read is specified by s. The number of data (1 to 32) is specified by n2. The number of the local station is specified in n1. The address area of the master station storing the read data is specified by d.

During the execution of the LRDP instruction the special relay M9200 in the master station is set. After completion of the instruction M9201 is set. Both special relays remain set after execution of the instruction. They must be reset through the sequence program.

Two or more LRDP instructions cannot be executed simultaneously. The LRDP instruction even cannot access a local station via an LWTP instruction at the same time.

**NOTE**

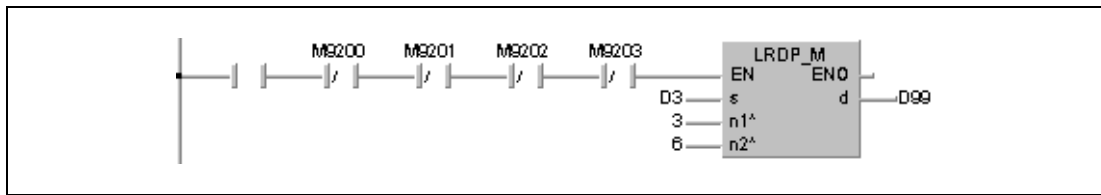
*The special relays M9200, M9201, M9202, and 9203 should be programmed as interlock and input condition for the LRDP or LWTP instruction to ensure that no other LRDP or LWTP instruction can be executed.*

The execution result of the LRDP instruction is returned through the data value in D9200 (see table below):

Data Value	Meaning
<b>LRDP D9200</b>	
0	Errorfree completion of instruction.
2	Operation error due to invalid addressing: The addresses in s and d exceed the relevant storage device range. The value in n1 exceeds the range of 1 to 64. The value in n2 exceeds the range of 1 to 32.
3	The addressed local station is offline and not accessible
4	There is no local station at the specified station number (error processing).



The following figure shows an interlock of the LRDP instruction:



### Operation Errors

In the following cases an operation error occurs and the error flag is set:

- There is no local station at the number specified by n1 or the specified number exceeds the range of 1 to 64.
- The addresses in s and d exceed the relevant storage device range.
- The number of data specified by n2 exceeds the range of 1 to 32.
- The LRDP instruction is executed in the program of a local station.

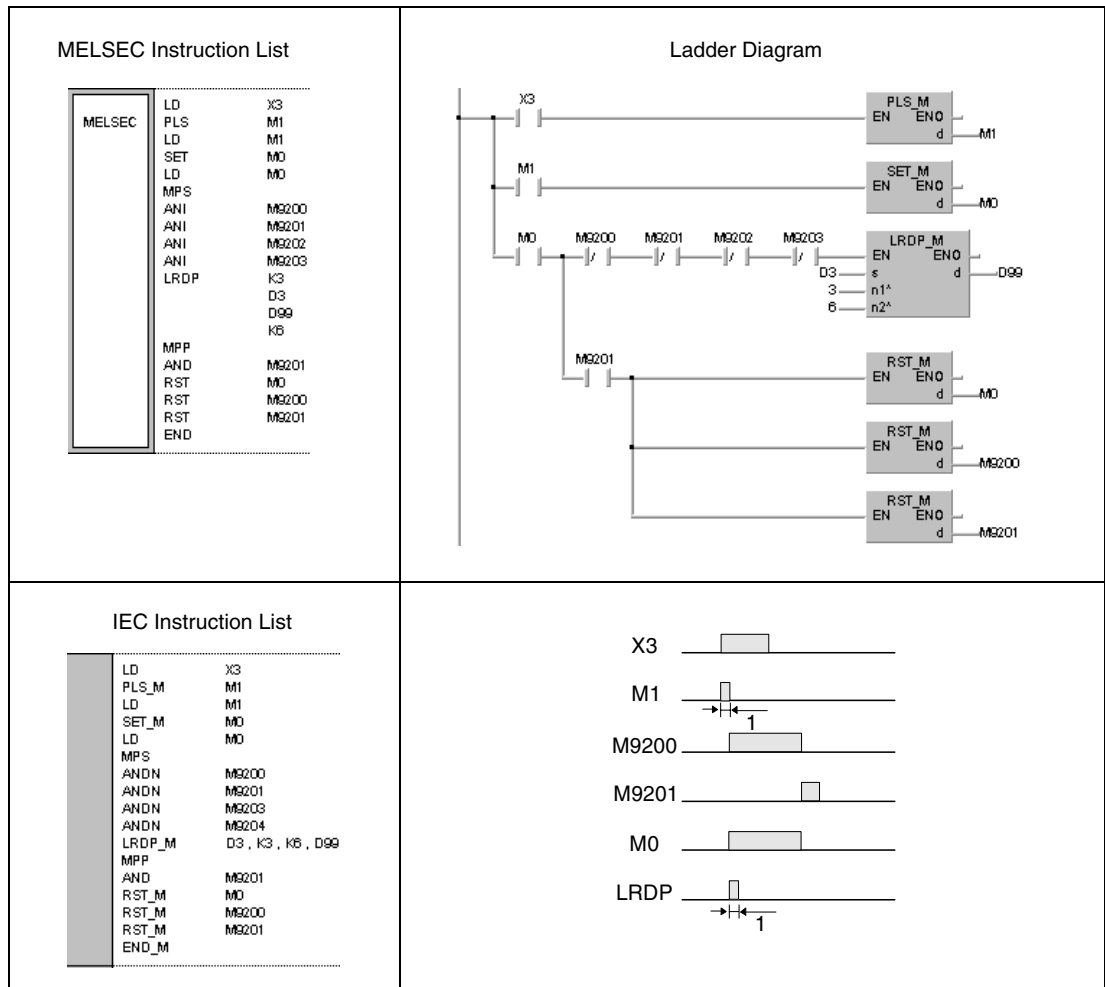
### NOTE

*If the CPU does not support data link operations or is set offline, the LRDP instruction will not be executed. No operation error occurs. However, M9200 is set.*

**Program Example**

**LRDP**

The following program reads data from D3 through D8 in the local station 3 and stores it in D99 through D104 in the master station. After switching X3 ON M0 is set and the LRDP instruction is executed. With the beginning data transfer M9200 is set. When the transfer is completed, M9201 is set. The LRDP instruction will not be executed, if another LRDP or LWTP instruction is already executed. After completion of the transfer (M9201 is set) in the further course of the program M0, M9200, and M9201 are reset.



<sup>1</sup> One single execution

*The contact corresponding to M1 should be converted into a pulse. Otherwise the LRDP instruction would not be executed completely.*

*The contact corresponding to M0 should be set via a SET instruction. If an OUT or PLS instruction is programmed instead of the SET instruction, errors might occur with the execution of the LRDP instruction.*

*In order to prevent the simultaneous execution of two LRDP instructions an interlock must be established through the special relays M9200 and M9201.*

*If within the same program a local station is accessed via an LWTP instruction the special relays M9202 and M9203 must be programmed as an interlock in addition.*

### 8.7.4 LWTP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

**Devices  
MELSEC A**

	Usable Devices																	Digit designation	Number of steps	Index	Carry Flag	Error Flag	
	Bit Devices							Word Devices (16-bit)							Constant		Pointer						Level
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K				H (16#)	P	I
n1																●	●						
d							●	●	●	●													
s							●	●	●	●												●	
n2																●	●						

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">LWTP</td> <td style="padding: 2px;">n1 d s n2</td> </tr> </table> </div>	MELSEC	LWTP	n1 d s n2	<p>Ladder Diagram</p> <div style="text-align: center; margin: 10px 0;"> </div>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="padding: 2px;">LWTP_M</td> <td style="padding: 2px;">s , n1 , n2 , d</td> </tr> </table> </div>		LWTP_M	s , n1 , n2 , d
MELSEC	LWTP	n1 d s n2						
	LWTP_M	s , n1 , n2 , d						

**Variables**

Set Data	Meaning	Data Type
n1	Number of local station.	BIN 16-bit
d	Address area of local station to be written to.	
s	Initial address of the data area in the master station to be written.	
n2	Send data length.	

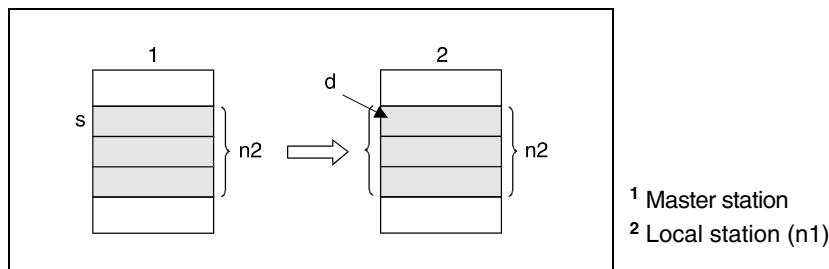
**Functions Writing data to a local station**

**LWTP Write instruction**

The LWTP instruction writes data from a master station to a specified address area in a local station. The initial address of the data area to be written is specified by s. The number of data (1 to 32) is specified by n2. The number of the local station is specified by n1. The address area of the local station to be written to is specified by d.

During the execution of the LWTP instruction the special relay M9202 in the master station is set. After completion of the instruction M9203 is set. Both special relays remain set after execution of the instruction. They must be reset through the sequence program.

Two or more LWTP instructions cannot be executed simultaneously. The LWTP instruction even cannot access a local station via an LRDP instruction at the same time.

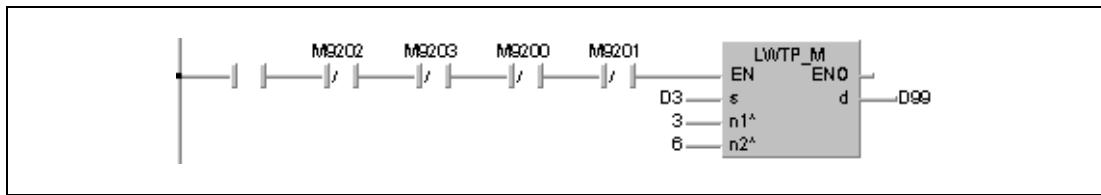


**NOTE** *The special relays M9200, M9201, M9202, and 9203 should be programmed as interlock and input condition for the LRDP or LWTP instruction to ensure that no other LRDP or LWTP instruction can be executed.*

The execution result of the LWTP instruction is returned through the data value in D9001 (see table below):

Data Value	Meaning
<b>LWTP D9201</b>	
0	Errorfree completion of instruction.
2	Operation error due to invalid addressing: The addresses in s and d exceed the relevant storage device range. The value in n1 exceeds the range of 1 to 64. The value in n2 exceeds the range of 1 to 32.
3	The addressed local station is offline and not accessible
4	There is no local station at the specified station number (error processing).

The following figure shows an interlock of the LWTP instruction:



### Operation Errors

In the following cases an operation error occurs and the error flag is set:

- There is no local station at the number specified by n1 or the specified number exceeds the range of 1 to 64.
- The addresses in s and d exceed the relevant storage device range.
- The number of data specified by n2 exceeds the range of 1 to 32.
- The LWTP instruction is executed in the program of a local station.

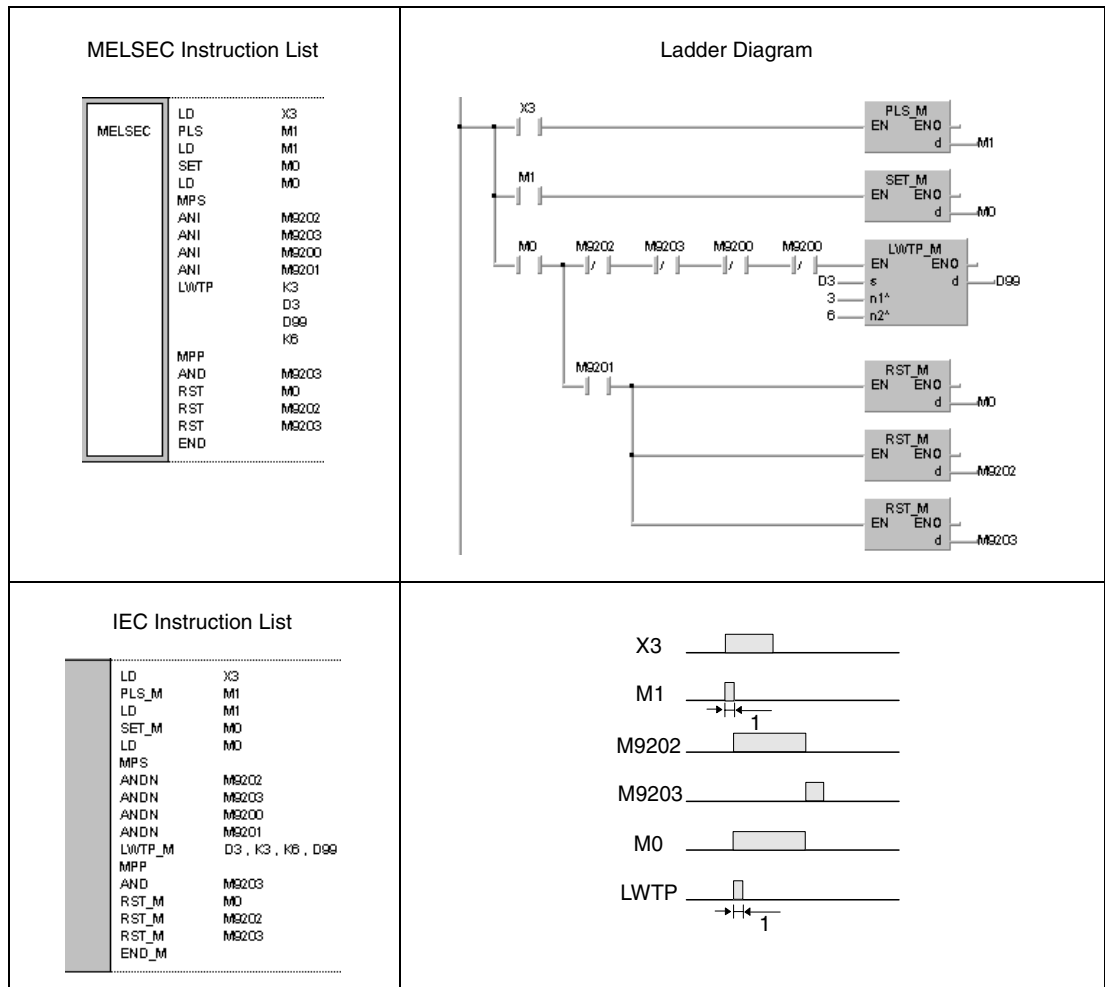
### NOTE

*If the CPU does not support data link operations or is set offline, the LWTP instruction will not be executed. No operation error occurs. However, M9202 is set.*

**Program Example**

**LWTP**

The following program writes data from D99 through D104 in the master station to D3 through D8 in the local station 3. After switching X3 ON M0 is set and the LWTP instruction is executed. With the beginning data transfer M9202 is set. When the transfer is completed, M9203 is set. The LWTP instruction will not be executed, if another LWTP or LRDP instruction is already executed. After completion of the transfer in the further course of the program M0, M9202, and M9203 are reset.



<sup>1</sup> One single execution

**NOTE**

*The contact corresponding to M1 should be converted into a pulse. Otherwise the LWTP instruction would not be executed completely.*

*The contact corresponding to M0 should be set via a SET instruction. If an OUT or PLS instruction is programmed instead of the SET instruction, errors might occur with the execution of the LWTP instruction.*

*In order to prevent the simultaneous execution of two LWTP instructions an interlock must be established through the special relays M9202 and M9203.*

*If within the same program a local station is accessed via an LRDP instruction the special relays M9200 and M9201 must be programmed as an interlock in addition.*

### 8.7.5 RFRP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	

**Devices  
MELSEC A**

	Usable Devices															Digit designation	Number of steps	Index	Carry Flag	Error Flag						
	Bit Devices					Word Devices (16-bit)					Constant		Pointer		Level											
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z				V	K	H (16#)	P	I	N	M9012	M9010 M9011
n1																●	●									
n2																●	●									
d										●														●	●	●
n3																●	●									

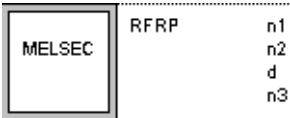
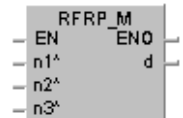
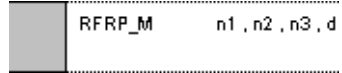
<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

**Devices  
MELSEC Q**

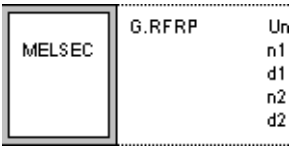
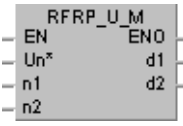
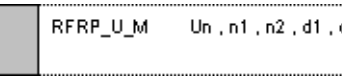
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
n1	●	●	●	—	—	—	—	●	—	SM0	9
d1	—	● <sup>1</sup>	—	—	—	—	—	—	—		
n2	●	●	●	—	—	—	—	●	—		
d2	●	●	●	—	—	—	—	—	—		

<sup>1</sup> Link registers only

**A series  
GX IEC  
Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

**QnA series  
GX IEC  
Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

## Variables

Set Data		Meaning	Data Type
A series	QnA series		
n1	Un	Head I/O number for special function module in the remote I/O station. ● <sup>1</sup>	BIN 16-bit
n2	n1	First number of buffer memory in special function module storing data to be read.	
d	d1	First number of link register in the host station storing read data.	Device number
n3	n2	Receive data length.	BIN 16-bit
	d2	Device set ON for 1 scan after completion of instruction.	Bit

## NOTE

- <sup>1</sup> The head I/O number of the network unit for the host station must range within 0 and FEH. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.



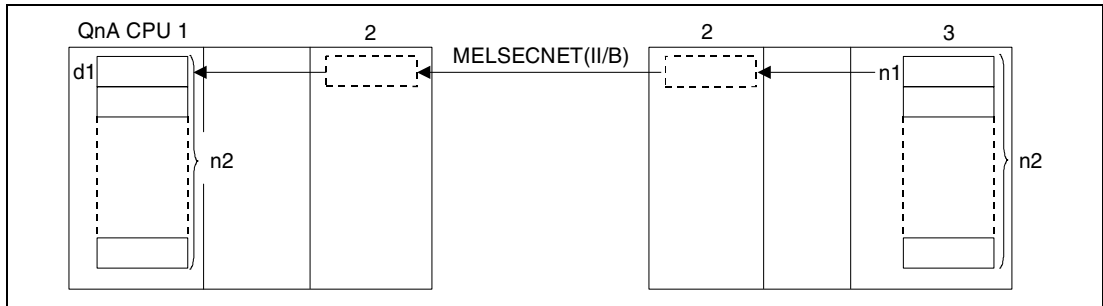
**Functions Reading data from a remote station**

**RFRP Read instruction**

The RFRP instruction reads data from the buffer memory in a special function module in a remote station connected to the MELSECNET.

The number of data words to be read is specified by n2 (A series = n3). The address area in the buffer memory is specified by n1 onwards (A series = n2). The I/O number of the connected special function module is specified by Un (A series = n1). The read data is stored in the link register specified by d1 onwards (A series = d) in the master station.

After the completion of the read operation in the remote I/O station the device specified in d2 is set (Q series only).



- 1 Host station (master station)
- 2 Data link module
- 3 Special function module (Object station/remote I/O station)

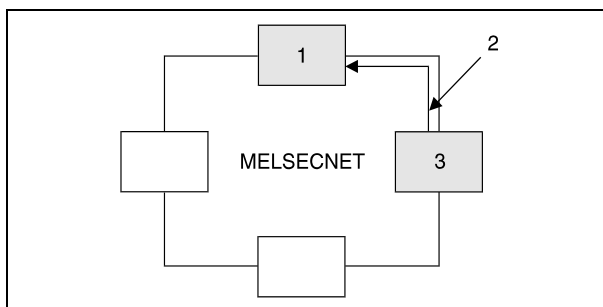
**NOTE**

*Even if only a read operation is executed the address area in the link registers in d1 (A series = d) must range within the MELSECNET parameterization of the remote and master station.*

In the following the I/O numbers of the special function modules for the QnA series are described. These specifications are valid for the A series as well except for the value n which has to be replaced by n1 (e.g. QnA series = Y(n+E) ⇒ A series = Y(n1+E)).

During the execution of the RFRP instruction the output Y(n+E) is set. X(n+1E) will be set as soon as the instruction is completed. Y(n+E) remains set after the execution completion and therefore has to be reset by the sequence program. The addressing applies automatically and must not be changed.

Read operations from remote I/O stations can be performed by a master station connected to the MELSECNET.



- 1 Station, executing the RFRP instruction (master station)
- 2 Read operation of the data from the special function module
- 3 Remote I/O station

If the RFRP instruction cannot be executed due to an error in the addressed special function module, X(n+1D) will be set. In this case the according module should be checked. X(n+1D) is reset once Y(n+D) is set.

The head I/O number of special function modules specified by Un in 4-digit format is stored in the upper 3 places. For example, the addresses X/Y0200 are specified 20 (QnA series only).

**NOTE**

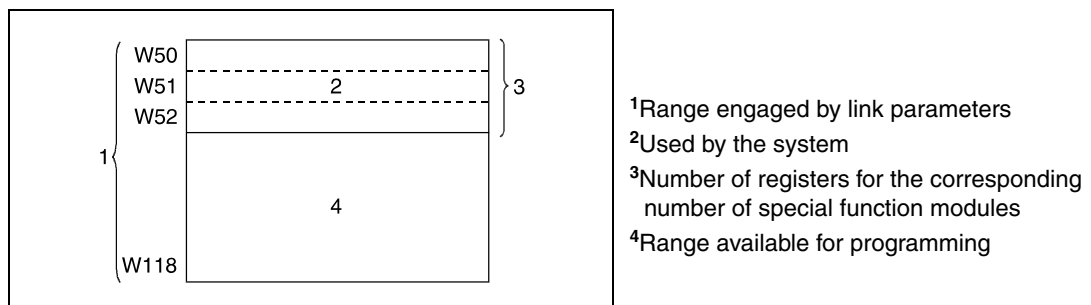
*Refer to the manual for the corresponding special function module for further information on the valid address range of the buffer memory in special function modules specified by n1 (A series = n2).*

The receive data length (number of data words) specified by n2 (A series = n3) may range from 1 to 16.

The address area of the link register in d1(A series = d) must range within the link parameter range of the remote and the master station.

The range of the link register Wxxx between master and remote station must be differentiated precisely. The number of link register addresses used by the operating system equals the number of special function modules contained in the remote stations of a network. The range available for data storage is the parameter range minus the link register addresses used by the operating system.

The example illustrated below shows the different areas of a link register. The area between master and remote station is specified W050 through W118 (A series = W09F) in the parameters. In this area 2 special function modules are allocated so the first two link registers W50 and W51 (2 addresses) are engaged by the operating system of the CPU. The available area for data storage therefore ranges from W52 to W118 (A series = W09F).



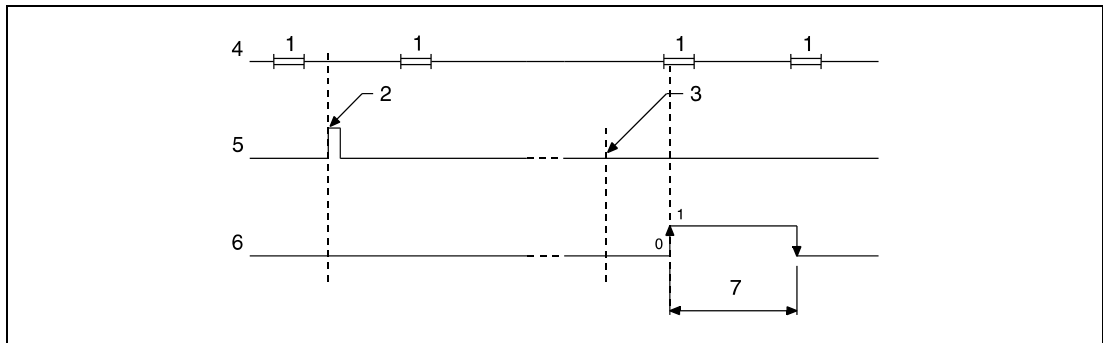
The RFRP and RTOP instructions cannot be executed from several locations simultaneously by the same special function module. A simultaneous execution from two or more locations is prevented through a handshake of the two active stations.

The inputs and outputs X(n+1E) and Y(n+E) should be programmed as interlock to ensure that no other RFRP or RTOP instruction can be executed.

The host station completion device (d2) is set with the execution of the END instruction within the program scan the read operation was completed in. The device is reset with the next END processing (QnA series only).

The MELSEC A series supplies numerous special registers for data transfer in the MELSEC-NET that register various communication states. For example, the status of the remote I/O stations is registered through the special registers D9228 through D9231. Parameter access is evaluated through the special registers M9224 through M9227 (A series only).

The following figure shows the operations of the host station during the execution of an RFRP instruction:



- 1 END processing
- 2 Execution of the RFRP instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 RFRP instruction
- 6 Host station completion device set after completion of the operation (d2) (QnA series only)
- 7 One scan

### Operation Errors

In the following cases an operation error occurs and the error flag is set:

- The I/O number specified by Un (A series = n1) is not that of a remote I/O station (QnA series = error code 4102).
- The I/O number specified by n1 (A series = n2) is not the head I/O number of a special function module (QnA series = error code 4102).
- The number of addresses specified by n2 (A series = n3) exceeds the address range specified from d1 onwards (A series = d, W0 through W3FF) (QnA series = error code 4101).
- The network specified by Un (A series = n1) does not exist (error code 2413).
- The value specified for n2 (A series = n3) exceeds the range of 1 to 16 (error code 4100).

**Program Example**

RFRP (A series)

The following program reads data from 10 successive addresses beginning at address 10 from a special function module (e.g. A68AD). The module is located at the second remote station. The addresses range from 140 through 15F. The read data is stored in the link registers W52 through W61 in the master station.

After switching X3 ON M0 is set and the RFRP instruction is executed. With the beginning data transfer Y(n+1+E) = Y14E is set. When the data transfer is completed X(n+1+E) = X15E is set. The RFRP instruction is not executed, if another RFRP or RTOP instruction is already executed.

<p style="text-align: center;">MELSEC Instruction List</p> <pre> MELSEC  LD      X003         PLS    M1         LD      M1         SET    M0         LD      M0         MPS         ANI    Y14E         ANI    X15E         ANI    Y14F         ANI    X15F         RFRP   H0140, K10, W052, K10         MPP         AND    X15E         RST    M0         RST    Y14E         END     </pre>	<p style="text-align: center;">Ladder Diagram</p>
<p style="text-align: center;">IEC Instruction List</p> <pre> LD      X3 PLS_M   M1 LD      M1 SET_M   M0 LD      M0 MPS ANDN    Y14E ANDN    X15E ANDN    Y14F ANDN    X15F RFRP_M  H0140, K10, W052, K10 MPP AND     X15E RST_M   M0 RST_M   Y14E END_M     </pre>	

<sup>1</sup> One single execution

**NOTE**

*The contact corresponding to M1 should be converted into a pulse. Otherwise the RFRP instruction would not be executed completely.*

*The contact corresponding to M0 should be set via a SET instruction. If an OUT or PLS instruction is programmed instead of the SET instruction, errors might occur with the execution of the RFRP instruction.*

*In order to prevent the simultaneous execution of two RFRP instructions an interlock must be established through the output Y14E and the input X15E.*

*If within the same program this station is accessed via an RTOP instruction the output Y14F and the input X15F must be programmed as an interlock in addition.*

### 8.7.6 RTOP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●	●	

**Devices  
MELSEC A**

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag		
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level								
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V				K	H (16#)	P	I
n1																●	●						
n2																●	●						
s									●												● <sup>1</sup>	●	●
n3																●	●						

<sup>1</sup> Refer to section "Programming an AnA, AnAS, and AnU CPU" in this manual for the according number of steps.

**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
n1	●	●	●	—	—	—	—	●	—	SM0	9
s	—	● <sup>1</sup>	—	—	—	—	—	—	—		
n2	●	●	●	—	—	—	—	—	—		
d	●	●	●	—	—	—	—	●	—		

<sup>1</sup> Link registers only

**A series  
GX IEC  
Developer**

<p>MELSEC Instruction List</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p>
--------------------------------	-----------------------	-----------------------------

**QnA series  
GX IEC  
Developer**

<p>MELSEC Instruction List</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p>
--------------------------------	-----------------------	-----------------------------

**Variables**

Set Data		Meaning	Data Type
A series	QnA series		
n1	Un	Head I/O number for special function module in the remote I/O station. ● <sup>1</sup>	BIN 16-bit
n2	n1	First number of buffer memory in special function module storing written data.	
s	s	First number of link register in the host station storing data to be written.	Device number
n3	n2	Send data length.	BIN 16-bit
	d	Device set ON for 1 scan after completion of instruction.	Bit

**NOTE**

- <sup>1</sup> The head I/O number of the network unit for the host station must range within 0 and FEH. Note, that the compiler expects a hexadecimal number for Un. A decimal number will be converted into a hexadecimal value automatically.

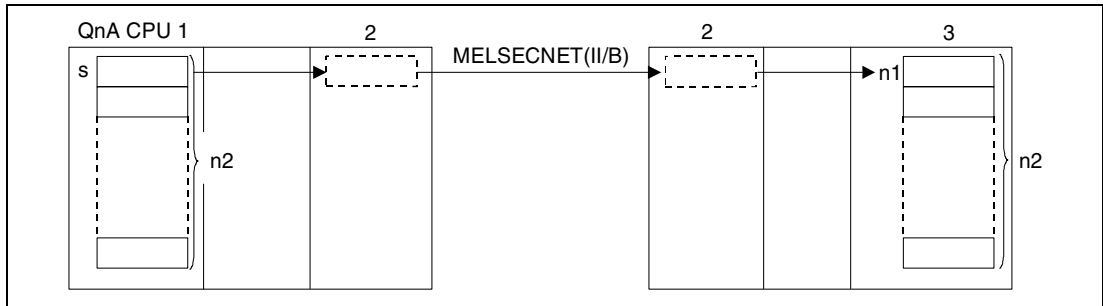
**Functions Writing data to a remote station**

**RTOP Write instruction**

The RTOP instruction writes data to the buffer memory in a special function module in a remote station connected to the MELSECNET.

The number of data words to be written is specified by n2 (A series = n3). The address area in the buffer memory is specified by n1 onwards (A series = n2). The I/O number of the connected special function module is specified by Un (A series = n1). The data to be written is stored in the link register specified by s in the master station.

After the completion of the write operation in the remote I/O station the device specified in d is set (QnA series only).



- 1 Host station (master station)
- 2 Data link module
- 3 Special function module (object station/remote I/O station)

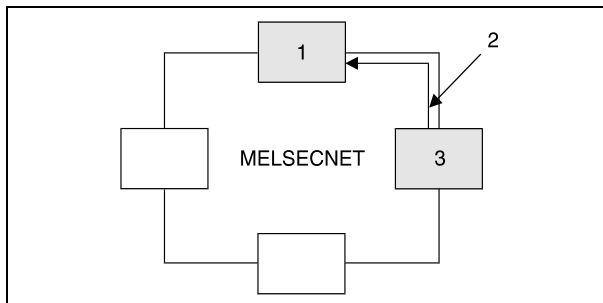
**NOTE**

*Even if only a write operation is executed the address area in the link registers in s must range within the MELSECNET parameterization of the remote and master station.*

In the following the I/O numbers of the special function modules for the QnA series are described. These specifications are valid for the A series as well except for the value n which has to be replaced by n1 (e.g. QnA series = Y(n+F) ⇒ A series = Y(n1+F)).

During the execution of the RTOP instruction the output Y(n+F) is set. X(n+1F) will be set as soon as the instruction is completed. Y(n+F) remains set after the execution completion and therefore has to be reset by the sequence program. The addressing applies automatically and must not be changed.

Write operations to remote I/O stations can be performed by a master station connected to the MELSECNET.



- 1 Station, executing the RTOP instruction (master station)
- 2 Write operation of the data to the special function module
- 3 Remote I/O station

If the RTOP instruction cannot be executed due to an error in the addressed special function module, X(n+1D) will be set. In this case the according module should be checked. X(n+1D) is reset once Y(n+D) is set.

The head I/O number of special function modules specified by Un in 4-digit format is stored in the upper 3 places. For example, the addresses X/Y0200 are specified 20 (QnA series only).

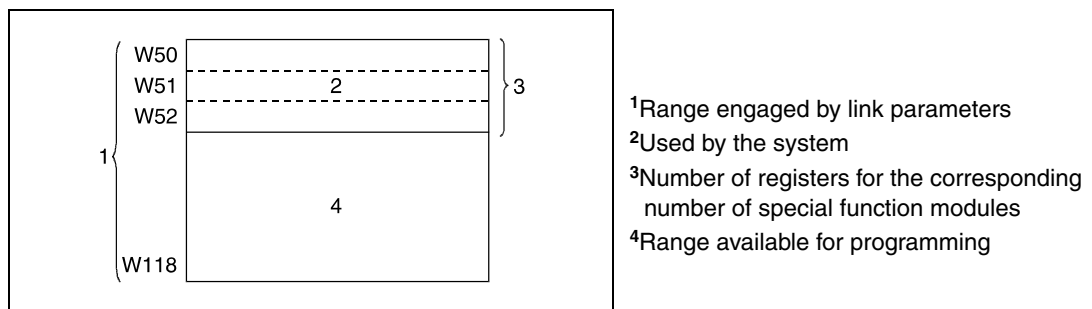
**NOTE**

*Refer to the manual for the corresponding special function module for further information on the valid address range of the buffer memory in special function modules specified by n1 (A series = n2).*

The send data length (number of data words) specified by n2 (A series = n3) may range from 1 to 16.

The range of the link register Wxxx between master and remote station must be differentiated precisely. The number of link register addresses used by the operating system equals the number of special function modules contained in the remote stations of a network. The range available for data storage is the parameter range minus the link register addresses used by the operating system.

The example illustrated below shows the different areas of a link register. The area between master and remote station is specified W050 through W118 (A series = W09F) in the parameters. In this area 2 special function modules are allocated so the first two link registers W50 and W51 (2 addresses) are engaged by the operating system of the CPU. The available area for data storage therefore ranges from W52 to W118 (A series = W09F).



The RTOP and RFRP instructions cannot be executed from several locations simultaneously by the same special function module. A simultaneous execution from two or more locations is prevented through a handshake of the two active stations.

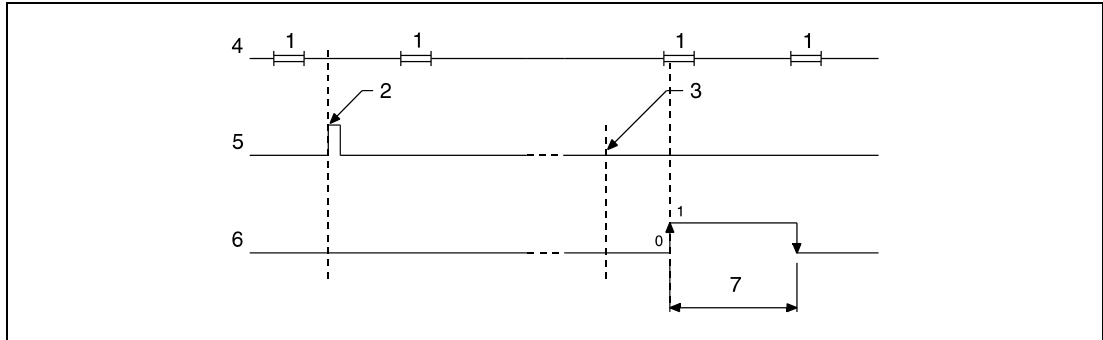
The inputs and outputs X(n+1F) and Y(n+F) should be programmed as interlock to ensure that no other RTOP or RFRP instruction can be executed.

The host station completion device (d) is set with the execution of the END instruction within the program scan the write operation was completed in. The device is reset with the next END processing (QnA series only).



The MELSEC A series supplies numerous special registers for data transfer in the MELSEC-NET that register various communication states. For example, the status of the remote I/O stations is registered through the special registers D9228 through D9231. Parameter access is evaluated through the special registers M9224 through M9227 (A series only).

The following figure shows the operations of the host station during the execution of the RTOP instruction:



- 1 END processing
- 2 Execution of the RTOP instruction
- 3 Completion of the operation
- 4 Program of the host station
- 5 RTOP instruction
- 6 Host station completion device set after completion of the operation (d) (QnA series only)
- 7 One scan

### Operation Errors

In the following cases an operation error occurs and the error flag is set:

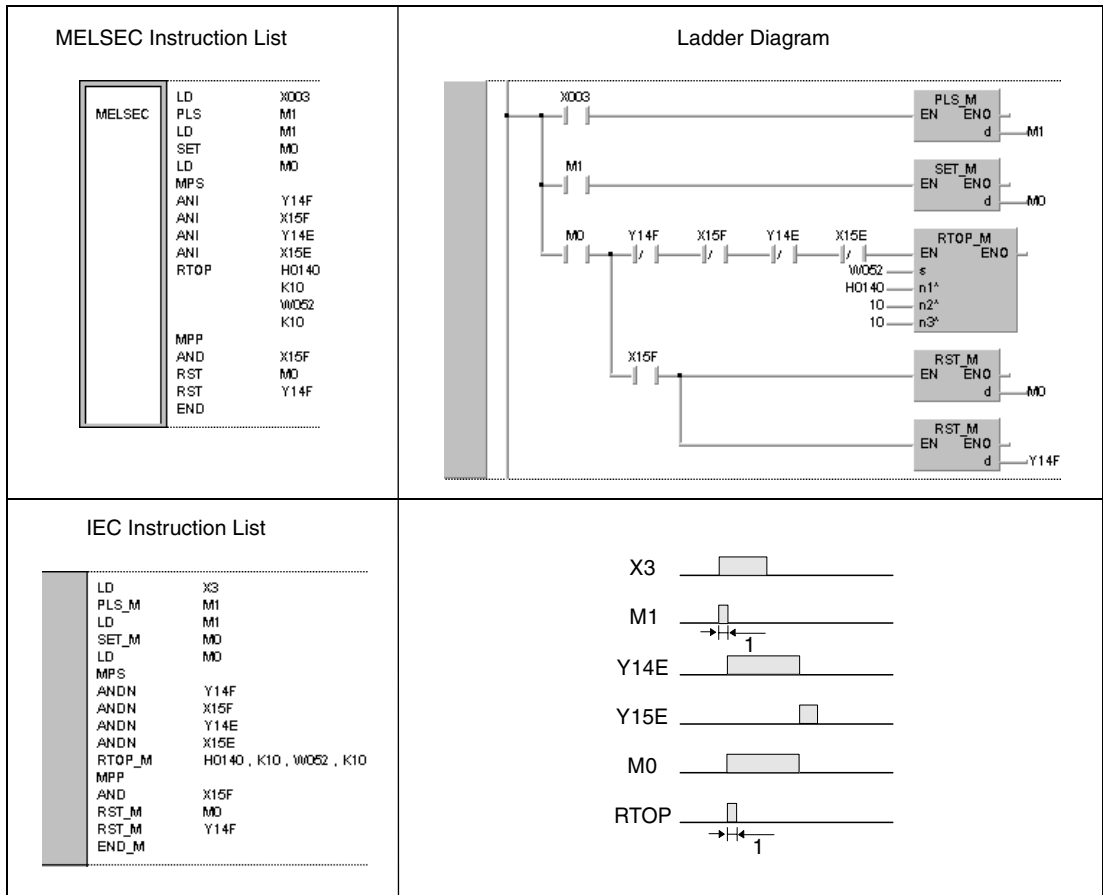
- The I/O number specified by Un (A series = n1) is not that of a remote I/O station (QnA series = error code 4102).
- The I/O number specified by n1 (A series = n2) is not the head I/O number of a special function module (error code 4102).
- The number of addresses specified by n2 (A series = n3) exceeds the address range specified from d1 onwards (A series = d, W0 through W3FF) (QnA series = error code 4101).
- The network specified by Un (A series = n1) does not exist (error code 2413).
- The value specified for n2 (A series = n3) exceeds the range of 1 to 16 (error code 4100).

**Program Example**

RTOP (A series)

The following program writes data from the link registers W52 through W61 in the master station to 10 successive addresses in a special function module (e.g. A68AD). The module is located at the second remote station. The addresses range from 140 through 15F. The written data is stored in the address area beginning with address number 10.

After switching X3 ON M0 is set and the RTOP instruction is executed. With the beginning data transfer Y(n1+F) = Y14F is set. When the data transfer is completed X(n1+1F) = X15F is set. The RTOP instruction is not executed, if another RTOP or RFRP instruction is already executed. After completion of the transfer in the further course of the program M0 and Y14F are reset.



<sup>1</sup> One single execution

**NOTE**

The contact corresponding to M1 should be converted into a pulse. Otherwise the RTOP instruction would not be executed completely.

The contact corresponding to M0 should be set via a SET instruction. If an OUT or PLS instruction is programmed instead of the SET instruction, errors might occur with the execution of the RTOP instruction.

In order to prevent the simultaneous execution of two RTOP instructions an interlock must be established through the output Y14F and the input X15F.

If within the same program this station is accessed via an RFRP instruction the output Y14E and the input X15E must be programmed as an interlock in addition.

## 8.8 Reading and writing routing information

These instructions read and write routing information. The routing parameters comprise network and station number of the relay station and the station number of the routing station.

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Read routing information	Z.RTREAD	RTREAD_M
	ZP.RTREAD	RTREADP_M
Write routing information	Z.RTWRITE	RTWRITE_M
	ZP.RTWRITE	RTWRITEP_M

8.8.1 RTREAD

CPU

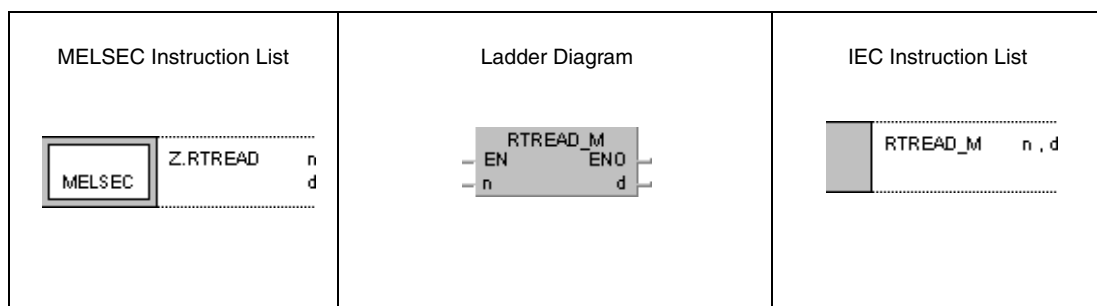
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

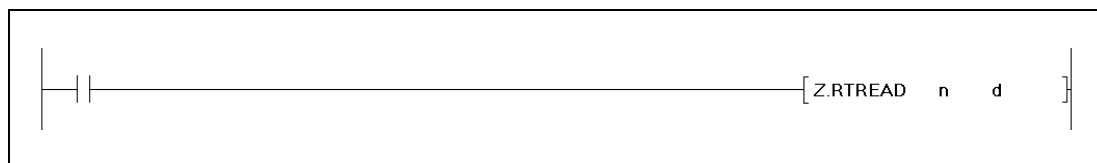
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
n	—	●	●	—	—	—	—	●	—	SM0	7
d	—	●	●	—	—	—	—	—	—		

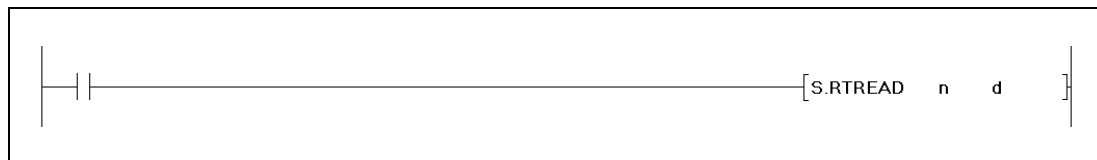
GX IEC Developer (QnA CPU)



GX Developer (QnA CPU)



GX Developer (System Q CPU)



Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
n	Destination network of transmission (1 to 239).	BIN 16-bit	ANY16
d	First number of device storing read routing information.	Device number	Array [1..3] of ANY16

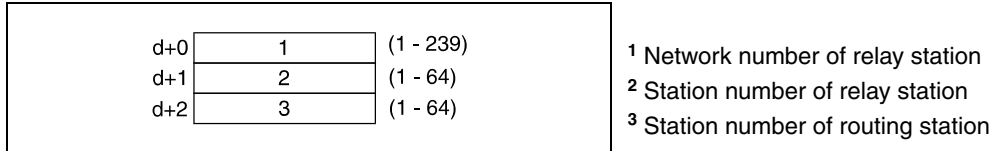
**Functions**      **Reading routing information**

**RTREAD      Read instruction**

The RTREAD instruction reads the routing information from the destination network specified by n. The routing information is stored in routing parameters. The read routing information is stored from d+0 (Array\_d[1]) onwards.

If no data is specified for the transmission the value 0 is written to the devices specified from d on (Array\_d[1] through Array\_d[3]).

The figure below shows the contents specified from d+0 (Array\_d[1]) on:

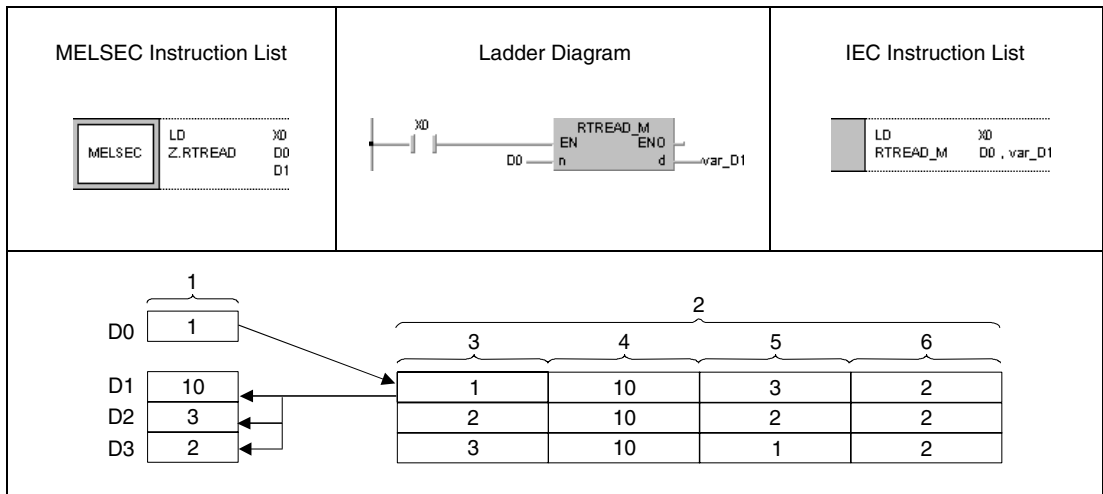


**Operation Errors**      In the following cases an operation error occurs and the error flag is set:

- The data value specified for n does not range within 1 and 239 (error code 4100).

**Program Example**      Z.RTREAD

While X0 is set, the following program reads the routing information from the network (11) specified by D0 and stores the data in D1 through D3 (var\_D1[1] through var\_D1[3]).



- 1 Operation
- 2 Contents of routing parameter settings
- 3 Network number of destination network for transmission
- 4 Network number of relay station
- 5 Station number of relay station
- 6 Station number of routing station

**NOTE**      *This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

8.8.2 RTWRITE

CPU

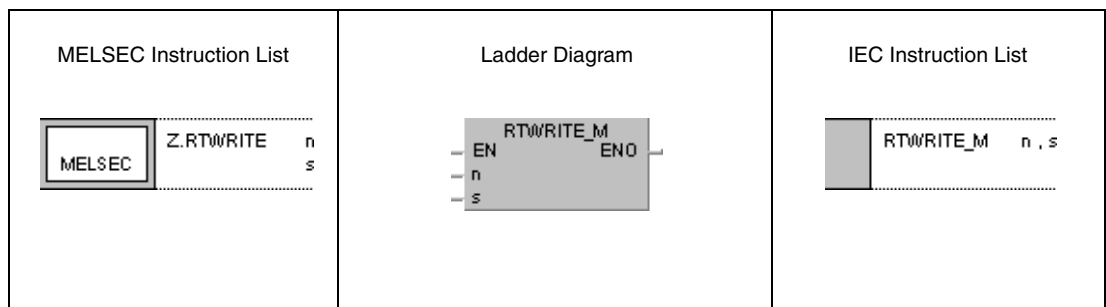
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

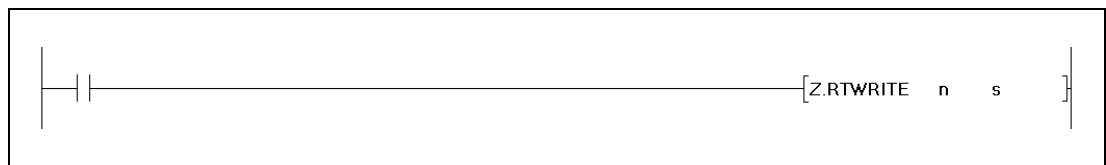
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
n	—	●	●	—	—	—	—	●	—	SM0	8
s	—	●	●	—	—	—	—	—	—		

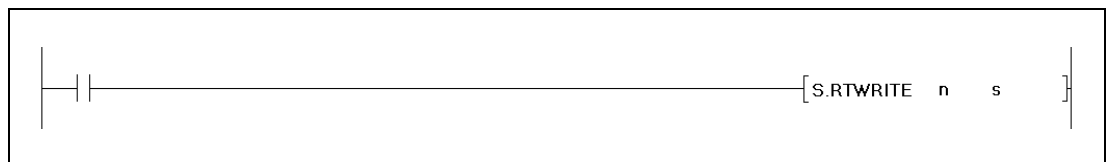
GX IEC Developer (QnA CPU)



GX Developer (QnA CPU)



GX Developer (System Q CPU)



Variables

Set Data	Meaning	Data Type	
		MELSEC	IEC
n	Destination network of transmission (1 to 239).	BIN 16-bit	ANY16
s	First number of device storing routing information to be written.	Device number	Array [1..3] of ANY16

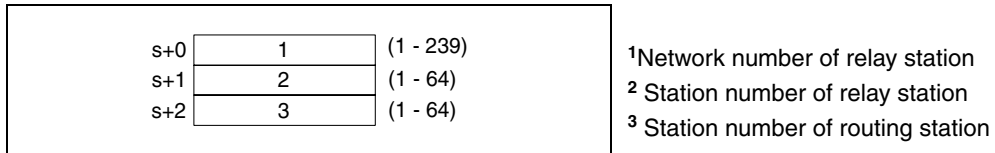
**Functions Writing routing information**

**RTWRITE Write instruction**

The RTWRITE instruction writes the routing information to the destination network specified by n. The routing information is stored in routing parameters. The read routing information is stored from s+0 (Array\_s[1]) onwards.

If data for the destination network is set in the routing parameters, it is used to refresh the data stored from s+0 (Array\_s[1]) on.

The figure below shows the contents specified from s+0 (Array\_d[1]) on:

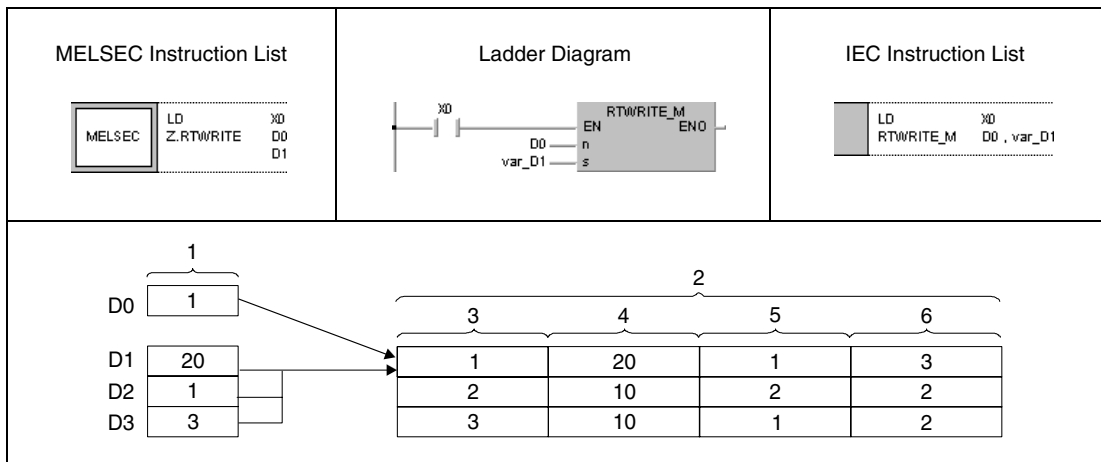


**Operation Errors** In the following cases an operation error occurs and the error flag is set:

- The data value specified for n does not range within 1 and 239 (error code 4100).
- The data specified by s exceed the relevant ranges (error code 4100).

**Program Example Z.RTWRITE**

While X0 is set, the following program writes the routing information stored in D1 through D3 (var\_D1[1] through var\_D1[3]) as routing parameters to the network (1) specified by D0.



- <sup>1</sup> Operation
- <sup>2</sup> Contents of routing parameter settings
- <sup>3</sup> Network number of destination network for transmission
- <sup>4</sup> Network number of relay station
- <sup>5</sup> Station number of relay station
- <sup>6</sup> Station number of routing station

**NOTE** *This program example will not run without variable definition in the header of the program organization unit (POU). It would cause compiler or checker error messages. For details see Chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*





## 9 Instructions for System Q CPUs

The following instructions are only available for a CPU of the System Q.

Function	MELSEC-Instruction in MELSEC-Editor	MELSEC-Instruction in IEC-Editor
Reading module information	UNIRD	UNIRD_M
	UNIRDP	UNIRDP_M
Debugging and failure diagnosis instructions	TRACE	TRACE_M
	TRACER	TRACER_M
Writing to and reading from a file	FWRITE	FWRITE_M
	FREAD	FREAD_M
Program instructions	PLOADP	PLOADP_M
	PUNLOADP	PUNLOADP_M
	PSWAPP	PSWAPP_M
Data transfer instructions	RBMOV	RBMOV_M
	RBMOVP	RBMOVP_M

To the System Q CPUs from function version B (Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU and Q25HCPU) the following instructions for use in a multi-CPU system are added:

Function	MELSEC-Instruction in MELSEC-Editor	MELSEC-Instruction in IEC-Editor
Write to CPU shared memory of host station	S.TO	TO_S_M
	S.TOP	TO_SP_M
Read from CPU shared memory of another station	FROM	FROM_M
	FROMP	FROMP_M

## 9.1 Reading Module Information

### 9.1.1 UNIRD, UNIRDP

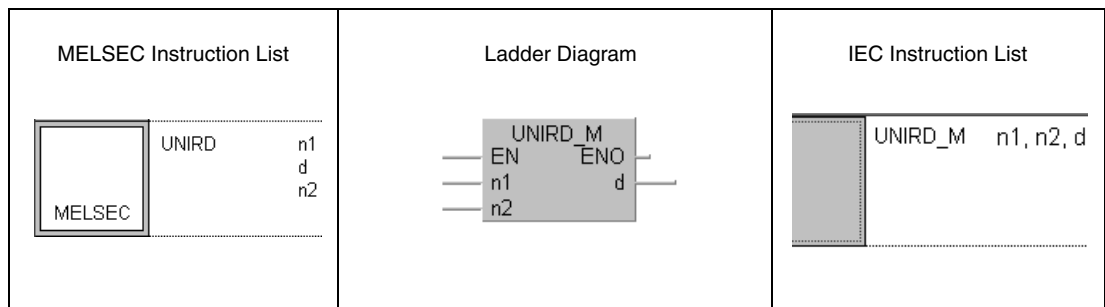
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

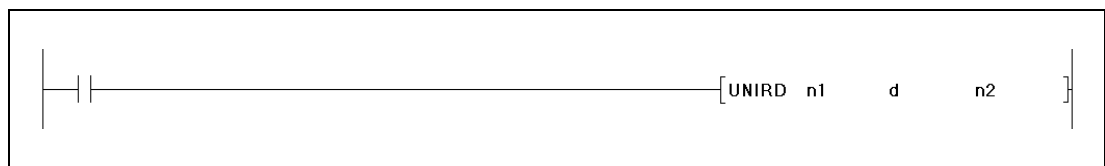
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
n1	●	●	●	—	—	—	—	●	—	SM0	4
d	—	●	●	—	—	—	—	—	—		
n2	●	●	●	—	—	—	—	●	—		

GX IEC Developer



GX Developer



Variables

Device	Meaning	Data Type
n1	Value obtained by dividing the head I/O number of the module from which module information is read by 16 (0 bis FF <sub>H</sub> ).	BIN 16-Bit
d	Head number of the device which stores module information.	Device name
n2	Number of points of read data (0 bis 256).	BIN 16-Bit

**Functions**     **Reading module information**

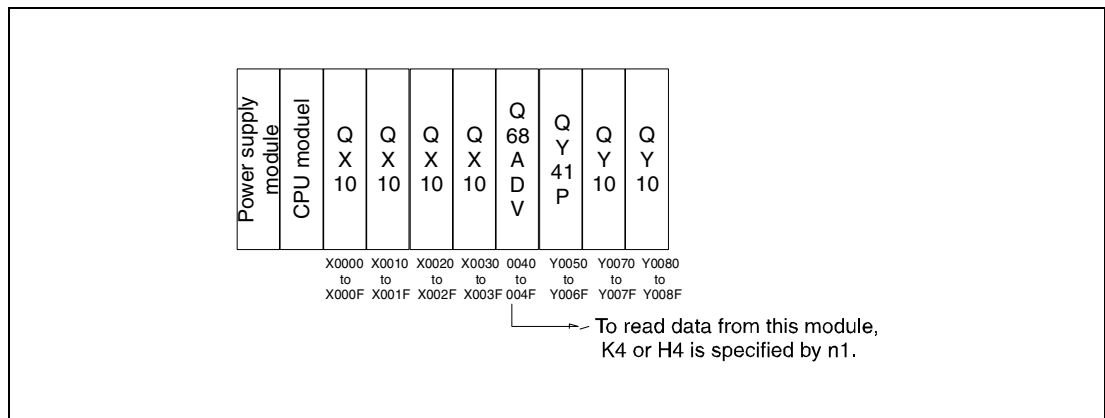
**UNIRD    Read instruction**

The UNIRD instruction reads the module information starting at the head I/O address, which is specified by n1 and stores the data at the address which is specified by d. The number of points is specified by n2. The value for n1 is calculated by dividing the head I/O number of the module by 16.

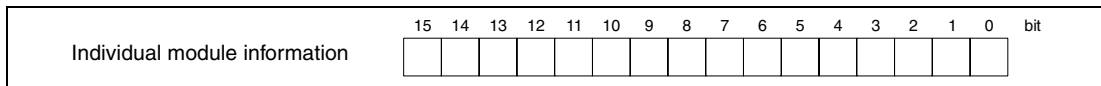
With the UNIRD instruction it is possible to read the statuses of the actually installed modules instead of the module type designated by I/O assignment.

**NOTE**

*The value of n1 is consists of the higher three digits of the head I/O number of the slot from which the module information is read. The head I/O number is expressed in 4 digits in hexadecimal notation.*



The details of the module information are described as follows:



Bit	Item	Meaning
0	Number of I/O points	000: 16      001: 32      010: 48      011: 64 100: 128    101: 256    110: 512    111: 1024
1		
2		
3	Module type	000: Input module 001: Output module 010: I/O mixed module 011: Intelligent function module
4		
5		
6	External power supply status (For future expansion)	ON: External power supply is connected OFF: External power supply is not connected
7	Fuse status	ON: Blown fuse OFF: Normal, no blown fuse
8	Vacant	
9	Light/medium error status	ON: Light/medium error has occurred OFF: Normal
10	Module error status	00: No module error 01: Light error 10: Medium error 11: Serious error
11		
12	Module standby status	ON: Normal OFF: Module error occurred
13	Vacant	
14	A-/Q-Modul	ON: The module is a A-series module OFF: The module is a Q-series module
15	Module installation status	ON: Modules are installed OFF: No Modules are installed

**Operation Errors**

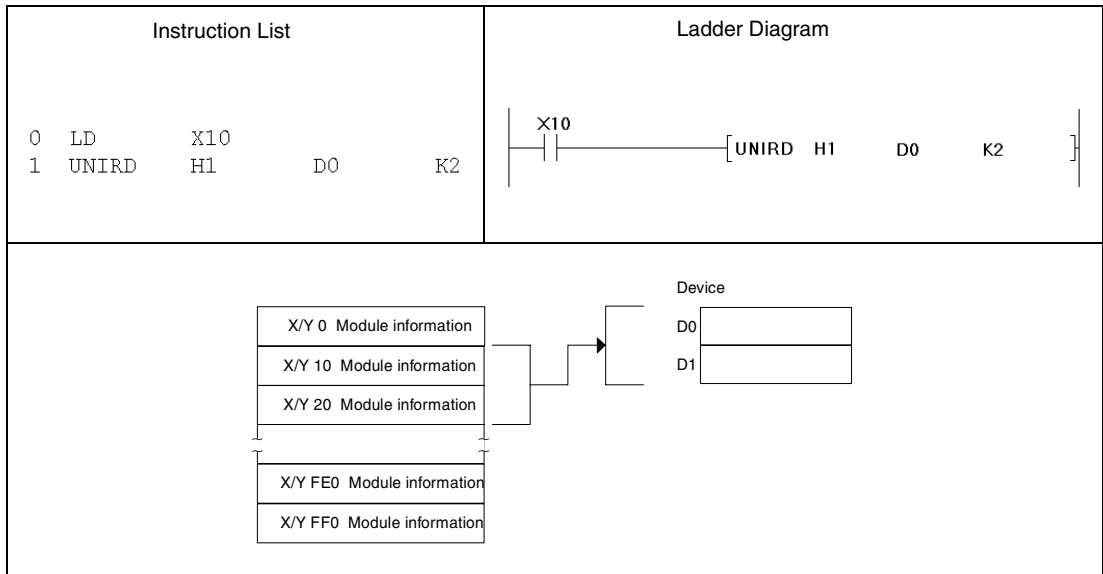
In the following cases an operation error occurs and the error flag is set:

- A value outside the relevant value range (0 through FF<sub>H</sub>) is specified in n1 (error code 4100).
- A value outside the relevant value range (0 through FF<sub>H</sub>) is specified in n2 (error code 4100).
- The sum of n1 and n2 is larger than 256 (error code 4100).

**Program Example**

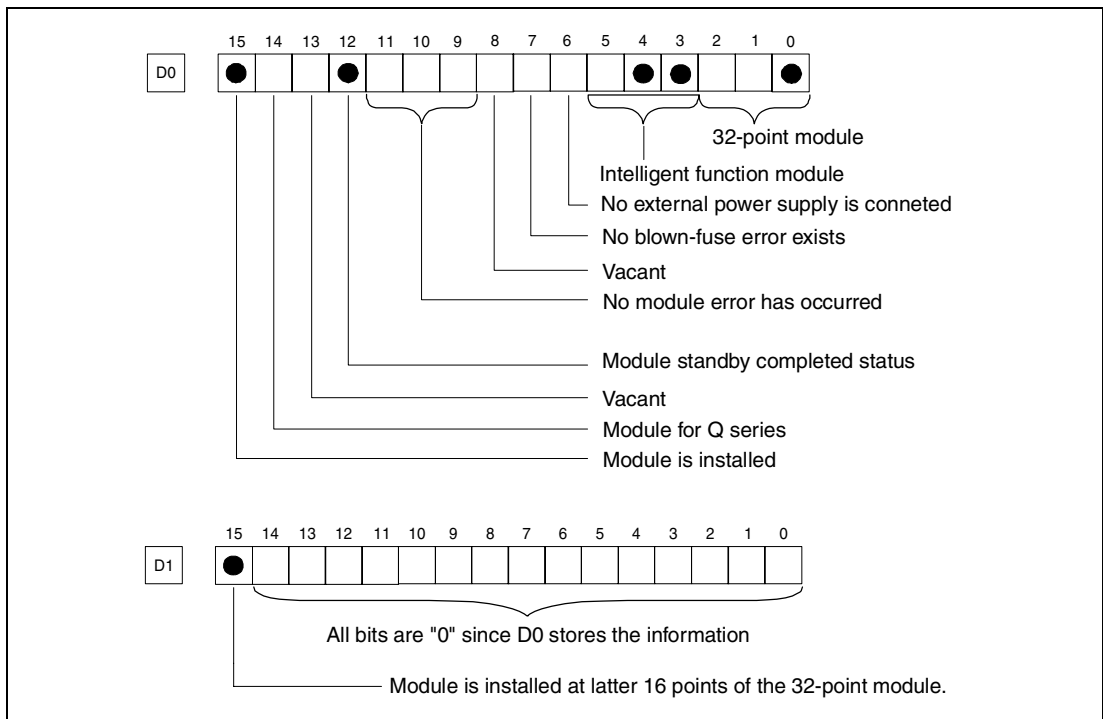
UNIRD

The following program stores the informations of the modules with the I/O numbers 10<sub>H</sub> through 2F<sub>H</sub> to D0 and D1, when X10 is turned ON.

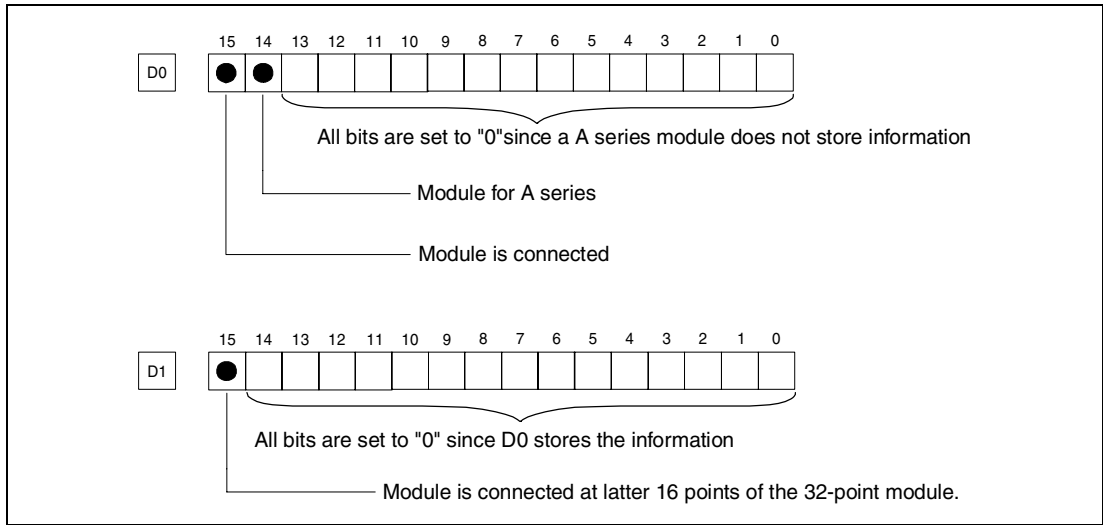


In this program example the module information is stored in D0 and D1. Readout results can be:

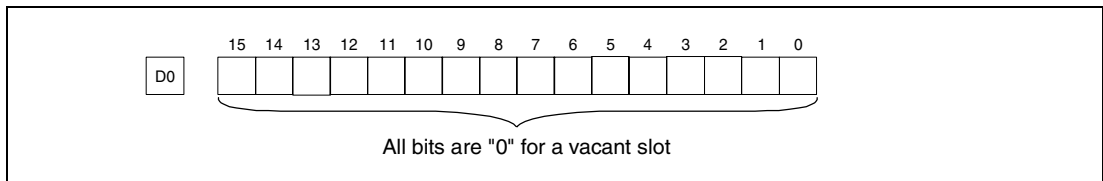
- For a 32-point intelligent function module of the System Q. With a 48- or 64-point module the same contents as stored in D1 is stored in D2 or D2 and D3 respectively.



- For a 32-point intelligent function module of the System Q. With a 48- or 64-point module the same contents as stored in D1 is stored in D2 or D2 and D3 respectively.



- Module information for a vacant slot:



## 9.2 Debugging and failure diagnosis instructions

### 9.2.1 TRACE, TRACER

**CPU**

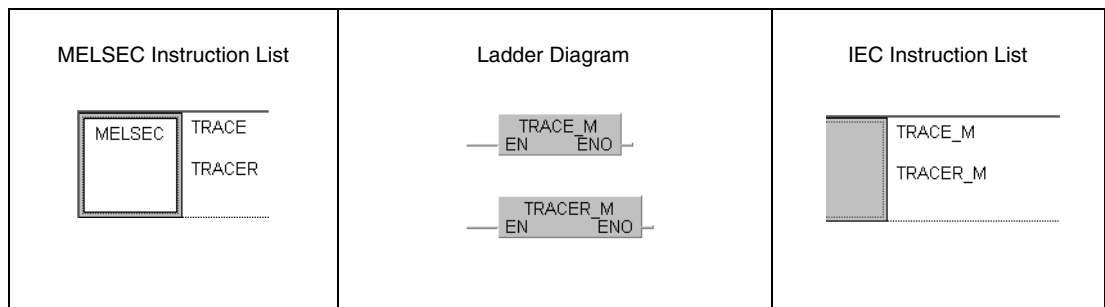
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

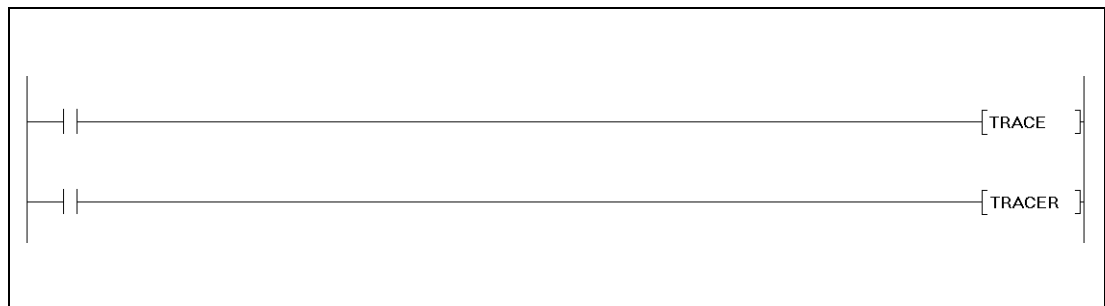
**Devices  
MELSEC Q**

Usable Devices										Error Flag	Number of steps
Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other			
Bit	Word		Bit	Word							
—	—	—	—	—	—	—	—	—	—	—	1

**GX IEC Developer**



**GX Developer**



**Variables**

Set data	Meaning	Data Type
—	—	—

**Functions**      **Trace set and trace reset**

**TRACE**      **Trace set**

The TRACE instruction stores the trace data designated by a peripheral device in the trace file in the memory card by the designated number when SM800, SM801, and SM802 turn ON. When the TRACE instruction is executed, SM803 turn ON. The sampling is repeated by the specified number of sampling trace after the TRACE instruction, then, data is latched and the trace is stopped.

The sampling is stopped if SM801 goes OFF during the trace execution.

After the TRACE instruction is executed and the trace is completed, SM805 turn ON.

During the execution of the TRACE instruction, other TRACE instructions are ignored. After the TRACE instruction is executed, the TRACE instruction is enabled again.

**TRACER**      **Trace reset**

The TRACER instruction resets the TRACE instruction and the flags SM803 through SM805. After the TRACER instruction is executed, the TRACE instruction is enabled again.

**NOTE**

*Please refer to the System Q CPU (Q mode) User's Manual (Functions/programming fundamentals) for more informations about trace.  
Please refer to the operating manuals for the GX Developer and GX IEC Developer for the execution of the trace with peripheral devices.*

**Program Example**

**TRACE, TRACER**

The following program executes the TRACE instruction when X0 is turned ON. When X1 is turned ON, the TRACE instruction is reset by the TRACER instruction.

Instruction List	Ladder diagram
<pre> 0  LD      X0 1  TRACE 2  LD      X1 3  TRACER                     </pre>	



## 9.3 Writing to and reading from files

### 9.3.1 FWRITE

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

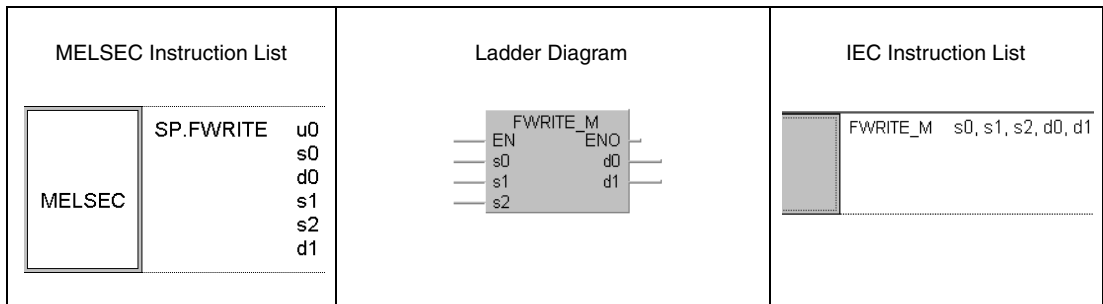
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

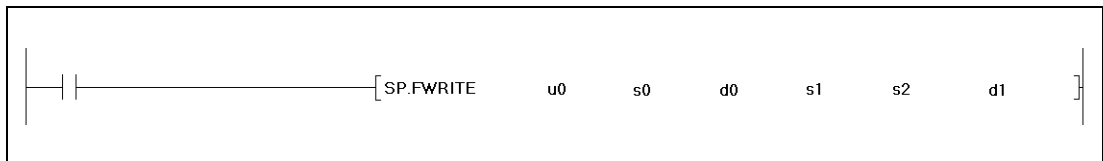
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s0	●	●	●	—	—	—	—	●	—	SM0	11
d0	—	●	●	—	—	—	—	—	—		
s1	—	●	●	—	—	—	—	—	—		
s2	—	●	●	—	—	—	—	—	●		
d1	●*	●*	●*	—	—	—	—	—	—		

\* Local devices and the devices designated for individual programs cannot be used.

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set data	Meaning	Setting Range	Set By	Data Type		
u0	Dummy	—	—	BIN 16-bit		
s0	Drive designation. Only the drive with ATA card can be designated (drive 2). A memory card (ROM) or standard RAM/ROM cannot be designated.	2	User			
d0	Head number of the device storing the control data. The following control data is required.				BIN 16-bit	
	Set data	Item	Meaning/Set Data	Setting Range		Set By
	(d0)	Execution type	Specifies the execution type: 0000 <sub>H</sub> : Write binary data 0100 <sub>H</sub> : Write data after conversion into CSV format	0000 <sub>H</sub> 0100 <sub>H</sub>		User
	(d0)+1	(Reserved)	Used by system	—		System
	(d0)+2	Writing result (Number of written data)	Contains the number of actually written data. The unit for the value is determined by word/byte unit designation.	—		System
	(d0)+3	Not used	—	—		—
	(d0)+4 (d0)+5	Location in file	Sets the location in the file to start writing when <b>binary data</b> is selected (d0 = 0000 <sub>H</sub> ). 00000000 <sub>H</sub> : From the beginning of the file 00000001 <sub>H</sub> to FFFFFFFE <sub>H</sub> : From the specified address. The unit for the value is determined by word/byte unit designation.  FFFFFFF <sub>H</sub> : Add to the ending of the file.  When data writing after <b>CSV format</b> conversion is selected (d0 = 0100 <sub>H</sub> ): For a CPU whose serial number is „01111“ or earlier in the upper 5 digits, always set the beginning of the file (00000000 <sub>H</sub> ). For a CPU whose serial number is „01112“ or later set the file position. 00000000 <sub>H</sub> to FFFFFFFE <sub>H</sub> : From the beginning of the file. FFFFFFF <sub>H</sub> : Add to the ending of the file.	00000000 <sub>H</sub> to FFFFFFF <sub>H</sub>		User
	(d0)+6	Number of columns	Sets the number of columns to write data in CSV format. 0: : No column setting. Data is shown in a single row. > 0 : Data is shown in the specified number of columns	0 to 65535		User
	(d0)+7	Word/Byte designation	0: Word 1: Byte	0, 1		User

## Variables

Set data	Meaning			Setting Range	Set By	Data Type
s1	Head number of the device storing a file name.					BIN 16-bit
	Set data	Item	Meaning/Set Data	Setting Range	Set By	
	(s1)+1 to (s1)+n	File name	The file name consists of up to 8 characters + period + extension (for example: ABD.BIN). The extension can be omitted. In this case, the period („.“) can also be omitted. When more than 8 characters are used, the extension is ignored regardless of its presence. The Extension „BIN“ or „CSV“ is assigned automatically.	Character string	User	
s2	Head number of the device storing the data.					BIN 16-bit
	Set data	Item	Meaning/Set Data	Setting Range	Set By	
	(s2)	Number of data to be written	Sets the number of data to be written (in units of words). This number should be designated in the unit of words even when byte is selected in (d0)+7.	1 to 480	User	
(s2)+1 to (s2)+n	Data to be written	Data requested to be written.	0000 <sub>H</sub> to FFFF <sub>H</sub>			
d1	Bit device that goes ON after the execution of the FWRITE instruction. When an error occurs, (d1)+1 goes ON.					Bit
	Set data	Item	Meaning/Set Data	Setting Range	Set By	
	(d1)	Completion signal	Indicates the completion of the FWRITE instruction. ON: Completed OFF: Not completed	—	System	
(d1)+1	Error completion signal	Indicates whether the FWRITE instruction is normally completed or abnormally completed. ON: Error completion OFF: Normal completion	—			

## NOTE

The data written in CSV format is expressed as decimal value by the programming software. For example, the character „A“ (41<sub>H</sub>) is written as 65. The available range is from -32768 to 32767.

**Functions Writing data to a designated file**

**FWRITE Write data**

The WRITE instruction writes a specified number of data to the ATA card. The user can select whether to write data as binary data without any conversion or to convert binary data into CSV-format data before writing it.

The completion signal bit device (d1)+0 automatically turns ON after the completion of the FWRITE instruction is detected and the END instruction is executed. The bit device turns OFF at the execution of the END instruction in the next scan.

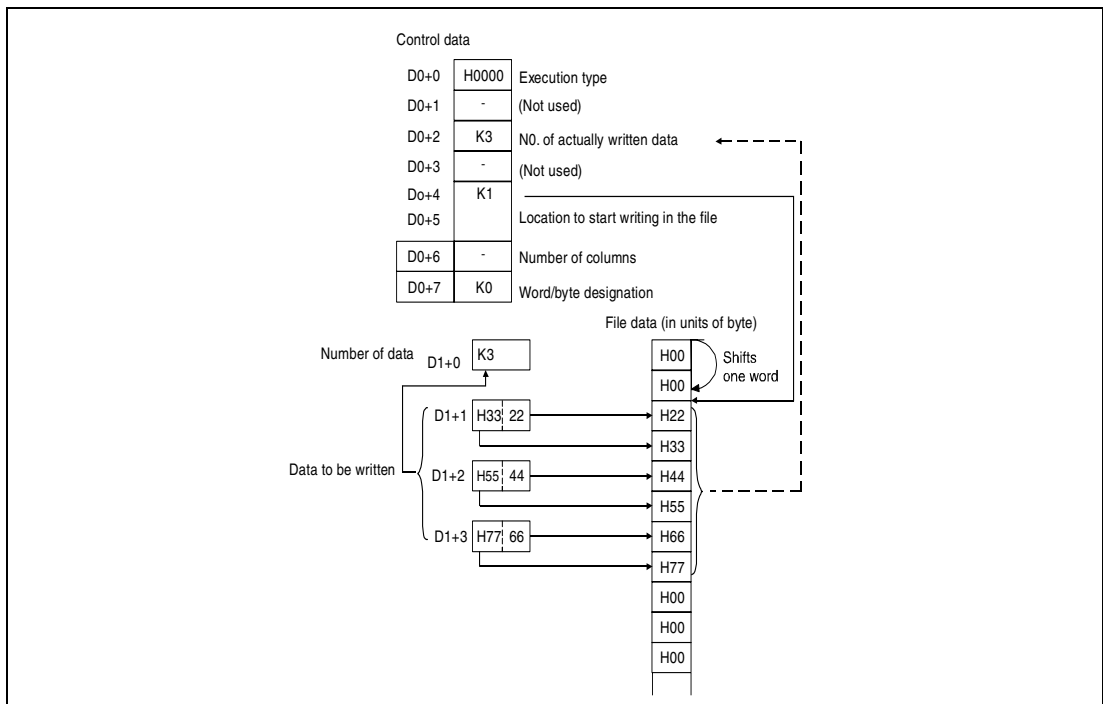
This bit device can be used as the execution completion flag for the FWRITE instruction.

When the FWRITE instruction is completed abnormally, the error completion device (d1)+1 turns ON/OFF in synchronization with the execution completion flag (d1)+0. This bit device can be used as error completion flag for the FWRITE instruction.

SM721 is on during the execution of the FWRITE instruction. SM721 is also used by other instruction such as S.FREAD, COMRD and PRC. The FWRITE instruction cannot be started while SM721 is ON. If an attempt is made, no processing is performed.

When an error is detected prior to the execution of the instruction (before SM721 goes ON), the execution completion device [(d1)+0], the error completion device [(d1)+1] and SM721 do not turn ON.

The unit for the number of data to be written [(s2)+0] is „word“, regardless of the setting in (d0)+7 (word/byte designation).



**Writing of binary data:**

If the extension of the object file is omitted, „.BIN“ is added as an extension.

When the designated file does not exist, a new file is created and the data is added and saved from the beginning of the file. The attributes of this new file are set using archive attributes.

When the size of the data exceeds that of the existing area in the file during the writing, the excess data is added at the end of the file.

An error occurs if the designated location in the file is larger than the file size. A CPU with the serial number 01111 or earlier (the upper 5 digits) will issue a error code. A CPU bearing the serial number 01112 will not write any data and will complete the instruction without an error message.

When the medium runs out of free space when data is added/saved, an error occurs. In such a case, the data that is successfully added/saved remains in the medium. The error completion is indicated after as much data as possible is added/saved.

**Writing of data after CSV format conversion**

If the extension of the object file is omitted, „.CSV“ is added as an extension.

When an existing file is designated and a CPU with the serial number 01111 or earlier (the upper 5 digits) is used, all the contents of the file is deleted and the designated data is saved starting from the beginning of the file.

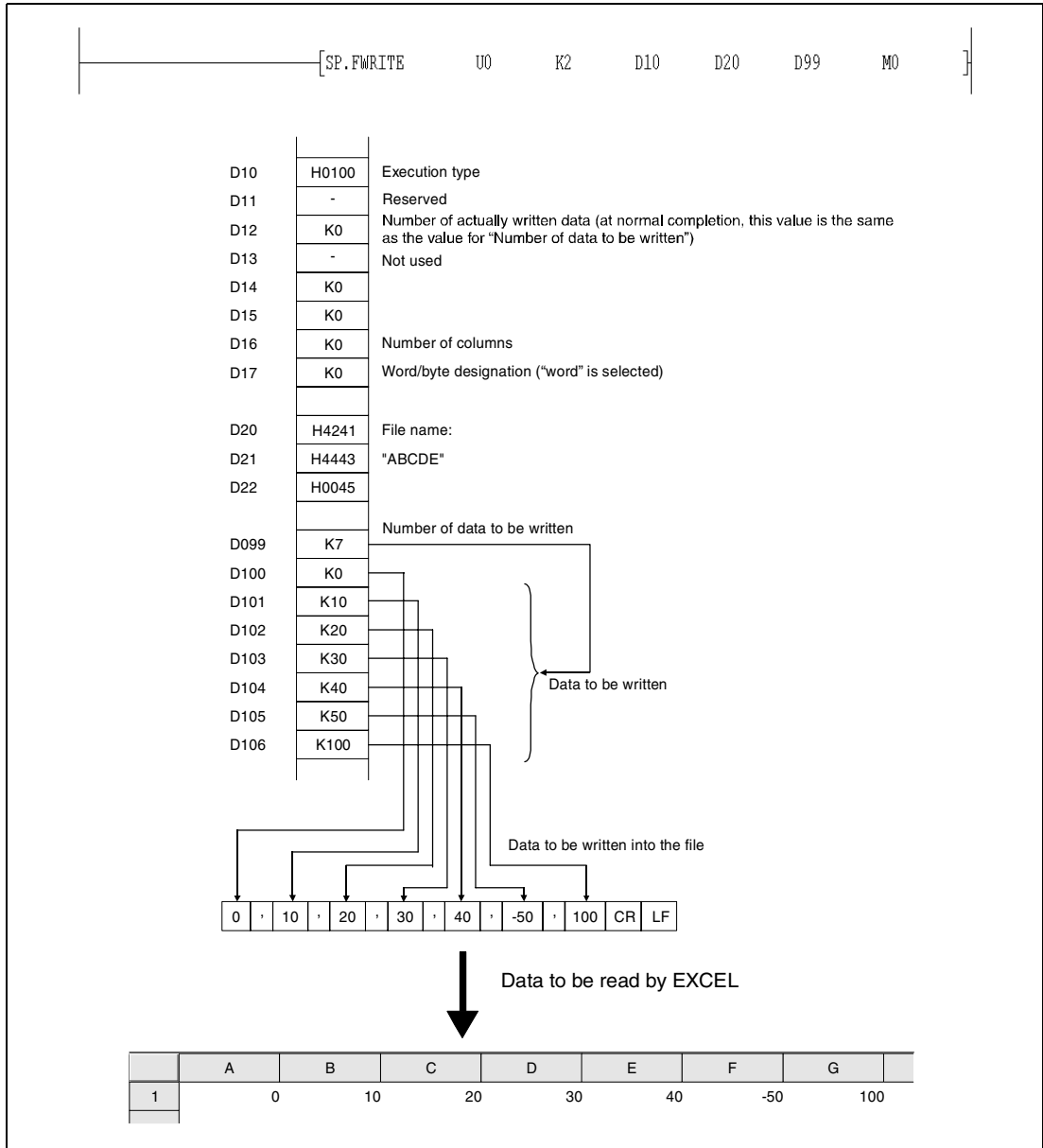
A later CPU (serial number 01112 or later in the upper 5 digits) will react depending of the value written in (d0)+4 and (d0)+5, when an existing file is designated:

When other than FFFFFFFFH is specified in (d0)+4 and (d0)+5, the file contents will be deleted and the data will be stored from the beginning of the file. When FFFFFFFFH is specified in (d0)+4 and (d0)+5, the data is added to the end of the file.

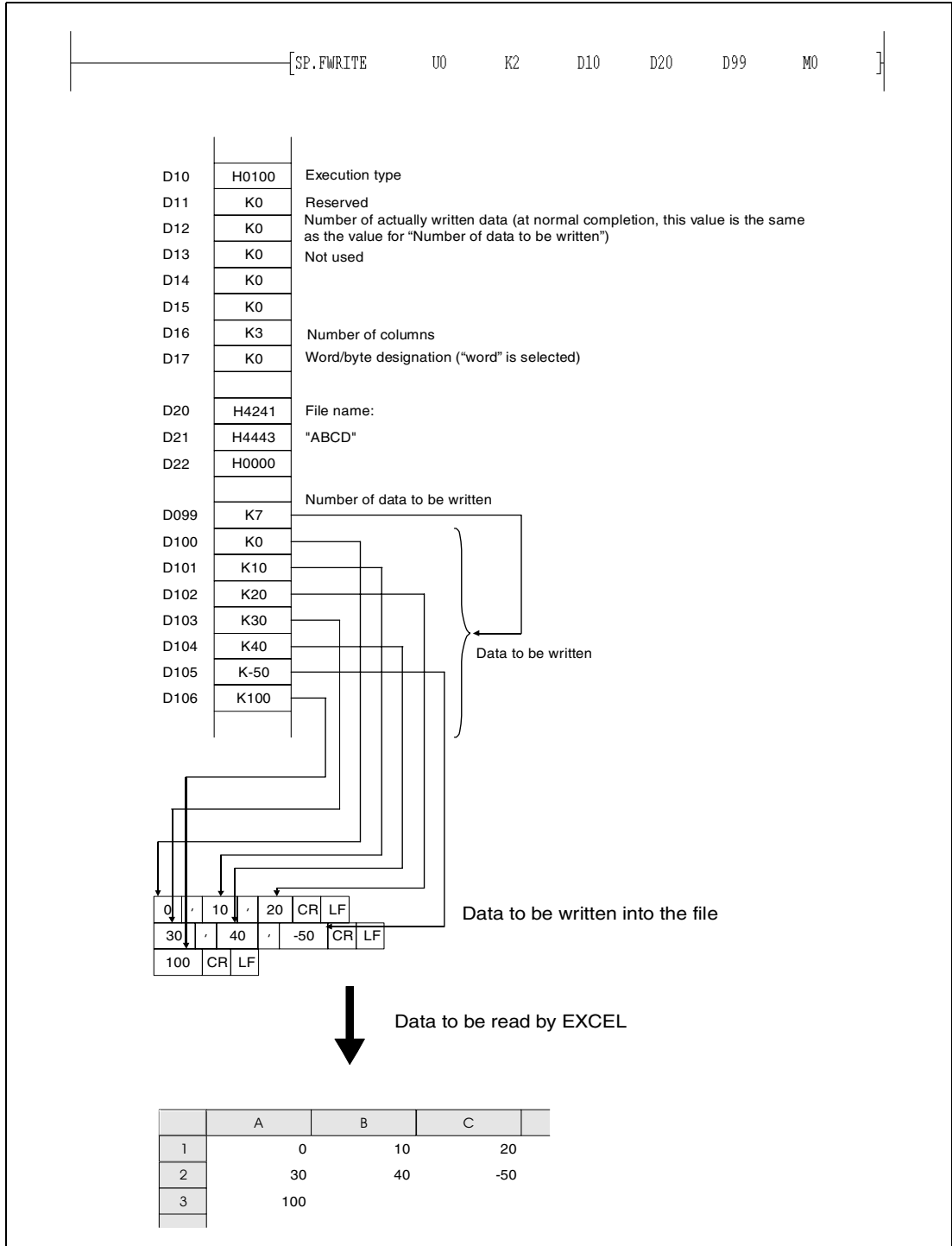
When the designated file does not exist, a new file is created and the data is added/saved from the beginning of the file. The attributes of this new file are set using archive attributes.

An error occurs when the medium runs out of free space when data is added/saved. In such a case, the data that is successfully added/saved remains in the medium. The error completion is indicated after as much data as possible is added/saved.

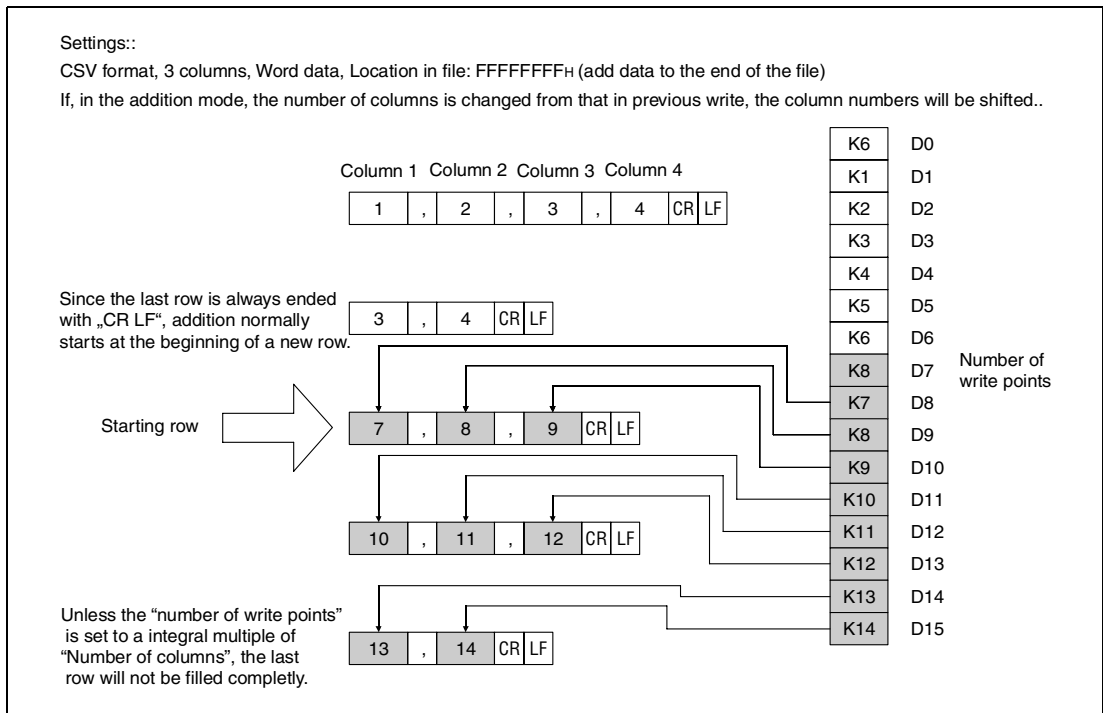
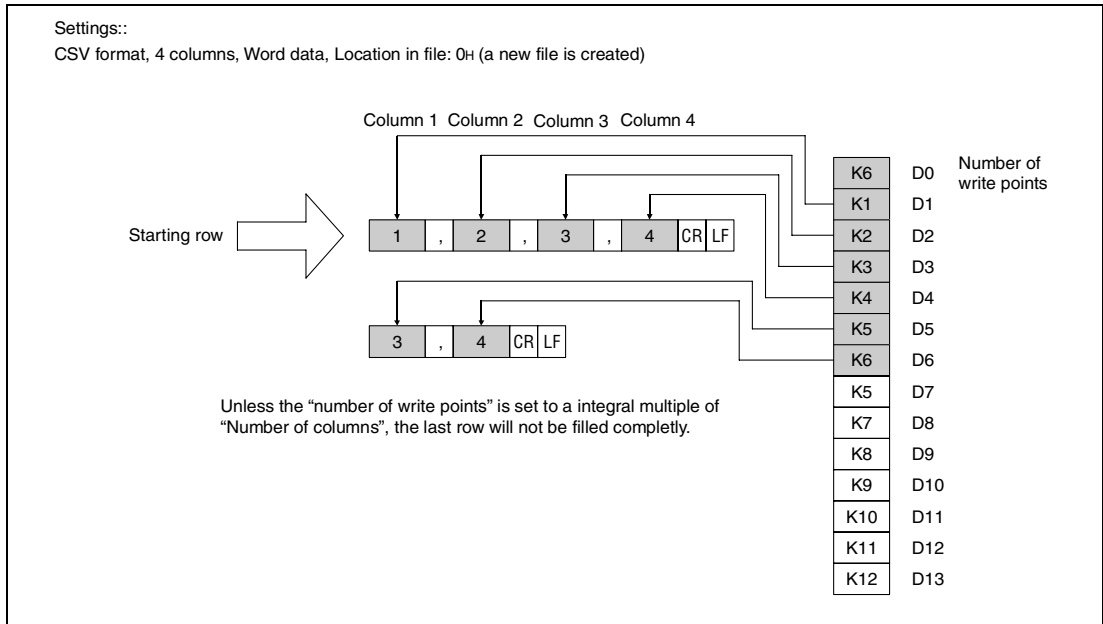
When the designated number of columns is „0“, the data is stored as single-row data in a CSV-format file. The figure on the following page indicates such a case:



When data is written after CSV format conversion and the designated number of columns is other than „0“, the data is stored as table data with the specified number of columns in a CSV format file. The following figure shows an example:



The following two figures are showing examples of writing data with a CPU whose serial number is „01112“ (upper 5 digits).



**NOTE**

*Do not execute the FWRITE instruction in an interrupt program.*



**Operation Errors**

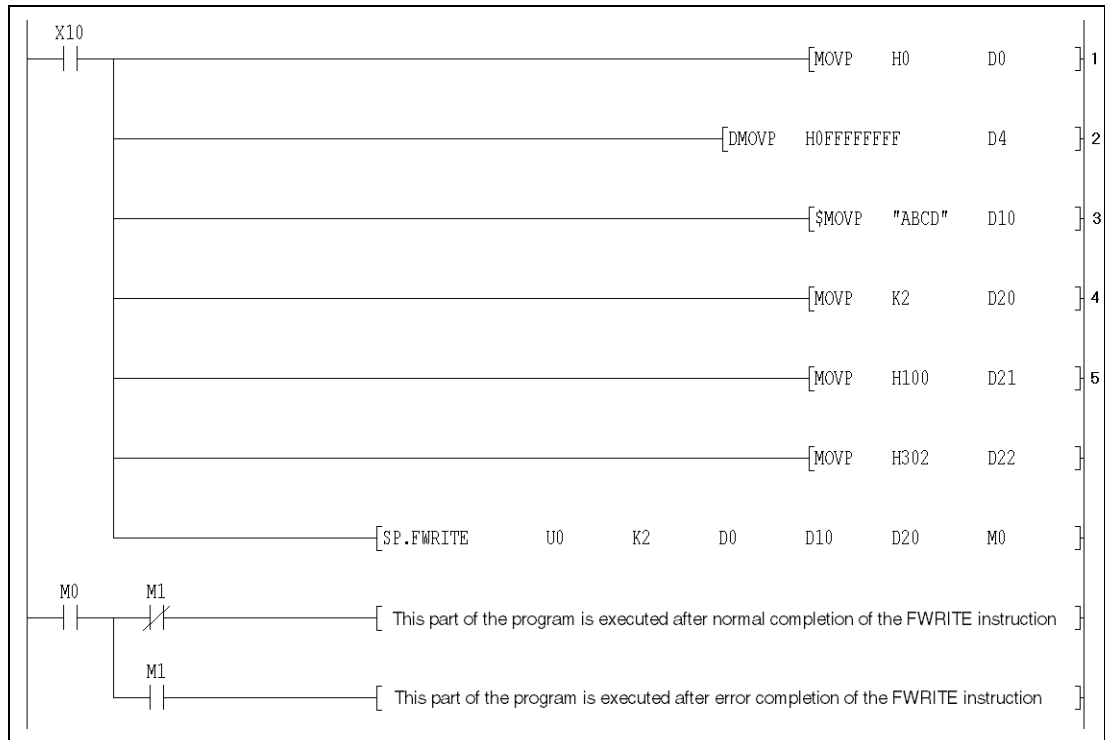
In the following cases an operation error occurs and the error flag is set:

- The drive specified by s0 contains a medium other than an ATA card (error code 4100).
- Values specified in the areas for control data are out of the setting range (error code 4100).
- The value „number of data to be written“ [(s2)+0] is out of the setting range, or is larger than the data stored in the area beginning with (s2)+1 (error code 4101).
- Free space in the medium is insufficient (error code 4100).
- No vacant entry is found when an attempt is made to create a new file (error code 4100).
- An invalid device is designated (error code 4104).

**Program Example 1**

**FWRITE**

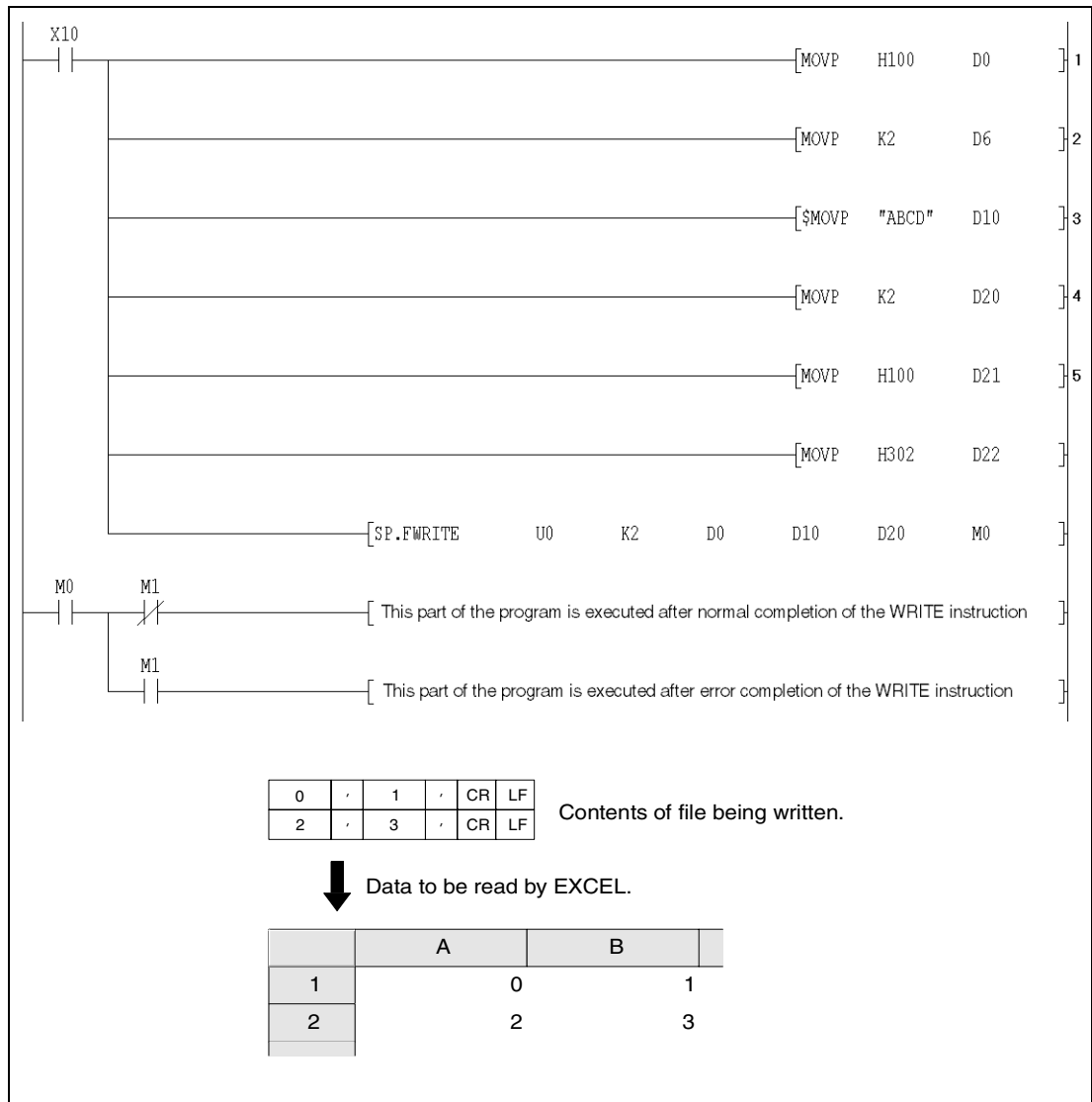
In the following program example, four bytes of binary data (00<sub>H</sub>, 01<sub>H</sub>, 02<sub>H</sub>, and 03<sub>H</sub>) are added to file „ABCD.BIN“ when X10 turn ON. The memory card is inserted in drive 2. Beginning with D0, eight points are reserved for control data.



- <sup>1</sup> Setting of the execution type (In this example: binary data)
- <sup>2</sup> Setting of the location in the file (In this example: data is added)
- <sup>3</sup> Setting of the file name, the extension „.BIN“ is added automatically.
- <sup>4</sup> Number of data to be written.
- <sup>5</sup> The data (00<sub>H</sub>, 01<sub>H</sub>, 02<sub>H</sub>, and 03<sub>H</sub>) is moved to the control data area.

**Program Example 2** FWRITE

When X10 is turned ON, the following program creates a file named „ABCD.CSV“ in the memory card inserted to drive 2. Then, four bytes of data (00<sub>H</sub>, 01<sub>H</sub>, 02<sub>H</sub> and 03<sub>H</sub>) are written as two-column table data in CSV format. Control data is stored from D0 onward (8 points).



- <sup>1</sup> Setting of the execution type (In this example: CSV format)
- <sup>2</sup> Setting of the number of columns
- <sup>3</sup> Setting of the file name, the extension „.CSV“ is added automatically.
- <sup>4</sup> Number of data to be written.
- <sup>5</sup> The data (00<sub>H</sub>, 01<sub>H</sub>, 02<sub>H</sub>, and 03<sub>H</sub>) is moved to the control data area.

9.3.2 FREAD

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

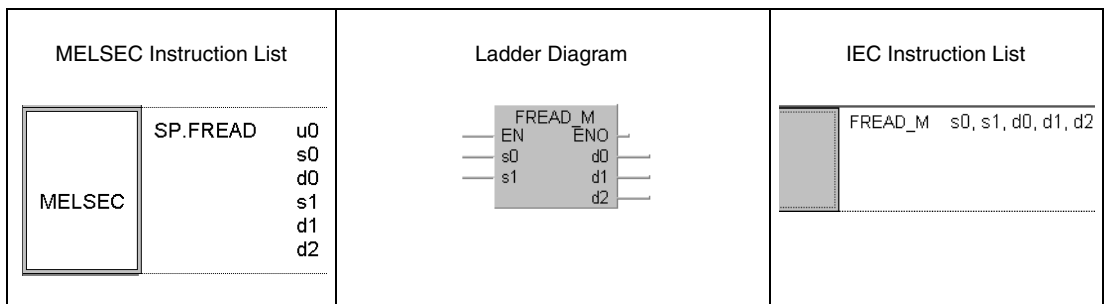
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices  
MELSEC Q

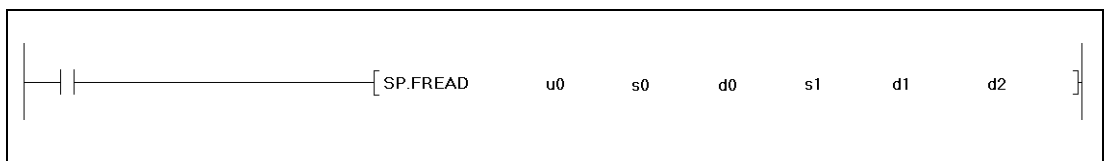
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s0	●	●	●	—	—	—	—	●	—	SM0	11
d0	—	●	●	—	—	—	—	—	—		
s1	—	●	●	—	—	—	—	—	—		
s2	—	●	●	—	—	—	—	—	●		
d1	●*	●*	●*	—	—	—	—	—	—		

\* Local devices and the devices designated for individual programs cannot be used.

GX IEC  
Developer



GX  
Developer



Variables

Set data	Meaning		Setting Range	Set By	Data Type
u0	Dummy		—	—	BIN 16-bit
s0	Drive designation. Only the drive with ATA card can be designated (drive 2). A memory card (ROM) or standard RAM/ROM cannot be designated.		2	User	
d0	Head number of the device storing the control data. The following control data is required.				
	Set data	Item	Meaning/Set Data	Setting Range	Set By
	(d0)	Execution type-	Specifies the execution type: 0000 <sub>H</sub> : Write binary data 0100 <sub>H</sub> : Write data after conversion into CSV format	0000 <sub>H</sub> 0100 <sub>H</sub>	User
	(d0)+1	(Reserved)	Used by system	—	System
	(d0)+2	Number of data to be read	Sets the number of data to be read (in units of words). This number should be designated in the unit of words even when byte is selected in (d0)+7.	1 to 480	User
	(d0)+3	Not used	—	—	—
	(d0)+4 (d0)+5	Location in file	Sets the location in the file to start reading when <b>binary</b> data is selected (d0 = 0000 <sub>H</sub> ). 00000000 <sub>H</sub> : From the beginning of the file 00000001 <sub>H</sub> to FFFFFFFC <sub>H</sub> : From the specified address. The unit for the value is determined by word/byte unit designation. FFFFFFFD <sub>H</sub> : Setting disabled  When data reading after <b>CSV format</b> conversion is selected (d0 = 0100 <sub>H</sub> ): For a CPU whose serial number is „01111“ or earlier in the upper 5 digits, always set the beginning of the file (00000000 <sub>H</sub> ). For a CPU with serial number „01112“ or later set the file position. 00000000 <sub>H</sub> : From the beginning of the file. 00000001 <sub>H</sub> : to FFFFFFFC <sub>H</sub> : From the specified address. FFFFFFFD <sub>H</sub> : Read continues, starting at the previous read position	00000000 <sub>H</sub> to FFFFFFFC <sub>H</sub> FFFFFFFD <sub>H</sub>	User
	(d0)+6	Number of columns	Sets the number of columns for the data to be read. 0: : No column setting. Data is considered to be in a single row. > 0 : Data is considered to be a table with the specified number of columns	0, 1 to 65535	User
(d0)+7	Word/Byte designation	0: Word 1: Byte	0, 1	User	

Variables

Set data	Meaning			Setting Range	Set By	Data Type
s1	Head number of the device storing a file name.					BIN 16-bit
	Set data	Item	Meaning/Set Data	Setting Range	Set By	
	(s1) to (s1)+n	File name	The file name consists of up to 8 characters + period + extension (for example: ABD.BIN). The extension can be omitted. In this case, the period („.“) can also be omitted. When more than 8 characters are used, the extension is ignored regardless of its presence. The Extension „BIN“ or „CSV“ is assigned automatically.	Character-string	User	
d1	Head number of the device storing the data.					BIN 16-bit
	Set data	Item	Meaning/Set Data	Setting Range	Set By	
	(d1)	Reading result (Number of read data)	Contains the number of actually read data. The unit for the value is determined by word/byte unit designation.	0 to 480	System	
(d1)+1 to (d1)+n	Data to be read	Data requested to be read	0000 <sub>H</sub> to FFFF <sub>H</sub>			
d2	Bit device that goes ON after the execution of the FREAD instruction. When an error occurs, (d1)+1 goes ON.					Bit
	Set data	Item	Meaning/Set Data	Setting Range	Set By	
	(d2)	Completion signal	Indicates the completion of the FREAD instruction. ON: Completed OFF: Not completed	—	System	
(d2)+1	Error completion signal	Indicates whether the FWRITE instruction is normally completed or abnormally completed. ON: Error completion OFF: Normal completion	—			

NOTE

The data written in CSV format is expressed as decimal value by the programming software. For example, the character „A“ (41<sub>H</sub>) is written as 65. The available range is from -32768 to 32767.

**Functions    Reading data from a designated file**

**FREAD    Read data**

The FREAD instruction reads a specified number of data from a file at the ATA card. The user can select whether to read data as binary data without any conversion or to convert data from the CSV-format into binary data before reading it.

The completion signal bit device (d2)+0 automatically turns ON after the completion of the FREAD instruction is detected and the END instruction is executed. The bit device turns OFF at the execution of the END instruction in the next scan.

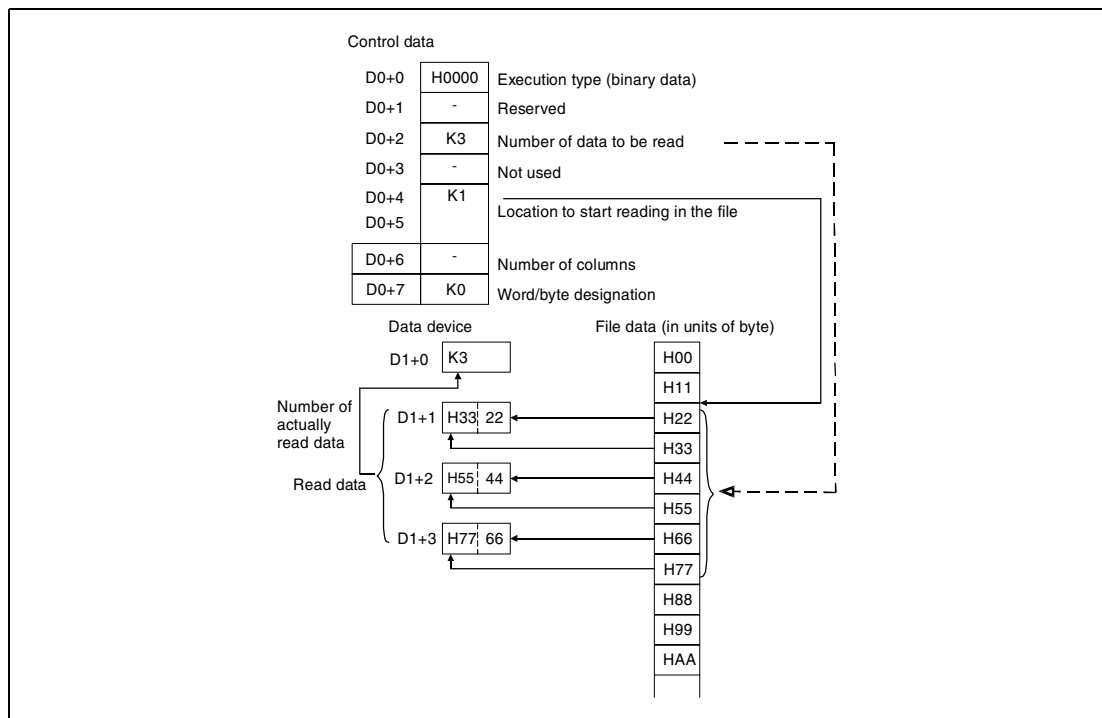
This bit device can be used as the execution completion flag for the FREAD instruction.

When the FREAD instruction is completed abnormally, the error completion device (d2)+1 turns ON/OFF in synchronization with the execution completion flag (d2)+0. This bit device can be used as error completion flag for the FREAD instruction.

SM721 is on during the execution of the FREAD instruction. SM721 is also used by other instruction such as S.FREAD, COMRD and PRC. The FREAD instruction cannot be started while SM721 is ON. If an attempt is made, no processing is performed.

When an error is detected prior to the execution of the instruction (before SM721 goes ON), the execution completion device [(d2)+0], the error completion device [(d2)+1] and SM721 do not turn ON.

The unit for the number of data to be read [(d0)+0] is „word“, regardless of the setting in (d0)+7 (word/byte designation). The following figure illustrates the reading of binary data:



**Reading of binary data:**

If the extension of the object file is omitted, „.BIN“ is added as an extension.  
When the designated file does not exist, an error occurs.

An error occurs if the designated location in the file is larger than the file size. A CPU with the serial number 01111 or earlier (the upper 5 digits) will issue a error code. A CPU bearing the serial number 01112 (or later) will not read any data and will complete the instruction without an error message.

**Reading of data after CSV format conversion**

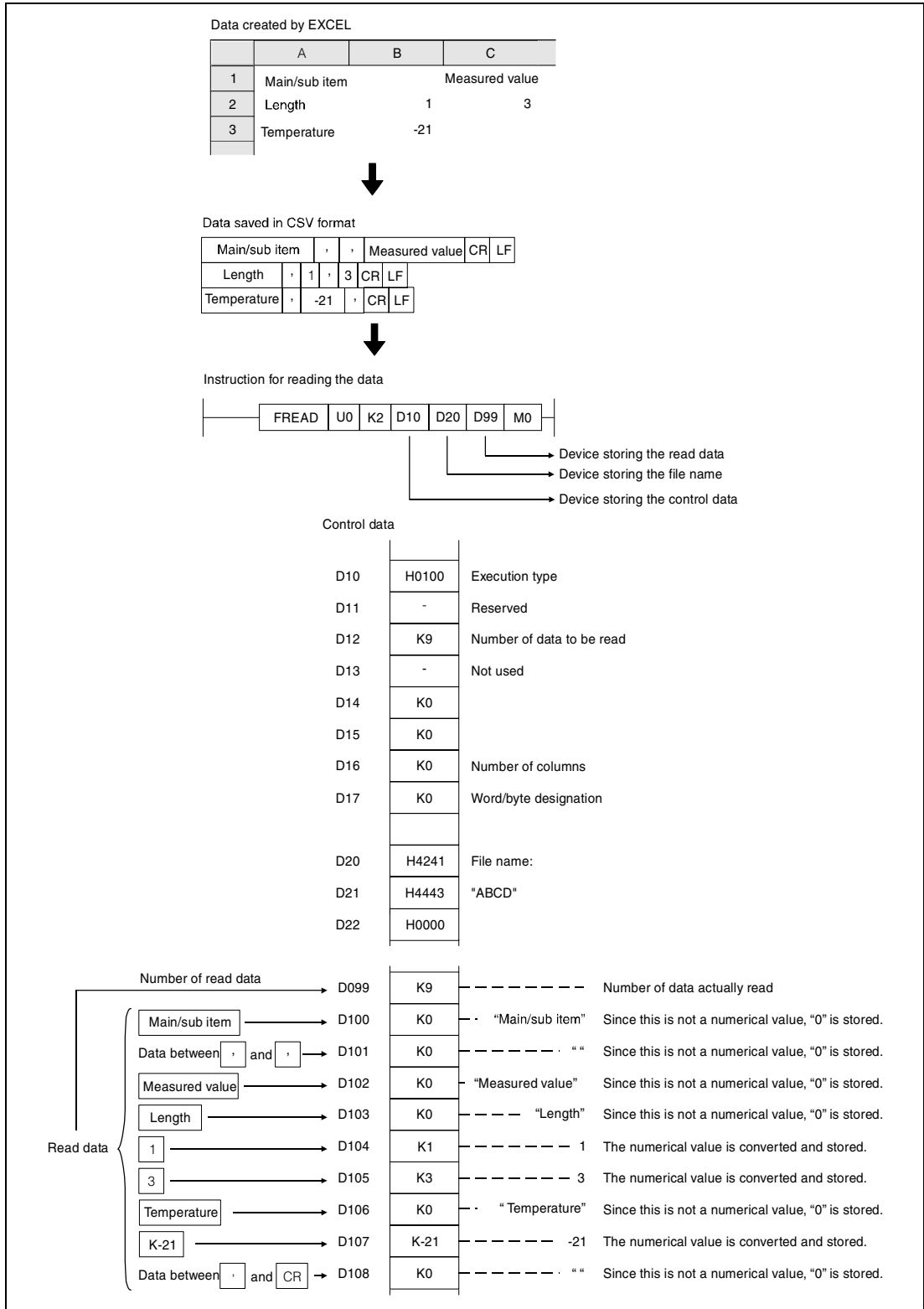
The elements in the CSV-format file (cells for EXCEL) are read row by row. The numerical values and character strings are converted into binary data and stored in the device.

If the extension of the file is omitted, „.CSV“ is added as an extension.  
When the designated file does not exist, an error occurs.

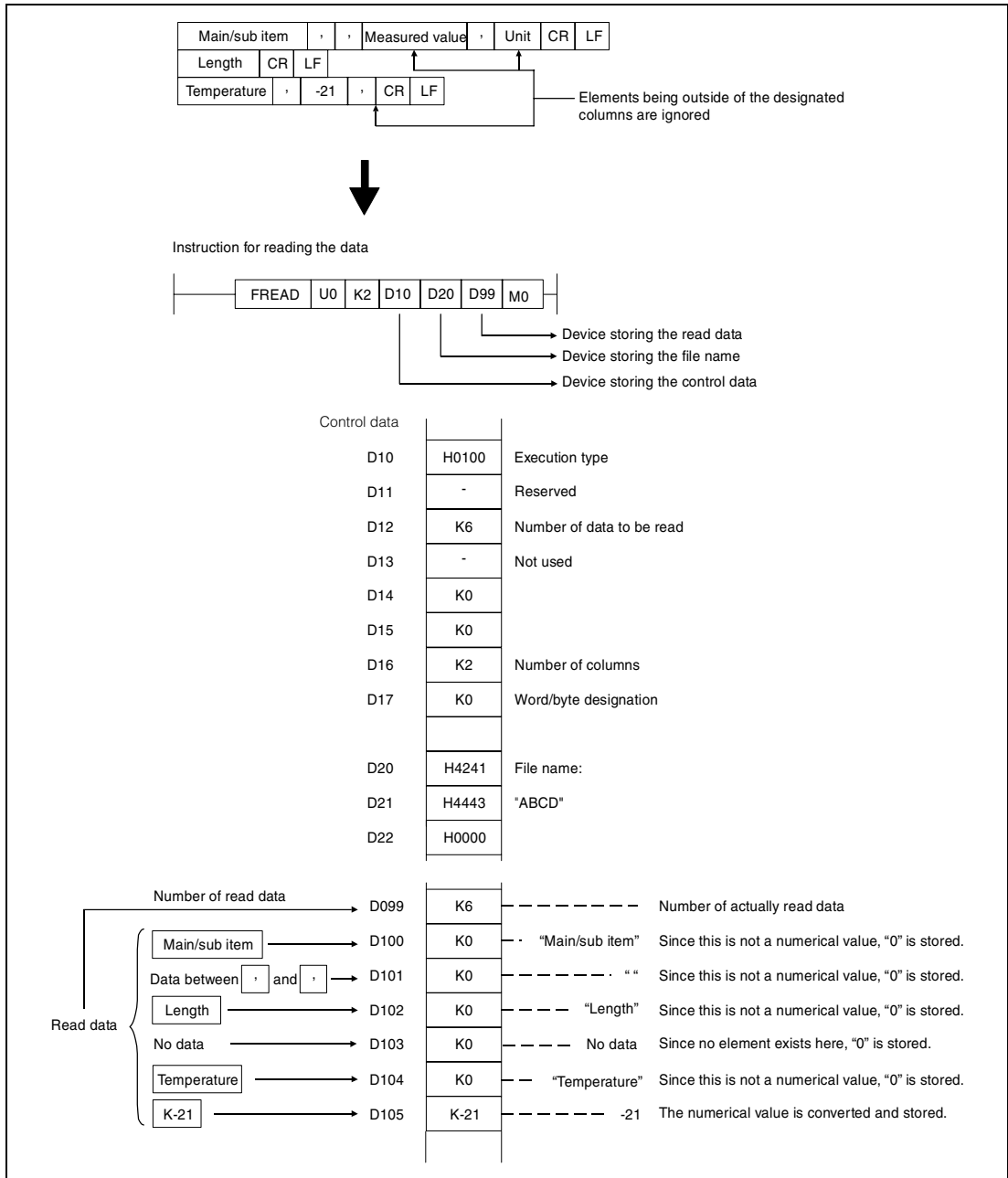
The reading starts at the specified position of the file. The number of elements to read is set in the control data with (d0)+2. When the last data of the file is reached before the specified number of data has been read, a CPU with the serial number 01111 or earlier (the upper 5 digits) will issue a error code. A CPU bearing the serial number 01112 or later will read the data that can be read.

When the specified number of columns is „0“, the data is read by ignoring the rows in a CSV-format file. The figure on the following page shows the handling of data in such a case.

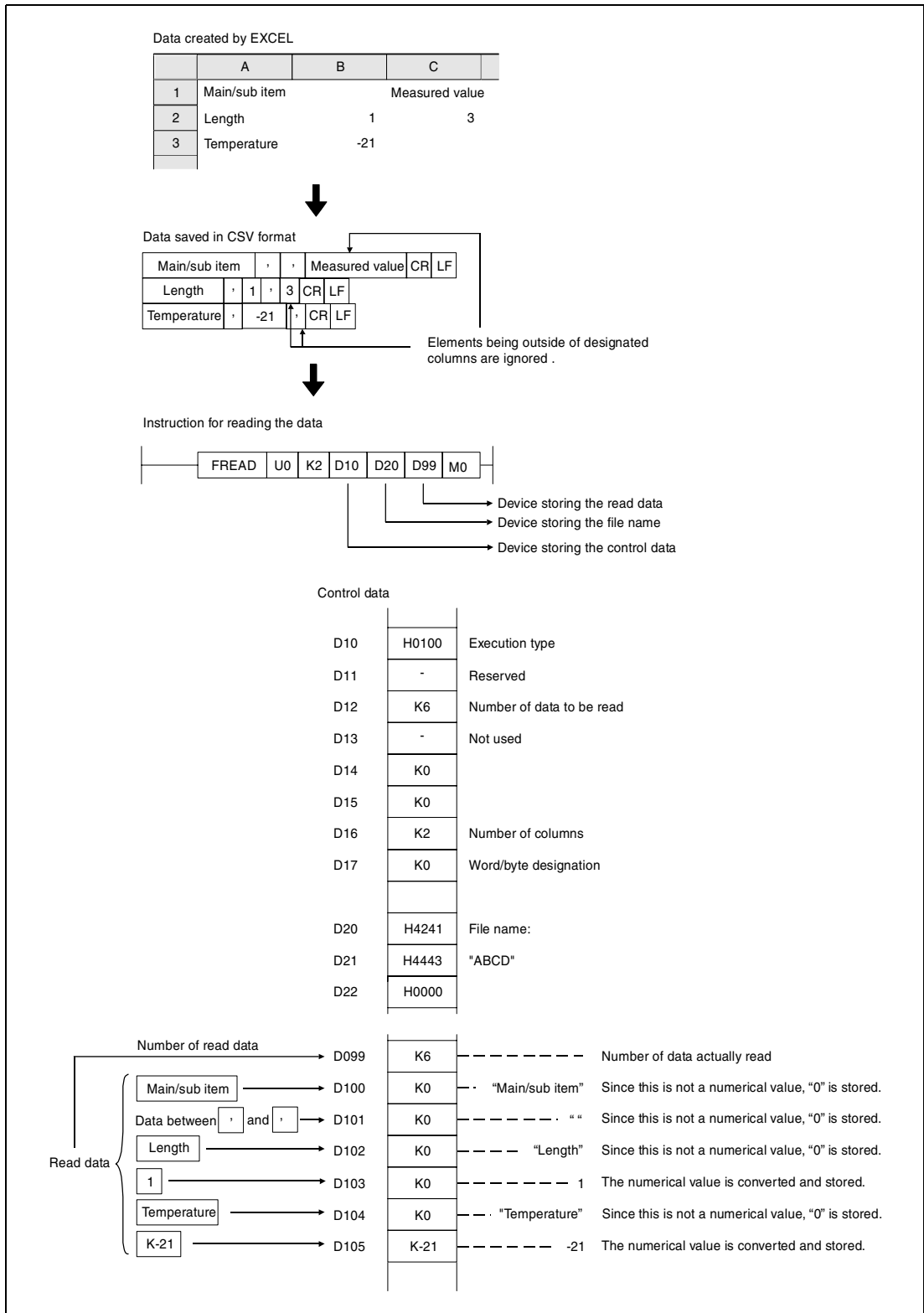




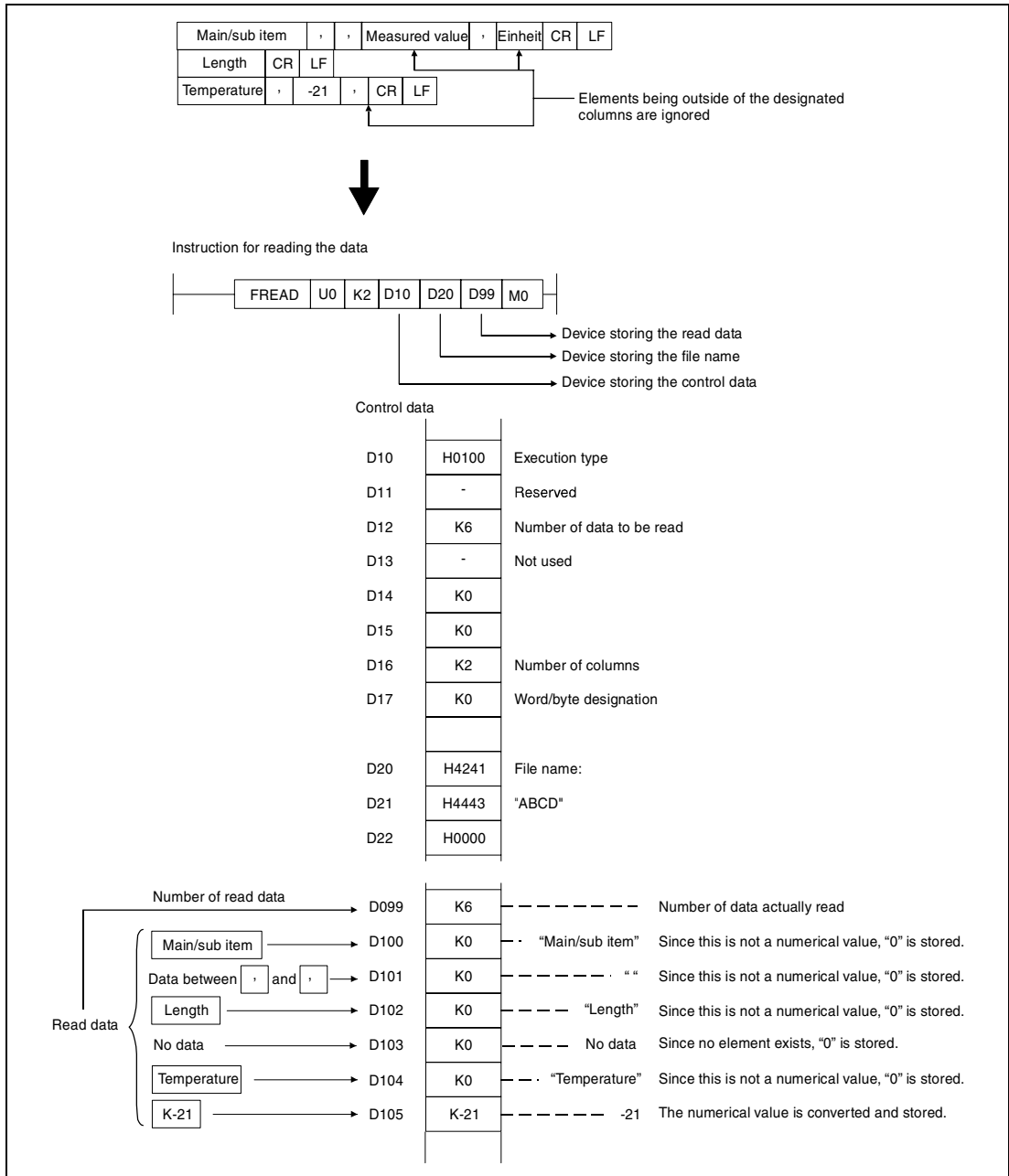
If the number of columns varies in each row, the data is also read by ignoring the rows. (EXCEL does not create such files. This happens when a user modifies a CSV file.)



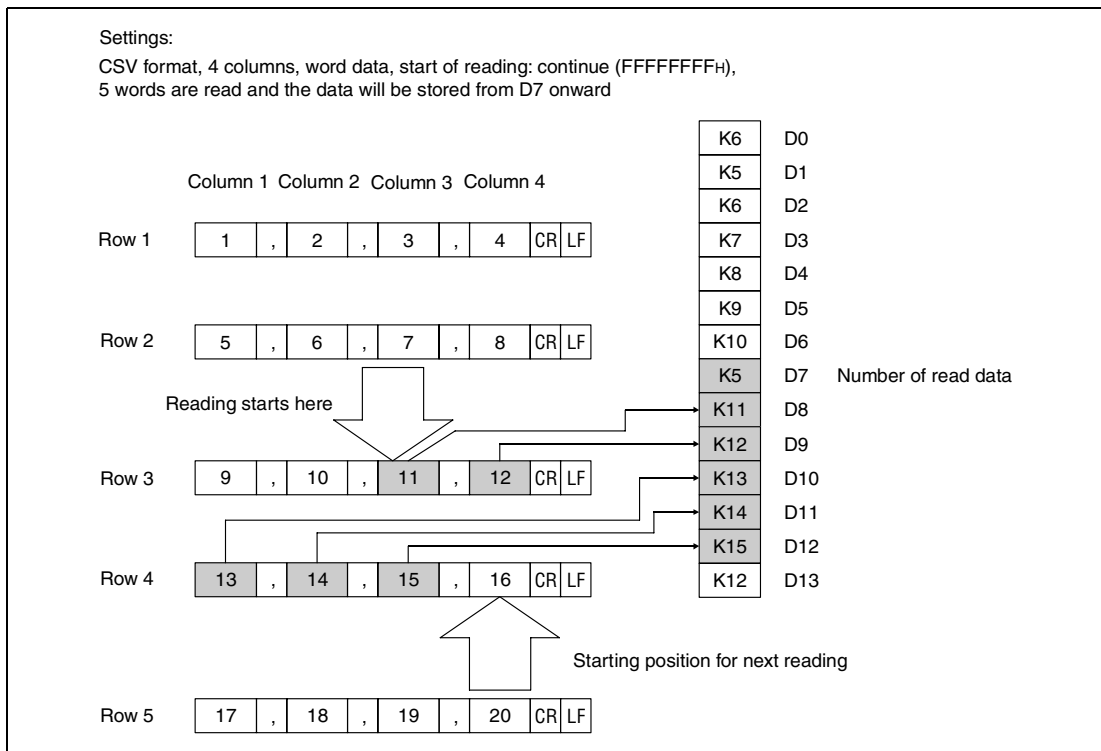
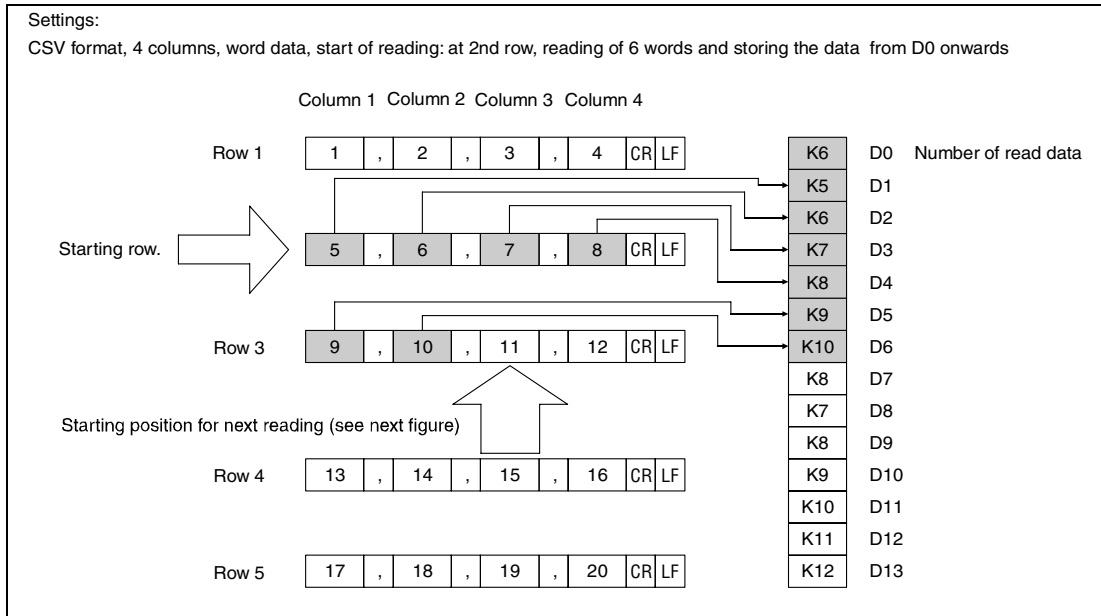
When data is read after CSV format conversion and the designated number of columns is other than „0“, the data is expected to be in a table with the specified number of columns. The elements being outside the specified columns are ignored. The following figure illustrates such a case:



If the number of columns varies in each row, the elements outside of the designated columns are ignored and „0“ is added to the places where elements do not exist.  
 If the number of rows in the file is less than specified by (d0)+2 (Number of data to be read) „0“ is added to the places where rows do not exist.



The following figures are to illustrate the case, when data is read separately several times from the same file (continuation mode) using a CPU bearing the serial number „01112“ or later in the upper 5 digits.



When read is performed in the continuation mode, the settings for data format, number of columns and word/byte designation must not differ from the settings for the previous reading. During reading in the continuation mode the execution of other FREAD or FWRITE instructions must be disabled.

When data is read after CSV format conversion, numerical values are read and converted as follows:

Numerical Values in CSV Format	Wort Device	
	Without Sign	With Sign
-32768	32768	-32768
-1	65535	-1
0	0	0
1	1	1
32767	32767	32767
32768	32768	-32768
65535	65535	-1

Numerical values which are out of range and elements other than numerical values in the object CSV file are converted into „0“.

**NOTE** *Do not execute the FREAD instruction in an interrupt program.*

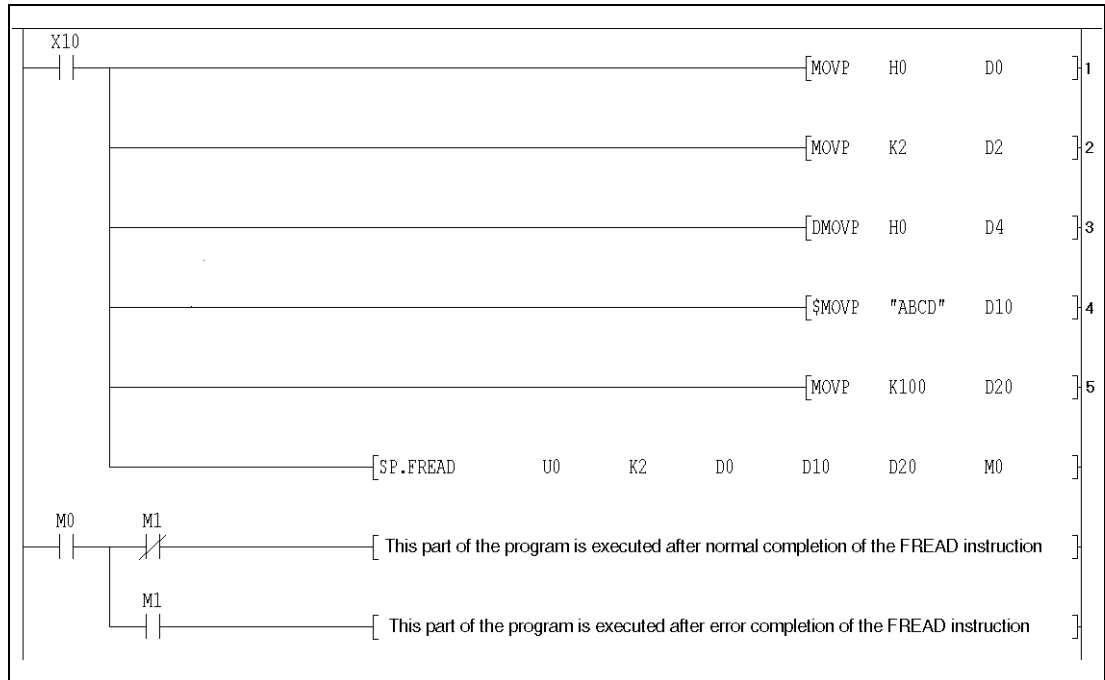
**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The drive specified by s0 contains a medium other than an ATA card (error code 4100).
- Values specified in the areas for control data are out of the setting range (error code 4100).
- The value „number of data to read“ [(d0)+0] is out of the setting range (error code 4101).
- An invalid device is designated (error code 4004).
- The file name specified by s1 does not exist in the designated drive (error code 2410).
- Size of read data exceeds the size of the reading device (error code 4101).
- When binary data is read, the number of data in the file is less than the size designated by the number of data to read [(d0)+2] (error code 4100).

**Program Example 1**      **FREAD**

When X10 is turned ON, four bytes of binary data are read from the beginning of the file „ABCD.BIN“. The file „ABCD.BIN“ is stored at a memory card which is inserted in drive 2. From D0 onward, eight points are reserved for control data. 100 bytes are reserved from D20 for the read data.

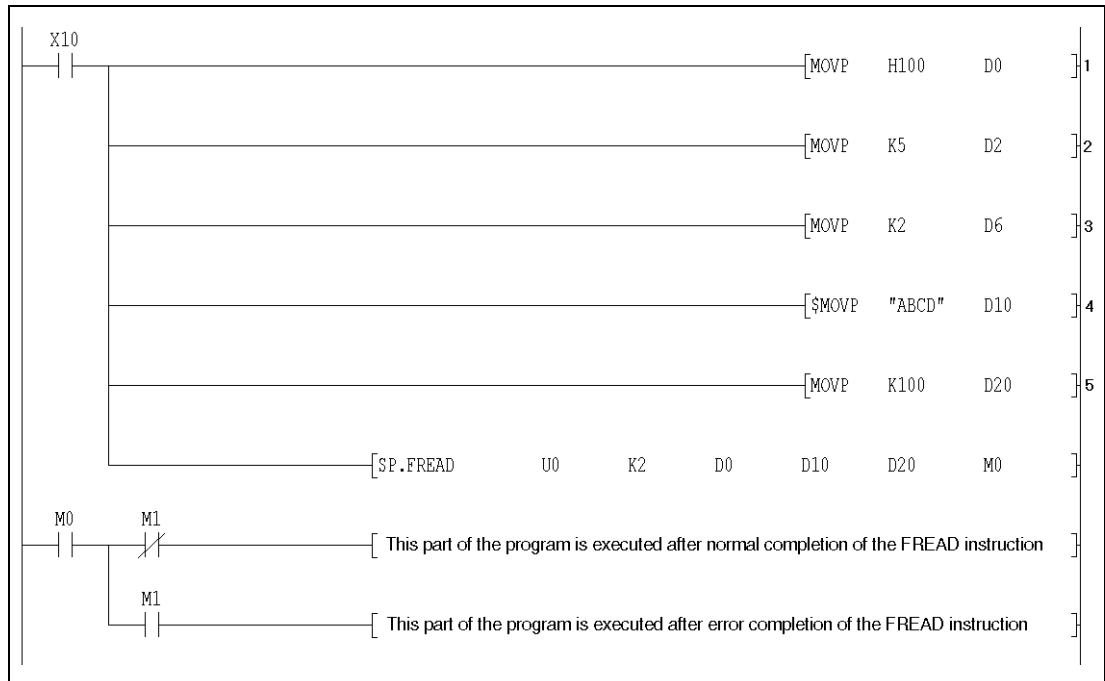


- <sup>1</sup> Setting of the execution type
- <sup>2</sup> Setting of the number of data to read
- <sup>3</sup> Head address in the file (start reading at the beginning of the file)
- <sup>4</sup> Transfer of the file name to the control data
- <sup>5</sup> Setting of the reading device size

**Program Example 2**

FREAD

The following program reads data from the file „ABCD.CSV“, which is stored at the memory card in drive 2 when X10 is turned ON. The contents of the file is two-column table data in CSV format. The file contains numerical values only. From D0 onward, eight points are reserved for control data. For the read data, 100 bytes are reserved from D20.



- 1 Setting of the execution type (CSV format for this example)
- 2 Setting of the number of data to read
- 3 Setting of the number of columns
- 4 Transfer of the file name to the control data
- 5 Setting of the reading device size



## 9.4 Program instructions

### 9.4.1 PLOADP

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

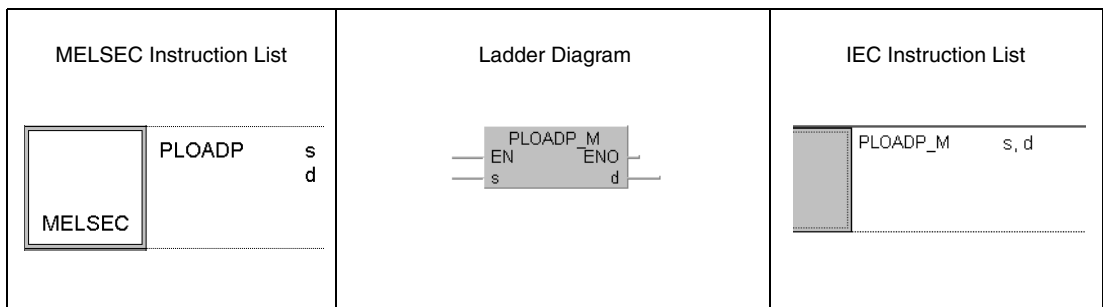
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

**Devices  
MELSEC Q**

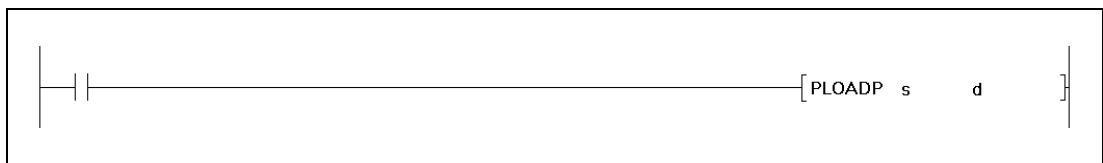
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	●	SM0	3
d	●*	—	—	—	—	—	—	—	—		

\* Local devices cannot be used.

**GX IEC Developer**



**GX Developer**



**Variables**

Set data	Meaning	Data Type
s	Drive number storing the program to be loaded, character string data of the file name, or head number of the device storing the character string data	BIN 16-bit
d	Device turned ON for 1 scan after completion of the instruction	Bit

**NOTE**

*The file system is not supported by the GX IEC Developer.*

**Functions Loading of a program from a memory card****PLOADP Load program**

The PLOADP instruction moves a program which is stored in a memory card or standard memory to the internal memory (drive 0) and places the program in the standby status. The memory card can be inserted in drive 1, 2 or 4. Drive 0 must have continuous free space.

It is unnecessary to designate the extension „.QPG“ to the file name.

The bit device specified by d goes ON during the END processing of the scan where the PLOADP instruction is completed. The bit device goes OFF during the next END processing.

The program instructions PLOADP, PUNLOADP and PSWAPP cannot be used simultaneously. If two or more instructions are executed, the instruction issued later will not be executed. Establish interlocks to avoid such a case.

The lowest program number in the CPU which is vacant is used as the program number of the added program. The program numbers can be checked with the GX Developer by reading the program list. A program number for the added program can be specified by storing a number in SD 720.

The PLOADP instruction cannot be executed during a interrupt program.

To execute the program that was transferred to the program memory with the PLOADP instruction, the PSCAN instruction must be executed.

The PLC file settings of the loaded program are set as follows:

File usage for each program:

All usage of the file register, device initial value, comment, and local device of the loaded program is set at „Follow PLC file setting“.

However, if „Use local device“ is designated in the PLC file setting and programs are loaded, an error occurs every time the number of executed programs exceeds the number of parameter-set programs. To use local devices in the loaded program, register a dummy file in the parameter, delete the dummy file with the PUNLOADP instruction, then load the program with the PLOADP instruction.

I/O refresh setting:

The I/O refresh setting for the loaded program is „Disabled“ for both input and output.

Writing during RUN is not executed during the execution of the PLOADP instruction, but executed after the instruction is completed. Conversely, the PUNLOADP instruction is not executed during the writing during RUN, but executed after the writing during RUN is completed.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The file name does not exist at the drive number specified by s (error code 2410).
- The drive number specified by s is invalid (error code 4100).
- There is not enough memory to load the specified program in drive 0 (error code 2413).
- The number of programs shown below are already registered in the program memory (error code 4101).
- The program number stored in SD720 is already used, or larger than the largest program number shown below (error code 4101).

Type of CPU	Program Memory (Number of files)	Largest Program Number
Q02(H)	28	28
Q06H	60	60
Q12H	124	124
Q25H	124	124

- A program file which has the same name as the program file to be loaded already exists. (error code 2410).
- The file size of the local devices cannot be reserved (error code 2401).

**Program Example**

**PLOADP**

When M0 is ON in the following program, the program „ABCD.QPG“ is transferred from drive 4 to drive 0 and placed in standby status.

Instruction List	Ladder Diagram
<pre> 0 LD      M0 1 PLOADP  "4:ABCD" M10 7 END                     </pre>	<p>The ladder diagram consists of two rungs. The first rung starts with a normally open contact labeled 'M0'. This contact is connected to a coil labeled 'PLOADP "4:ABCD" M10'. The second rung consists of a single coil labeled 'END'.</p>

9.4.2 PUNLOADP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

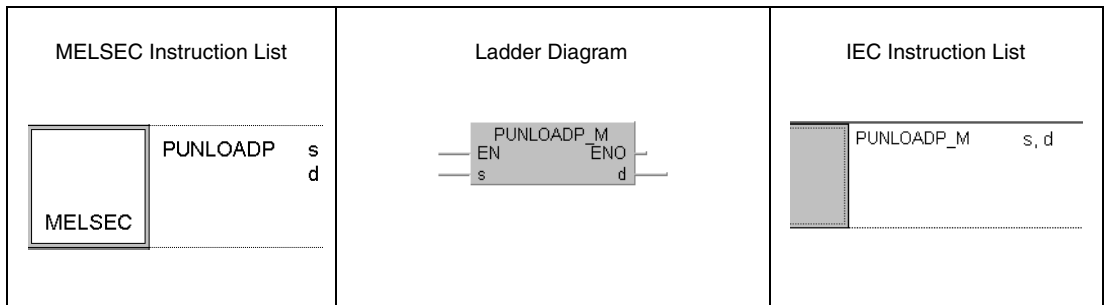
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices  
MELSEC Q

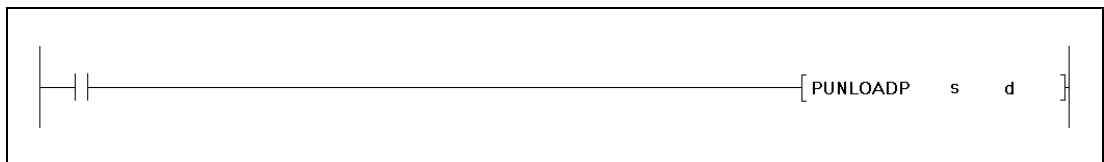
	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	●	SM0	3
d	●*	—	—	—	—	—	—	—	—		

\* Local devices cannot be used.

GX IEC Developer



GX Developer



Variables

Set data	Meaning	Data type
s	Character string data of the program file name to be unloaded, or head number of the device storing the character string data	BIN 16-bit
d	Device turned ON for 1 scan after completion of the instruction	Bit

NOTE

The file system is not supported by the GX IEC Developer.

**Functions Unloading of a program from program memory**

**PUNLOADP Unload program**

The PUNLOADP instruction is used to delete a standby program stored in the program memory (drive 0). The standby program being executed by the PSCAN instruction cannot be deleted.

It is unnecessary to designate the extension „.QPG“ to the file name.

The bit device specified by d goes ON during the END processing of the scan where the PUNLOADP instruction is completed. The bit device goes OFF during the next END processing.

The program instructions PLOADP, PUNLOADP and PSWAPP cannot be used simultaneously. If two or more instructions are executed, the instruction issued later will not be executed. Establish interlocks to avoid such a case.

If the power supply for the CPU is switched OFF and then turned ON again, or the CPU module is reset after the program deletion, „FILE SET ERROR (error code 2400)“ occurs. To solve this problem, delete the name of the deleted program from the program setting of the parameter.

The PUNLOADP instruction cannot be executed during a interrupt program.

The program to be deleted from the program memory with the PUNLOADP instruction should be placed in standby status with the PSTOP instruction before.

Writing during RUN is not executed during the execution of the PUNLOADP instruction, but executed after the instruction is completed. Conversely, the PUNLOADP instruction is not executed during the writing during RUN, but executed after the writing during RUN is completed.

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

- The file name specified by s does not exist (error code 2410).
- The program designated by s is not in standby status or is being executed (error code 4101).
- The program specified by s is the only one in the program memory (error code 4101).

**Program Example**

**PUNLOADP**

The following program deletes the program „ABCD.QPG“ stored in drive 0 from the memory when M0 turns from OFF to ON.

Instruction List	Ladder Diagram
<pre> 0 LD      M0 1 PUNLOADP "ABCD" M10 6 END                     </pre>	

9.4.3 PSWAPP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

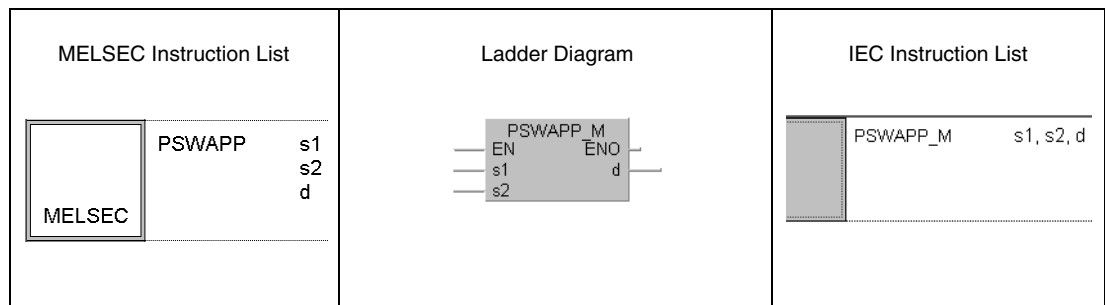
<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

Devices  
MELSEC Q

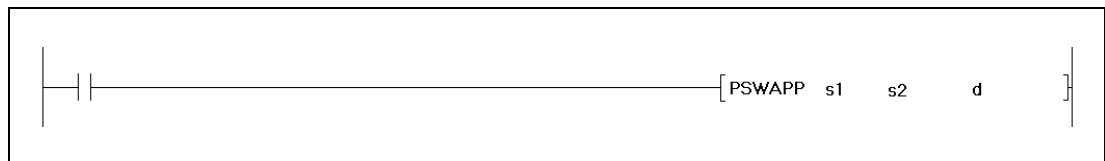
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	●	SM0	3
s2	—	●	●	—	—	—	—	—	●		
d	●*	—	—	—	—	—	—	—	—		

\* Local devices cannot be used.

GX IEC Developer



GX Developer



Variables

Set data	Meaning	Data Type
s1	Character string data of the program file name to be unloaded, or head number of the device storing the character string data.	BIN 16-bit
s2	Drive number storing the program to be loaded, character string data of the file name, or head number of the device storing the character string data	BIN 16-bit
d	Device turned ON for 1 scan after completion of the instruction	Bit

NOTE

The file system is not supported by the GX IEC Developer.

**Functions Unloading of a program from program memory and loading of a program****PSWAPP Unload program and load program**

The PSWAPP instruction deletes (unloads) a standby program from the program memory (drive 0). The program to be deleted is specified by s1. The standby program being executed by the PSCAN instruction cannot be deleted. After the deletion, a program stored in drive 1, 2, or 4 is transferred to the program memory and placed in standby status. This program is specified by s2. The program memory drive 0 must have continuous free space before loading the program.

It is unnecessary to designate the extension „.QPG“ to the file name.

The bit device specified by d goes ON during the END processing of the scan where the PSWAPP instruction is completed. The bit device goes OFF during the next END processing.

The program instructions PLOADP, PUNLOADP and PSWAPP cannot be used simultaneously. If two or more instructions are executed, the instruction issued later will not be executed. Establish interlocks to avoid such a case.

The program number of the deleted program is used for the loaded program.

If the power supply for the CPU is switched OFF and then turned ON again, or the CPU module is reset after the program swap, „FILE SET ERROR (error code 2400)“ occurs. To solve this problem, change the name of the deleted program in the program setting of the parameter to the name of the swapped program.

The PSWAPP instruction cannot be executed during an interrupt program.

The PLC file settings of the loaded program are set as follows:

- All usage of the file register, device initial value, comment, and local device of the swapped program is set to „Follow PLC file setting“.
- The I/O refresh setting for the swapped program is „Disabled“ for both input and output.

Writing during RUN is not executed during the execution of the PSWAPP instruction, but executed after the instruction is completed. Conversely, the PSWAPP instruction is not executed during the writing during RUN, but executed after the writing during RUN is completed.

**Operation Errors**

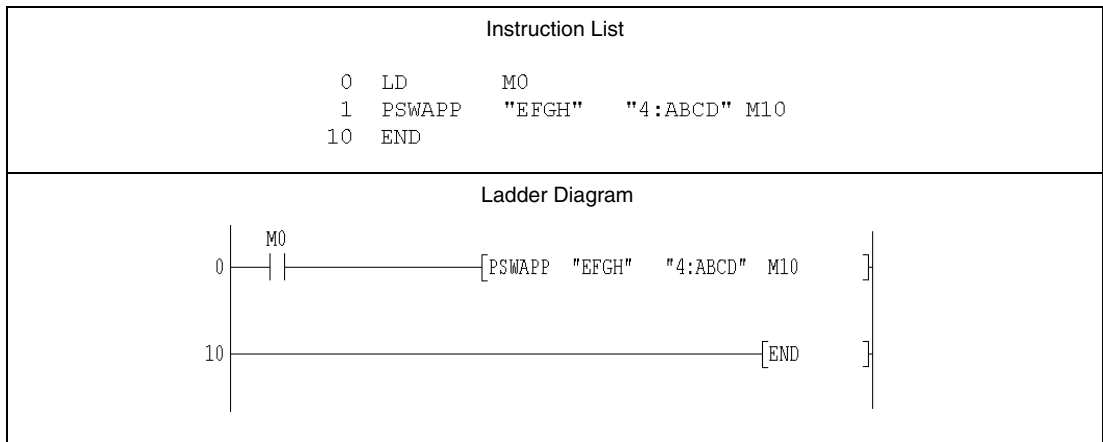
In the following cases an operation error occurs and the error flag is set:

- The drive number or the file specified by s1 or s2 does not exist (errorcode 2410).
- The drive number specified by s2 is invalid (errorcode 4100).
- There is not enough capacity in the program memory (drive 0) to load the specified program (errorcode 2413).
- The program designated by s1 is not in standby status or is being executed (error code 4101).

**Program Example**

PSWAPP

When M0 turns from OFF to ON in the following program example, the program „EFGH.QPG“ is deleted from the program memory. Then the program „ABCD.QPG“ is loaded from drive 4, stored in the program memory, and placed in standby status.





## 9.5 Data transfer instructions

### 9.5.1 RBMOV, RBMOVP

**CPU**

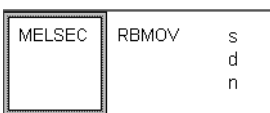
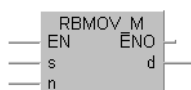
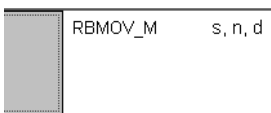
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

<sup>1</sup> Not available for Q00JCPU, Q00CPU and Q01CPU

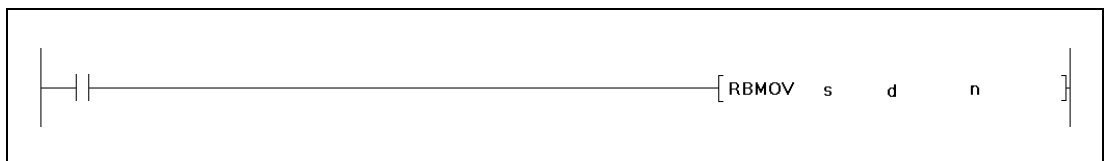
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constant K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	●	●	●	●	●	—	—	—	SM0	4	
d	●	●	●	●	●	—	—	—			
n	●	●	●	●	●	●	●	—			

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	---	---

**GX  
Developer**



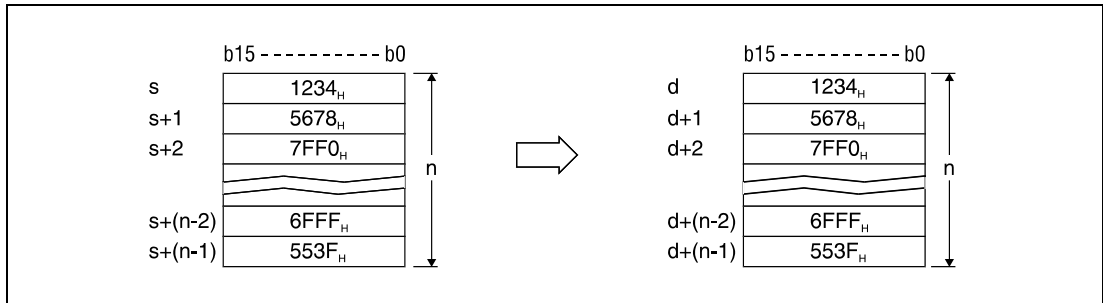
**Variables**

Set data	Meaning	Data type
s	Head number of the device storing the data to be transferred	BIN 16-bit
d	Head number of the destination device	
n	Number of data to be transferred	

**Functions High-speed block transfer of file register**

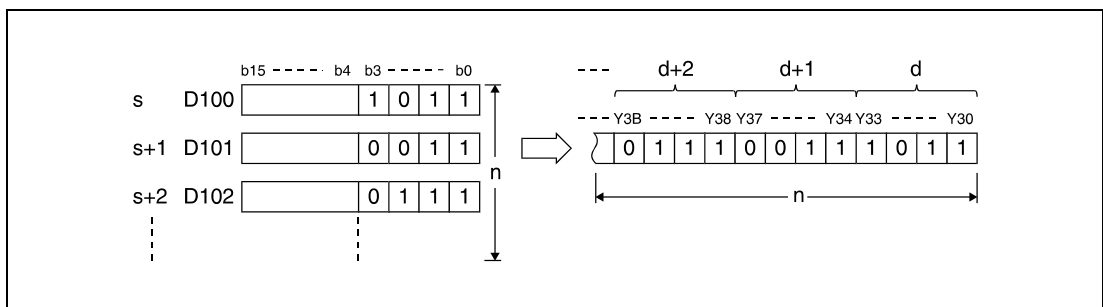
**RBMOV/RBMOV P Block transfer**

The RBMOV instruction batch transfers „n“ points of 16-bit data starting from the device specified by s to the area of „n“ points starting from the device specified by d.



The transfer is possible even if there is an overlap between the source and destination devices. For the transmission to the smaller devices, the data is transferred from s. For the transmission to the larger device number, the data is transferred from s+(n-1).

If s is a word device and d is a bit device, the object for the word device will be the number of bits designated by the bit device digit designation. For example, when „K1Y30“ is specified by d, the lower four bits of the word device specified by s are the object.



If bit devices are specified by s and d, the number of digits must be the same for s and d.

**NOTE**

The RBMOV and the RBMOVP instructions are useful to batch transfer a large quantity of file register data with a high performance System Q CPU. With a Q02CPU, this instruction is similar to the BMOV instruction. The comparison of processing speed between RBMOV and BMOV instructions is as follows:

Q02HCPU, Q06HCPU, Q12HCPU, Q25HCPU				
Object memory	RBMOV		BMOV	
	Time (μs) to transfer		Time (μs) to transfer	
	100 words	1000 words	100 words	1000 words
SRAM	56,30	367,77	44,37	393,14
Built-in RAM	44,37	393,14		
Flash ROM	29	308		

Q02CPU				
Object memory	RBMOV-Anweisung		BMOV-Anweisung	
	Time (μs) to transfer		Time (μs) to transfer	
	100 words	1000 words	100 words	1000 words
SRAM	115,89	579,47	115,89	535,23
Built-in RAM				
Flash ROM				

**Operation Errors**

In the following cases an operation error occurs and the error flag is set:

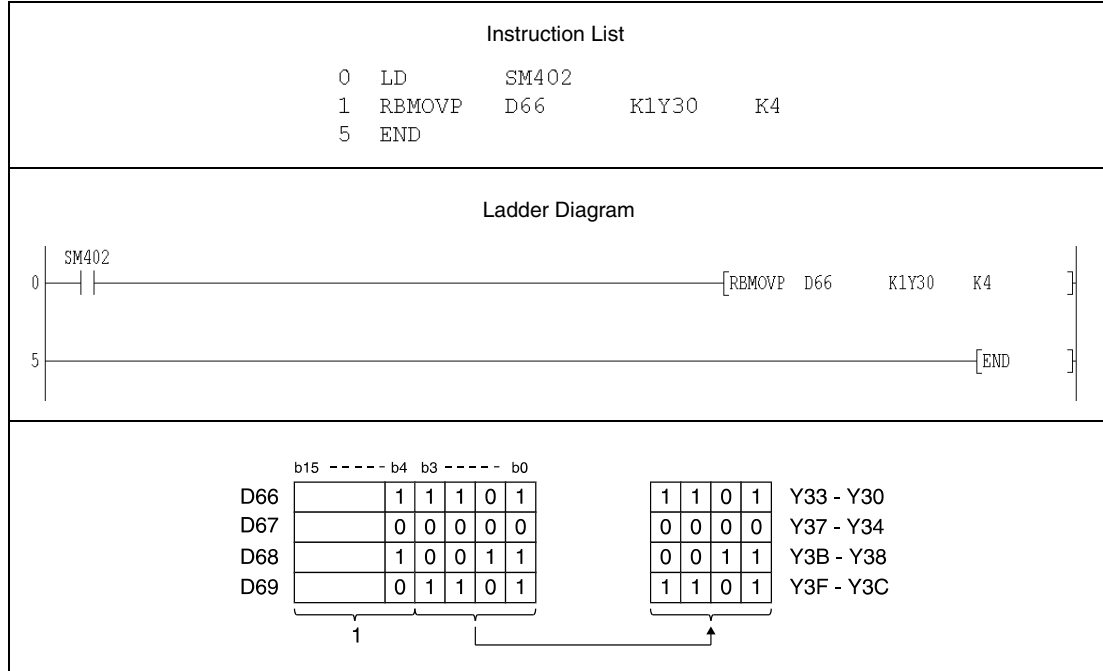
- The device range of „n“ points starting from s or d exceeds the available device (errorcode 4101).
- The file register is not designated for both s and d (errorcode 4101).

**Program Example 1**

**RBMOV P**

The following program transfers the lower four bits (b0 through b3) of data in D66 through D69 to the outputs Y30 through Y3F with the rising edge of SM402. The number of data (4 blocks) is specified by n.

The bit patterns show the structure of bits before and after the transfer.

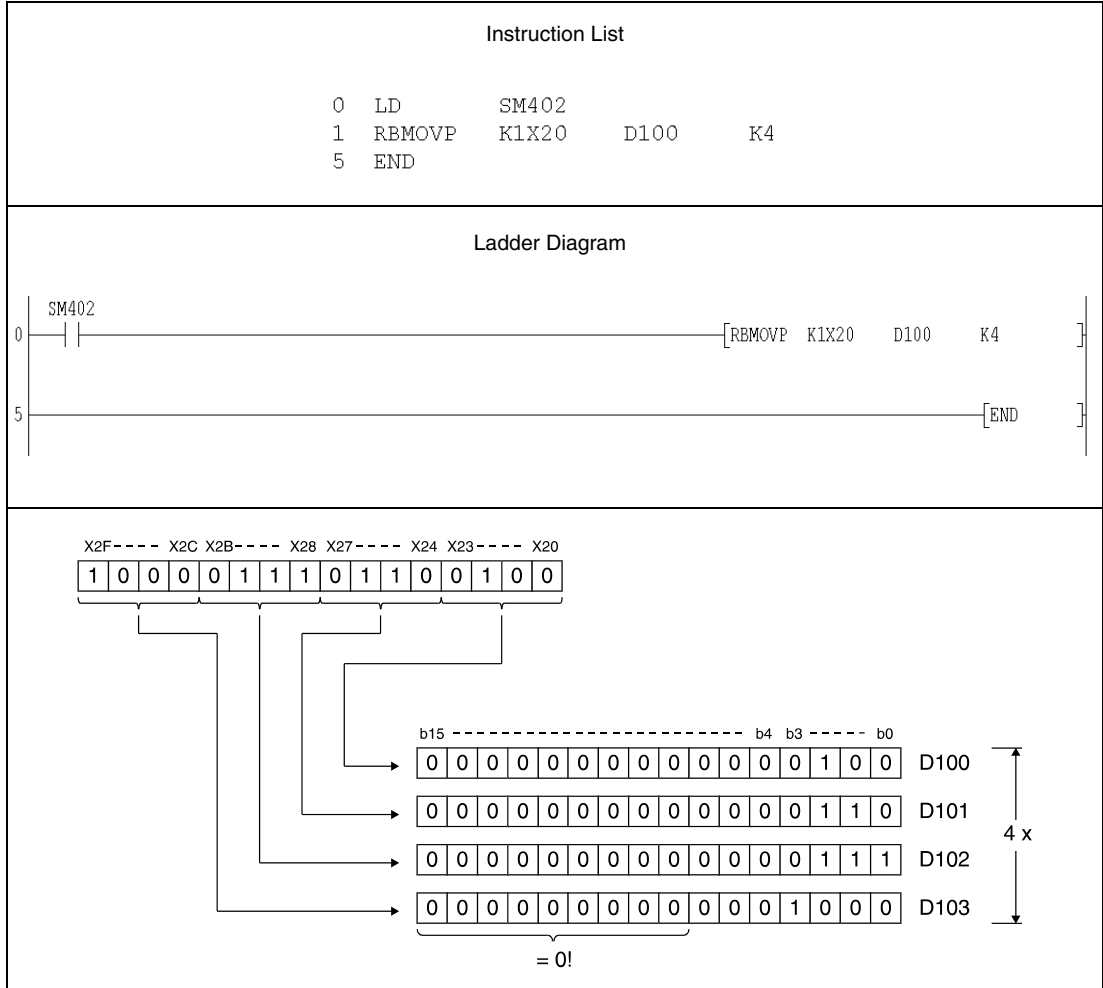


<sup>1</sup> These bits are ignored.

**Program Example 2** RBMOVP

With leading edge from SM402, the following program transfers data at X20 through X2F to D100 through 103. The number of blocks (4) to be transferred is determined by the constant K4.

The bit patterns show the structure of bits before and after the transfer.



## 9.6 Instructions for use in a Multi-CPU System

### 9.6.1 S.TO, SP.TO

**CPU**

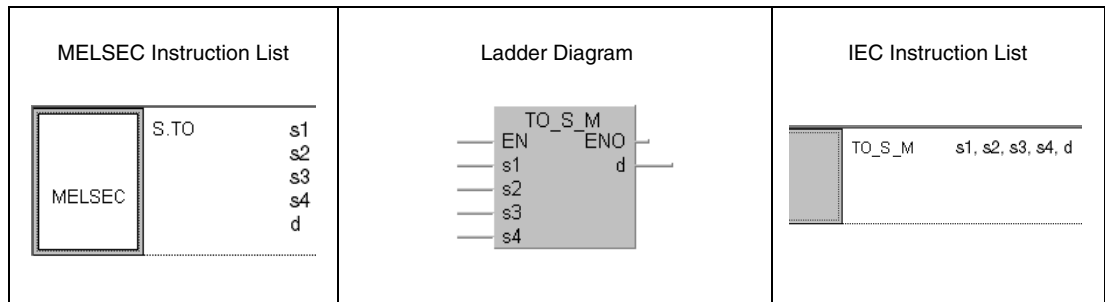
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

<sup>1</sup> For Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU and Q25HCPU from function version B or later only.

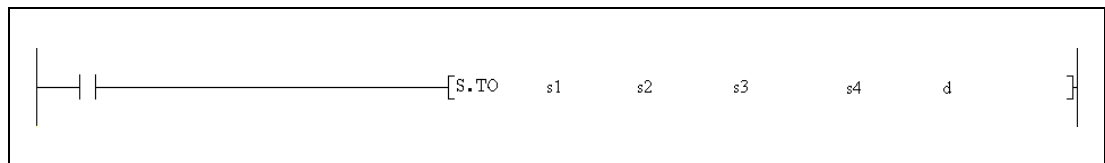
**Devices  
MELSEC Q**

	Usable Devices								Error Flag	Steps	
	Internal Devices (System, User)		File Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)			Other
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—	SM0	5
s2	—	●	●	—	—	—	—	●	—		
s3	—	●	●	—	—	—	—	—	—		
s4	—	●	●	—	—	—	—	●	—		
d	●	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

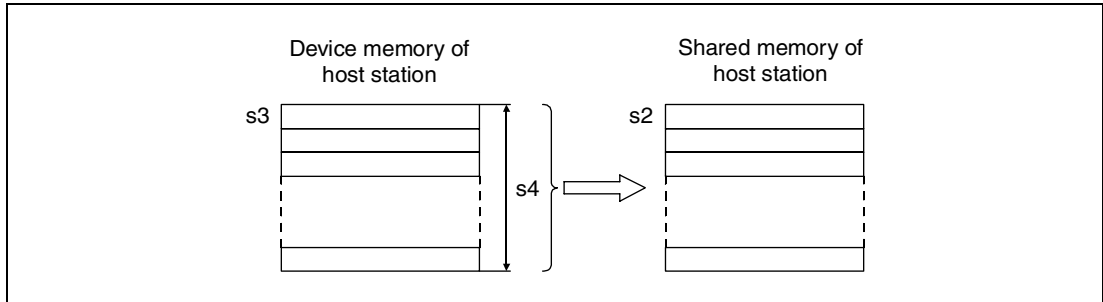
Set Data	Meaning	Data Type
s1	Head I/O number of the CPU which executes the S.TO instruction	BIN16-bit
s2	First number of CPU shared memory address area to be written to (800 <sub>H</sub> to 0FFF <sub>H</sub> ).	
s3	First number of device area storing data to be written.	
s4	Number of data to be written (1 to 256)	
d	Bit device which is turned ON for one scan after the instruction is executed	Bit

**Functions Writing data to the CPU shared memory**

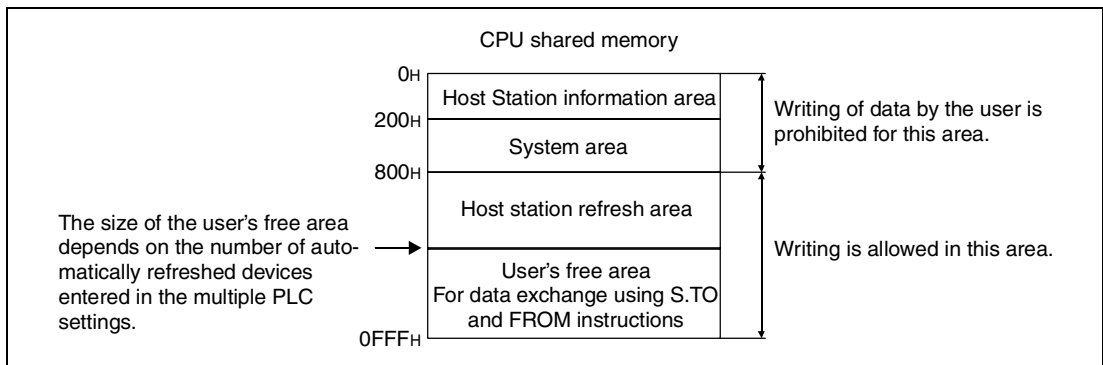
**S.TO/SP.TO Write data**

The S.TO instruction writes data to the user's area in the shared memory of the CPU which is executing the S.TO instruction (host station). The destination address in the shared memory is entered in s2. The data is taken from a device area in the same CPU, starting from the number specified in s3. The number of data words is specified in s4.

The S.TO instruction cannot be used for writing data directly to another CPU in a multi-CPU system.



The CPU shared memory is used for data exchange with other CPUs in a multi-CPU system. The automatic refresh area begins at the address 800H, followed by the user's free area.



The head I/O number of the CPU is determined by the slot in which the CPU module is loaded. Only the first 3 digits of the head I/O number are entered in s1.

Slot of the base unit	CPU	0	1	2
Number of the CPU in multi-CPU system	1	2	3	4
Head I/O number	3E00	3E10	3E20	3E30
Contents of s1	3E0	3E1	3E2	3E3

When the number of write points is entered in s4 as „0“, Processing of the instruction is not performed and the completion device, specified in d, does not turn on, either.

**NOTE**

*Only one S.TO instruction may be executed in one scan by each CPU. However, automatic handshaking makes sure that only the instruction called first will be processed, if two or more S.TO instructions are enabled simultaneously.*

**Operation Errors**

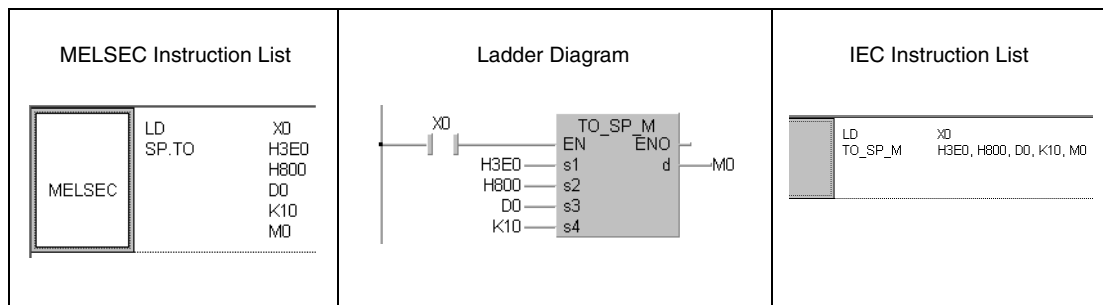
In the following cases an operation error occurs, the error flag is set, and the corresponding error code is stored in SD0:

- The number of write points specified in s4 is other than 0 to 256 (error code 4101).
- The beginning of the CPU shared memory specified in s2 is larger than the CPU shared memory address range (error code 4101).
- The beginning of the CPU shared memory specified in s2 plus the number of write points specified in s4 exceeds the CPU shared memory address range (error code 4101).
- The first device number (s3) where the data to be written is stored plus the number of write points specified in s4 exceeds the device range (error code 4101).
- The value stored in s1 is not the head I/O-number of the CPU performing the S.TO instruction (error code 2107).
- The number stored in s1 is other than a correct head I/O number (3E0<sub>H</sub>, 3E1<sub>H</sub>, 3E2<sub>H</sub> or 3E3<sub>H</sub>)(error code 4100).
- The specified instruction is improper (error code 4002).
- The specified number of devices is wrong (error code 4003).
- An unusable device was specified (error code 4002).

**Program Example**

SP.TO

The data stored in CPU1 in the data registers D0 to D9 is written into the shared memory of the same CPU, beginning at address Adresse 800<sub>H</sub> when X0 turns ON.





**9.6.2 FROM, FROMP**

**CPU**

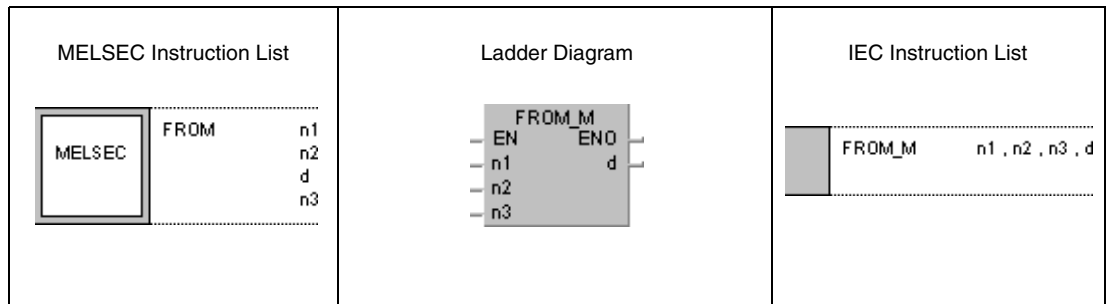
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					● <sup>1</sup>

<sup>1</sup> For Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU and Q25HCPU from function version B or later only.

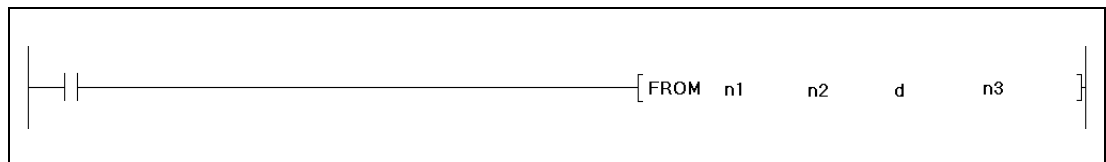
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Schritte
	Internal Devices (System, User)		File Register	MELSECNET/10 Direkt J□□		Special Function Module U□G□	Index Register Zn	Constant K, H (16#)	Andere U		
	Bit	Word		Bit	Word						
n1	—	●	●	●	●	●	●	●	●	SM0	5
n2	—	●	●	●	●	●	●	—	—		
d	—	●	●	—	—	—	—	—	—		
n3	—	●	●	●	●	●	●	—	—		

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data type
n1	Head I/O adress of the CPU which stores the data to be read	BIN 16 bit
n2	First address of data to be read in CPU shared memory (800 <sub>H</sub> to 0FFF <sub>H</sub> ).	
d	First number of memory address area where the read data will be stored	
n3	Number of data words to read (1 to 6144)	

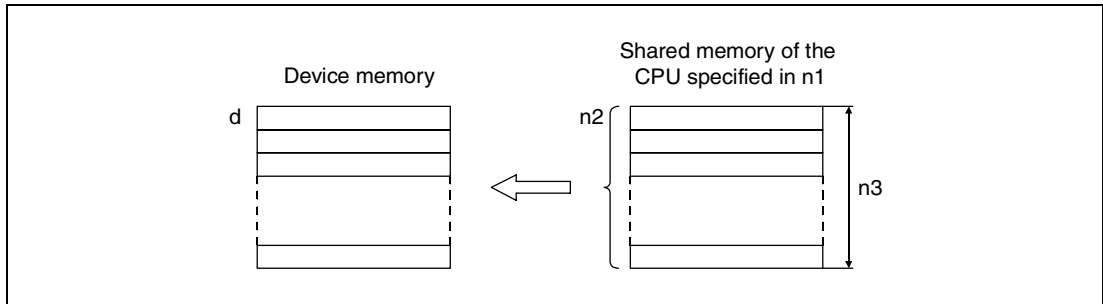
**NOTE**

See chapter 7.8.1 for details of using the FROM instruction for reading data from the buffer memory of special function modules.

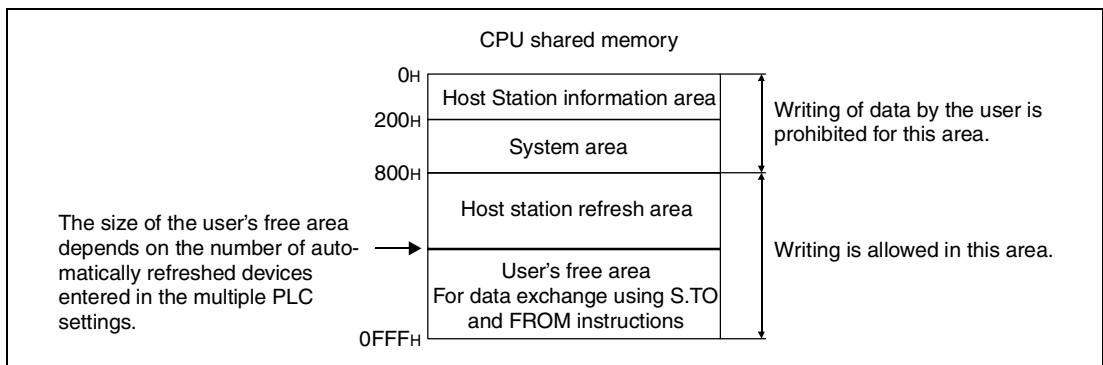
**Functions    Reading from shared memory of another CPU**

**FROM/FROMP    Read word data**

In a multi-CPU system the FROM instruction is used to read word data from the user's free area of the shared memory of another CPU. The head address of this CPU is specified in n1. Enter the number of words to be read in n3. The starting address in the shared memory of the other CPU is specified in n2. The data will be stored in the CPU which executes the FROM instruction starting from the device specified in d.



The CPU shared memory is used for data exchange with other CPUs in a multi-CPU system. The automatic refresh area begins at the address 800H, followed by the user's free area.



The head I/O number of the CPU is determined by the slot in which the CPU module is loaded. Only the first 3 digits of the head I/O number are entered in n1.

Slot of the base unit	CPU	0	1	2
Number of the CPU in multi-CPU system	1	2	3	4
Head I/O number	3E00	3E10	3E20	3E30
Contents of n1	3E0	3E1	3E2	3E3

The special relay SM390 turn ON after reading of the data. SM390 turn not ON if the CPU specified in n1 was in reset status. No error does occur in this case.

Processing of the instruction is not performed when the number of read data is entered in n3 as „0“.

**Operation Errors**

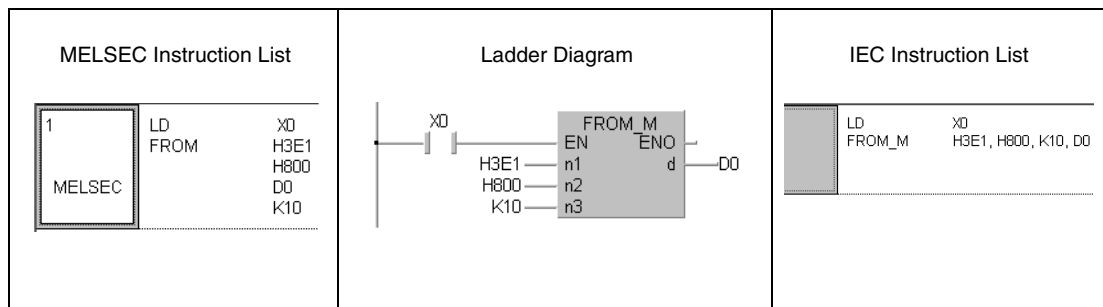
In the following cases an operation error occurs, the error flag is set, and the corresponding error code is stored in SD0:

- The beginning of the CPU shared memory address (n2) from where read will be performed is greater than the CPU shared memory range. (error code 4101).
- The in n2 specified beginning of the CPU shared memory plus the number of read points (n3) exceeds the CPU shared memory range (error code 4101).
- The read data storage device number (d) plus the number of read points (n3) is greater than the specified device range. (error code 4101).
- The head I/O number specified in n1 is the head I/O number of the CPU performing the FROM instruction (error code 2114).
- No CPU module exist in the position specified with the head I/O number in n1 (error code 2110).

**Program Example**

**FROM**

When X0 is set, 10 datawords are read from the shared memory of CPU 2, starting from adress 800<sub>H</sub>. The data is stored in the data registers D0 to D9 of the CPU processing the FROM instruction.





---

# 10 Instructions for Q4ARCPU

Two Q4ARCPU modules can form a redundant PLC system in which one CPU takes over from the other CPU if that module fails. By doing so the seamless continuation of control is possible. Typical applications for a redundant PLC are e.g. power stations, the chemical industry or the water supply of communities.

The following instructions are available for a Q4ARCPU only.

Function	MELSEC Instruction in MELSE Editor	MELSEC Instruction in IEC Editor
Setting of start up mode of the CPU	S.STMODE	STMODE_S_M
Setting of mode for switching of the CPUs	S.CGMODE	CGMODE_S_M
Data transfer to the standby system	S.TRUCK	TRUCK_S_M
Transfer of batch data in and out of the buffer memory of special function modules	S.SPREF	SPRE_S_M

# 10.1 Mode setting instructions

## 10.1.1 STMODE

### CPU

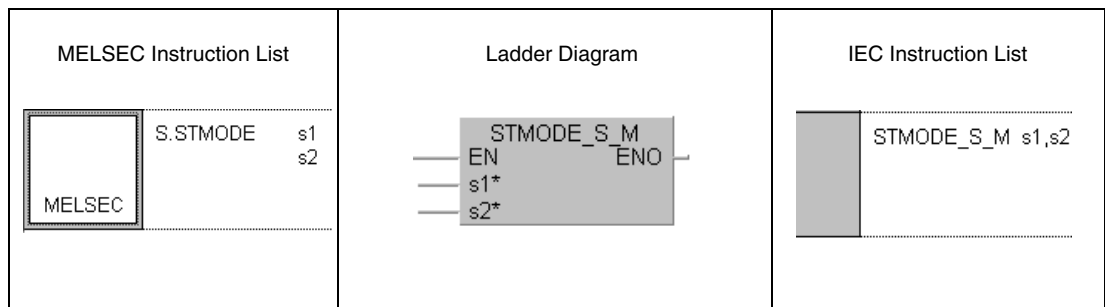
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				● <sup>1</sup>	

<sup>1</sup> For Q4AR only

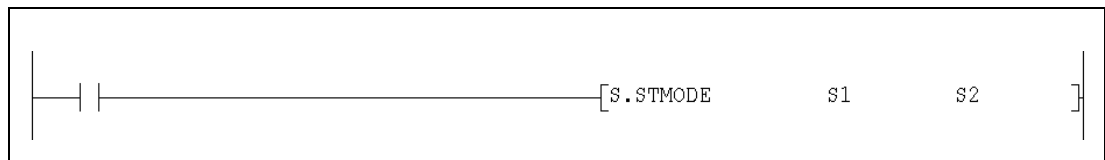
### Devices MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	—	—	—	—	—	—	●	—	—	
s2	—	—	—	—	—	—	—	●	—		

### GX IEC Developer



### GX Developer



### Variables

Set Data	Meaning	Data Type
s1	Setting for the start up mode of the CPU (0 = Initial start mode, 1 = Hot start mode)	BIN 16-bit
s2	Maximum power out time. After this time a hot start will be performed.	BIN 16-bit

**Functions      Operation mode setting for CPU start up**

**STMODE              Operation mode setting**

The contents of s1 selects whether the CPU devices are cleared (Initial start mode) or not (Hot start mode) when the power supply of the PLC is switched on.

When specifying the hot start mode, an automatic data clear and restart can be done when a temporary power outage of a specified time occurs. In this case, specify the switch power out time in s2.

This instruction is executed when the power supply is turned on. For this reason, there is no problem even if the instruction point is turned off. The instruction point will become a dummy point. NOP processing will be conducted when the instruction point is turned on during program execution.

One of these instructions should be created in each system. If there are multiple program files, this instruction is only needed in one file. If more than one of these instructions exists then operation cannot be guaranteed.

The contents of s1 can either be 0 or 1:

0: Initial start mode (Clears devices outside the latch range)

1: Hot start mode (The devices are not cleared but index registers and signal flow (Operation results) are cleared. In addition, the special relay SM and special register SD are preset.

The time in s2 is specified in seconds (0 bis 65535). When specifying 0, the initial start mode cannot be executed. If a number that exceeds 32767 is set, then please do it using a hexadecimal number.

**Operation Error**

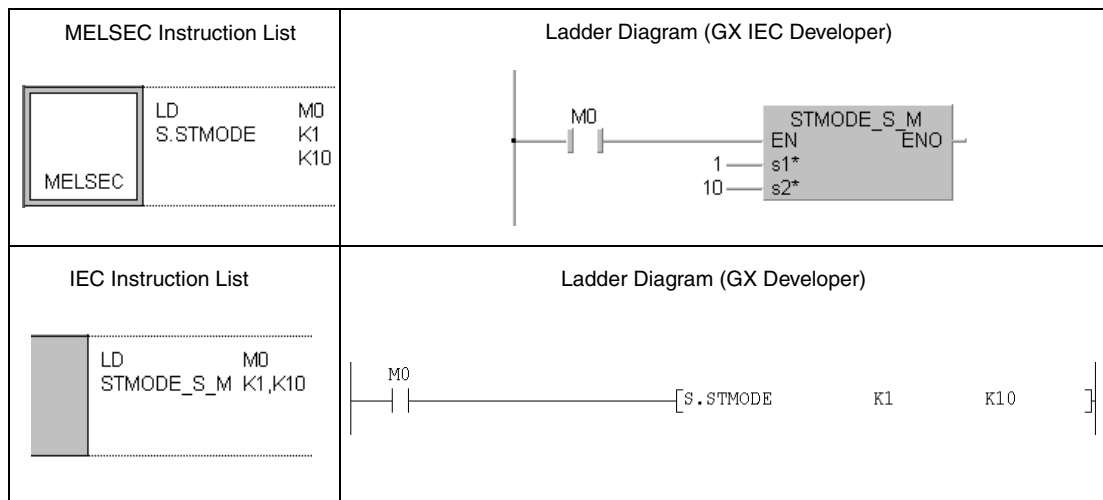
In the following case an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When a value that exceeds the specificable range is set for s1 or s2 (Error code: 4104).

**Program Example**

**STMODE**

The following program starts up the CPU in the hot start mode when the power is turned on. When the power supply of the PLC was off for more than 10 seconds a initial start will be performed.



10.1.2 CGMODE

CPU

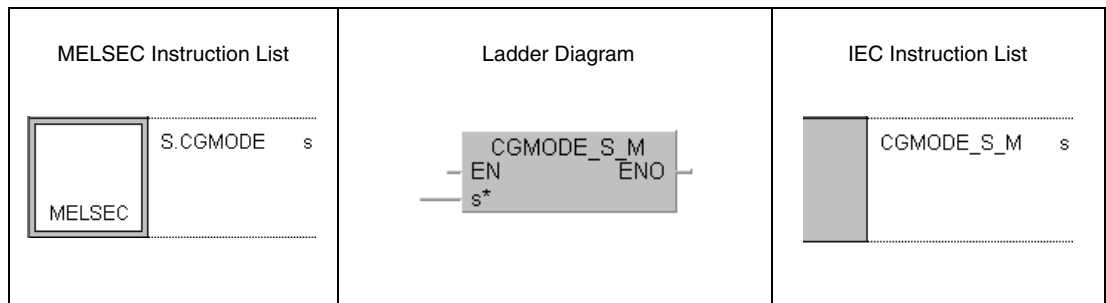
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				● <sup>1</sup>	

<sup>1</sup> For Q4AR only

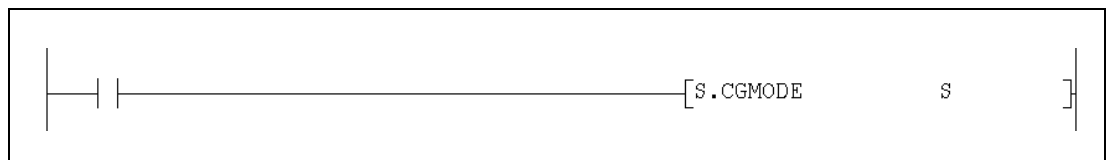
Devices  
MELSEC Q

	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)			Other
	Bit	Word		Bit	Word						
s	—	—	—	—	—	—	—	●	—	—	

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
s	Operation mode setting	BIN 16-bit



**Functions Mode during switching from control system to standby system**

**CGMODE Operation mode setting**

This instruction specifies whether the CPU devices will be cleared or not when control is switched from the control system to the standby system. This specification is made in s.

This instruction is changing from STOP to RUN when the power is turned on. For this reason there is no problem even if the instruction contact is turned off. The instruction contact becomes a dummy contact. NOP processing will be conducted when the instruction contact is turned on during program execution.

Only one of these instructions can be created in one system. Only create this instruction even if there are multiple program files. If more than one of these instructions exists then operation cannot be guaranteed.

The contents of s1 can either be 0 or 1:

0: Initial start mode (Clears devices outside the latch range)

1: Hot start mode (Devices and all signal flows (Operation results) are not cleared as in the initial start mode. Special relay SM and special register SD are preset)

**Operation Errors**

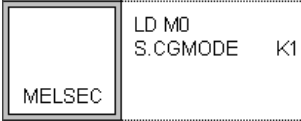
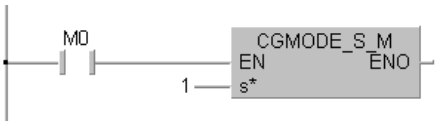
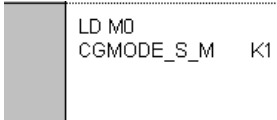

In the following case an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When a value other than 0 or 1 is specified for s (Error code: 4104).

**Program Example**

**CGMODE**

This program starts up the CPU in hot start mode when switching from the control system to the standby system.

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram (GX IEC Developer)</p> 
<p>IEC Instruction List</p> 	<p>Ladder Diagram (GX Developer)</p> 

## 10.2 Instructions for data transfer

### 10.2.1 TRUCK

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				● <sup>1</sup>	

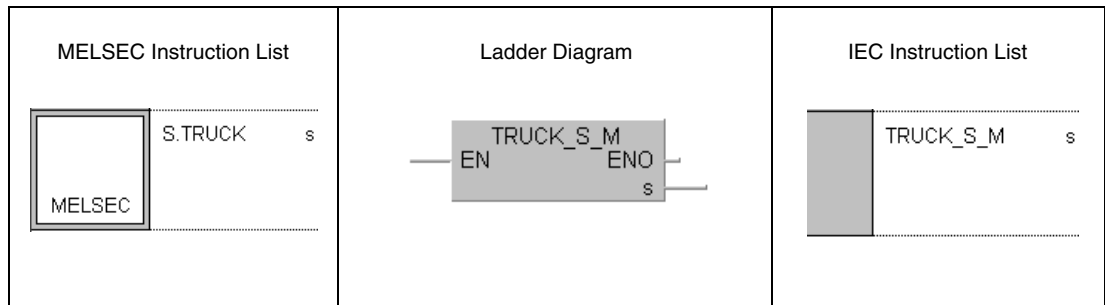
<sup>1</sup> For Q4AR only

Devices  
MELSEC Q

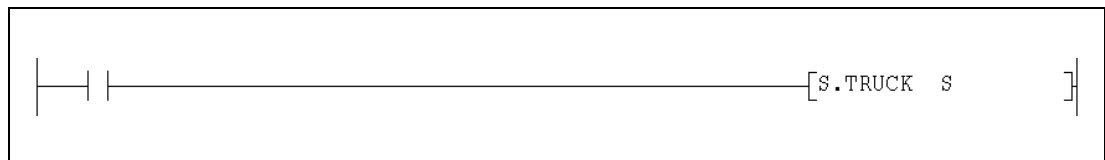
	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direkt J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	● <sup>1</sup>	●	—	—	—	—	—	—	—	—

<sup>1</sup> Latched devices only

GX IEC Developer



GX Developer



Variables

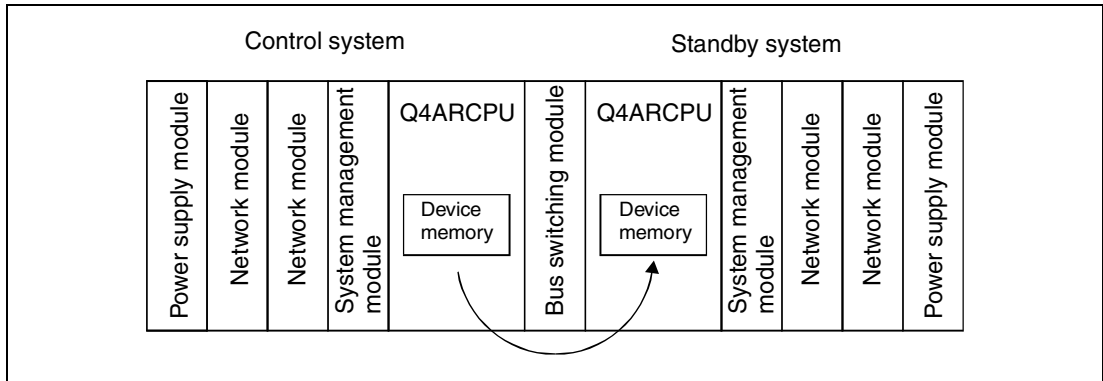
Set Data	Meaning	Data type
s	Parameter block header device	BIN 16-bit

**Functions Data transfer to the standby CPU of a redundant PLC System**

**TRUCK Data trucking instruction**

Trucking is the function that transmits the data from the control system Q4ARCPU device memory to the standby system Q4ARCPU device memory.

The Q4ARCPU conducts device memory tracking following the parameter block data contents stored in the devices from that specified in s during the END processing for each scan executed by this instruction.



Only create one of these instructions in one system. Only create this instruction in one file even if there are multiple program files. If more than one of these instructions exists operation cannot be guaranteed.

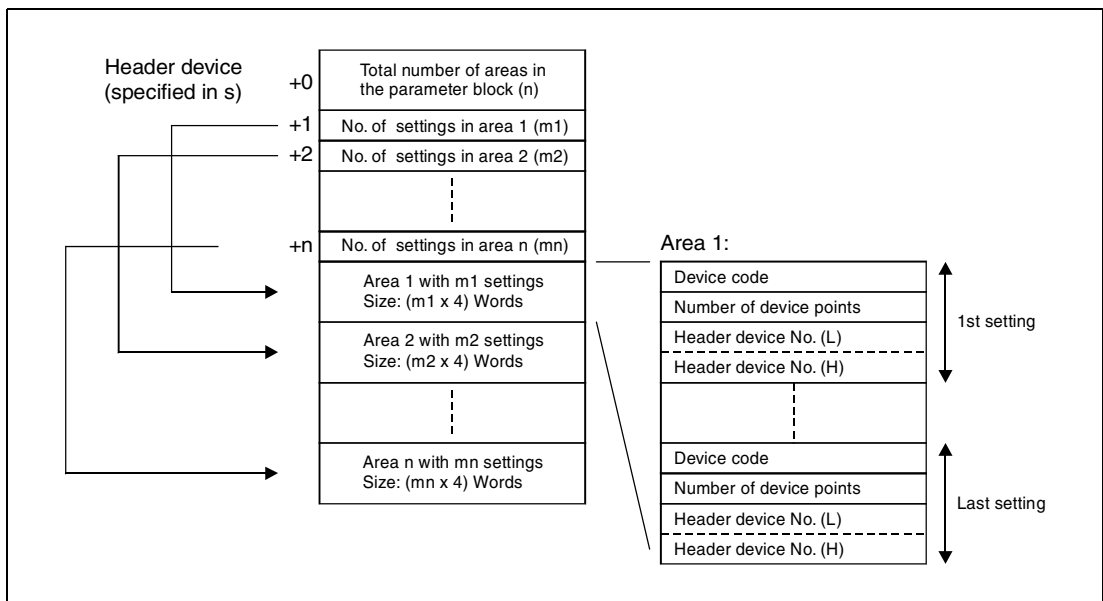
The Q4ARCPU reads the parameter block contents at power-on or reset. (Therefore, if the parameter block contents are changed the system must be restarted up.) The parameter block is configured of multiple areas. The transfer of data in each area is controlled by one of the special relays SM1520 to SM1583.

When SM1520 is set the data specified in the first area is send to the standby system, SM1521 sends the 2nd area etc. Execute this instruction after setting the block special relays SM1520 to SM1583.

**NOTE**

*For the TRUCK instruction the same transmission triggers (SM1520 to SM1583) can be used as for the SPREF instruction.*

The parameter block has the following configuration:



**Contents of the parameter block:**

- Total number of areas (n)  
The parameter block is a collection of multiple setting areas. This sets how many areas are contained in the parameter block.
- Number of settings in each area (m1 to mn)  
Multiple settings are possible in one area. Each setting consists of the device code, the number of devices and address of the header device.
- Setting areas  
Each setting occupies 4 words:  
1st word: Device code (see the following table)

Device	Code	Device	Code	Device	Code	Device	Code
X	0	B	5	C <sup>1</sup>	10	Z	15
Y	1	F	6	D	11	SB	16
M	2	V	7	W	12	SW	17
L	3	ST	8	R	13	SM	18
S	4	T <sup>1</sup>	9	ZR	14	SD	19

<sup>1</sup> For timer (T) and counter (C) the contact, the coil and the current value is included.

**NOTE**

*Devices specified as local devices are not trucked.*

2nd word: Number of devices

Setting is done in either decimal or hexadecimal. The bit devices are set in multiples of 16.

3rd and 4th word: Header device number, low (L) and high (H)

The settings are done in two words in either decimal or hexadecimal. Bit devices are set either to 0 or in multiples of 16 (e.g. 0, 16, 32, ...).

**NOTES**

*The following restrictions apply when setting parameter blocks:*

- *A maximum of 64 setting areas can exist in one parameter block ( $n \leq 64$ ).*
- *The total number of settings must not exceed 2048 ( $m_1+m_2+\dots+m_n \leq 2048$ ).*
- *When the number of settings ( $m_1$  to  $m_n$ ) is 0, the number of areas is set to 0. Setting to 0 the number of areas for which setting will not be done makes it possible to skip an area.*
- *The number of points in one block for which trucking can be done during one scan END processing is 48k words. If this number is exceeded an error will be detected and trucking cannot be executed.*
- *When specifying a bit device as the device for which trucking will be conducted, set the device number of points and header device No. to multiples of 16.*
- *When a timer or counter is specified as the device to be trucked, the following formula is used to calculate the actual number of devices that will be trucked.*

$$\text{Trucking device number of points} = \text{Set device number of points} \times (1 + 1/8)$$

*The „1“ in parentheses represents the word information of present value data, the fraction „1/8“ represents the bit information of contact or coil.*

**Modes for trucking**

With the special relay SM1518 two types of trucking can be selected. The selection is valid after the scan END processing that turns SM1518 off/on.

a) Batch transmission mode (SM1518 = 0)

If the standby system is using the trucking memory when the trucking is executed, the control system will execute the trucking processing after waiting for the standby system process to end. If control system CPU generates trucking processing wait time, so this amount of time will increase the scan time.

b) Repeat mode (SM1518 = 1)

If the standby system is using the trucking memory when trucking is executed, the control system will repeatedly conduct the following END processing without executing trucking processing. The following trucking requests cannot be received while trucking processing is being repeated. The control system CPU will not generate trucking processing wait time, so the scan time is not lengthened.

**Trucking end flag**

When a specified block trucking processing has been completed a special relay (SM1712 to SM1775) for each area is set for one program scan. (Area 1: SM1712, Area 2: SM1713 .... Area 64: SM1775)

**Operation Errors**

In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- The file for the file register does not exist even when file register R is specified in the parameter block. (Error code: 2402)
- When a value that exceeds the specification allowable range is specified. (Error code: 4104)
- When the device number of points to be trucked exceeds 48k words. (Error code: 4104)

**Program Example**

**TRUCK**

The state of the relays M0 to M95 and M320 to M639 together with the contents of the data register D0 to D29 and D600 to D699 is transmitted to the standby system. The parameter block starts at R100 and contains two areas: In the first area the internal relays are specified and in the second area are settings for the transmission of the data register. The sending of these areas is triggered by the special relays SM1520 and SM1521.

Parameter block		Meaning	Remark	
Device	Contents			
R100	2	Number of areas	—	
R101	2	Settings in area 1	—	
R102	2	Settings in area 2	—	
R103	2	Device code (2 = M)	Area 1	Setting 1
R104	96	Number of devices		
R105	0	Header device number (M0)		
R106	0			
R107	2	Device code (2 = M)		Setting 2
R108	320	Number of devices		
R109	320	Header device number (M320)		
R110	0		Area 2	Setting 1
R111	11	Device code (11 = R)		
R112	30	Number of devices		
R113	0	Header device number (D0)		
R114	0			
R115	11	Device code (11 = R)		Setting 2
R116	100	Number of devices		
R117	600	Header device number (D600)		
R118	0			

<p><b>MELSEC Instruction List</b></p> <table border="1"> <tr> <td>MELSEC</td> <td>LD</td> <td>M0</td> </tr> <tr> <td></td> <td>OUT</td> <td>SM1520</td> </tr> <tr> <td></td> <td>OUT</td> <td>SM1521</td> </tr> </table> <table border="1"> <tr> <td>MELSEC</td> <td>LD</td> <td>M10</td> </tr> <tr> <td></td> <td>S.TRUCK</td> <td>R100</td> </tr> </table>	MELSEC	LD	M0		OUT	SM1520		OUT	SM1521	MELSEC	LD	M10		S.TRUCK	R100	<p><b>Ladder Diagram (GX IEC Developer)</b></p>
MELSEC	LD	M0														
	OUT	SM1520														
	OUT	SM1521														
MELSEC	LD	M10														
	S.TRUCK	R100														
<p><b>IEC Instruction List</b></p> <table border="1"> <tr> <td></td> <td>LD</td> <td>M0</td> </tr> <tr> <td></td> <td>ST</td> <td>SM1520</td> </tr> <tr> <td></td> <td>ST</td> <td>SM1521</td> </tr> </table> <table border="1"> <tr> <td></td> <td>LD</td> <td>M10</td> </tr> <tr> <td></td> <td>TRUCK_S_M</td> <td>R100</td> </tr> </table>		LD	M0		ST	SM1520		ST	SM1521		LD	M10		TRUCK_S_M	R100	<p><b>Ladder Diagram (GX Developer)</b></p>
	LD	M0														
	ST	SM1520														
	ST	SM1521														
	LD	M10														
	TRUCK_S_M	R100														

### 10.2.2 SPREF

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				● <sup>1</sup>	

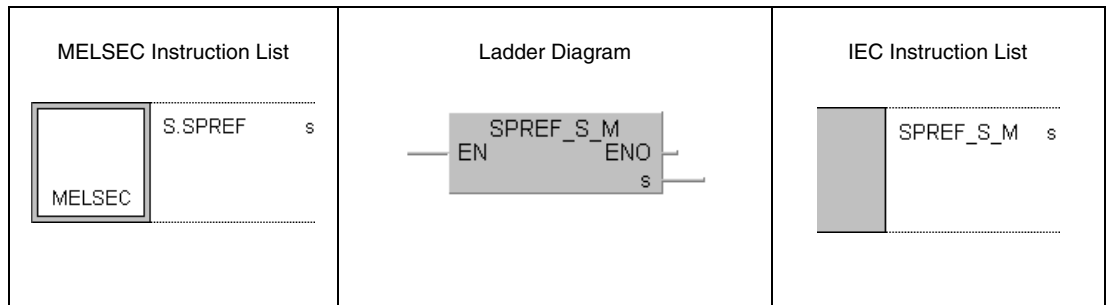
<sup>1</sup> For Q4AR only

**Devices  
MELSEC Q**

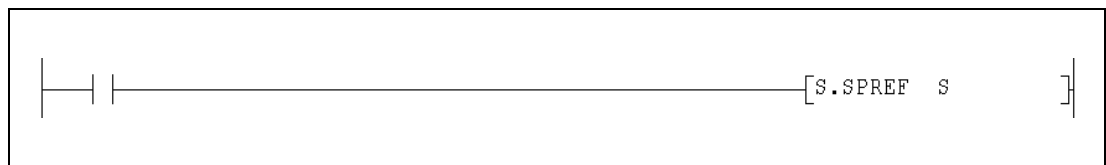
s	Usable Devices								Error Flag	Number of steps	
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)			Other
	Bit	Word		Bit	Word						
—	● <sup>1</sup>	●	—	—	—	—	—	—	—	—	

<sup>1</sup> Latched devices only.

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
s	Parameter block header device	BIN 16-bit

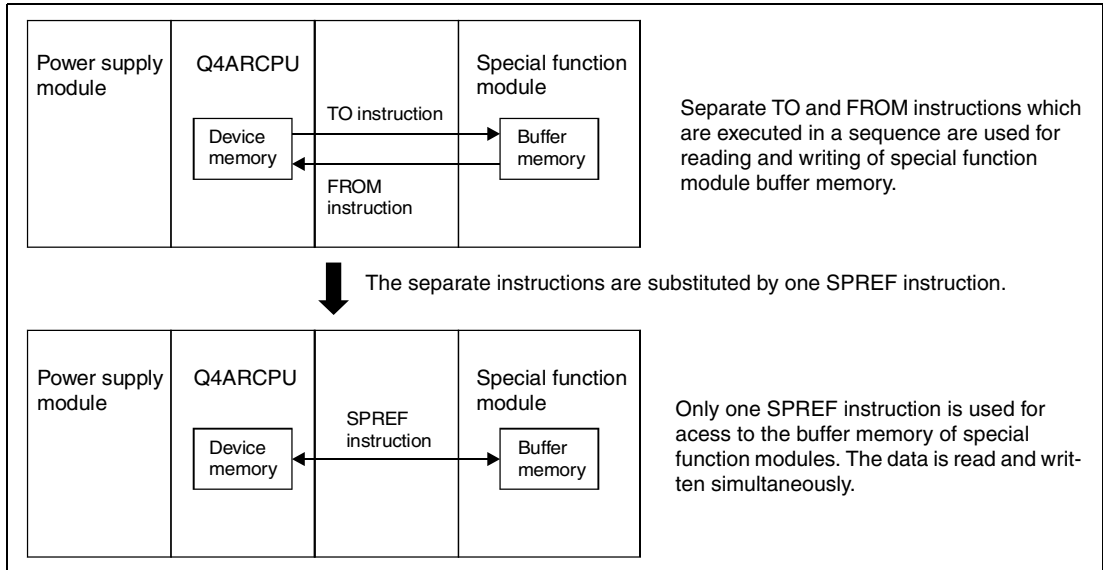
**Functions Buffer memory batch refresh instruction**

**S.SPREF Buffer memory refresh**

With the SPREF instruction the buffer memory contents of one or even multiple special function modules is batch read or written.

**NOTE**

*The buffer memory batch refresh instruction cannot be executed for the special function modules of remote I/O stations in MELSECNET (II), /B, /10 or the MELSECNET/MINI-S3.*

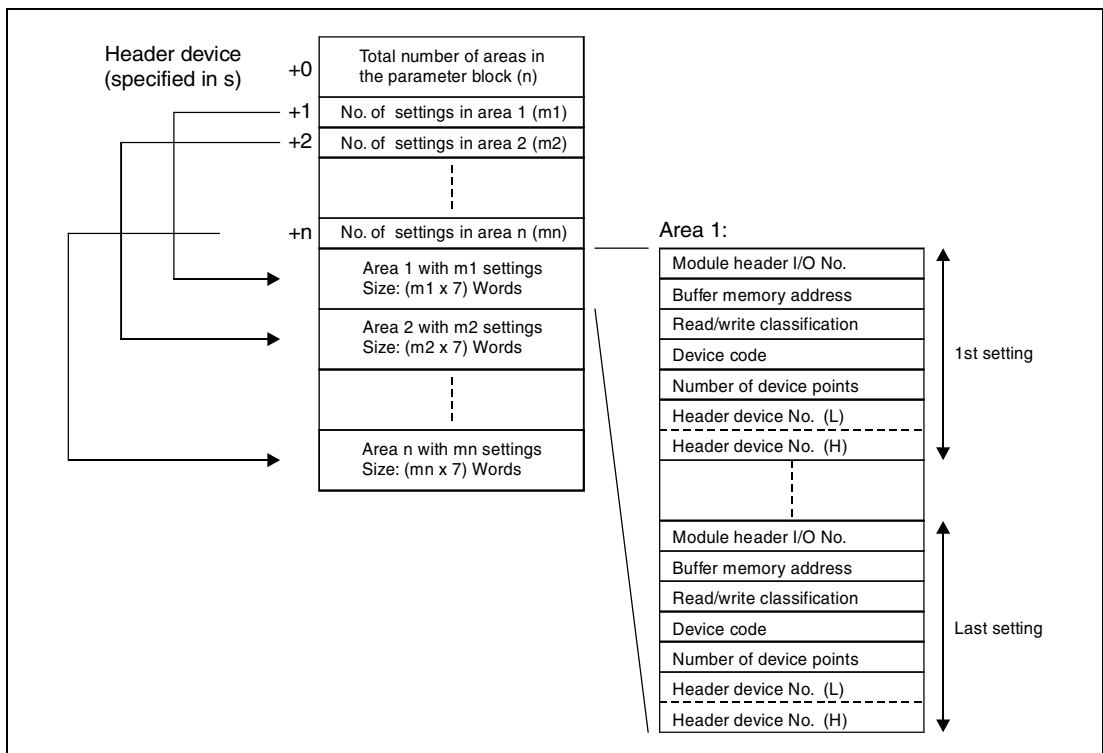


In s the header device of a parameter block with settings for the data transfer is stored. The parameter block contents should be set before the SPREF instruction is executed. The parameter block is configured of multiple areas. The transfer of data specified in each area is controlled by one of the special relays SM1520 to SM1583. When SM1520 is set the data specified in the first area is read or written, with SM1521 the specifications in the 2nd area are executed etc. The special relays SM1520 to SM1583 must be set before the execution of the SPREF instruction.

**NOTE**

For the SPREF instruction the same transmission triggers (SM1520 to SM1583) can be used as for the TRUCK instruction.

The parameter block has the following configuration:





**Contents of the parameter block:**

- Total number of areas (n)  
The parameter block is a collection of multiple setting areas. Each of these areas stores information about read/write specification, device memory type, number of points, header No. etc. This sets how many areas are contained in the parameter block.
- Number of settings in each area (m1 to mn)  
Multiple settings are possible in one area. Each setting consists of the following items.
- Setting area  
Each setting occupies 7 words of an area:

1st word: Module header I/O number

This specifies the header I/O No. for the object special function module. The setting is done with the first 2 digits when the number is expressed in a 3-digit hexadecimal number. (Example: A header I/O no. of X/Y100 is entered as 10H).

2nd word: Buffer memory address

Set the header address of the buffer memory in decimal or hexadecimal.

3rd word: read/write classification

The read/write classification sets whether to read or to write the buffer memory.

0 = Read (from buffer memory to the CPU), 1 = Write (from the CPU to the buffer memory)

4th word: Device code (see the following table)

Device	Code	Device	Code	Device	Code	Device	Code
X	0	B	5	C <sup>1</sup>	10	Z	15
Y	1	F	6	D	11	SB	16
M	2	-	-	W	12	SW	17
L	3	ST	8	R	13	SM	18
-	-	T <sup>1</sup>	9	ZR	14	SD	19

<sup>1</sup> For timer (T) and counter (C) only the current value is included.

5th word: Number of devices

Setting is done in either decimal or hexadecimal. The bit devices are set in multiples of 16.

6th and 7th word: Header device number, low (L) and high (H)

The settings are done in two words in either decimal or hexadecimal. Bit devices are set either to 0 or in multiples of 16 (e.g. 0, 16, 32, ...).

**NOTES**

*The following restrictions apply when setting parameter blocks:*

- A maximum of 64 setting areas can exist in one parameter block ( $n \leq 64$ ).
- The total number of settings must not exceed 2048 ( $m1+m2+...mn \leq 2048$ ).
- When the number of settings (m1 to mn) is 0, the number of areas is set to 0. Setting to 0 the number of areas for which setting will not be done makes it possible to skip an area.

**Operation Errors**

In the following case an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When a value that exceeds the specificable range is specified. (Error code: 4104)

**Program Example**

SPREF

The following program transfers data between the CPU and the buffer memory of two special function modules. For each module an area exists in the parameter block which is stored from File-Register R100 onwards:

- 1st area: Communication with the special function module with the module header I/O address X/Y20

The specifications in this area are fulfilled when SM1520 is set.

The contents of the buffer memory addresses 0 to 3 is read and stored in the file registers R0 to R3.

The contents of the file register R10 and R11 is written to the buffer memory addresses 10 and 11 of the special function module.

- 2nd area: Communication with the special function module with the module header I/O address X/Y100

The specifications in this area are fulfilled when SM1520 is set.

The contents of the buffer memory addresses 110 to 119 is read and stored in data registers D110 to D113.

The parameter block for this example contains the following constants:

Parameter block		Meaning	Remark	
Device	Contents			
R100	2	Number of areas	—	
R101	2	Settings in area 1	—	
R102	1	Settings in area 2	—	
R103	2	Module header I/O number (X/Y20)	Area 1	Setting 1
R104	0	Buffer memory header address		
R105	0	Read/write classification (0 = Read)		
R106	13	Device code (13 = R)		
R107	4	Number of devices		
R108	0	Header device number (R0)		
R109	0			
R110	2	Module header I/O number (X/Y20)		Setting 2
R111	10	Buffer memory header address		
R112	1	Read/write classification (1 = Write)		
R113	13	Device code (13 = R)		
R114	2	Number of devices		
R115	10	Header device number (R10)		
R116	0			
R117	10	Module header I/O number (X/Y100)	Area 2	Setting 1
R118	110	Buffer memory header address		
R119	0	Read/write classification (0 = Read)		
R120	11	Device code (11 = D)		
R121	10	Number of devices		
R122	110	Header device number (D110)		
R123	0			

<p style="text-align: center;"><b>MELSEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: center;">MELSEC</td> <td style="width: 35%;">LD OUT OUT</td> <td style="width: 50%;">M0 SM1520 SM1521</td> </tr> <tr> <td style="text-align: center;">MELSEC</td> <td>LD S.SPREF</td> <td>M10 R100</td> </tr> </table>	MELSEC	LD OUT OUT	M0 SM1520 SM1521	MELSEC	LD S.SPREF	M10 R100	<p style="text-align: center;"><b>Ladder Diagram (GX IEC Developer)</b></p>
MELSEC	LD OUT OUT	M0 SM1520 SM1521					
MELSEC	LD S.SPREF	M10 R100					
<p style="text-align: center;"><b>IEC Instruction List</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"></td> <td style="width: 35%;">LD ST ST</td> <td style="width: 50%;">M0 SM1520 SM1521</td> </tr> <tr> <td></td> <td>LD SPREF_S_M</td> <td>M10 R100</td> </tr> </table>		LD ST ST	M0 SM1520 SM1521		LD SPREF_S_M	M10 R100	<p style="text-align: center;"><b>Ladder Diagram (GX Developer)</b></p>
	LD ST ST	M0 SM1520 SM1521					
	LD SPREF_S_M	M10 R100					



---

# 11 Instructions for Special Function Modules

Instructions	Function
Instructions for serial communication modules	Reading of received data in an interrupt program; Reading, registration or deletion of user frames; Transmission of data using user frames
Instructions for PROFIBUS/DP interface modules	Reading or writing of data from and to the buffer memory of a PROFIBUS/DP interface module
Instructions for ETHERNET interface modules	Writing and reading of data to and from fixed buffer; Opening and closing of connections, Clearing of error codes; Re-initialization of the ETHERNET interface module
Instructions for MELSECNET/10	Setting of stations for duplex network
Instructions for CC-Link	Parameter setting, Setting of automatic refresh parameters Reading of data from the buffer memory of a station connected to CC-Link or from the PLC CPU of this station; Writing of data to the buffer memory of a station connected to CC-Link or to the PLC CPU of this station; Reading and writing from and to the automatic updated buffer memory

## 11.1 Instructions for Serial Communication Modules

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Reading of received data from a QJ71C24 in an interrupt program	Z.BUFRCVS	BUFRCVS_M
Reading of user registered frames	G.GETE	GETE_M
	GP.GETE	GETEP_M
Registration or deletion of user registered frames	G.PUTE	PUTE_M
	GP.PUTE	PUTEP_M
Transmission of user frames	G.PRR	PRR_M
	GP.PRR	PRRP_M

**11.1.1 BUFRCVS**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

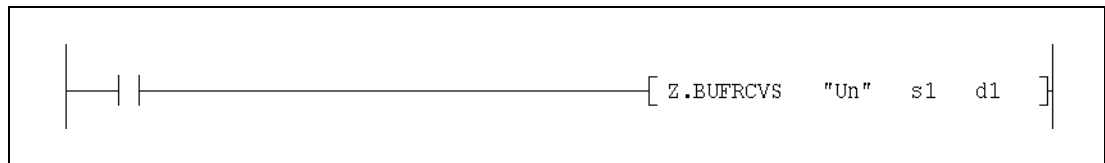
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
d1	●	●	●	—	—	—	—	—	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p>
--------------------------------	-----------------------	-----------------------------

**GX Developer**



**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the serial communication module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Reception channel number 1: Channel 1 (CH1) 2: Channel 2 (CH2)	1 or 2			
Head number of the devices that stores received data					
d1	<b>Set Data</b>	<b>Meaning</b>	<b>Description</b>	<b>Range</b>	<b>Contents is stored by</b>
	(d1)+0	Data length	Length of the received data The unit (bytes or words) is set in the parameters.	—	System
	(d1)+1 to (d1)+n	Received data	In this area the data read from the receive area of the buffer memory is stored sequentially in ascending order.		

**Functions Reading of received data from the QJ71C24**

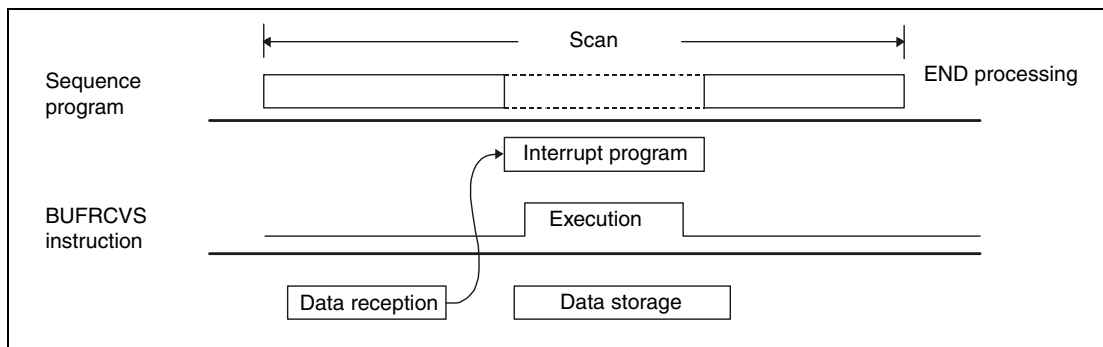
**BUFRCVS Data read**

The BUFRCVS instruction reads data sent from an external device to the communication module QJ71C24 from the buffer memory of the QJ71C24 and stores the data in the CPU module.

The BUFRCVS instruction can identify the address of the reception area in the buffer memory and read relative receive data to the area designated with d1.

When the data transfer is completed, the reception data read request (X3/XA) or the reception abnormal detection signal (X4/XB) is turned off automatically. It is not necessary to turn on the reception data completion signal (Y1/Y8) when received data is read by the BUFRCVS instruction.

The BUFRCVS instruction is used by an interrupt program and its processing is completed in one scan. The following figure shows the timing when the BUFRCVS instruction is being executed:



**NOTES**

*When received data is read with a BUFRCVS instruction in an interrupt program, the data of the same interface can not be read again in the main program. Thus the BUFRCVS instruction can not used together with the following instructions:*

- the INPUT instruction
- the BIDIN instruction
- the FROM instruction in combination with input/output signals of the communication module

*The BUFRCVS and the CSET instruction cannot be executed at the same time.*

*The area specified with d1 in the PLC CPU must be large enough to store all data sent from the external device. If this area is too small, the data that can not be stored, is lost.*

**Operation Errors**

When the BUFRCVS instruction is completed abnormally, the error flag SM0 is set, and an error code is stored in SD0. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, please refer to chapter 13 of this manual for error diagnostics.
- When the error code is 7000<sub>H</sub> or higher, please refer to the user's manual of the serial communication module QJ71C24.

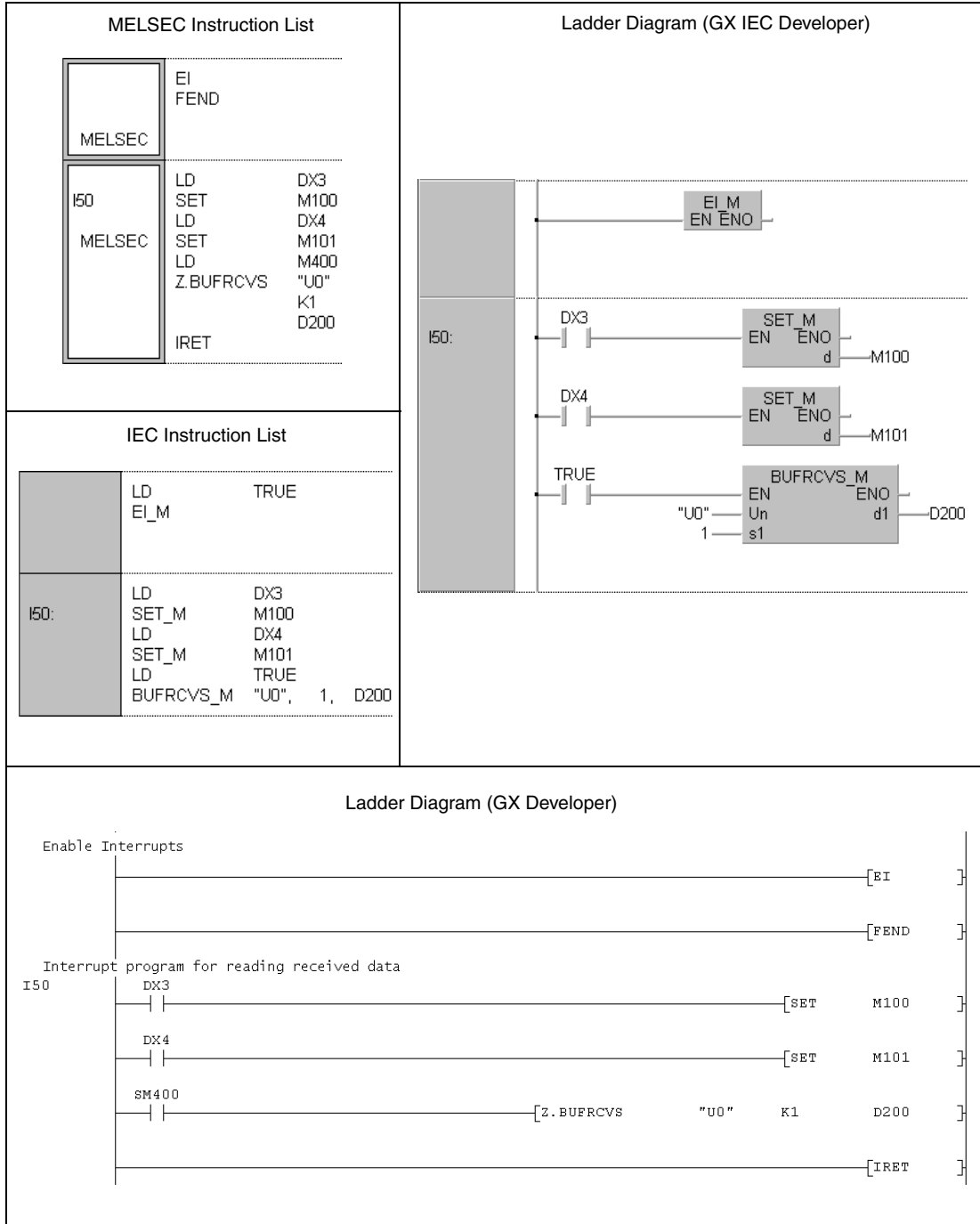
If an error occurs during data reception (indicated by the input signals X4 and XB), the error code is written to the buffer memory addresses 258<sub>H</sub> and 268<sub>H</sub> of the communication module and can be used for diagnostics.



**Program Example**

**BUFRCVS**

The following program reads the data received via channel 1 of a QJ71C24 with the head address X/Y0 and stores the data from D200 onward. Only channel 1 issues an interrupt. When data is received, the interrupt program 50 (I50) is processed. The internal relays M100 and M101 are used as interface with the main program. If data was received correctly, M100 is set. When an error occurs during reception of the data, M101 is set. Both relays are reset in the main program.



11.1.2 GETE, GETEP

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

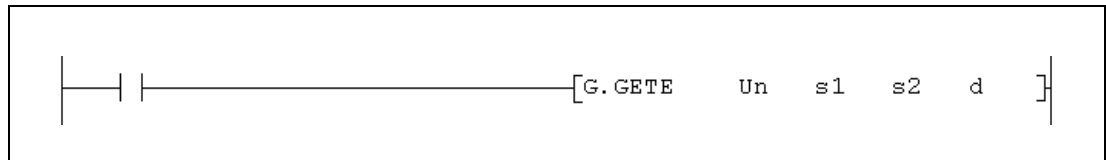
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
s2	—	●	●	—	—	—	—	—	—		
d	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <p style="text-align: center;">MELSEC</p> </div> <p>G.GETE      Un                  s1                  s2                  d</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <p style="text-align: center;">GETE_M      Un, s1, s2, d</p> </div>
---	-----------------------	--

GX Developer



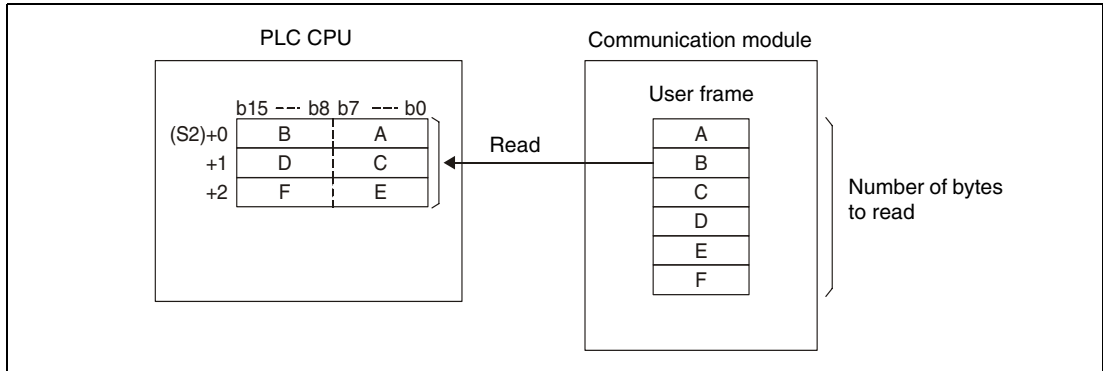
**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
Un	Head I/O address of the serial communication module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Head number of the devices that store control data				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s1)+0	Dummy	Used by the system	0	—
	(s1)+1	Read result	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred and the stored value is an error code.		System
	(s1)+2	Frame number	Number of the user frame	1000 to 1199	User
	(s1)+3	Number of bytes to read	Max. number of bytes of the user frame that can be stored in the area specified by s2	1 to 80	
Number of read bytes		Number of bytes of the user frame that has been read	1 to 80	System	
s2	Head number of the devices that store the read data		User System	Address	
d	Bit device which is set for one scan after completion of the GETE instruction. (d)+1 indicates an abnormal completion of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d)+0	Instruction completed	Indicates the completion of the GETE instruction ON: Instruction completed OFF: Instruction not completed	—	System
	(d)+1	Instruction completed with error	Indicates the abnormal completion of the GETE instruction. ON: Abnormal completion OFF: Normal completion	—	

**Functions Reading of user registered frames**

**GETE Data read**

The GETE instruction reads data from a user frame in a serial communication module and stores the data in the PLC CPU. The head address of the communication module is specified with Un.

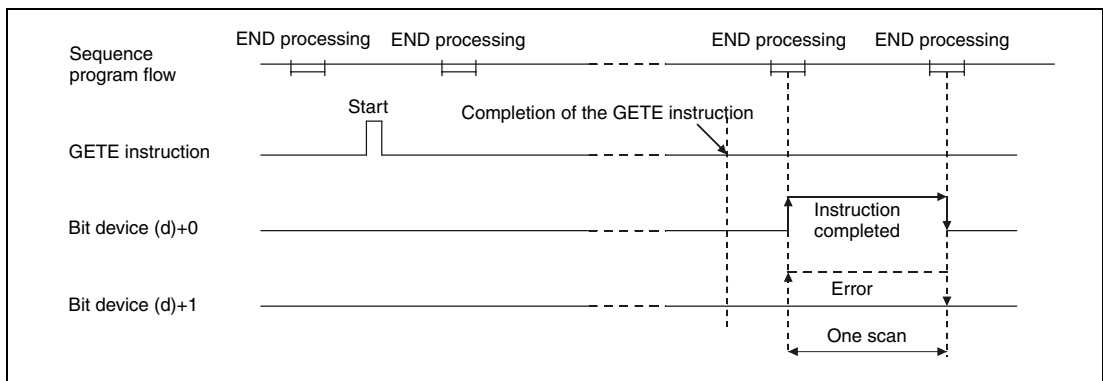


During GETE instruction execution, another GETE or PUTE instruction cannot be executed. If an attempt is made to execute a GETE or PUTE instruction during execution of a GETE instruction, the system waits until the execution of the instruction already being processed is completed.

Whether the execution of the GETE instruction has been finished can be checked with the devices (d)+0 and (d)+1:

- The bit device (d)+0 turns ON at the END processing of the scan in which the GETE instruction has been completed and turns OFF at the next END processing.
- The bit device (d)+1 indicates an error during execution of the GETE instruction. When the instruction is completed normal, this device stays OFF. When an error occurs during execution of the GETE instruction, (d)+1 turns ON at the END processing of the scan in which the GETE instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the GETE instruction is being executed:



**Operation Errors**

When an error occurs during execution of the GETE instruction, the bit device (d)+1 is set and an error code is written to (s1)+1. For more information about the error codes please refer to the following manuals:

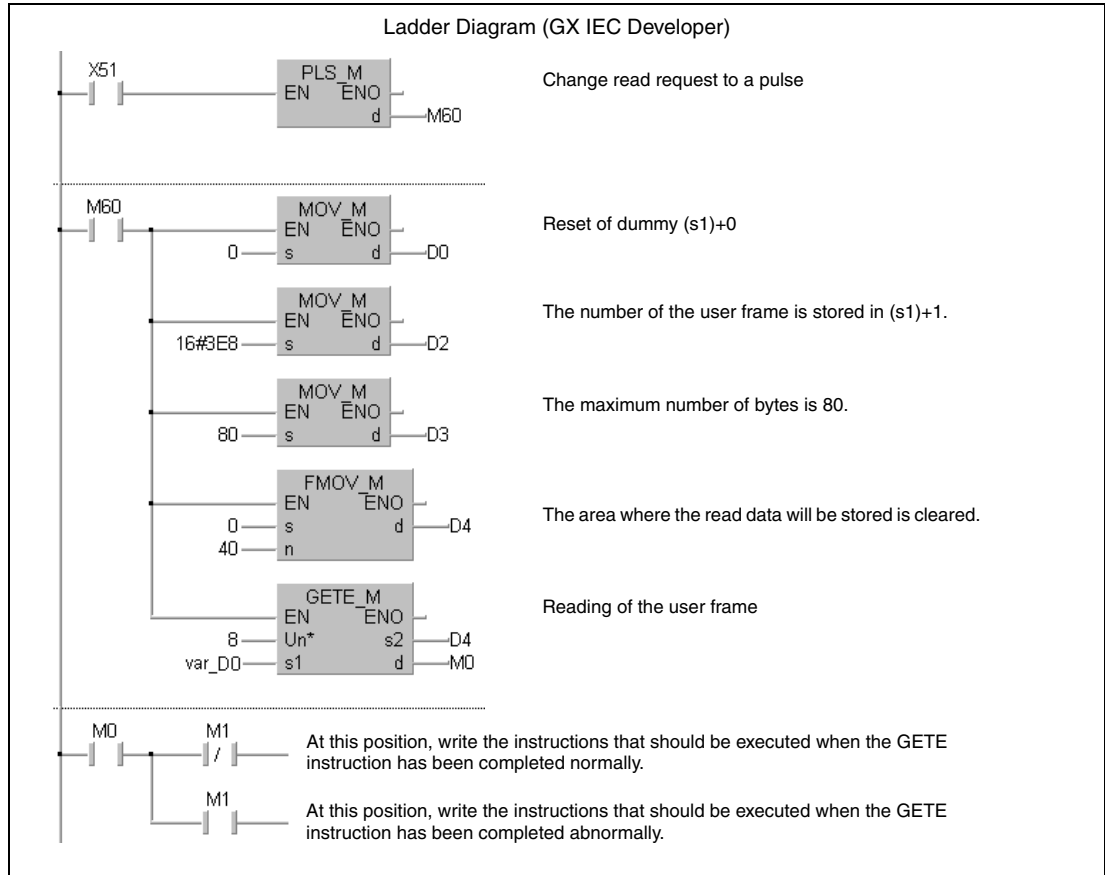
- When the error code is 4FFF<sub>H</sub> or less, refer to chapter 13 of this manual for error diagnostics.
- When the error code is 7000<sub>H</sub> or higher, you will find more information in the user's manual of the serial communication module.

**Program Example**

**GETE**

The following program reads data of the user frame with the number 3E8<sub>H</sub> from a QJ71C24 and stores the data in the QCPU from data register D4 onward. The communication module occupies the input/output signals from X/Y80 to X/Y9F.

- IEC editors



IEC Instruction List

LD	X51				
PLS_M	M60				
LD	M60				
MOV_M	0,	D0			
MOV_M	16#3E8,	D2			
MOV_M	80,	D3			
FMOV_M	0,	40,	D4		
GETE_M	8,	var_D0,	D4,	M0	
LD	M0				
ANDN	M1				
An instruction at this position will be executed when the GETE instruction has been completed normally.					
LD	M0				
AND	M1				
An instruction at this position will be executed when the GETE instruction has been completed with an error.					

The devices and instructions used are explained in the above ladder diagram.

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

At this position, write the instructions that should be executed when the GETE instruction has been completed normally.  
 At this position, write the instructions that should be executed when the GETE instruction has been completed not normally.

---

MELSEC Instruction List

MELSEC	LD X51 PLS M60
MELSEC	LD M60 MOV K0 D0 MOV H3E8 D2 MOV K80 D3 FMOV K0 D4 K40 G.GETE U8 D0 D4 M0
MELSEC	LD M0 MPS ANI M1 An instruction at this position will be executed when the GETE instruction has been completed normally MPP AND M1 An instruction at this position will be executed when the GETE instruction has been completed with an error.

**11.1.3 PUTE, PUTEP**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

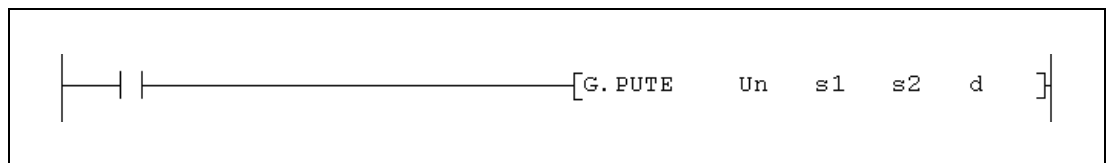
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
s2	—	●	●	—	—	—	—	—	—		
d	●	●	●	—	—	—	—	—	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>MELSEC</p> </div> <p>G.PUTE      Un                  s1                  s2                  d</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>PUTE_M      Un, s1, s2, d</p> </div>
---	-----------------------	--

**GX Developer**



Variables

Set Data	Meaning			Range	Contents is stored by	Data Type
Un	Head I/O address of the serial communication module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)			0 to FE <sub>H</sub>	User	BIN 16-bit
s1	Head number of the devices that store control data					
	Set Data	Meaning	Description	Range	Contents is stored by	BIN 16-bit
	(s1)+0	Selection: Register or delete user frame	Designate whether to register or to delete the user frame specified by (s1)+2: • 1: Register • 3: Delete	1 or 3	User	
	(s1)+1	Register/delete result	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred and the stored value is an error code.	—	System	
	(s1)+2	Frame No.	Number of the user frame to register or to delete	1000 to 1199	User	
(s1)+3	Number of bytes to register	Number of bytes of the user frame to be registered. Please set also a value between 1 and 80 as dummy when deleting a user frame [(s1)+0 = 3].	1 to 80			
s2	Head number of the devices that store the data to be registered.				User	Address
d	Bit device which is set for one scan after completion of the PUTE instruction. (d)+1 indicates an abnormal completion of the instruction.					
	Operand	Meaning	Description	Range	Contents is stored by	Bit
	(d)+0	Instruction completed	Indicates the completion of the PUTE instruction ON: Instruction completed OFF: Instruction not completed	—	System	
(d)+1	Instruction completed with error	Indicates the abnormal completion of the PUTE instruction ON: Abnormal completion OFF: Normal completion	—			



**Functions Registration or deletion of user frames**

**PUTE Register or delete user frames**

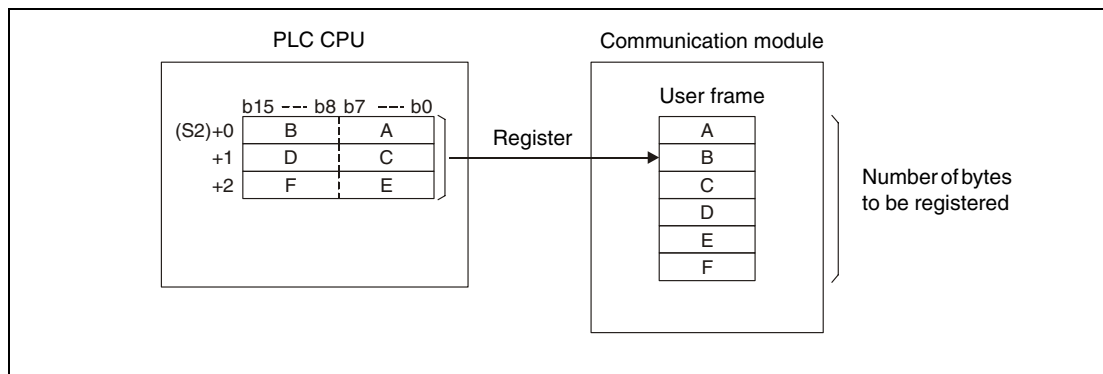
The PUTE instruction is used to register or delete user frames in a serial communication module. The head address of the serial communication module is specified with Un.

**Registering a user frame**

When registering a user frame, write „1“ to the device designated with (s1)+0. Data from the devices starting with the device designated by s2 will be registered in accordance with the control data.

Since each device can store two bytes of data, the number of necessary devices equals half the number of data bytes.

If for instance six bytes are to be registered in a user frame, two additional devices must be reserved after s2:



**Deletion of a user frame**

To delete the user frame, whose number is written in (s1)+2, write „3“ to the device designated with (s1)+0.

Although the number of bytes [(s1)+3] and the area specified with s2 are not used during deletion, these settings are required for the PUTE instruction format. Write any value between 1 and 80 to the device designated by (s1)+3 and choose a dummy for s2.

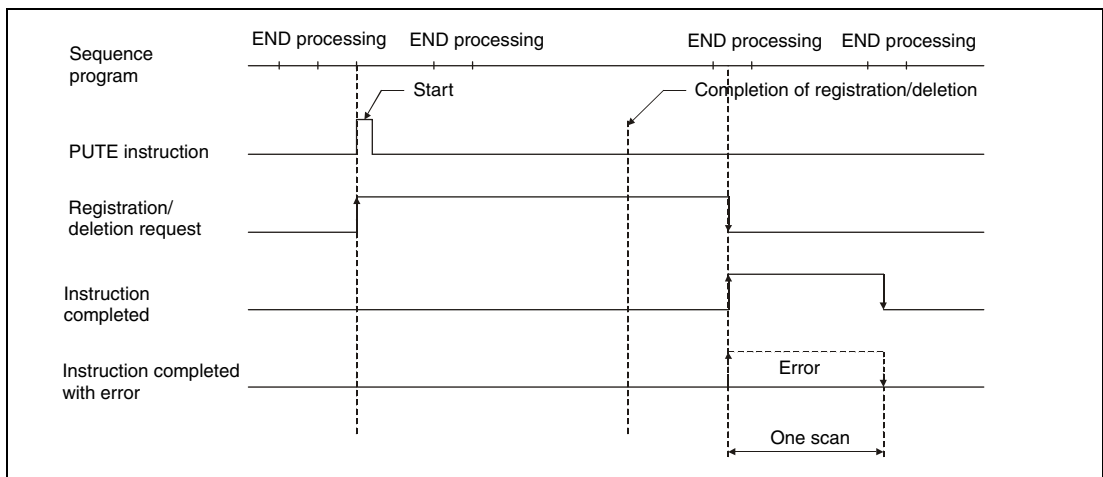
**Operation conditions**

During execution of a PUTE instruction, it is not possible to execute another PUTE or GETE instruction. If an attempt is made to execute one of these instructions when a PUTE instruction is already being executed, the system waits until the execution of the instruction already being processed is completed.

Whether the execution of the PUTE instruction has been finished or not can be checked with the devices (d)+0 and (d)+1:

- The bit device (d)+0 turns ON with the END processing of the scan in which the PUTE instruction has been completed and turns OFF at the next END processing.
- The bit device (d)+1 indicates an error during execution of the PUTE instruction. When the instruction is completed normal, this device stays OFF. When an error occurs during execution of the PUTE instruction, (d)+1 turns ON at the END processing of the scan in which the PUTE instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing for the PUTE instruction:



**Operation Error**

When an error occurs during execution of the PUTE instruction, the bit device (d)+1 is set and an error code is written to (s1)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, please refer to chapter 13 of this manual for error diagnostics.
- When the error code is 7000<sub>H</sub> or higher, please refer to the user's manual of the serial communication module.

**Program Example**

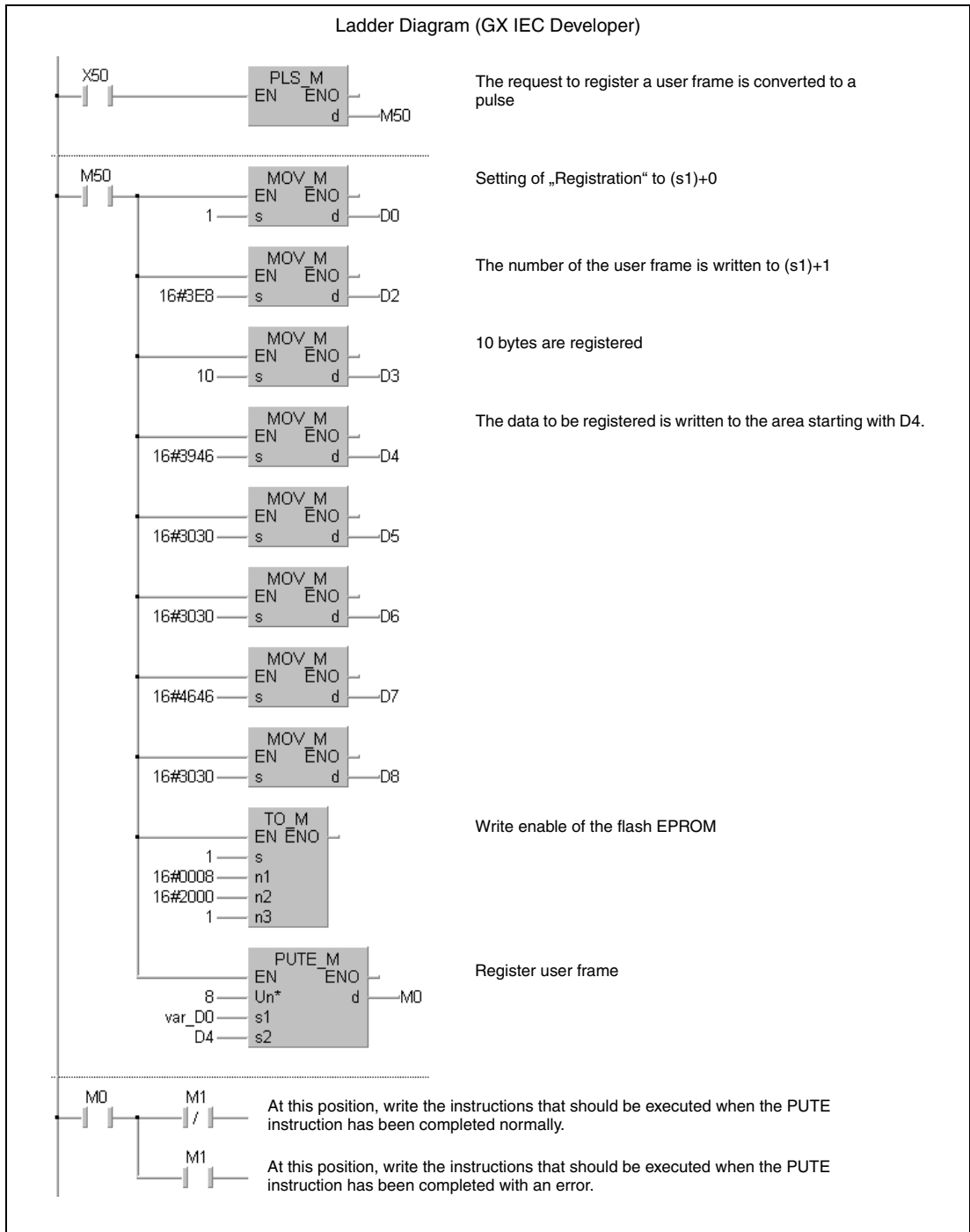
PUTE

The following program registers data to the user frame with the number 3E8<sub>H</sub>. A QJ71C24 is used as communication module. It occupies the input/output signals from X/Y80 to X/Y9F.

**NOTE**

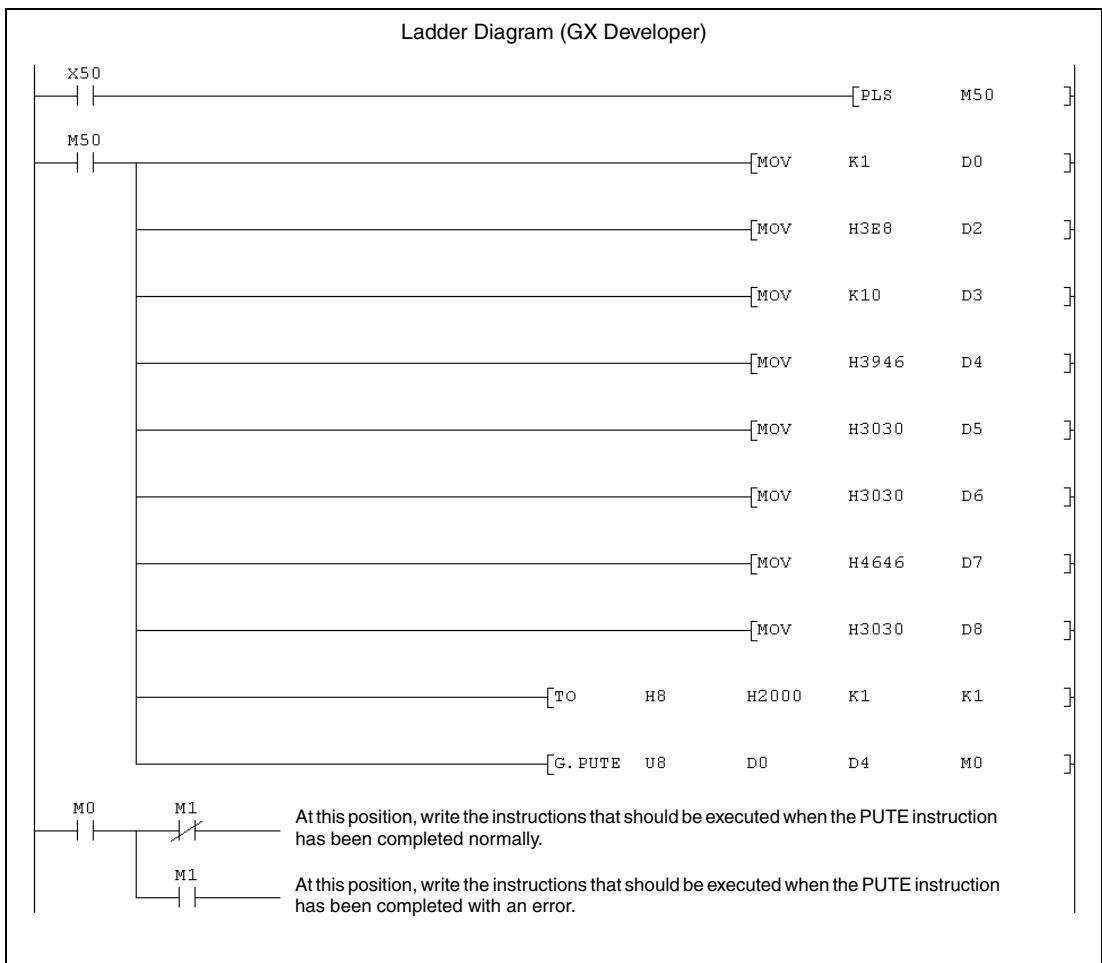
When using the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- IEC editors



IEC Instruction List				
LD	X50			
PLS_M	M50			
For an explanation of the devices and instructions used please see the ladder diagram on the previous page.				
LD	M50			
MOV_M	1,	D0		
MOV_M	16#3E8,	D2		
MOV_M	10,	D3		
MOV_M	16#3946,	D4		
MOV_M	16#3030,	D5		
MOV_M	16#3030,	D6		
MOV_M	16#4646,	D7		
MOV_M	16#3030,	D8		
TO_M	1,	16#0008,	16#2000,	1
PUTE_M	8,	var_DO,	D4,	M0
LD	M0			
ANDN	M1			
An instruction at this position will be executed when the PUTE instruction has been completed normally.				
LD	M0			
AND	M1			
An instruction at this position will be executed when the PUTE instruction has been completed with an error.				

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous pages.



MELSEC Instruction List					
MELSEC	LD	X50			
	PLS	M50			
MELSEC	LD	M50			
	MOV	K1	D0		
	MOV	H3E8	D2		
	MOV	K10	D3		
	MOV	H3946	D4		
	MOV	H3030	D5		
	MOV	H3030	D6		
	MOV	H4646	D7		
	MOV	H3030	D8		
	TO	H8	H2000	K1	K1
	G.PUTE	U8	D0	D4	M0
MELSEC	LD	M0			
	MPS				
	ANI	M1			
	An instruction at this position will be executed when the PUTE instruction has been completed normally.				
	MPP				
	AND	M1			
	At this position, write the instructions that should be executed when the PUTE instruction has been completed with an error.				

11.1.4 PRR, PRRP

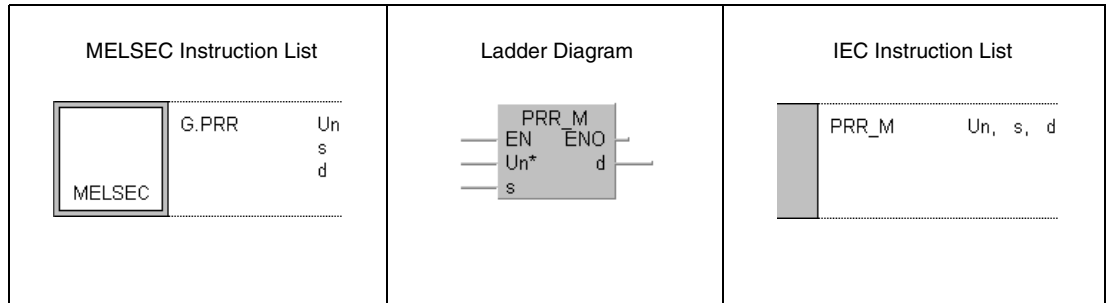
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

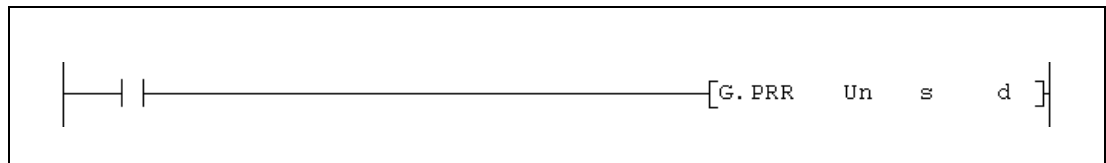
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
d1	●	●	●	—	—	—	—	—	—		

GX IEC  
Developer



GX  
Developer



**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
Un	Head I/O address of the serial communication module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s	Head number of the devices that store control data.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s)+0	Transmission channel	Designation of the channel used to transmit data 1: Channel 1 (CH1) 2: Channel 2 (CH2)	1 or 2	User
	(s)+1	Transmission result	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred and the stored value is an error code.	—	System
	(s)+2	Addition of CR/LF	Designate whether or not to add CR/LF to the transmission data 0: Do not add CR/LF 1: Add CR/LF	0 or 1	User
	(s)+3	Transmission pointer	Pointer to the first address of the device area which stores the data to be transmitted.	1 to 100	
(s)+4	Number of user frames	Designation of the number of user frames to be transmitted.	1 to 100		
d	Bit device which is set for one scan after completion of the PRR instruction. (d)+1 indicates an abnormal completion of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d)+0	Instruction completed	Indicates the completion of the PRR instruction ON: Instruction completed OFF: Instruction not completed	—	System
	(d)+1	Instruction completed with error	Indicates the abnormal completion of the PRR instruction ON: Abnormal completion OFF: Normal completion	—	

**Functions**      **Transmission of user frames**

**PRR**                      **Transmit user frames**

The PRR instruction transmits data using user frames to the communication module designated by Un. Information about the processing of the instruction are stored from the device designated by s. The contents of the user frames has to be set in the communication module before the PRR instruction is executed.

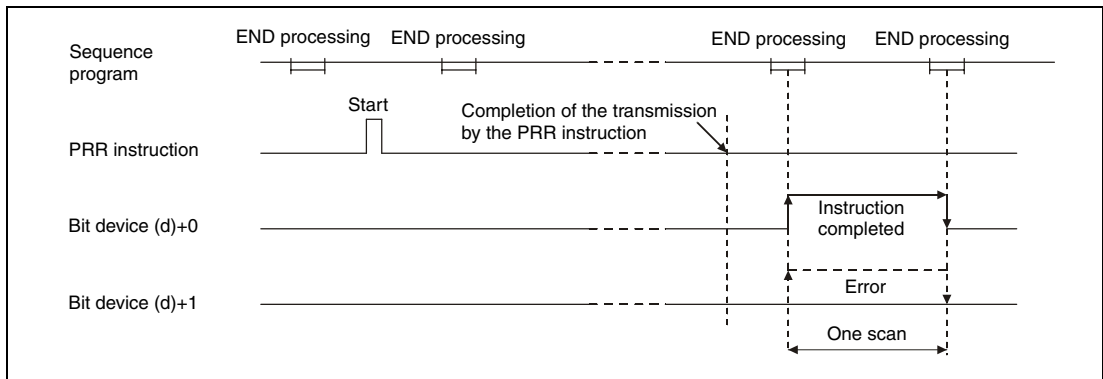
While a PRR instruction is being executed the following instructions cannot be executed for the same channel of the communication module:  
 OUTPUT instruction, ONDEMAND instruction, BIDOUT instruction and other PRR instructions.

If an attempt is made to execute any of the above instructions while an PRR instruction is being executed, the system waits until the PRR instruction already being executed is completed.

Whether the execution of the PRR instruction has been finished can be checked with the devices (d)+0 and (d)+1:

- The bit device (d)+0 turns ON with the END processing of the scan in which the PRR instruction has been completed and turns OFF at the next END processing.
- The bit device (d)+1 indicates an error during execution of the PRR instruction. When the instruction is completed normal, this device stays OFF. When an error occurs during the execution of the PRR instruction, (d)+1 turns ON at the END processing of the scan in which the PRR instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing for the PRR instruction:



**Operation Error**

When an error occurs during execution of the PUTE instruction, the bit device (d)+1 is set and an error code is written to (s1)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, please refer to chapter 13 of this manual for error diagnostics.
- When the error code is 7000<sub>H</sub> or higher, please refer to the user's manual of the serial communication module.



**Program Example**

PRR

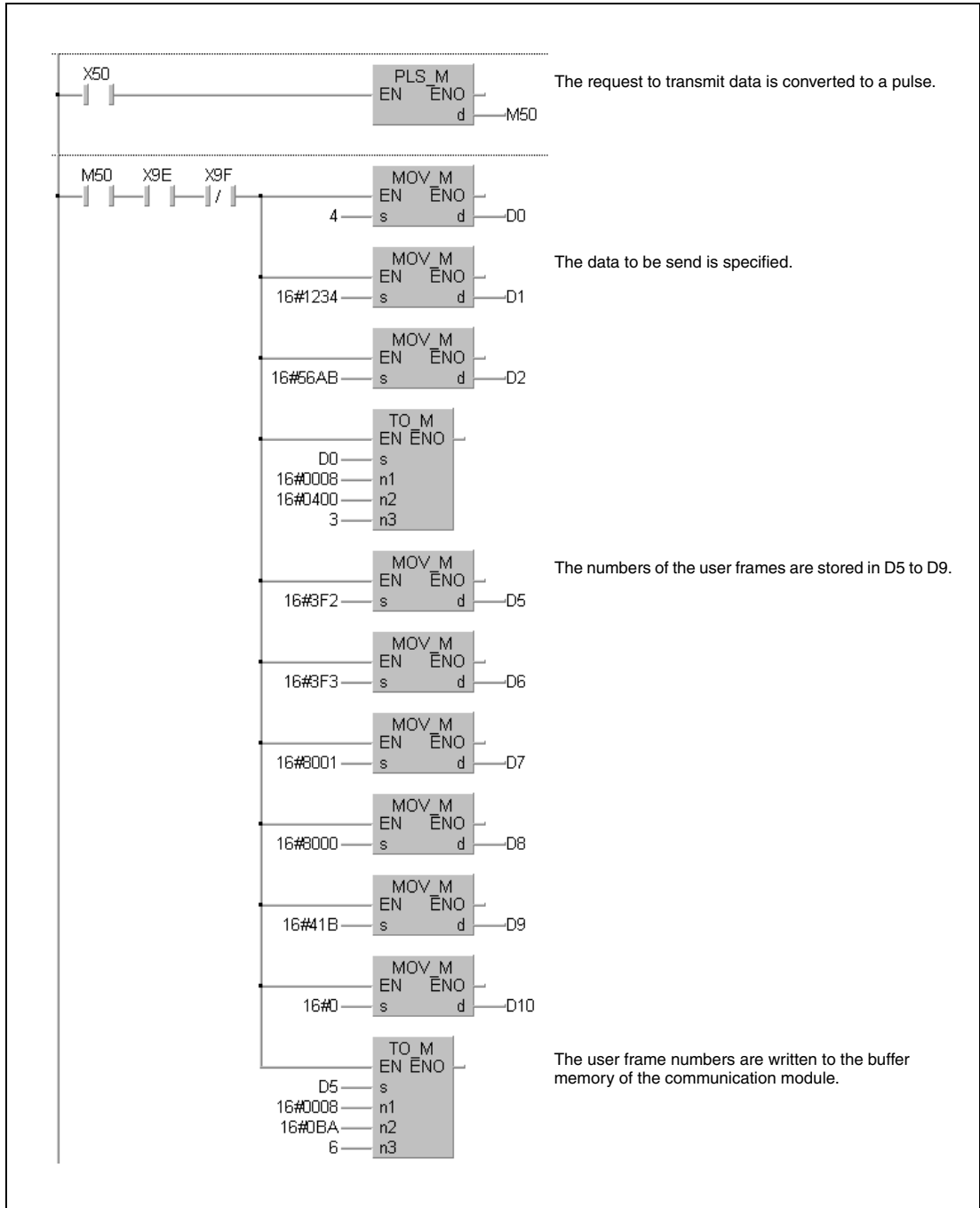
The program for this example transmits data and the first five user frames. The communication module QJ71C24 is used. It occupies the input/output signals from X/Y80 to X/Y9F. The following data registers are used in the program:

Data register	Contents	Meaning	
D0	0004 <sub>H</sub>	Number of bytes to send	
D1	3412 <sub>H</sub>	Data to be send	
D2	AB56 <sub>H</sub>		
D5	03F2 <sub>H</sub>	Numbers of the user frames	
D6	03F3 <sub>H</sub>		
D7	8001 <sub>H</sub>		
D8	8000 <sub>H</sub>		
D9	041B <sub>H</sub>		
D10	0000 <sub>H</sub>		
D11	0001 <sub>H</sub>	(s)+0	Interface: CH1
D12	0000 <sub>H</sub> or error code	(s)+1	Transmission result
D13	0000 <sub>H</sub>	(s)+2	CR/LF is not added
D14	0001 <sub>H</sub>	(s)+3	Transmission pointer
D15	0005 <sub>H</sub>	(s)+4	Number of data frames to transmit

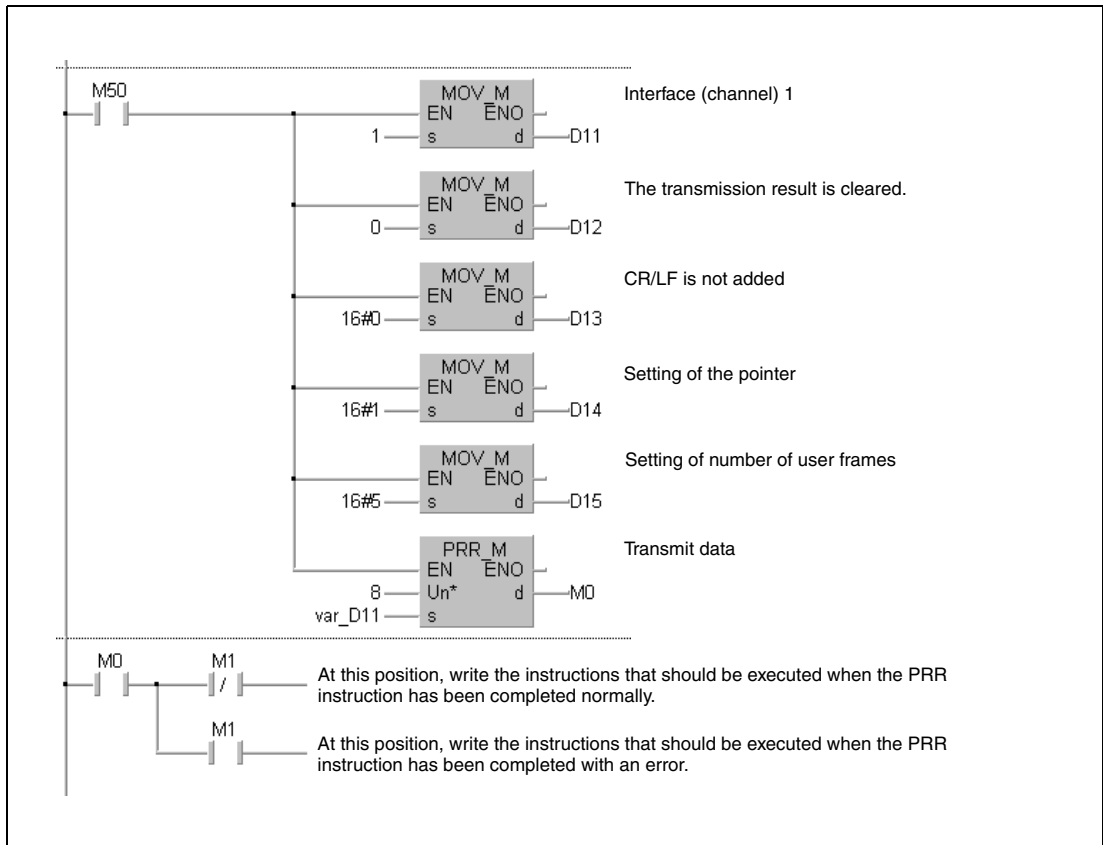
**NOTE**

When using the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- IEC editors  
Ladder Diagram of the GX IEC Developer (part 1)



Ladder Diagram of the GX IEC Developer (continued)

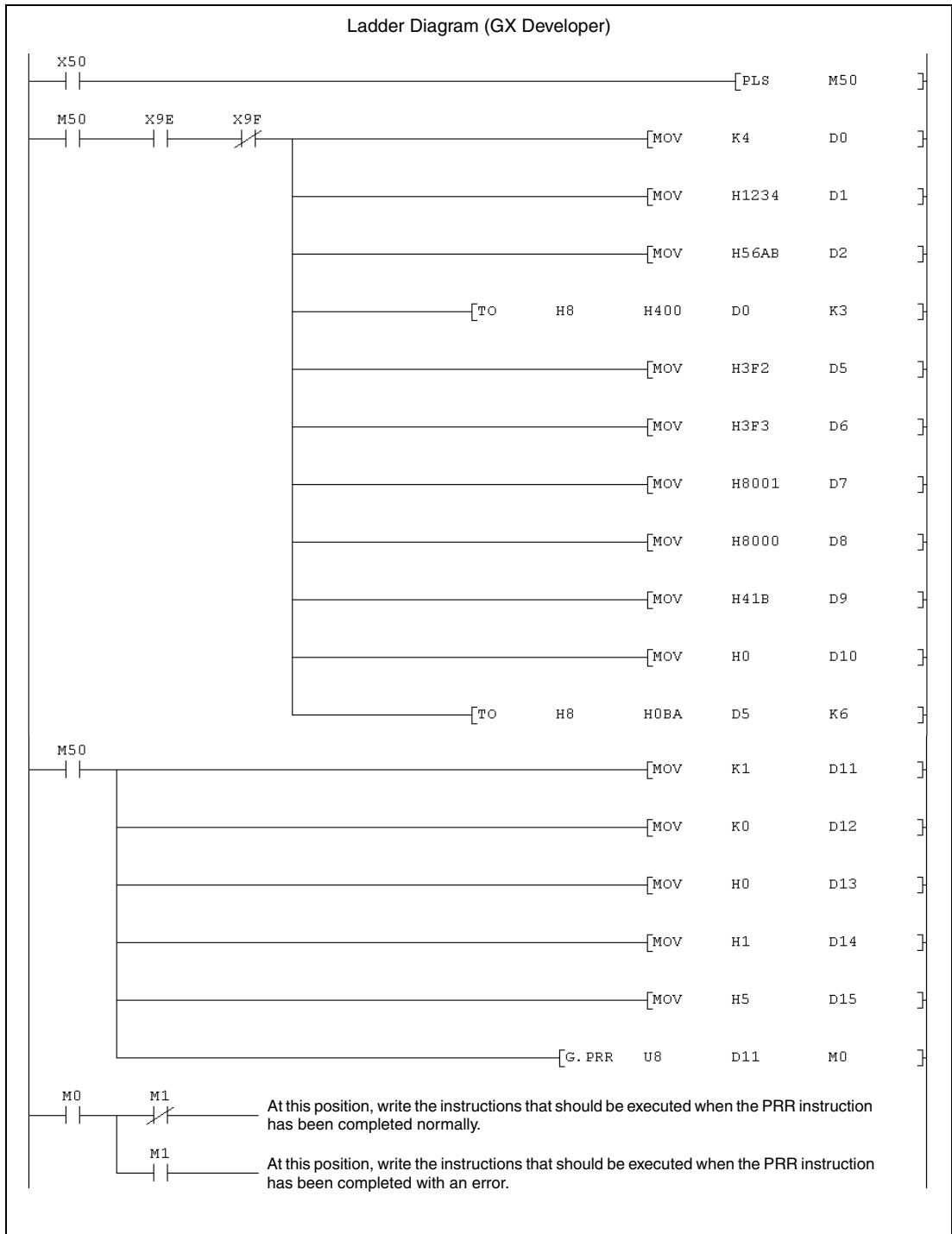


IEC Instruction List

LD	X50		
PLS_M	M50		
-----			
LD	M50		
AND	X9E		
ANDN	X9F		
MOV_M	4,	D0	
MOV_M	16#1234,	D1	
MOV_M	16#56AB,	D2	
TO_M	D0,	16#0008,	16#0400, 3
MOV_M	16#3F2,	D5	
MOV_M	16#3F3,	D6	
MOV_M	16#0001,	D7	
MOV_M	16#0000,	D8	
MOV_M	16#41B,	D9	
MOV_M	16#0,	D10	
TO_M	D5,	16#0008,	16#0BA, 6
-----			
LD	M50		
MOV_M	1,	D11	
MOV_M	0,	D12	
MOV_M	16#0,	D13	
MOV_M	16#1,	D14	
MOV_M	16#5,	D15	
PRR_M	8,	var_D11,	M0
-----			
LD	M0		
ANDN	M1		
An instruction at this position will be executed when the PRR instruction has been completed normally.			
LD	M0		
AND	M1		
An instruction at this position will be executed when the processing of the PRR instruction has been completed with an error.			

For an explanation of the devices and instructions used please see the above ladder diagram.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer.



MELSEC Instruction List					
MELSEC	LD	X50			
	PLS	M50			
MELSEC	LD	M50			
	AND	X9E			
	ANI	X9F			
	MOV	K4	D0		
	MOV	H1234	D1		
	MOV	H56AB	D2		
	TO	H8	H400	D0	K3
	MOV	H3F2	D5		
	MOV	H3F3	D6		
	MOV	H8001	D7		
	MOV	H8000	D8		
	MOV	H41B	D9		
	MOV	H0	D10		
TO	H8	H0BA	D5	K6	
MELSEC	LD	M50			
	MOV	K1	D11		
	MOV	K0	D12		
	MOV	H0	D13		
	MOV	H1	D14		
	MOV	H5	D15		
G.PRR	U8	D11	M0		
MELSEC	LD	M0			
	MPS				
	ANI	M1			
	MPP				
	AND	M1			

An instruction at this position will be executed when the PRR instruction has been completed normally.

At this position, write the instructions that should be executed when the PRR instruction has been completed with an error.

## 11.2 Instructions for PROFIBUS/DP interface modules

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Reading of data from the buffer memory of a PROFIBUS/DP interface module	G.BBLKRD	BBLKRD_M
	GP.BBLKRD	BBLKRDP_M
Writing of data to the buffer memory of a PROFIBUS/DP interface module	G.BBLKWR	BBLKWR_M
	GP.BBLKWR	BBLKWRP_M

**11.2.1 BBLKRD, BBLKRD P**

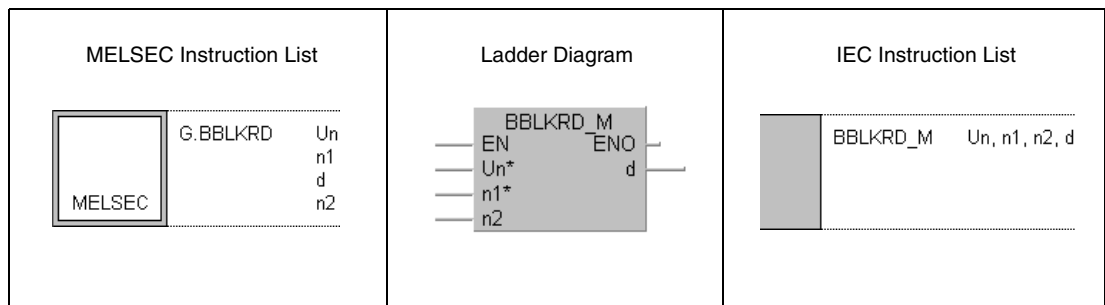
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

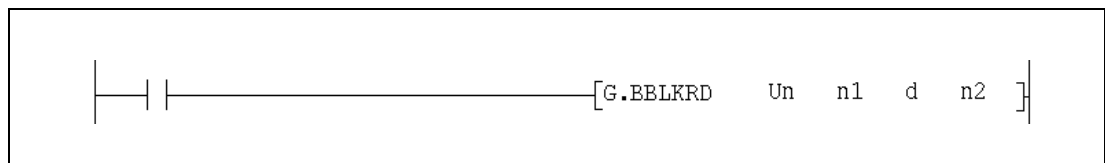
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
n1	—	●	●	—	—	—	—	●	—	SM0	
d	—	●	●	—	—	—	—	—	—		
n2	—	●	●	—	—	—	—	●	—		

**GX IEC Developer**



**GX Developer**



**Variables**

Set Data	Meaning	Data Type
Un	Head I/O number of the PROFIBUS interface module on the base unit	BIN 16-bit
n1	Head address of the buffer memory of the PROFIBUS interface module from where the reading of the data is started.	
d	Head address of the device area in the PLC CPU where the read data is stored	Device name
n2	Number of data to read	BIN 16-bit

**Functions**      **Reading of data from the buffer memory of a PROFIBUS interface module****BBLKRD / BBLKRD**                      **Reading of data**

The BBLKRD instruction is used to read data from the buffer memory of the PROFIBUS interface modules QJ71PB92D and QJ71PB93D. While reading, data separation is prevented.

The QJ71PB93 must be prepared for the BBLKRD instruction by setting of the output signal Y0A. When the PROFIBUS module in turn sets the input signal X0A, the BBLKRD instruction can be executed. The output signal Y0A must be reset when the reading of the buffer memory is completed.

Allowable ranges and designation of the devices:

- Un (Head I/O address of the PROFIBUS interface module): 0 to FF<sub>H</sub>  
(Only the upper two digits of the 3-digit-address are used. E. g. the head address X/Y100 is set as 10<sub>H</sub>.)
- n1 (Head address in the buffer memory): The specified address must be exist.
- d (Head address of the target area): The designated device must be exist.
- n2 (Number of data to read)  
For a QJ71PB92D: 1 to 960 words (1 to 3C0<sub>H</sub>)  
For a QJ71PB93D: 1 to 122 words (1 to 7A<sub>H</sub>)

**NOTES**

*Only a single BBLKRD instruction can be executed in one scan.*

*The BBLKRD and the BBLKWR instruction (chapter 11.2.2) are working independently.*

*The transmission delay time increases when the BBLKRD instruction is used.*

*The BBLKRD instruction is not executed when the output module has not been set in the data module setting in the master station parameter.*

**Operation Error**

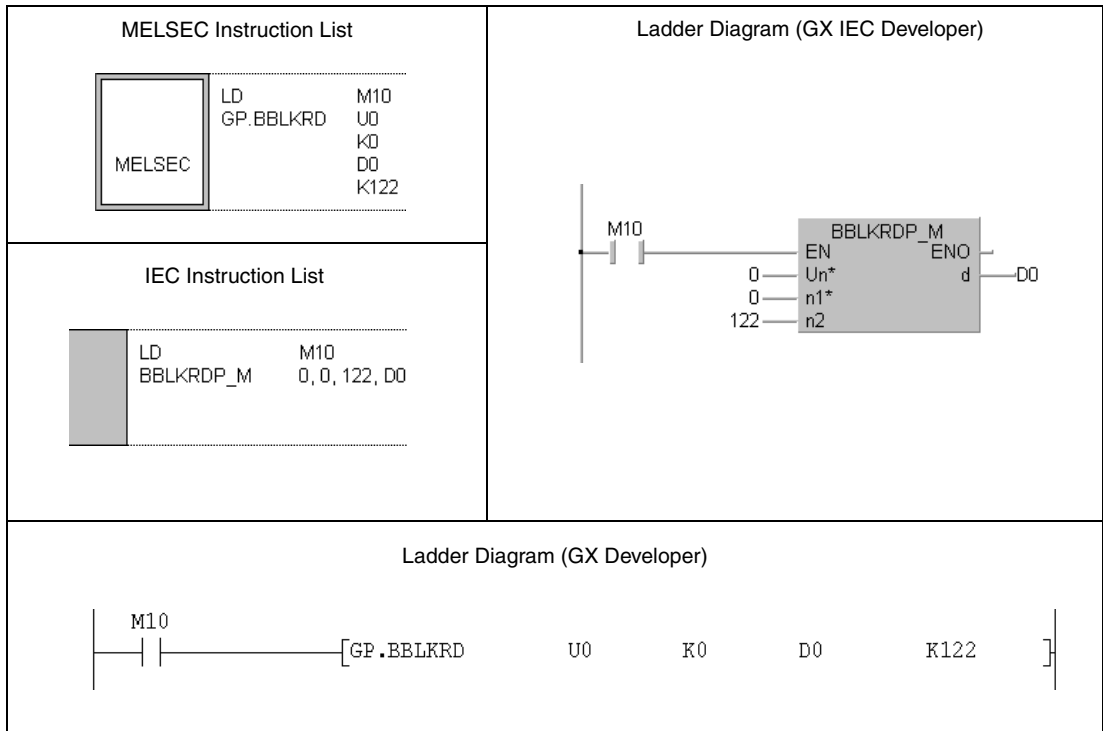
In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When a value that exceeds the specificable range is set for the set data (error code: 4101).
- By the addition of the head address of the buffer memory designated by n1 and the number of data to be read designated by n2 the size of the buffer memory is exceeded (error code: 4101).
- The number of data to be read (designated by n2) is larger than the available device area starting with the head address designated by d (error code: 4101).



**Program Example** BBLKRD

When the relay M10 is set, 122 words of data are read from the buffer memory of the PROFIBUS interface module with the head I/O address X/Y0. The reading is started at the buffer memory address 0 while the storage of the data is started from register D0 onward.



11.2.2 BBLKWR, BBLKWRP

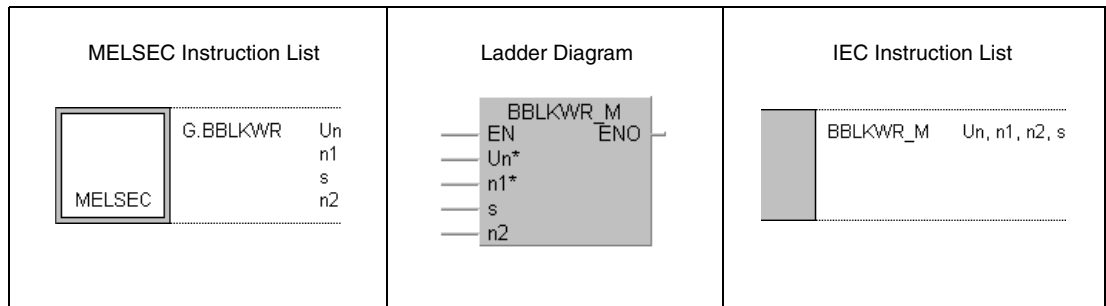
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

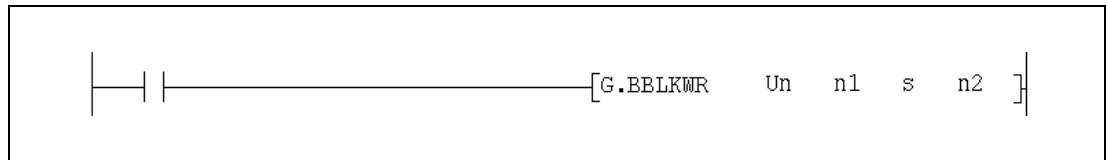
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
n1	—	●	●	—	—	—	—	●	—	SM0	
s	—	●	●	—	—	—	—	—	—		
n2	—	●	●	—	—	—	—	●	—		

GX IEC Developer



GX Developer



Variables

Set Data	Meaning	Data Type
Un	Head I/O number of the PROFIBUS interface module on the base unit	BIN 16-bit
n1	Head address of the buffer memory of the PROFIBUS interface module from where the writing of the data is started.	
s	Head address of the device area in the PLC CPU where the data is stored that is to be send to the PROFIBUS interface module.	Device name
n2	Number of data to be send to the PROFIBUS interface module	BIN 16-bit

**Functions**      **Writing of data to the buffer memory of a PROFIBUS interface module****BBLKWR / BBLKWRP**      **Writing of data**

The BBLKWR instruction writes data to the buffer memory of the PROFIBUS interface modules QJ71PB92D and QJ71PB93D. Data separation is prevented during the write operation.

The QJ71PB93 must be prepared for the BBLKWR instruction by setting of the output signal Y0B. When the PROFIBUS module in turn sets the input signal X0B, the BBLKWR instruction can be executed. After completion of the writing to the buffer memory the output signal Y0B must be reset.

Allowable ranges and designation of the devices:

- Un (Head I/O address of the PROFIBUS interface module): 0 to FF<sub>H</sub>  
(Only the upper two digits of the 3-digit-address are used. E. g. the head address X/Y100 is set as 10<sub>H</sub>.)
- n1 (Head address in the buffer memory): The specified address must be exist.  
  
The head address for the QJ71PB93 has an offset of 100<sub>H</sub>. Thus, 100<sub>H</sub> must be subtracted from the desired head address when designating n1. For example the head address 100<sub>H</sub> is specified as „0<sub>H</sub>“ and the head address 120<sub>H</sub> is specified as „20<sub>H</sub>“.
- d (Head address of the source area): The designated device must be exist.
- n2 (Number of data to write)  
For a QJ71PB92D: 1 to 960 words (1 to 3C0<sub>H</sub>)  
For a QJ71PB93D: 1 to 122 words (1 to 7A<sub>H</sub>)

**NOTES**

*Only a single BBLKWR instruction can be executed in one scan.*

*The BBLKRD and the BBLKWR instruction (chapter 11.2.1) are working independently.*

*The transmission delay time increases when the BBLKWR instruction is used.*

*The BBLKRD instruction is not executed when the input module has not been set in the data module setting in the master station parameter.*

**Operation Error**

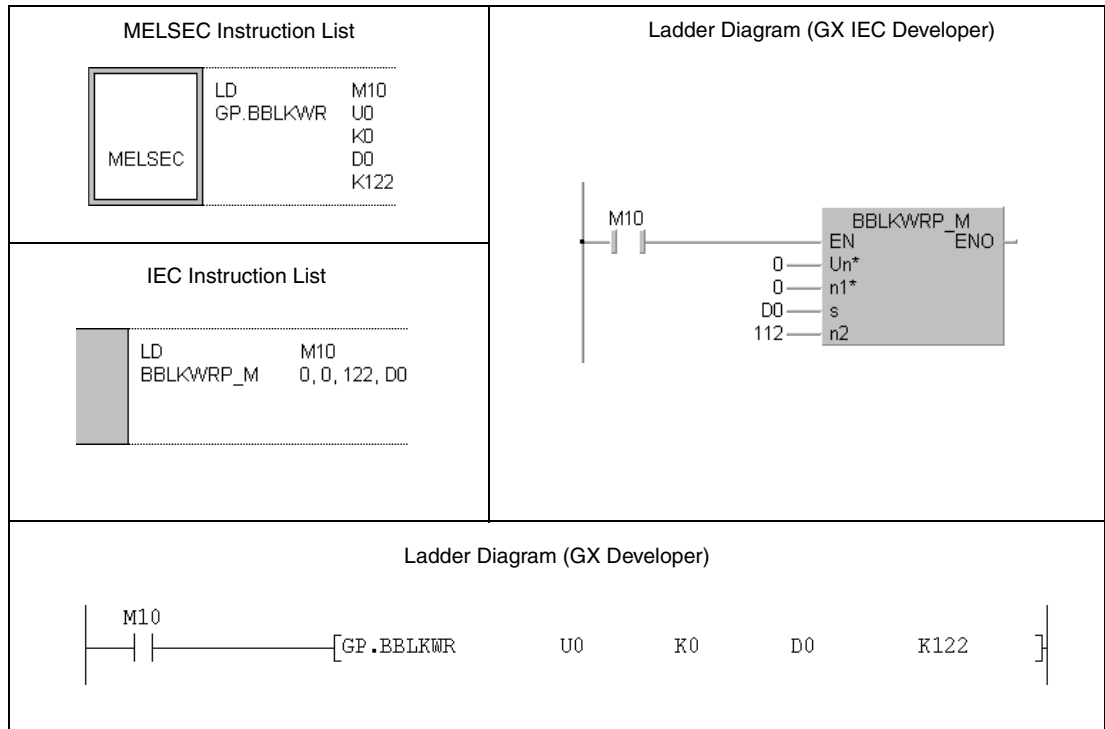
In the following cases an operation error occurs, the error flag SMO is set, and an error code is stored in SDO:

- When a value that exceeds the specificable range is set for the set data. (error code: 4101).
- By the addition of the head address of the buffer memory designated by n1 and the number of data to write (designated by n2) the size of the buffer memory is exceeded (error code: 4101).
- The number of data to be write (designated by n2) is larger than the available device area starting with the head address designated by d (error code: 4101).

**Program Example**

**BBLKWRP**

After the relay M10 is set, the contents of the data registers D0 to D121 (122 words) is written to the input area of the PROFIBUS/DP slave module QJ71PB93D. The input area starts at the buffer memory address 100<sub>H</sub>. Please note that the head address designated by n1 is specified with „0<sub>H</sub>“ in this case. The head I/O number of the PROFIBUS/DP slave module is X/Y0.



## 11.3 Instructions for ETHERNET interface modules

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Reading of received data from fixed buffers	ZP.BUFRVCV	BUFRVCV_M
	Z.BUFRCVS	BUFRCVS_M
Sending of data to fixed buffers	ZP.BUFSND	BUFSND_M
Opening of a connection	ZP.OPEN	OPEN_M
Closing of a connection	ZP.CLOSE	CLOSE_M
Clearing of error information	ZP.ERRCLR	ERRCLR_M
Reading of error information	ZP.ERRRD	ERRRD_M
Reinitialization of a ETHERNET interface module	ZP.UINI	UINI_M

11.3.1 BUFRCV

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

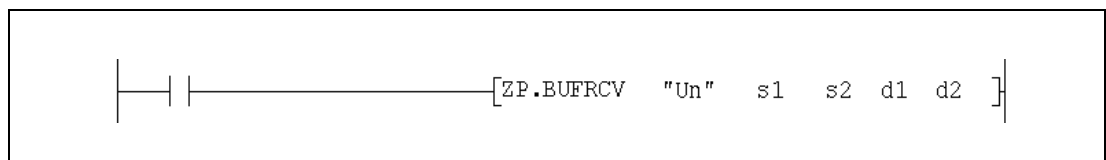
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—		
s2	—	●	●	—	—	—	—	—	—		
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>MELSEC</p> </div> <p>ZP.BUFRCV "Un" s1 s2 d1 d2</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>BUFRCV_M "Un", s1, s2, d1, d2</p> </div>
--	-----------------------	--

GX Developer



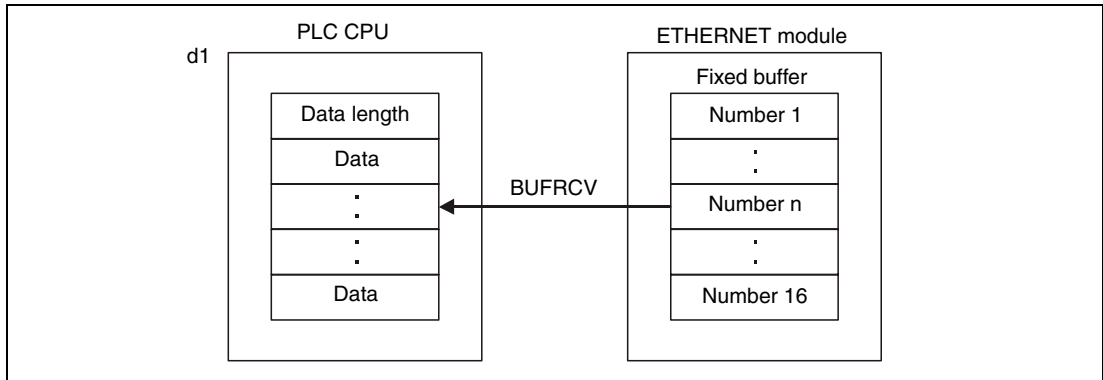
**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Connection number	1 to 16			
s2	Head number of the devices where control data for execution of this instruction is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s2)+0	System area	Used by the system	—	System
(s2)+1	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained either in the user's manual of the ETHERNET interface module or in chapter 13 of this manual.			
d1	Head number of the device area where the received data is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Length of the received data	With procedure (binary data): Number of words read from the fixed buffer	1 to 1017 words	System
			With procedure (ASCII data): Number of words read from the fixed buffer	1 to 508 words	
Without procedure (binary data): Number of bytes read from the fixed buffer			1 to 2016 bytes		
(d1)+1 to (d1)+n	Received data	In this area the data read from the fixed buffer is stored sequentially in ascending order.	—		
d2	Bit device which is set for one scan after completion of the BUFRCV instruction. (d)+1 indicates an abnormal completion of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d2)+0	Instruction completed	Indicates the completion of the BUFRCV instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d2)+1	Instruction completed with error	Indicates the abnormal completion of the BUFRCV instruction ON: Abnormal completion OFF: Normal completion	—		

**Functions**      **Reading of received data from fixed buffer (Execution of the instruction in the main program)**

**BUFRCV Data read**

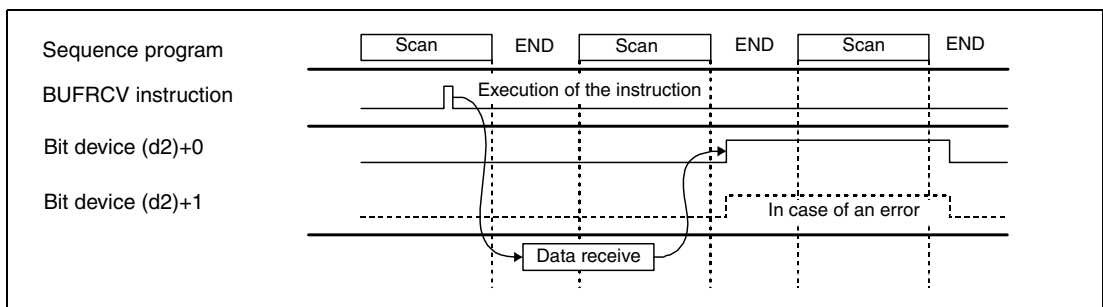
With the BUFRCV instruction, Data sent by an external Station to an ETHERNET interface module via fixed buffer communication can be read from the ETHERNET module and stored in the PLC CPU. The BUFRCV instruction is executed in the main program, whereas the BUFRCVS instruction is used in an interrupt program. Where the data should be stored is specified with d1:



Whether the execution of the BUFRCV instruction has been finished can be checked with the devices (d2)+0 and (d2)+1:

- The bit device (d2)+0 turns ON with the END processing of the scan in which the BUFRCV instruction has been completed and turns OFF at the next END processing.
- The bit device (d2)+1 indicates an error during execution of the BUFRCV instruction. When the instruction is completed normal, this device stays OFF. When an error occurs during the execution of the BUFRCV instruction, (d2)+1 turns ON at the END processing of the scan in which the BUFRCV instruction has been completed and turns OFF at the next END processing.

The timing for the PRR instruction is shown in the following figure:



The BUFRCV instruction can be executed when the ETHERNET interface module indicates that data has been received. One bit is reserved in the buffer memory address 5005<sub>H</sub> for each of the 16 possible connections and is set when data has been received.

**NOTE**      *It is not possible to read received data of the same connection with the BUFRCV instruction in the main programm and the BUFRCVS instruction in an interrupt program.*

**Operation Error**      When the BUFRCV instruction is completed abnormally, the bit device (d2)+1 is set, and an error code is stored in (s2)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please refer to the user's manual of the ETHERNET interface module.



**Program Example**

**BUFRCV**

The following program reads received data from the fixed buffer for connection number 1. The input/output points X/Y0 to X/Y1F are occupied by the ETHERNET module.

- IEC editors (This program example is shown on the next page for the MELSEC instruction list and the ladder diagram of the GX Developer.)

Ladder Diagram (GX IEC Developer)

IEC Instruction List

	LD TRUE		
	MOV_M	U0\G20480, K4M0	
	MOV_M	U0\G20482, K4M20	
	MOV_M	U0\G20485, K4M40	
	LD	X19	
	AND	M0	
	AND	M40	
	PLS_M	M100	
	LD	M100	
	BUFRCV_M	"U0", K1, var_D5000, D500, var_M500	
For an explanation of the devices and instructions used please see the above ladder diagram			
	LD	M500	
	ANDN	M501	
	At this position, write the instructions that should be executed when the BUFRCV instruction has been completed normally.		
	LD	M500	
	AND	M501	
	At this position, write the instructions that should be executed when the execution of the BUFRCV instruction has been resulted in an error.		

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

MELSEC Instruction List

MELSEC	LD SM400
	MOV U0\G20480 K4M0
	MOV U0\G20482 K4M20
	MOV U0\G20485 K4M40
	LD X19
	AND M0
	AND M40
	PLS M100
	LD M100
	ZP.BUFRCV "U0" K1 D5000 D500 M500
MELSEC	LD M500
	ANI M501
	At this position, write the instructions that should be executed when the BUFRCV instruction has been completed normally.
MELSEC	LD M500
	AND M501
	At this position, write the instructions that should be executed when the execution of the BUFRCV instruction has been resulted in an error.

### 11.3.2 BUFRCVS

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

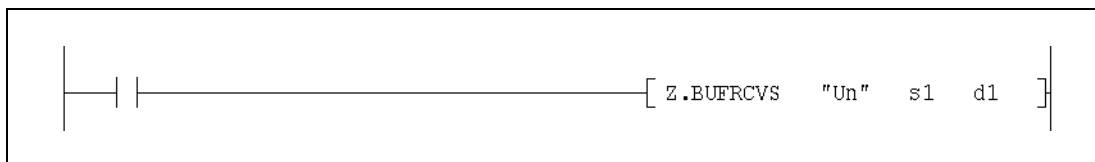
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—	SM0	
d1	—	●	●	—	—	—	—	—	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p>
--------------------------------	-----------------------	-----------------------------

**GX Developer**



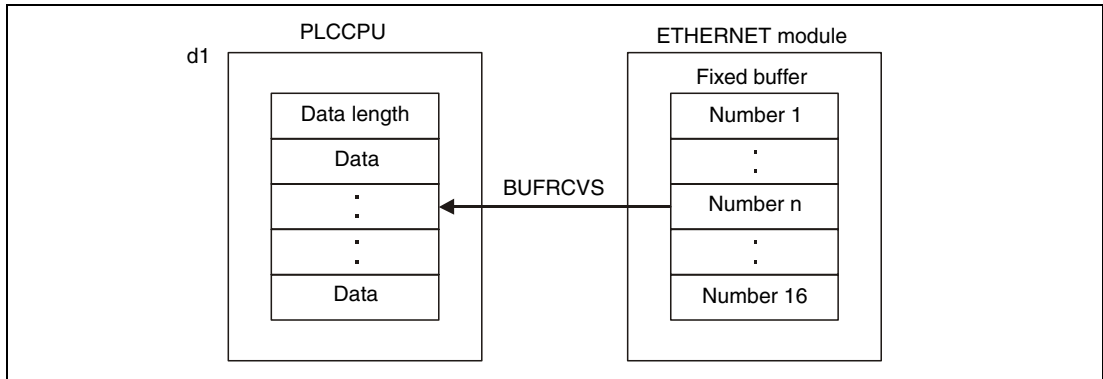
**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Connection number	1 to 16			
d1	Head number of the device area where the received data is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Length of the received data (Number of word or bytes read from the fixed buffer)	With procedure (binary data):)	1 to 1017 words	System
			With procedure (ASCII data):	1 to 508 words	
Without procedure (binary data):			1 to 2016 bytes		
(d1)+1 to (d1)+n	Received data	In this area the data read from the fixed buffer is stored sequentially in ascending order.	—		

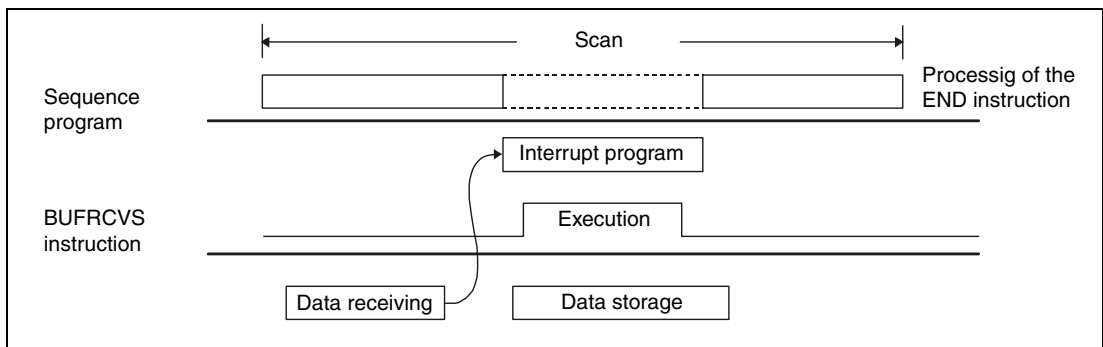
**Functions**     **Reading of received data from fixed buffer (Execution of the instruction in an interrupt program)**

**BUFRCVS     Data read**

With the BUFRCVS instruction, Data sent by an external Station to an ETHERNET interface module via fixed buffer communication can be read from the ETHERNET module and stored in the PLC CPU. The BUFRCVS instruction is executed in an interrupt program, whereas the BUFRCV instruction is used in the main program. Where the data should be stored is specified with d1:



The processing of the BUFRCVS instruction is completed within one scan. The following figure shows the timing of the BUFRCVS instruction:



In order to read receive data with an interrupt program, it is necessary to perform both the interrupt settings and interrupt pointer settings with parameter settings of GX (IEC) Developer.

**NOTES**

*It is not possible to read received data of the same connection with the BUFRCV instruction in the main program and the BUFRCVS instruction in an interrupt program.*

*The BUFRCVS instruction can also be used for a serial communication module QJ71C24 (see chapter 11.1.1).*

**Operation Error**

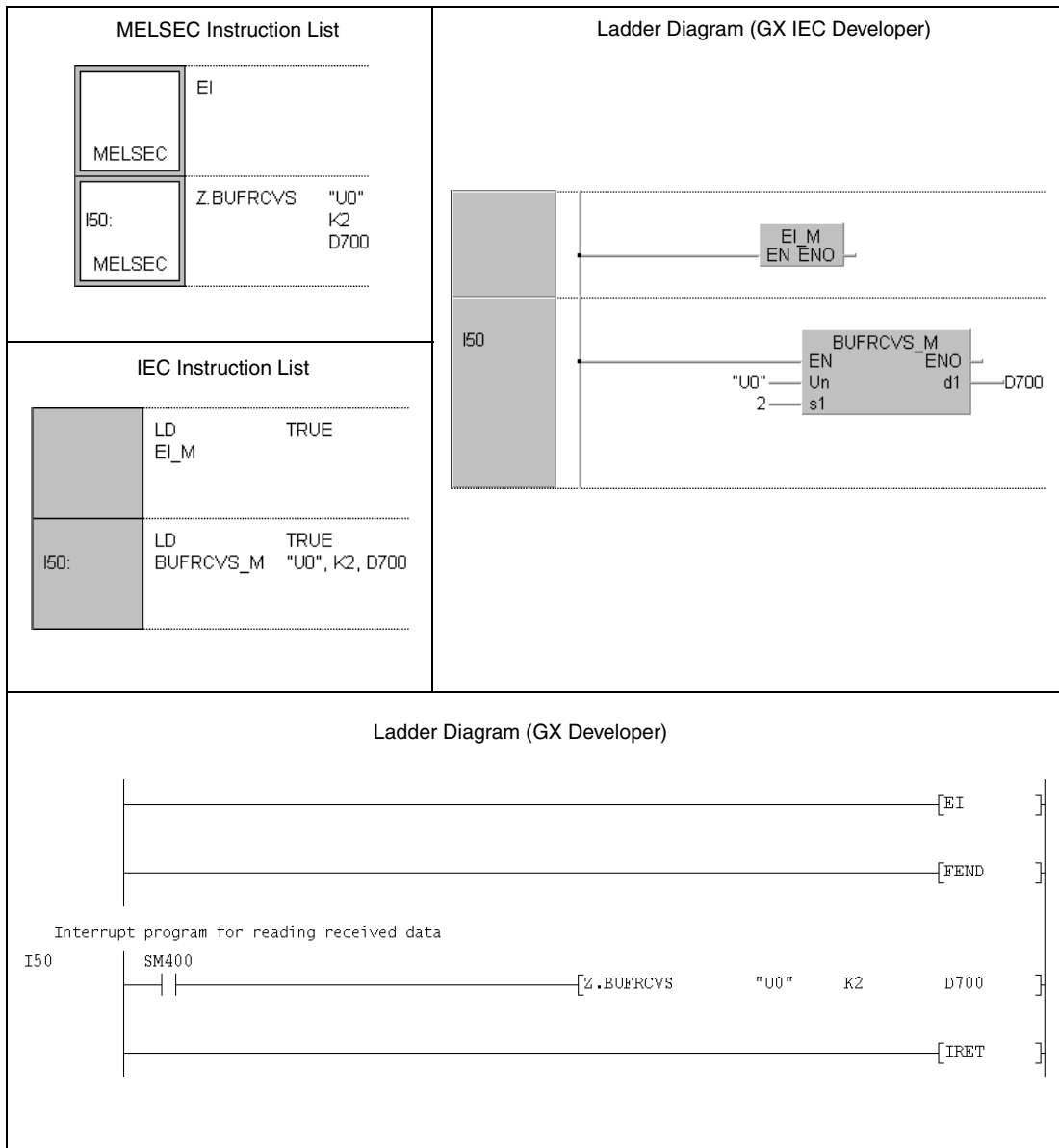
When the BUFRCV instruction is completed abnormally, the error flag SM0 is set, and an error code is stored in SD0. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please refer to the user's manual of the ETHERNET interface module.

**Program Example**

**BUFRCVS**

The following program reads received data from the fixed buffer for connection number 2. The head I/O number of the ETHERNET module is X/Y0.



11.3.3 BUFSND

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

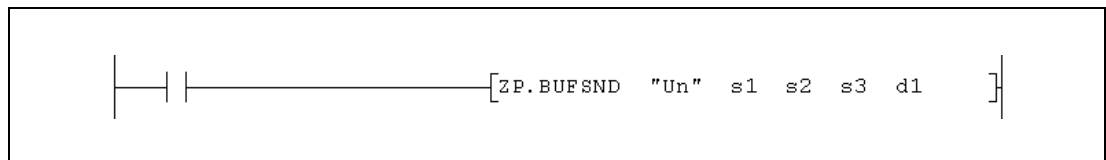
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—		
s2	—	●	●	—	—	—	—	—	—		
s3	—	●	●	—	—	—	—	—	—		
d1	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p style="text-align: center;">MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p style="margin: 0;">MELSEC</p> </div> <table style="margin-left: 20px; border-collapse: collapse;"> <tr><td style="border-right: 1px dotted black; padding-right: 5px;">ZP.BUFSND</td><td style="padding-left: 5px;">"Un"</td></tr> <tr><td style="border-right: 1px dotted black; padding-right: 5px;"></td><td style="padding-left: 5px;">s1</td></tr> <tr><td style="border-right: 1px dotted black; padding-right: 5px;"></td><td style="padding-left: 5px;">s2</td></tr> <tr><td style="border-right: 1px dotted black; padding-right: 5px;"></td><td style="padding-left: 5px;">s3</td></tr> <tr><td style="border-right: 1px dotted black; padding-right: 5px;"></td><td style="padding-left: 5px;">d1</td></tr> </table>	ZP.BUFSND	"Un"		s1		s2		s3		d1	<p>Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table style="margin-left: 20px; border-collapse: collapse;"> <tr><td style="border-right: 1px dotted black; padding-right: 5px;">BUFSND_M</td><td style="padding-left: 5px;">"Un", s1, s3, s2, d1</td></tr> </table>	BUFSND_M	"Un", s1, s3, s2, d1
ZP.BUFSND	"Un"													
	s1													
	s2													
	s3													
	d1													
BUFSND_M	"Un", s1, s3, s2, d1													

GX Developer



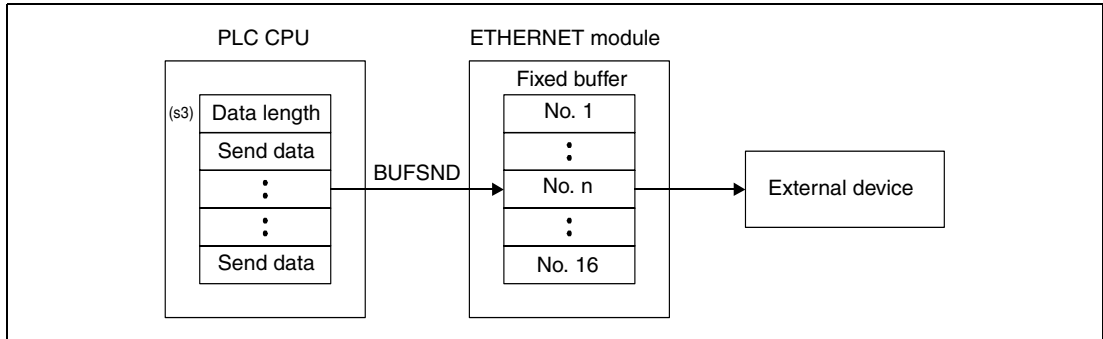
**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Connection number	1 to 16			
s2	Head number of the devices where control data for execution of this instruction is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s2)+0	System area	Used by the system	—	System
(s2)+1	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained either in the user's manual of the ETHERNET interface module or in chapter 13 of this manual.			
s3	Head number of the devices where the send data is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s3)+0	Length of the data to be send	Designation of the amount of data that is to be transferred to the fixed buffer when a procedure (binary data) is used for communication.	1 to 1017 words	User
			Designation of the amount of data that is to be transferred to the fixed buffer when a procedure (ASCII data) is used for communication.	1 to 508 words	
			Designation of the amount of data that is to be transferred to the fixed buffer when a non procedure protokoll (binary data) is used for communication.	1 to 2046 bytes	
(s3)+1 to (s3)+n	Data to be send	The data stored in this are is send to the ETHERNET module.	—		
d1	Bit device which is set for one scan after completion of the BUFSND instruction. (d)+1 indicates an abnormal completion of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Instruction completed	Indicates the completion of the BUFSND instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d1)+1	Instruction completed with error	Indicates the abnormal completion of the BUFSND instruction ON: Abnormal completion OFF: Normal completion	—		

**Functions**     **Sending of data to fixed buffer**

**BUFSND Data send**

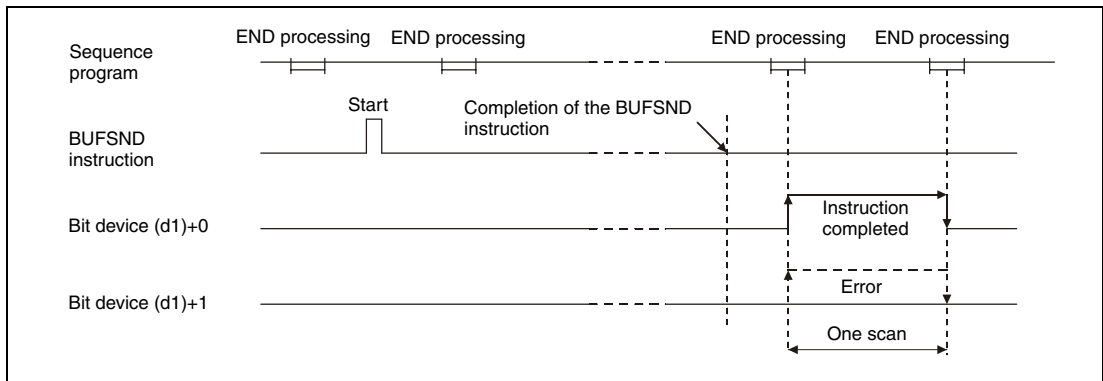
Data which is to be send through fixed buffer communication to an external device connected to an ETHERNET interface module is send to this module by the BUFSND instruction in advance. The data is stored in the PLC CPU from the device designated by (s3)+1 onward:



Whether the execution of the BUFSND instruction has been finished can be checked with the devices (d)+0 and (d)+1:

- The bit device (d)+0 turns ON at the END processing of the scan in which the BUFSND instruction has been completed and turns OFF at the next END processing.
- The bit device (d)+1 indicates an error during execution of the BUFSND instruction. When the instruction is completed normal, this device stays OFF. When an error occurs during execution of the BUFSND instruction, (d)+1 turns ON at the END processing of the scan in which the BUFSND instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the BUFSND instruction is being executed:



The BUFSND instruction is executed when the command for this instruction switches from off to on.

**Operation Error**

When the BUFRVC instruction is completed abnormally, the bit device (d1)+1 is set, and an error code is stored in (s2)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please refer to the user's manual of the ETHERNET interface module.



**Program Example**

**BUFSND**

The following program writes data to the fixed buffer for connection 1. The head I/O number of the ETHERNET module is X/Y0.

- IEC editors (On the next page the same program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

**Ladder Diagram (GX IEC Developer)**

**PLS\_M**  
EN ENO  
d M3000  
Pulse forming (X19 = 1: Start up of the module has been completed successfully; M0 = 1: Opening of connection 1 completed)

**MOV\_M**  
EN ENO  
s d D300  
K6  
6 words will be send

**MOV\_M**  
EN ENO  
s d D301  
K1234  
Registration of send data

**MOV\_M**  
EN ENO  
s d D302  
K5678

**MOV\_M**  
EN ENO  
s d D303  
K8901

**BUFSND\_M**  
EN ENO  
Un s2 var\_D3000  
1 s1 d1 var\_M300  
D300 s3  
The data is send to the buffer memory of the ETHERNET module.

**M300** / **M301**  
At this position, write the instructions that should be executed when the BUFSND instruction has been completed normally.

**M301**  
At this position, write the instructions that should be processed when the execution of the BUFSND instruction has been resulted in an error.

---

**IEC Instruction List**

LD	M600	
AND	X19	
AND	M0	
PLS_M	M3000	
LD	M3000	
MOV_M	K6, D300	
MOV_M	K1234, D301	
MOV_M	K5678, D302	
MOV_M	K8901, D303	
BUFSND_M	"U0", K1, D300, var_D3000, var_M300	

TFor an explanation of the devices and instructions used please see the above ladder diagram.

---

LD	M300	
ANDN	M301	

At this position, write the instructions that should be executed when the BUFSND instruction has been completed normally.

---

LD	M300	
AND	M301	

At this position, write the instructions that should be processed when the execution of the BUFSND instruction has been resulted in an error.

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

MELSEC Instruction List

MELSEC	LD M6000							
	AND X19							
	AND M0							
	PLS M3000							
	LD M3000							
	MOV K6	D300						
	MOV K1234	D301						
	MOV K5678	D302						
	MOV K8901	D303						
	ZP.BUFSND "U0"	K1	D3000	D300	M300			
.....								
MELSEC	LD M300							
	ANI M301							
	At this position, write the instructions that should be executed when the BUFSND instruction has been completed normally.							
.....								
MELSEC	LD M300							
	AND M301							
	At this position, write the instructions that should be processed when the execution of the BUFSND instruction has been resulted in an error.							
.....								

### 11.3.4 OPEN

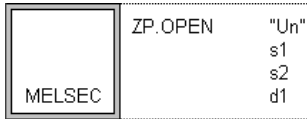
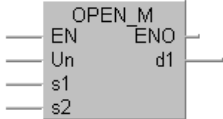
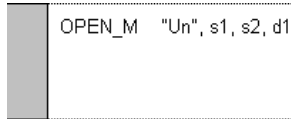
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

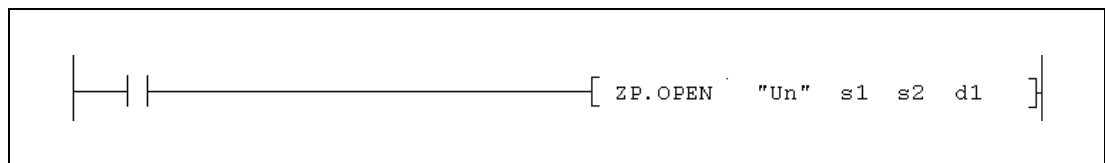
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—		
s2	—	●	●	—	—	—	—	—	—		
d1	●	●	●	—	—	—	—	—	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	---	--

**GX Developer**



**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit
s1	Connection number	1 to 16		

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type
Head number of the devices where control data for the execution of this instruction is stored.				
Set Data	Meaning	Description	Range	Contents is stored by
(s2)+0	Source of the parameter settings	Designate which settings are used to open the connection: <ul style="list-style-type: none"> <li>• 0000<sub>H</sub>: The connection will be opened with the settings made in GX (IEC) Developer.</li> <li>• 8000<sub>H</sub>: The connection will be opened with the settings stored in the devices (s2)+2 to (s2)+9.</li> </ul>	0000 <sub>H</sub> or 8000 <sub>H</sub>	User
(s2)+1	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained either in the user's manual of the ETHERNET interface module or in chapter 13 of this manual.	—	System
s2	Application setting area	The bits of this device are used to make settings for the connection: <ul style="list-style-type: none"> <li>• Bit 0: Usage of fixed buffers                              0: The buffer is used for transmission, fixed buffer communication is not executed                              1: The buffer is used for receiving</li> <li>• Bit 1: Destination existence confirmation                              0: No confirm                              1: Confirm</li> <li>• Bit 7: Pairing open setting                              0: No pairs                              1: Pairs</li> <li>• Bit 8: Communication method (protocol)                              0: TCP/IP                              1: UDP/IP</li> <li>• Bit 9: Fixed buffer communication                              0: With procedure                              1: Without procedure</li> <li>• Bit 13: Issue an interrupt when receiving fixed buffer                              0: No interrupt                              1: An Interrupt is issued</li> <li>• Bits 14 und 15: Active or passive opening                              Bit 15/14 = 00: Active open or UDP/IP                              Bit 15/14 = 10: Unpassive open                              Bit 15/14 = 11: Full passive open</li> </ul>	Please see the description on the left.	User
				BIN 16-bit

Variables

Set Data	Meaning			Range	Contents is stored by	Data Type
	Set Data	Meaning	Description			
s2	(s2)+3	Port No. of the ETHERNET module	Designate the port No. of the ETHERNET interface module.	408 <sub>H</sub> to 1388 <sub>H</sub> 138B <sub>H</sub> to FFFE <sub>H</sub>	User	BIN 16-bit
	(s2)+4 (s2)+5	Destination IP address	IP address of the external device to communicate with. When the IP address FFFFFFFF <sub>H</sub> is set, data is exchanged with simultaneous broadcast.	1 <sub>H</sub> to FFFFFFF <sub>H</sub>		
	(s2)+6	Destination Port No.	Port No. of the external device to communicate with. (FFFF <sub>H</sub> = Simultaneous broadcast)	401 <sub>H</sub> to FFFF <sub>H</sub>		
	(s2)+7 to (s2)+9	Destination ETHERNET address	When the external device supports the ARP function set either 000000000000 <sub>H</sub> or FFFFFFFF <sub>H</sub> . When the external device support does not support the ARP function set the destination ETHERNET address.	Please see the description on the left.		
	Bit device which is set for one scan after completion of the OPEN instruction. (d)+1 indicates an abnormal completion of the instruction.					
d1	Set Data	Meaning	Description	Range	Contents is stored by	
	(d1)+0	Instruction completed	Indicates the completion of the OPEN instruction ON: Instruction completed OFF: Instruction not completed	—	System	Bit
	(d1)+1	Instruction completed with error	Indicates the abnormal completion of the OPEN instruction ON: Abnormal completion OFF: Normal completion	—		

**Functions**      **Opening of a connection**

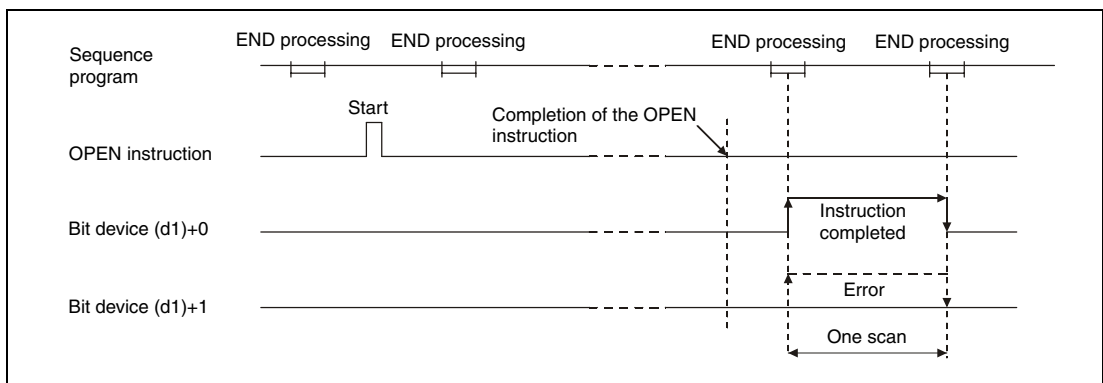
**OPEN      Open connection**

This instruction performs the open processing for a connection specified by s1 for the module designated by Un.

Whether the execution of the OPEN instruction has been finished can be checked with the devices (d1)+0 and (d1)+1:

- The bit device (d1)+0 turns ON at the END processing of the scan in which the OPEN instruction has been completed and turns OFF at the next END processing.
- The bit device (d1)+1 indicates an error during execution of the OPEN instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the OPEN instruction, (d1)+1 turns ON at the END processing of the scan in which the OPEN instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the OPEN instruction is being executed:



The OPEN instruction is executed when the command for this instruction switches from off to on.

**NOTE**      *Never execute the open/close processing using input/output signals and the OPEN or CLOSE dedicated instructions simultaneously for the same connection. It will result in malfunctions.*

**Operation Error**      When an error occurs during the processing of the OPEN instruction, the bit device (d1)+1 is set, and an error code is stored in (s2)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please refer to the user's manual of the ETHERNET interface module.

**Program Example**

OPEN

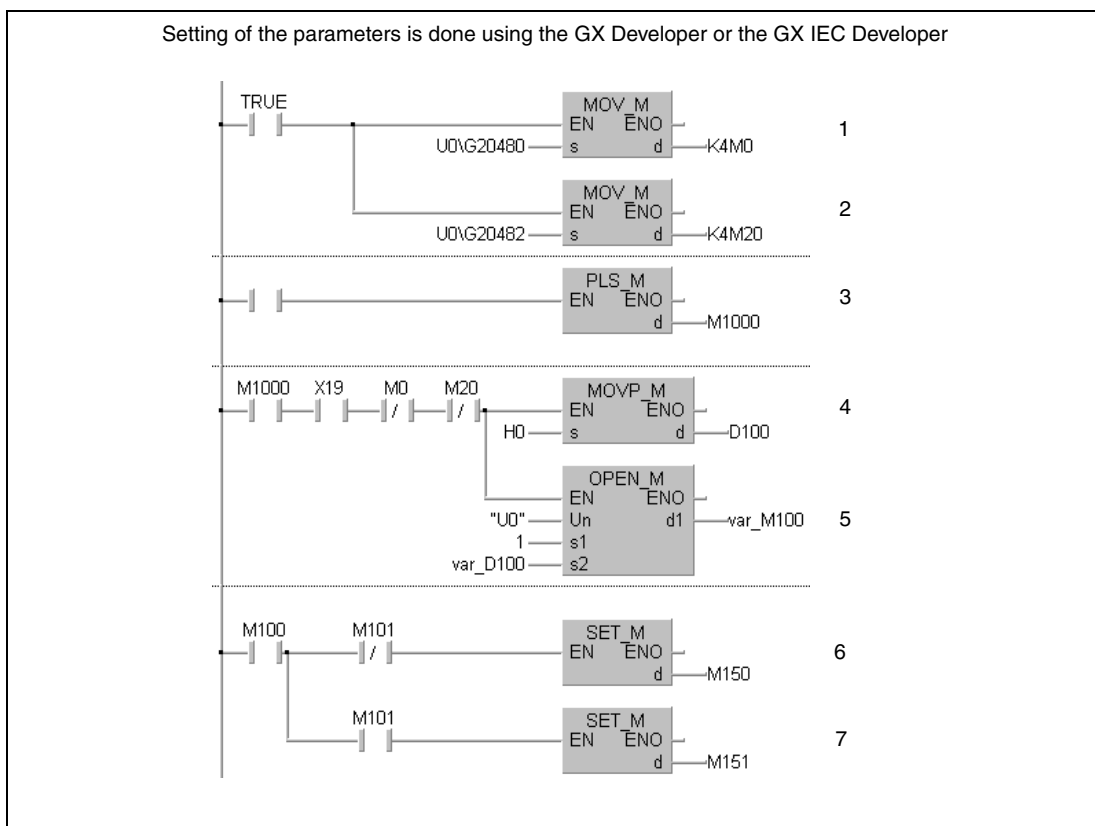
The following program active opens the connection number 1 for TCP/IP communication. The head I/O address of the is X/Y0.

**NOTE**

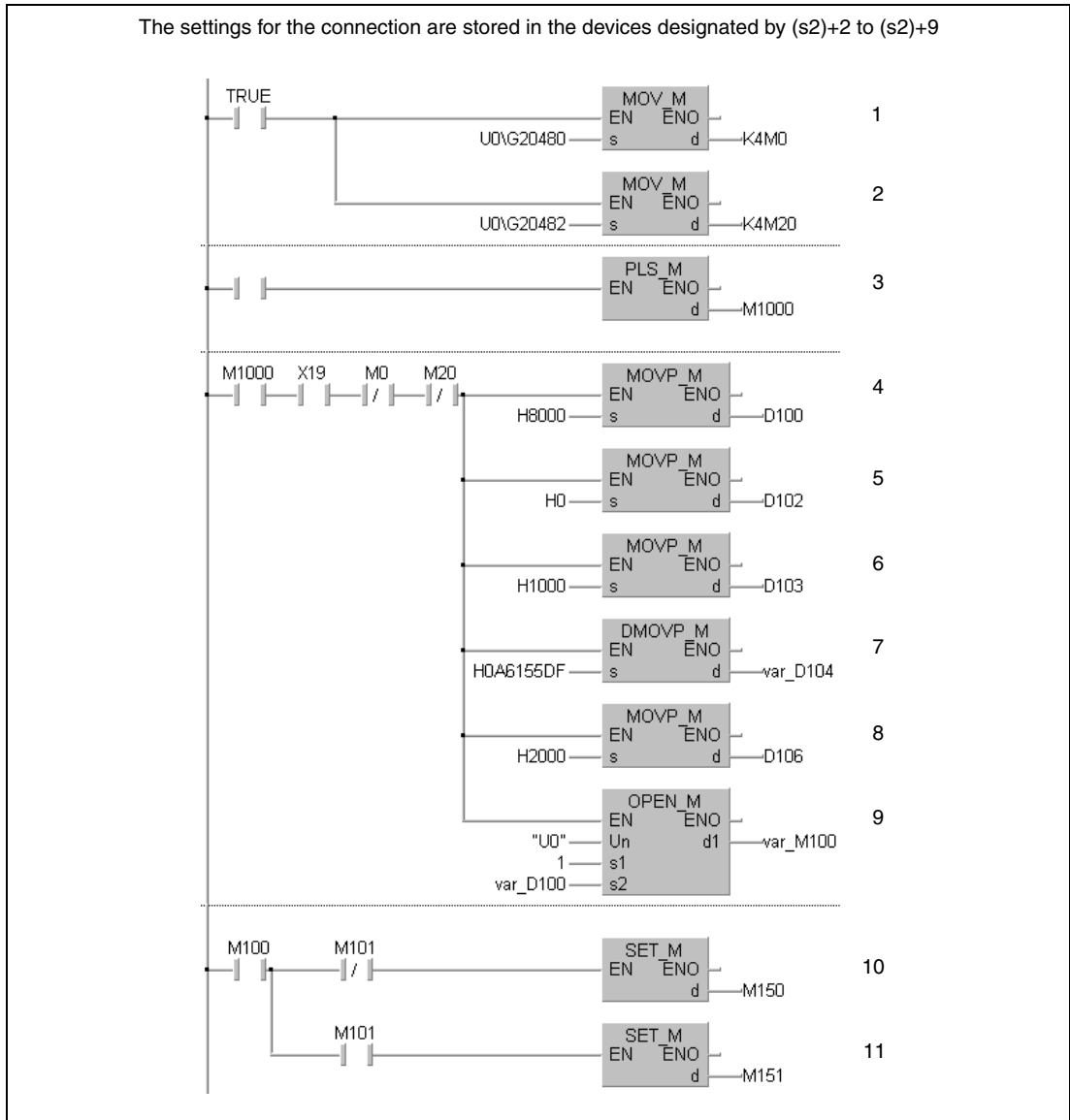
For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

● Ladder Diagram (GX IEC Developer)

For the following example it is necessary to set the parameters with the GX (IEC) Developer in advance. Another example where the settings are made with the OPEN instruction is shown on the next page.



- 1 Reading of the connection status (M0 = 1: Opening of connection 1 has been completed)
- 2 Reading of the open request (M20 = 1: Opening of connection 1 is requested)
- 3 The signal to open the connection is converted to a pulse.
- 4 The source for the parameters is set (0000<sub>H</sub> = External setting).
- 5 Opening of connection 1
- 6 M150 is set when the opening of the connection has been completed without an error.
- 7 M151 is set when an error has occurred during the opening of the connection.



- 1 Reading of the connection status (M0 = 1: Opening of connection 1 has been completed)
- 2 Reading of the open request (M20 = 1: Opening of connection 1 is requested)
- 3 The signal to open the connection is converted to a pulse.
- 4 The source for the parameters is set (8000<sub>H</sub> = Parameters are stored in (s2)+2 to (s2)+9))
- 5 The application setting is stored in (s2)+2.
- 6 The port No. of the ETHERNET module is written to (s2)+3.
- 7 The IP address (10.97.85.223) of the external device is stored in (s2)+4 and (s2)+5.
- 8 In (s2)+6 the port No. of the external device is stored.
- 9 Opening of connection 1
- 10 M150 is set when the opening of the connection has been completed without an error.
- 11 M151 is set when an error has occurred during the opening of the connection.



● IEC Instruction List

Setting of the parameters is done using the GX Developer or the GX IEC Developer

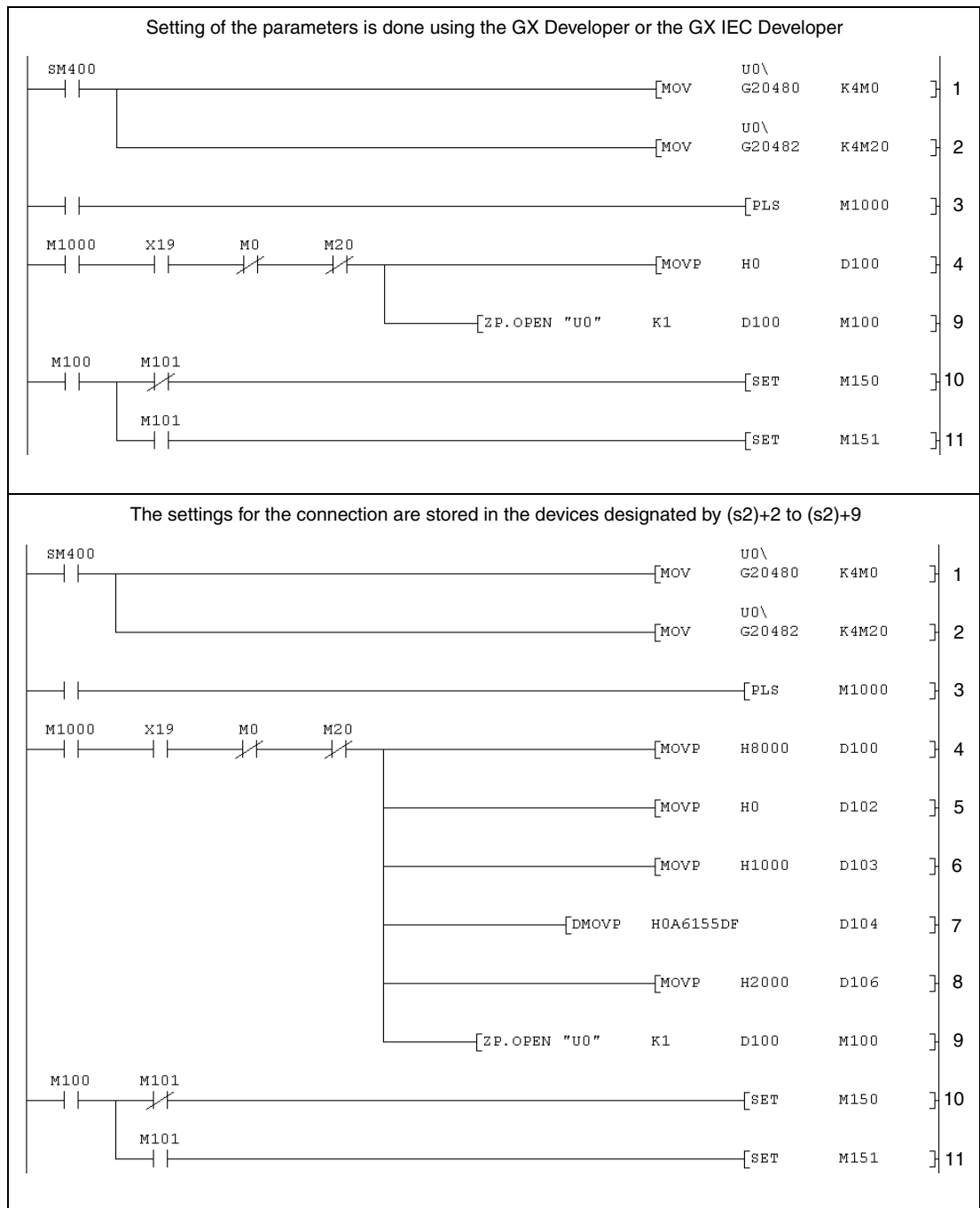
	LD	TRUE		
	MOV_M	U0\G20480, K4M0	_____	1
	MOV_M	U0\G20482, K4M20	_____	2
	LD	M123		
	PLS_M	M1000	_____	3
	LD	M1000		
	AND	X19		
	ANDN	M0		
	ANDN	M20		
	MOVP_M	H0, D100	_____	4
	OPEN_M	"U0", K1, var_D100, var_M100	_____	9
<hr/>				
	LD	M100		
	ANDN	M101		
	SET_M	M150	_____	10
	LD	M100		
	AND	M101		
	SET_M	M151	_____	11

The settings for the connection are stored in the devices designated by (s2)+2 to (s2)+9

	LD	TRUE		
	MOV_M	U0\G20480, K4M0	_____	1
	MOV_M	U0\G20482, K4M20	_____	2
	LD	M123		
	PLS_M	M1000	_____	3
	LD	M1000		
	AND	X19		
	ANDN	M0		
	ANDN	M20		
	MOVP_M	H8000, D100	_____	4
	MOVP_M	H0, D102	_____	5
	MOVP_M	H1000, D103	_____	6
	DMOVP_M	H0A6155DF, var_D104	_____	7
	MOVP_M	H2000, D106	_____	8
	OPEN_M	"U0", K1, var_D100, var_M100	_____	9
<hr/>				
	LD	M100		
	ANDN	M101		
	SET_M	M150	_____	10
	LD	M100		
	AND	M101		
	SET_M	M151	_____	11

- 1 Reading of the connection status (M0 = 1: Opening of connection 1 has been completed)
- 2 Reading of the open request (M20 = 1: Opening of connection 1 is requested)
- 3 The signal to open the connection is converted to a pulse.
- 4 The source for the parameters is set (0000<sub>H</sub> = External, 8000<sub>H</sub> = Devices (s2)+2 to(s2)+9)
- 5 The application setting is stored in (s2)+2.
- 6 The port No. of the ETHERNET module is written to (s2)+3
- 7 The IP address (10.97.85.223) of the external device is stored in (s2)+4 and (s2)+5.
- 8 In (s2)+6 the port No. of the external device is stored.
- 9 Opening of connection 1
- 10 M150 is set when the opening of the connection has been completed without an error.
- 11 M151 is set when an error has occurred during the opening of the connection.

● Ladder Diagram (GX Developer)



- 1 Reading of the connection status (M0 = 1: Opening of connection 1 has been completed)
- 2 Reading of the open request (M20 = 1: Opening of connection 1 is requested)
- 3 The signal to open the connection is converted to a pulse.
- 4 The source for the parameters is set (0000<sub>H</sub> = External, 8000<sub>H</sub> = Devices (s2)+2 to (s2)+9)
- 5 The application setting is stored in (s2)+2.
- 6 The port No. of the ETHERNET module is written to (s2)+3
- 7 The IP address (10.97.85.223) of the external device is stored in (s2)+4 and (s2)+5.
- 8 In (s2)+6 the port No. of the external device is stored.
- 9 Opening of connection 1
- 10 M150 is set when the opening of the connection has been completed without an error.
- 11 M151 is set when an error has occurred during the opening of the connection.

● MELSEC Instruction List

Setting of the parameters is done using the GX Developer or the GX IEC Developer

MELSEC	LD	SM400			
	MOV	UDVG20480	K4M0	_____	1
	MOV	UDVG20482	K4M20	_____	2
	LD	M123			
	PLS	M1000		_____	3
	LD	M1000			
	AND	X19			
	ANI	M0			
	ANI	M20			
	MOVP	H0	D100	_____	4
	ZP.OPEN	"U0"	K1	D100 M100	9
MELSEC	LD	M100			
	ANI	M101			
	SET	M150		_____	10
MELSEC	LD	M100			
	AND	M101			
	SET	M151		_____	11

The settings for the connection are stored in the devices designated by (s2)+2 to (s2)+9

MELSEC	LD	SM400			
	MOV	UDVG20480	K4M0	_____	1
	MOV	UDVG20482	K4M20	_____	2
	LD	M123			
	PLS	M1000		_____	3
	LD	M1000			
	AND	X19			
	ANI	M0			
	ANI	M20			
	MOVP	H8000	D100	_____	4
	MOVP	H0	D102	_____	5
	MOVP	H1000	D103	_____	6
	DMOVP	HA6155DF	D104	_____	7
	MOVP	H2000	D106	_____	8
	ZP.OPEN	"U0"	K1	D100 M100	9
MELSEC	LD	M100			
	ANI	M101			
	SET	M150		_____	10
MELSEC	LD	M100			
	AND	M101			
	SET	M151		_____	11

- 1 Reading of the connection status (M0 = 1: Opening of connection 1 has been completed)
- 2 Reading of the open request (M20 = 1: Opening of connection 1 is requested)
- 3 The signal to open the connection is converted to a pulse.
- 4 The source for the parameters is set (0000<sub>H</sub> = External, 8000<sub>H</sub> = Devices (s2)+2 to(s2)+9)
- 5 The application setting is stored in (s2)+2.
- 6 The port No. of the ETHERNET module is written to (s2)+3
- 7 The IP address (10.97.85.223) of the external device is stored in (s2)+4 and (s2)+5.
- 8 In (s2)+6 the port No. of the external device is stored.
- 9 Opening of connection 1
- 10 M150 is set when the opening of the connection has been completed without an error.
- 11 M151 is set when an error has occurred during the opening of the connection.

11.3.5 CLOSE

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

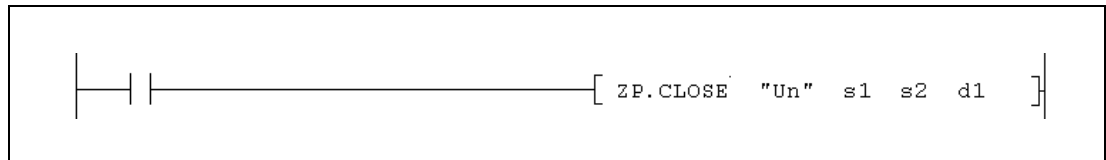
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	●	—		
s2	—	●	●	—	—	—	—	—	—		
d1	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <table style="border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">MELSEC</td> <td style="padding: 2px;">ZP.CLOSE</td> <td style="padding: 2px;">"Un" s1 s2 d1</td> </tr> </table> </div>	MELSEC	ZP.CLOSE	"Un" s1 s2 d1	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <table style="border-collapse: collapse;"> <tr> <td style="width: 20px; height: 20px; background-color: #cccccc;"></td> <td style="padding: 2px;">CLOSE_M</td> <td style="padding: 2px;">"Un", s1, s2, d1</td> </tr> </table> </div>		CLOSE_M	"Un", s1, s2, d1
MELSEC	ZP.CLOSE	"Un" s1 s2 d1						
	CLOSE_M	"Un", s1, s2, d1						

GX Developer



**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Number of the connection	1 to 16			
s2	Head number of the devices where control data for the execution of this instruction is stored				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s2)+0	System area	Used by the system	—	System
(s2)+1	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained either in the user's manual of the ETHERNET interface module or in chapter 13 of this manual.			
d1	Bit device which is set for one scan after completion of the CLOSE instruction. (d)+1 indicates an abnormal completion of the instruction				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Instruction completed	Indicates the completion of the CLOSE instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d1)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the CLOSE instruction ON: Abnormal completion OFF: Normal completion	—		

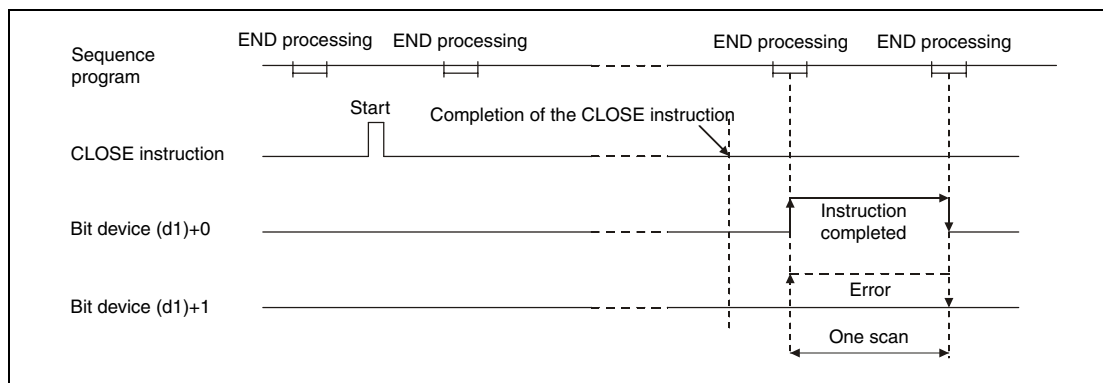
**Functions**    **Closing of a connection****CLOSE**    **Close connection**

This instruction closes the connection specified by s1 for the module designated by Un (disconnecting connections).

Whether the execution of the CLOSE instruction has been finished can be checked with the devices (d1)+0 and (d1)+1:

- The bit device (d1)+0 turns ON at the END processing of the scan in which the CLOSE instruction has been completed and turns OFF at the next END processing.
- The bit device (d1)+1 indicates an error during execution of the CLOSE instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the CLOSE instruction, (d1)+1 turns ON at the END processing of the scan in which the CLOSE instruction has been completed and turns OFF at the next END processing.

The timing for the CLOSE instruction is shown in the following figure:



The CLOSE instruction is executed when the command for this instruction switches from off to on.

**NOTE**

*Never execute the open/close processing using input/output signals and the OPEN or CLOSE dedicated instructions simultaneously for the same connection. It will result in malfunctions.*

**Operation Error**

When an error occurs during the processing of the CLOSE instruction, the bit device (d1)+1 is set, and an error code is stored in (s2)+1. For more information about the error codes please refer to the following manuals:

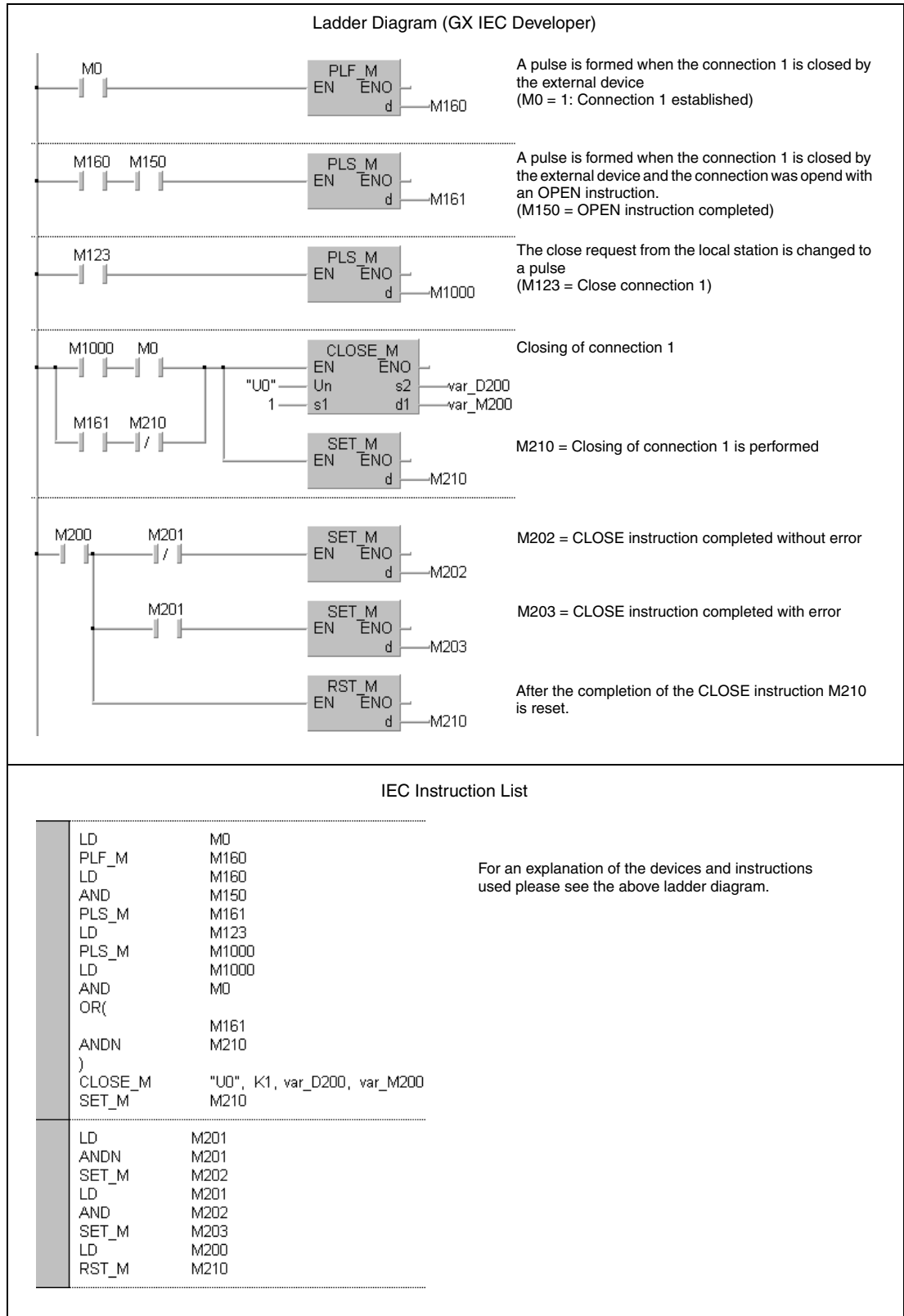
- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please refer to the user's manual of the ETHERNET interface module.

**Program Example**

**CLOSE**

The following program closes the connection number 1 of the ETHERNET module with the head I/O address X/Y0.

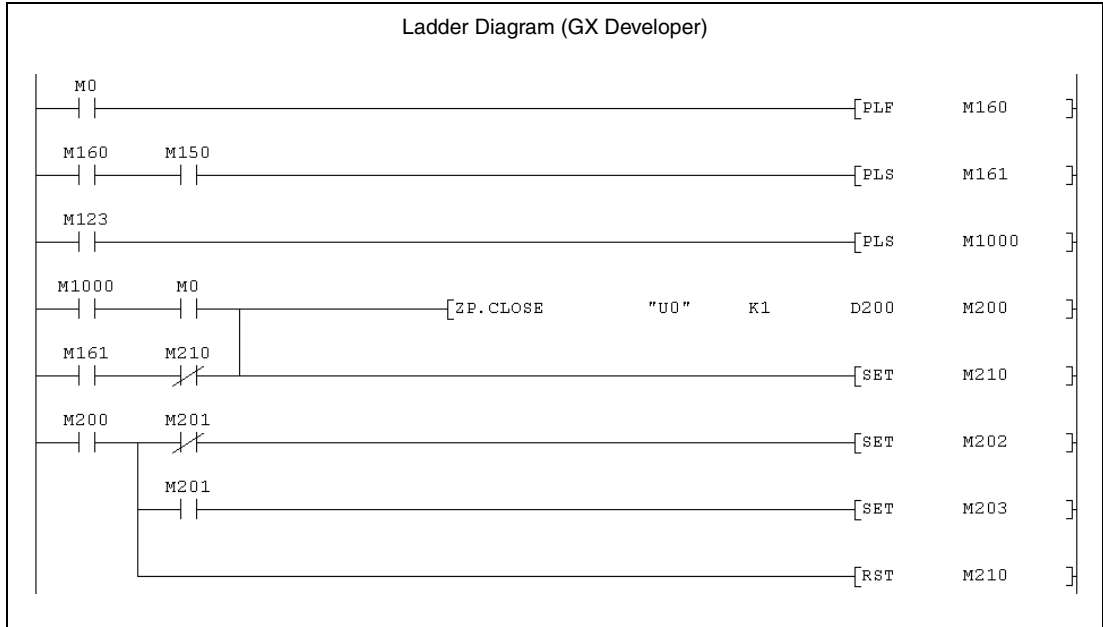
- IEC editors (On the next page the same program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)



**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.



MELSEC Instruction List

MELSEC	LD	M0			
	PLF	M160			
	LD	M160			
	AND	M150			
	PLS	M161			
	LD	M123			
	PLS	M1000			
	LD	M1000			
	AND	M0			
	LD	M161			
	ANI	M210			
	ORB				
	ZP.CLOSE	"U0"	K1	D200	M200
	SET	M210			
MELSEC	LD	M200			
	MPS				
	ANI	M201			
	SET	M202			
	MRD				
	AND	M201			
	SET	M203			
	MPP				
RST	M210				



### 11.3.6 ERRCLR

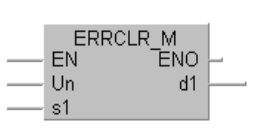
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

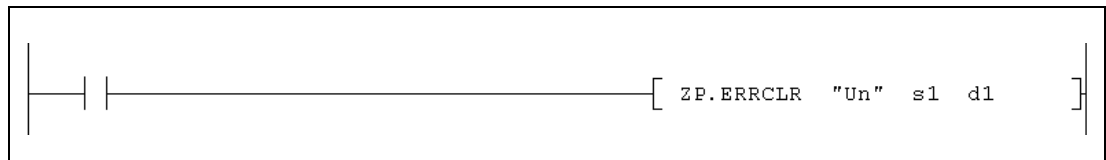
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
d1	●	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <div style="border: 1px solid black; padding: 2px; display: inline-block;">MELSEC</div> <div style="margin-left: 10px;">                 ZP.ERRCLR "Un"                  s1                  d1             </div> </div>	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;">                 ERRCLR_M "U0", s1, d1             </div>
--	--	--

**GX  
Developer**



## Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Head number of the devices where control data for the execution of this instruction is stored				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s1)+0	System area	Used by the system	—	System
	(s1)+1	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained either in the user's manual of the ETHERNET interface module or in chapter 13 of this manual.		
	(s1)+2	Error information to be cleared	Depending on the entered value an error code stored in the buffer memory is cleared and the „ERR.“ LED of the ETHERNET module is switched off. Please refer to the description on the next page.	0000 <sub>H</sub> 0001 <sub>H</sub> to 0016 <sub>H</sub> 0100 <sub>H</sub> 0101 <sub>H</sub> 0102 <sub>H</sub> 0103 <sub>H</sub> FFFF <sub>H</sub>	User
	(s1)+3	Function	Choose between clearing of an error code and switching off of the „ERR.“ LED. Please refer to the description on the next page.	0000 <sub>H</sub> or FFFF <sub>H</sub>	System
(s1)+4 to (s1)+7	System area	Used by the system	—		
d1	Bit device which is set for one scan after completion of the ERRCLR instruction. (d)+1 indicates an abnormal completion of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Instruction completed	Indicates the completion of the ERRCLR instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d1)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the ERRCLR instruction ON: Abnormal completion OFF: Normal completion	—		

**Functions Clearing of errorcode and turning off the „ERR.“ LED**

**ERRCLR Clearing operation**

The ERRCLR instruction clears an error code stored in the buffer memory of the ETHERNET interface module. When the „ERR.“ LED at the front side of the module is lit, this indicator is turned off after processing of the ERRCLR instruction as well. This instruction also clears the areas in the buffer memory where the communication status is stored.

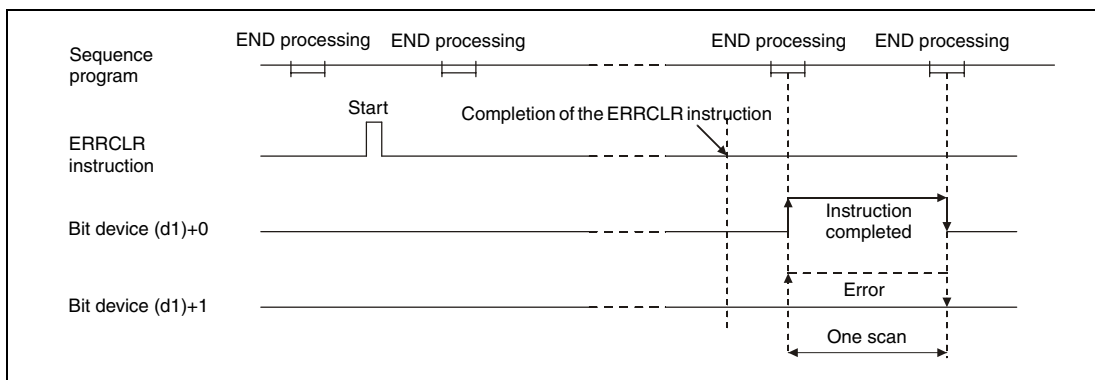
Which area of the buffer memory is cleared depends on the contents of the devices designated by (s1)+2 and (s1)+3:

Error or communication status area		Contents of		Action that will be performed
		(s1)+2	(s2)+3	
Initial error		0000 <sub>H</sub>	0000 <sub>H</sub>	<ul style="list-style-type: none"> <li>The buffer memory address 69<sub>H</sub> is cleared.</li> <li>The „ERR.“ LED is switched off.</li> </ul>
Error when opening an connection		0001 <sub>H</sub> to 0016 <sub>H</sub> (Number of the connection)		<ul style="list-style-type: none"> <li>The buffer memory address where an errorcode for the faulty connection is stored is cleared (7C<sub>H</sub>, 86<sub>H</sub>...).</li> <li>The „ERR.“ LED is switched off.</li> </ul>
Error log		0100 <sub>H</sub>	FFFF <sub>H</sub>	<ul style="list-style-type: none"> <li>The error log (addresses E3<sub>H</sub> to 174<sub>H</sub>) is cleared.</li> </ul>
Communication status	Status of the protocols	0101 <sub>H</sub>		<ul style="list-style-type: none"> <li>The buffer memory addresses from 178<sub>H</sub> to 1FF<sub>H</sub> are cleared.</li> </ul>
	E-mail receive status	0102 <sub>H</sub>		<ul style="list-style-type: none"> <li>The buffer memory addresses from 5871<sub>H</sub> to 5B38<sub>H</sub> are cleared.</li> </ul>
	E-mail send status	0103 <sub>H</sub>		<ul style="list-style-type: none"> <li>The buffer memory addresses from 5B39<sub>H</sub> to 5CA0<sub>H</sub> are cleared.</li> </ul>
All stored error codes or communication status areas		FFFF <sub>H</sub>	<ul style="list-style-type: none"> <li>All above mentioned buffer memory areas are cleared.</li> <li>The „ERR.“ LED is switched off.</li> </ul>	

Whether the execution of the ERRCLR instruction has been finished can be checked with the devices (d1)+0 and (d1)+1:

- The bit device (d1)+0 turns ON at the END processing of the scan in which the ERRCLR instruction has been completed and turns OFF at the next END processing.
- The bit device (d1)+1 indicates an error during execution of the ERRCLR instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the ERRCLR instruction, (d1)+1 turns ON at the END processing of the scan in which the ERRCLR instruction has been completed and turns OFF at the next END processing.

The timing when executing the ERRCLR instruction is shown below:



**Operation  
Error**

When an error occurs during the processing of the ERRCLR instruction, the bit device (d1)+1 is set, and an error code is stored in (s1)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please refer to the user's manual of the ETHERNET interface module.

**Program Example**

**ERRCLR**

The following program is used to clear the error code issued for connection 1. The ETHERNET module occupies the inputs and outputs from X/Y0.

- IEC editors (On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

**Ladder Diagram (GX IEC Developer)**

Clearing of the error code for connection 1 (1<sub>H</sub> is entered in (s1)+2)

Selection of the function: Clearing of the error code and turning off the „ERR.“ LED (Moving of 0<sub>H</sub> to (s1)+3)

The signal to start the ERRCLR instruction is set.

---

Execution of the ERRCLR instruction

---

When the ERRCLR instruction has been completed normally the data register D100 is cleared. In D100 the error code of the ERRCLR instruction is stored.

When an error has occurred during execution of the ERRCLR instruction the error code is moved from (s1)+1 (register D1) to D100.

The start signal for the ERRCLR instruction is reset.

MOV\_P\_M EN ENO d D2

MOV\_P\_M EN ENO d D3

SET\_M EN ENO d M1

---

ERRCLR\_M EN ENO d1 var\_M10

Un "UD"

s1 var\_D0

---

MOV\_M EN ENO d D100

MOV\_M EN ENO d D100

RST\_M EN ENO d M1

**IEC Instruction List**

	LD	M0			
	ANDN	M1			
	MOV_P_M	H1,	D2		
	MOV_P_M	H0,	D3		
	SET_M	M1			
	LD	M1			
	ERRCLR_M	"UD",	var_D0,	var_M10	
<hr/>					
	LD	M10			
	ANDN	M11			
	MOV_M	K0,	D100		
	LD	M10			
	AND	M11			
	MOV_M	D1,	D100		
	LD	M10			
	RST_M	M1			

For an explanation of the devices and instructions used please see the above ladder diagram.

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

The ladder diagram consists of three main rungs. Rung 1: M0 (NO) and M1 (NC) in series, leading to three parallel branches: [MOV H1 D2], [MOV H0 D3], and [SET M1]. Rung 2: M1 (NO) leading to [ZP.ERRCLR "U0" D0 M10]. Rung 3: M10 (NO) and M11 (NC) in series, leading to three parallel branches: [MOV K0 D100], [MOV D1 D100], and [RST M1].

MELSEC Instruction List

MELSEC	LD	M0		
	ANI	M1		
	MOV	H1	D2	
	MOV	H0	D3	
	SET	M1		
	LD	M1		
	ZP.ERRCLR	"U0"	D0	M10
MELSEC	LD	M10		
	MPS			
	ANI	M11		
	MOV	K0	D100	
	MRD			
	AND	M11		
	MOV	D1	D100	
	MPP			
RST	M1			

**11.3.7 ERRRD**

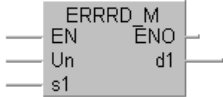
**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

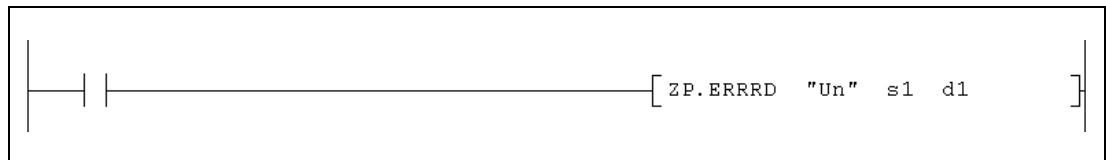
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
d1	●	●	●	—	—	—	—	—	—		

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p style="margin: 0;">ZP.ERRRD "Un" s1 d1</p> <p style="margin: 0; text-align: center;">MELSEC</p> </div>	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p style="margin: 0;">ERRRD_M "Un", s1, d1</p> </div>
--	--	---

**GX  
Developer**



## Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Head number of the devices where control data for the execution of this instruction is stored				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s1)+0	System area	Used by the system	—	System
	(s1)+1	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained either in the user's manual of the ETHERNET interface module or in chapter 13 of this manual.		
	(s1)+2	Error code to be read	Depending on the entered value an error code stored in the buffer memory is read: • 0000 <sub>H</sub> : Initial error code which is entered in the buffer memory address 69 <sub>H</sub> . • 0001 <sub>H</sub> to 0016 <sub>H</sub> : Error code for the connection with this number (Buffer memory address 7C <sub>H</sub> , 86 <sub>H</sub> ...)	0000 <sub>H</sub> 0001 <sub>H</sub> to 0016 <sub>H</sub>	User
	(s1)+3	Function	Reading of the last issued error code	0000 <sub>H</sub>	System
	(s1)+4	Read error code	Stores the error code read from the ETHERNET module 0000 <sub>H</sub> = No error Other than 0000 <sub>H</sub> : error code	—	
(s1)+5 to (s1)+7	System area	Used by the system	—		
d1	Bit device which is set for one scan after completion of the ERRRD instruction. (d)+1 indicates an abnormal completion of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Instruction completed	Indicates the completion of the ERRRD instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d1)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the ERRRD instruction ON: Abnormal completion OFF: Normal completion	—		



**Functions    Reading of an error code from an ETHERNET module**

**ERRRD    Read error code**

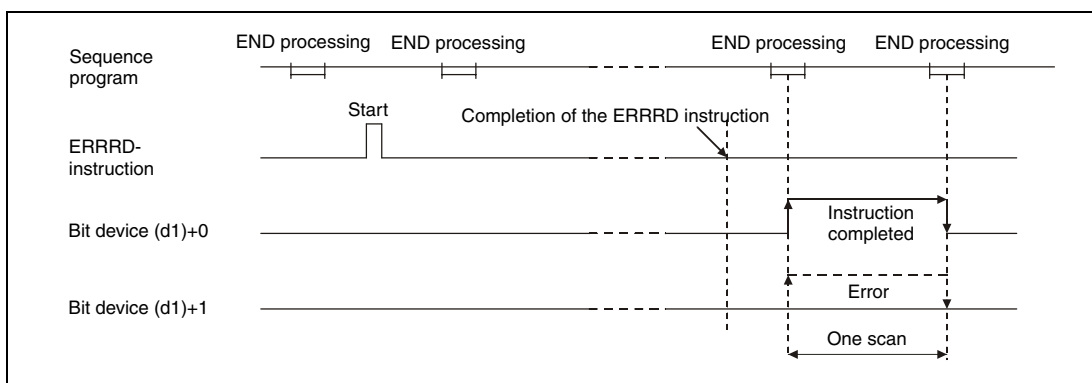
This instruction reads error code which is stored in the buffer memory of the ETHERNET interface module with the head I/O number designated by „Un“.

The device designated by (s1)+2 stores information about the buffer memory address to read from.

Whether the execution of the ERRRD instruction has been finished can be checked with the devices (d1)+0 and (d1)+1:

- The bit device (d1)+0 turns ON at the END processing of the scan in which the ERRRD instruction has been completed and turns OFF at the next END processing.
- The bit device (d1)+1 indicates an error during execution of the ERRRD instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the ERRRD instruction, (d1)+1 turns ON at the END processing of the scan in which the ERRRD instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the ERRRD instruction is being executed:



**Operation Error**

When an error occurs during the processing of the ERRRD instruction, the bit device (d1)+1 is set, and an error code is stored in (s1)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please refer to the user’s manual of the ETHERNET interface module.

**Program Example**

**ERRRD**

The following program reads the error code which is issued if the opening of connection 1 has failed. The ETHERNET module has the head I/O address X/Y0.

- IEC editors (On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

Ladder Diagram (GX IEC Developer)

Reading of the error code for connection 1 (1<sub>H</sub> is moved to (s1)+2)

Setting of the function: Reading of the last error code (0<sub>H</sub> is stored in (s1)+3)

The start signal for the ERRRD instruction is set.

---

The ERRRD instruction is executed when M1 changes from off to on.

---

When the ERRRD instruction has been completed normally the data register D101 is cleared. In D101 the error code of the ERRCLR instruction is stored.

When an error has occurred during execution of the ERRCLR instruction the error code is moved from (s1)+1 (register D1) to D101.

The start signal for the ERRRD instruction is reset.

MOVP\_M EN ENO d D2

MOVP\_M EN ENO d D3

SET\_M EN ENO d M1

ERRRD\_M EN ENO d1 var\_M10

MOV\_M EN ENO d D101

MOV\_M EN ENO d D101

RST\_M EN ENO d M1

---

IEC Instruction List

LD	M0		
ANDN	M1		
MOV_P_M	H1,	D2	
MOV_P_M	H0,	D3	
SET_M	M1		
LD	M1		
ERRRD_M	"U0",	var_D0,	var_M10
LD	M10		
ANDN	M11		
MOV_M	K0,	D101	
LD	M10		
AND	M11		
MOV_M	D1,	D101	
LD	M10		
RST_M	M1		

For an explanation of the devices and instructions used please see the above ladder diagram.

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

MELSEC Instruction List

MELSEC	LD	M0		
	ANI	M1		
	MOVP	H1	D2	
	MOVP	H0	D3	
	SET	M1		
	LD	M1		
	ZP.ERRRD	"U0"	D0	M10
MELSEC	LD	M10		
	MPS			
	ANI	M11		
	MOV	K0	D101	
	MRD			
	AND	M11		
	MOV	D1	D101	
MPP				
RST	M1			

11.3.8 UINI

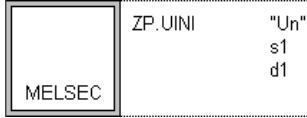
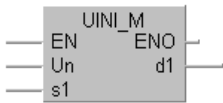
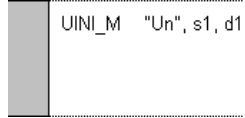
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

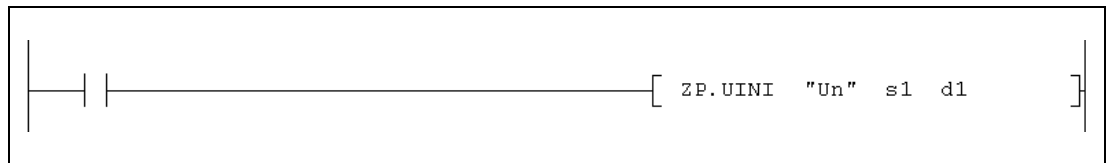
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
d1	●	●	●	—	—	—	—	—	—		

GX IEC  
Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
--	--	---

GX  
Developer



Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
„Un“	Head I/O address of the ETHERNET interface module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as „U10“)	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Head number of the devices where control data for the execution of this instruction is stored				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s1)+0	System area	Used by the system		
	(s1)+1	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained either in the user's manual of the ETHERNET interface module or in chapter 13 of this manual.	—	System
	(s1)+2	Target of change	The bits 0 and 1 of this word device are used to specify the parameters to be changed: <ul style="list-style-type: none"> <li>• Bit 0: Change of the IP address of the local station (The new address is entered in (s1)+3 and (s1)+4.) 0: The IP address is not changed 1: Change the IP address</li> <li>• Bit 1: Change of the operation settings (Enter the new settings in (s1)+5.) 0: Settings are not changed 1: The Settings are changed</li> </ul> Make sure to set all other bits (b2 to b15) to „0“.	0000 <sub>H</sub> to 0003 <sub>H</sub>	
	(s1)+3 (s1)+4	IP address of the locale station	New IP address of the local station	0000001 <sub>H</sub> to FFFFFFFE <sub>H</sub>	
(s1)+5	Operation settings	The bits of this word device specify the operation settings: <ul style="list-style-type: none"> <li>• Bit 1: Communication data code setting 0: Communication in binary code 1: Communication in ASCII code</li> <li>• Bit 5: Send frame setting 0: ETHERNET frame 1: IEEE802.3 frame</li> <li>• Bit 6: Enable/disable writing of program when the CPU is in RUN mode 0: Writing disabled 1: Writing enabled</li> <li>• Bit 8: Initial time setting 0: Do not wait for open (Communication is impossible when the CPU is stopped.) 1: Always wait for open (Communication is possible when the CPU is stopped.)</li> </ul> All other bits of this device must be reset (to „0“).	—	User	

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
d1	Bit device which is set for one scan after completion of the UINI instruction. (d)+1 indicates that an error has occurred during execution of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Instruction completed	Indicates the completion of the UINI instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d1)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the UINI instruction ON: Abnormal completion OFF: Normal completion	—		

NOTE

When performing re-initial processing of the ETHERNET module only, i.e., without changing the local station IP address and operation settings, the control data should be specified so that the value (0<sub>H</sub>) is stored in (s1)+2, the specification of target of change, before executing the UINI instruction.

The ETHERNET module clears external device address information that it has been maintaining and performs re-initial processing in order to allow data communication to restart. (The initial normal completion signal (X19) is on.)

Functions

Re-initial processing of an ETHERNET interface module

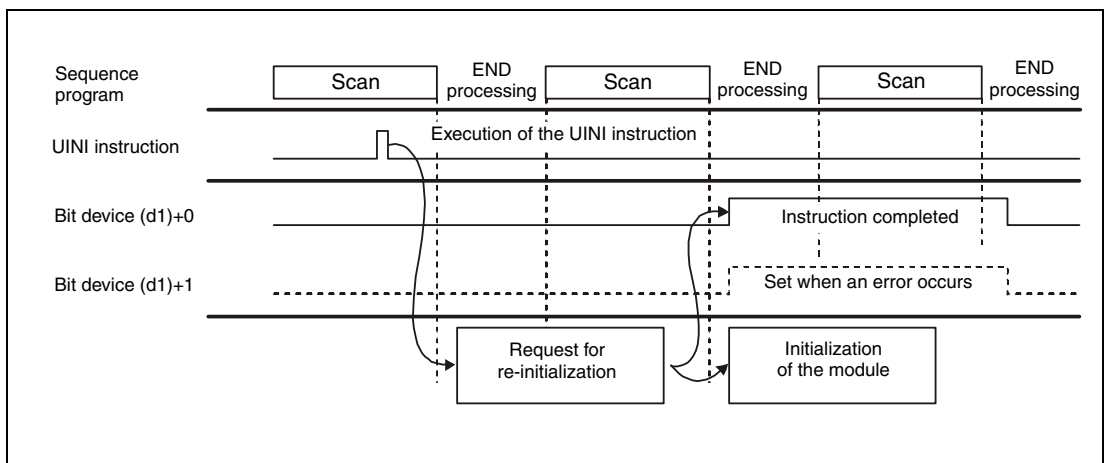
UINI Start re-initialization

The UINI instruction performs the re-initial processing of the ETHERNET module specified with Un.

Whether the execution of the UINI instruction has been finished can be checked with the devices (d1)+0 and (d1)+1:

- The bit device (d1)+0 turns ON at the END processing of the scan in which the UINI instruction has been completed and turns OFF at the next END processing.
- The bit device (d1)+1 indicates an error during execution of the UINI instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the UINI instruction, (d1)+1 turns ON at the END processing of the scan in which the UINI instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the UINI instruction is being executed:



**NOTES**

*Please keep the following points in mind when reinitializing an ETHERNET module. (Failure to do so may cause errors in the data communication with the external devices.)*

- *Be sure to end all current data communication with external devices and close all connections before performing a re-initial process.*
- *Do not mix a re-initial processing done by writing directly into buffer memory, for instance by using a TO instruction, with a re-initial processing via UINI instruction. Also, do not request another re-initial processing while an UINI instruction is already being executed.*
- *Be sure to reset external devices if the IP address of the ETHERNET module has been changed. (If an external device maintains the ETHERNET address of a device with which it communicates, the communication may not be continued after the IP address of the ETHERNET module has been changed.)*

**Operation  
Error**

When an error occurs during the processing of the UINI instruction, the bit device (d1)+1 is set, and an error code is stored in (s1)+1. For more information about the error codes please refer to the following manuals:

- When the error code is 4FFF<sub>H</sub> or less, you will find more information in chapter 13 of this manual.
- When the error code is C001<sub>H</sub> or higher, please see the user's manual of the ETHERNET interface module.

**Program Example**

UINI

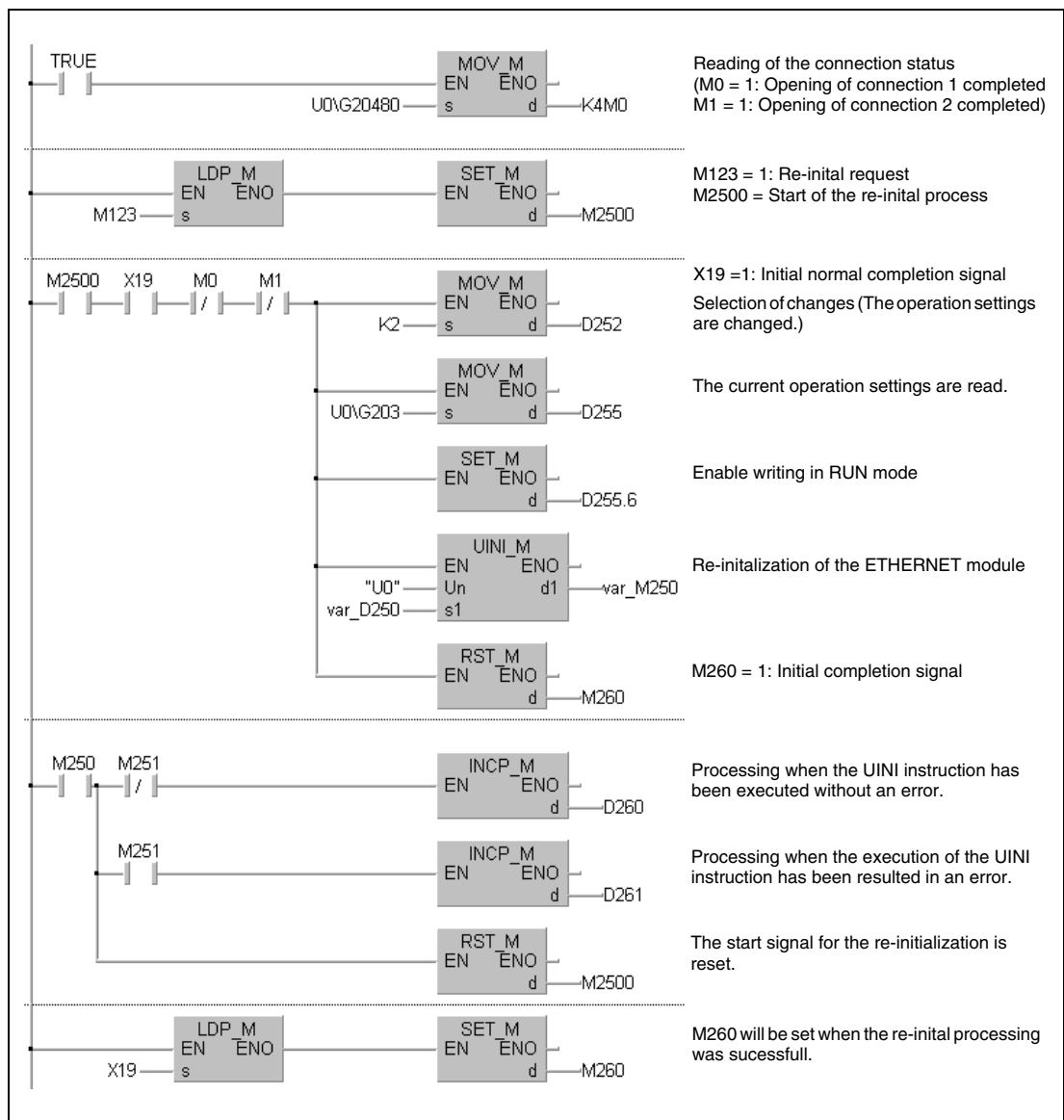
For the ETHERNET module with the head I/O address X/Y0 (Range from X/Y0 to X/Y1F) a re-initial process is performed.

**NOTES**

Only the connections 1 and 2 are used for this program example. When other connections are used the corresponding signals must be used.

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

● Ladder Diagram (GX IEC Developer)





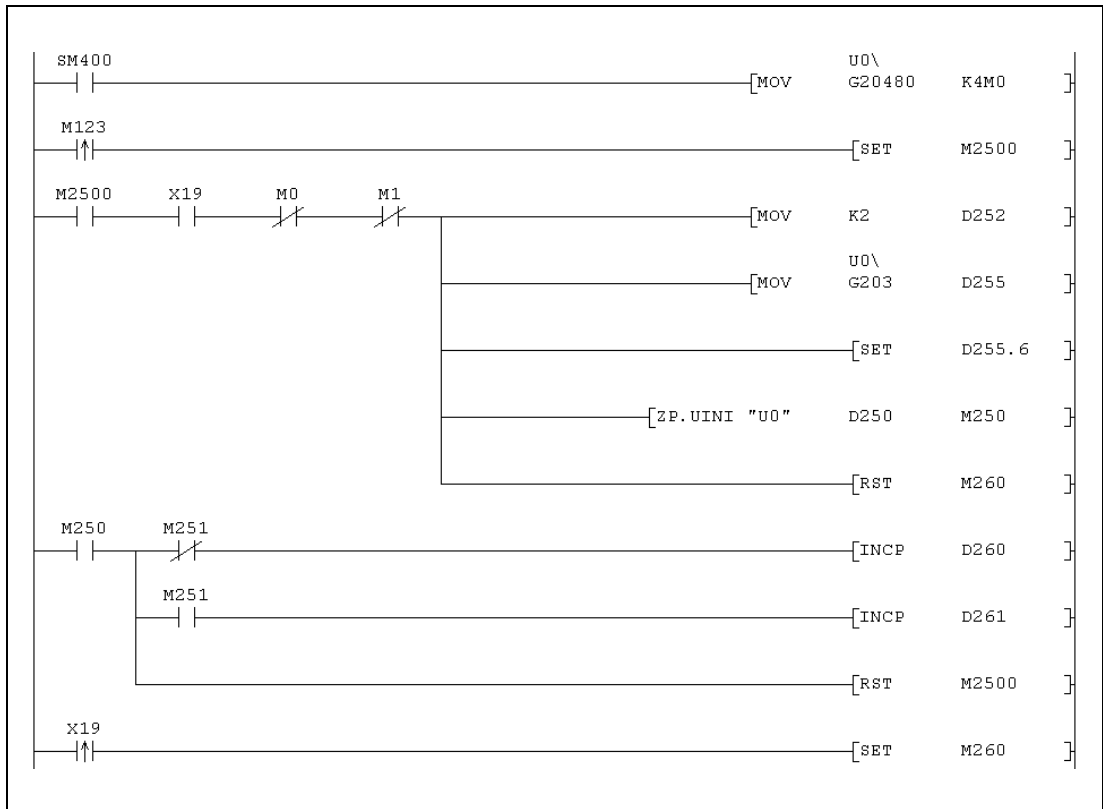
● IEC Instruction List and MELSEC Instruction List

For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

IEC Instruction List				MELSEC Instruction List			
LD	TRUE			MELSEC	LD	SM400	
MOV_M	U0\G20480, K4M0				MOV	U0\G20480	K4M0
PLS_M	M123				LDP	M123	
SET_M	M2500				SET	M2500	
AND	M2500				LD	M2500	
AND	X19				AND	X19	
ANDN	M0				ANI	M0	
ANDN	M1				ANI	M1	
MOV_M	K2, D252				MOV	K2	D252
MOV_M	U0\G203, D255				MOV	U0\G203	D255
SET_M	D255.6				SET	D255.6	
UINI_M	"U0", var_D250, var_M250				ZP.UINI	"U0"	D250 M250
RST_M	M260				RST	M260	
LD	M250			MELSEC	LD	M250	
ANDN	M251				MPS		
INCP_M	D260				ANI	M251	
LD	M250				INCP	D260	
AND	M251				MRD		
INCP_M	D261				AND	M251	
LD	M250				INCP	D261	
RST_M	M2500				MPP		
PLS_M	X19				RST	M2500	
SET_M	M260				LDP	X19	
					SET	M260	

● Ladder Diagram (GX Developer)

The devices and instructions used are explained with the program example for the ladder diagram of the GX IEC Developer shown on the previous page.



## 11.4 Instructions for MELSECNET/10

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Setting of stations for duplex network	J.PAIRSET	PAIRSET_M

### 11.4.1 PAIRSET

**CPU**

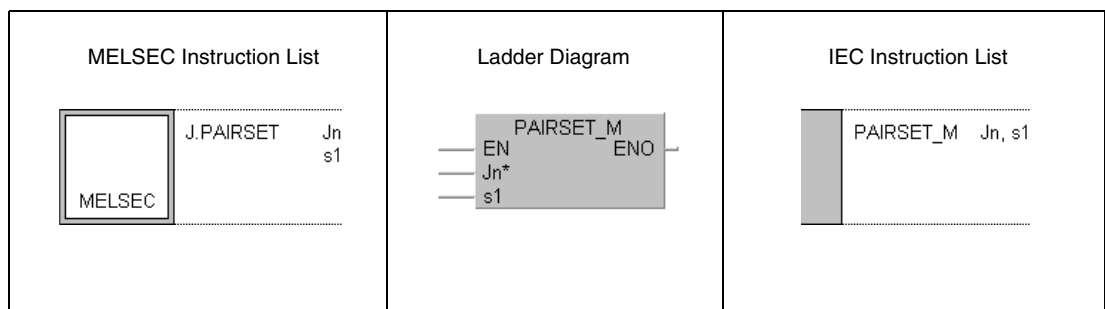
AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				● <sup>1</sup>	

<sup>1</sup> For Q4AR only

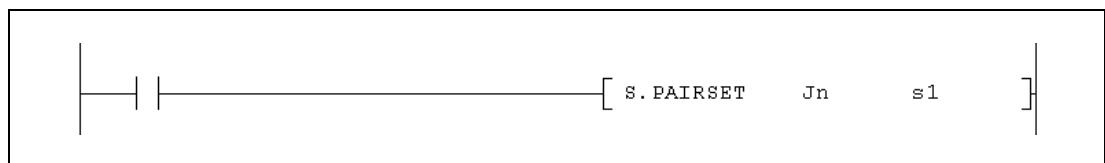
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□G□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	

**GX IEC  
Developer**



**GX  
Developer**



**Variables**

Set Data	Befehlswert	Data Type
Jn	Number of the network (1 to 239)	BIN 16-bit
s1	Head address of the device area where the settings for pairing are stored. File register (R, ZR) or the devices T, ST, C, D and W set in latch range can be used. When file registers are used, a memory card is required.	

**Functions**     **Pairing setting of stations**

**PAIRSET**     **Pairing setting instruction**

This instruction specifies which station numbers are paired (duplexed). It is required to set up on the control station.

**Structure of the device area storing the settings**

- The setting of the stations in the devices designated by s1 cannot be done in a sequence program. It is necessary to load them in the PLC CPU by peripheral devices in advance.
- Four words are used regardless of the number of stations connected.
- It is only possible to pair two stations with neighbouring station numbers. For pairing, set in s1 the bit designating the station with the higher number.
- Each bit in the devices designated by (s1)+0 to (s1)+3 stands for a station number between 1 and 64:

Set Data	Bits															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(s1)+0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
(s1)+1	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
(s1)+2	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
(s1)+3	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49

**NOTES**

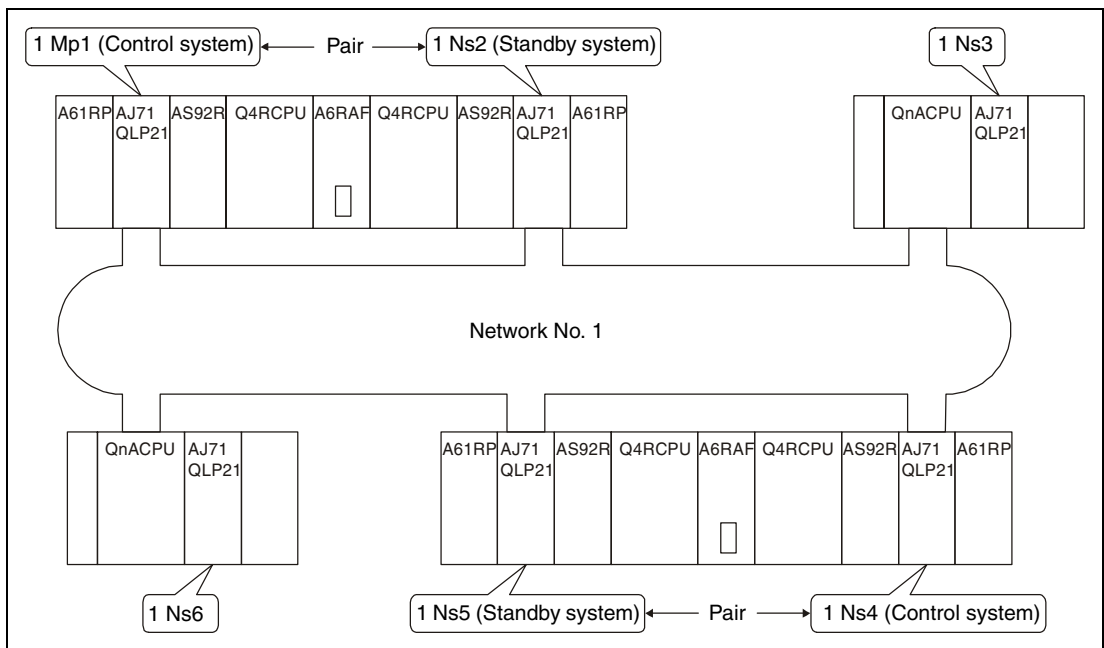
*The pairing setting instruction is valid only on control stations. Any settings on normal stations are voided.*

*If in a redundant system consisting of Q4ARCPUs the control systems network module fails to data-link due to cable connection breakage, switching from control system to standby system is done only when pairing setting has been performed.*

**Program Example**

**PAIRSET**

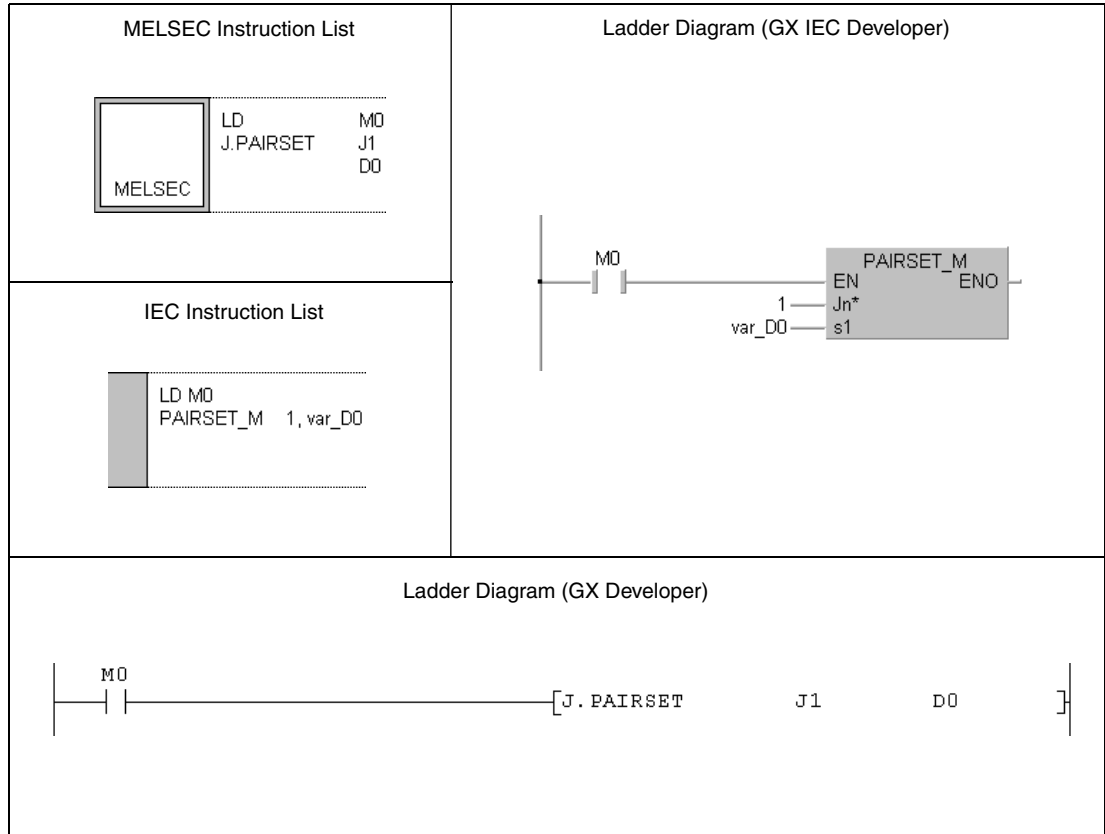
Pairing is performed for the stations 1 and 2 as well as for the stations 4 and 5 of a redundant system:



The settings are stored in the data registers D0 to D3. Bit 1 (b1) of D0 is set for the pairing of

the stations 1 and 2 whereas b4 is set for the pairing of the stations 4 and 5:

Set Data	Bits															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



**NOTE**

*For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

## 11.5 Instructions for CC-Link

Function	MELSEC Instruction in MELSEC Editor	MELSEC Instruction in IEC Editor
Parameter setting for a CC-Link network (A series)	G.RLPA	RLPA_MD
	GP.RLPA	RLPA_P_MD
Parameter setting for a CC-Link network and start of the data link (System Q)	G.RLPASET	RLPASET_MD
	GP.RLPASET	RLPASET_P_MD
Setting of automatic refresh parameters (A series)	G.RRPA	RRPA_MD
	G.RRPA	RRPA_P_MD
Reading from the buffer memory of an intelligent device station or the device memory of the PLC CPU	G.RIRD	RIRD_MD
	GP.RIRD	RIRD_P_MD
Writing to the buffer memory of an intelligent device station or the device memory of the PLC CPU	G.RIWT	RIWT_MD
	GP.RIWT	RIWT_P_MD
Reading from the buffer memory of an intelligent device station (with handshake)	G.RIRCV	RIRCV_MD
	GP.RIRCV	RIRCV_P_MD
Writing to the buffer memory of an intelligent device station (with handshake)	G.RISEND	RISEND_MD
	GP.RISEND	RISEND_P_MD
Write to the automatic updated buffer memory	G.RITO	RITO_MD
	GP.RITO	RITO_P_MD
Read from the automatic updated buffer memory	G.RIFR	RIFR_MD
	GP.RIFR	RIFR_P_MD

**11.5.1 RLPA (A series)**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

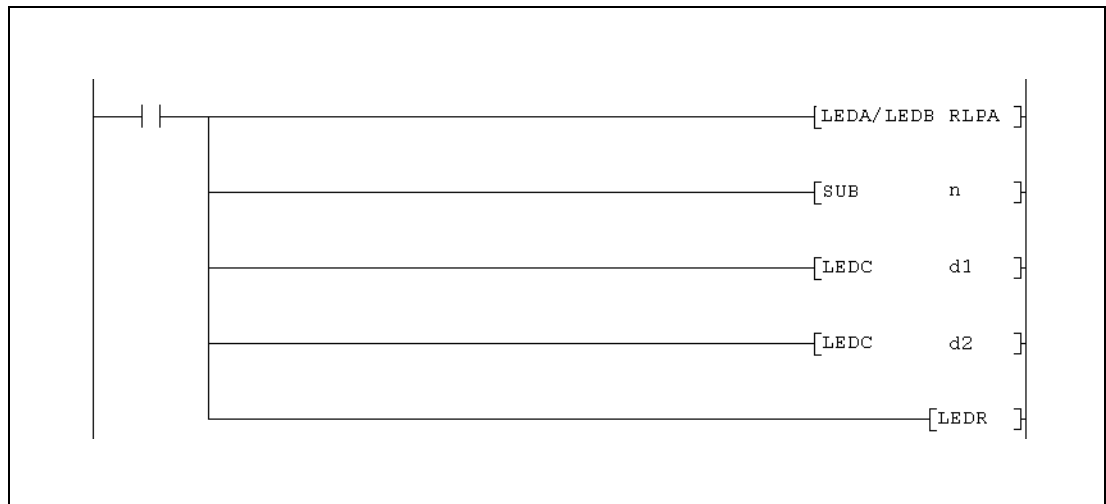
**Devices  
MELSEC A**

	Usable Devices																Blockänge	Number of steps	Index	Carry Flag M9012	Error Flag M9011			
	Bit Devices							Word Devices (16-bit)							Constant	Pointer						Level		
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P
n1																	●	●				23		●
d1							●	●	●	●	●													
d2	●	●	●	●	●																			

**GX IEC  
Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <table style="border-collapse: collapse;"> <tr><td style="border: 1px solid black; padding: 2px;">MELSEC</td></tr> </table> </div> <table style="width: 100%; border-collapse: collapse;"> <tr><td style="border-bottom: 1px dotted black;">LEDA/LEDB</td><td style="border-bottom: 1px dotted black;">RLPA</td></tr> <tr><td style="border-bottom: 1px dotted black;">SUB</td><td style="border-bottom: 1px dotted black;">n</td></tr> <tr><td style="border-bottom: 1px dotted black;">LEDC</td><td style="border-bottom: 1px dotted black;">d1</td></tr> <tr><td style="border-bottom: 1px dotted black;">LEDC</td><td style="border-bottom: 1px dotted black;">d2</td></tr> <tr><td style="border-bottom: 1px dotted black;">LEDR</td><td></td></tr> </table>	MELSEC	LEDA/LEDB	RLPA	SUB	n	LEDC	d1	LEDC	d2	LEDR		<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table style="width: 100%; border-collapse: collapse;"> <tr><td style="border-bottom: 1px dotted black;">RLPA_MD</td><td style="border-bottom: 1px dotted black;">n, d1, d2</td></tr> </table>	RLPA_MD	n, d1, d2
MELSEC															
LEDA/LEDB	RLPA														
SUB	n														
LEDC	d1														
LEDC	d2														
LEDR															
RLPA_MD	n, d1, d2														

**GX  
Developer**



For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
n	Head I/O address of the CC-Link master module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
d1	Head number of devices which store link parameters				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Synchronous mode	<ul style="list-style-type: none"> <li>0: No synchronous mode</li> <li>1: Synchronous mode</li> </ul>	0 or 1	User
	(d1)+1	Number of connected stations	Set the number of slave stations connected to the master module of CC-Link.	1 to 64	User
	(d1)+2	Slave station settings (1st station)	Please see the table on the next page.		
	(d1)+3	Slave station settings (2nd station)			
	•	•	•	•	
	•	•	•	•	
		Slave station settings (Last station)	Please see the table on the next page.		
		F i r s t	Sending buffer size	Number of devices exchanged between master station and local or intelligent device stations.	
			Receiving buffer size		
		l o c a l	Automatic updating buffer size	Number of devices of the automatic updating buffer used for communication between master station and local or intelligent device stations.	Depends on the module used
•	•	•	•	•	
•	•	•	•	•	
(d1)+(n-2)	L a s t	Sending buffer size	The same as for station 1	Depends on the module used	
(d1)+(n-1)		Receiving buffer size			
(d1)+n		Automatic updating buffer size			
d2	Bit device which is set for one scan after completion of the RLPA instruction.	0 or 1	System	Bit	

Number of points required for d1:

Two points are occupied for the selection of the synchronous mode in (d1)+0 and the designation of the number of connected stations in (d1)+1. For each station one point is required for the station settings. In addition three points are used for each local or intelligent device station to set the buffer sizes.



**Slave station settings**

For each station a word device ((d1)+2, (d1)+6, (d1)+10, ...) is reserved which contains settings for this station:

Meaning	Description	Wertebereich
Slave station settings	<p style="text-align: center;">             b15 ----- b12 b11 ----- b8 b7 ----- b0  <span style="border: 1px solid black; padding: 2px;">Station type</span>   <span style="border: 1px solid black; padding: 2px;">Occupied stations</span>   <span style="border: 1px solid black; padding: 2px;">Station number</span> </p> <p>             0: Remote I/O station              1: Remote device station              2: Intelligent device station              (including local stations and              standby master station)         </p> <p style="text-align: center;">             Number of stations occupied by the slave station:              1: 1 station are occupied              2: 2 stations are occupied              3: 3 stations are occupied              4: 4 stations are occupied         </p> <p style="text-align: center;">             Set the number of the station              in the range from 1 to 64.         </p>	<p>b0 to b7: 1 – 64 (01<sub>H</sub> – 40<sub>H</sub>)</p> <p>b8 to b11: 1 – 4</p> <p>b12 to b15: 0 – 2</p>

**Functions**      **Parameter setting for a CC-Link network**

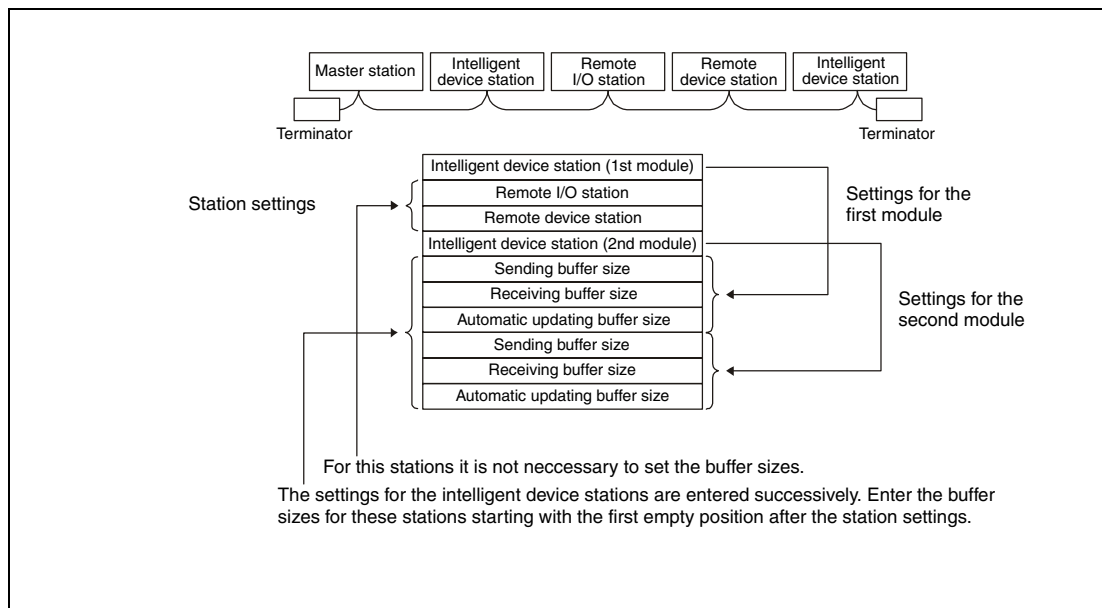
**RLPA**      **Parameter setting instruction**

When the RLPA instruction is executed, the network parameter data set to the devices beginning with the one specified at (d1) is send to the the master module specified at (n).

**NOTE**

*Use the RLPA instruction only to set the synchronous mode, the number of connected stations, the station settings and the buffer sizes. For all other parameters, initial values are forcibly set. If both the RLPA instruction and the TO instruction are used for setting the parameters, the parameters set with the TO instruction are disregarded.*

When the slave station type specified is a local/intelligent device station, it is necessary to set the "sending buffer size", "receiving buffer size" and "automatic updating buffer size". When the slave station type is a remote I/O station or a remote device station, these settings are not necessary. An example is shown in the following picture:



**NOTES**

*For the sending/receiving buffer size, specify a number 7 words larger than the size actually required for communication.*

*For the automatic updating buffer size, allocate the necessary size for the individual intelligent device station.*

*Among the intelligent device station, set 0 for the automatic updating buffer size for the stations where the automatic updating function is not provided or not used.*

If the RLPA instruction is executed again in RUN mode to change the network parameters, the new data is not used for communication with the slave station. After the A series CPU has been switched to STOP/PAUSE and then to RUN, the new network parameters will be used for communication with the slave stations.

Execution of the RLPA instruction automatically starts the data link.

When the RLPA instruction is executed, interlocking must be provided using the unit error signal (Xn0) and the unit ready signal (XnF) which indicate whether the CC-Link unit is ready.

**Execution Conditions**

When the LEDA instruction is used, the RLPA instruction is executed every scan while the write command is ON.

When the LEDB instruction is used, the RLPA instruction is executed only one scan on the leading edge (OFF -> ON) of the write command.

**Program Example**

RLPA

This program sets the following network parameters to the CC Link master module with the head I/O number of X/Y000.

Parameter	Setting	Value	Device for storing data
Synchronous mode	Valid	1	D1000
Number of connected stations	1 module	1	D1001
Stations settings	Slave station type	Intelligent device station	D1002
	No. of occupied stations	1 station	
	Station number	No. 1	
Sending buffer size	128 words	80 <sub>H</sub>	D1003
Receiving buffer size	128 words	80 <sub>H</sub>	D1004
Automatic updating buffer	960 words	600 <sub>H</sub>	D1005

- IEC editors (On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

Ladder Diagram (GX IEC Developer)

16#0001 — s — MOV\_P\_M — ENO — d — D1000

16#0001 — s — MOV\_P\_M — ENO — d — D1001

16#2101 — s — MOV\_P\_M — ENO — d — D1002

16#0080 — s — MOV\_P\_M — ENO — d — D1003

16#0080 — s — MOV\_P\_M — ENO — d — D1004

16#0600 — s — MOV\_P\_M — ENO — d — D1005

0 — n1\* — RLPA\_P\_MD — ENO — d2 — L1000

D1000 — d1

The settings for the synchronous mode are moved to (d1)+0.

The number of connected Stations is written to (d1)+1.

The station settings are moved to (d1)+2.

The size of the sending buffer is specified.

Setting of the receiving buffer size

The size of the automatic updated buffer is set.

The parameters are transmitted to the master station.

---

IEC Instruction List

LD	X20	
ANDN	X0	
AND	X0	
MOV_P_M	16#0001,	D1000
MOV_P_M	16#0001,	D1001
MOV_P_M	16#2101,	D1002
MOV_P_M	16#0080,	D1003
MOV_P_M	16#0080,	D1004
MOV_P_M	16#0600,	D1005
RLPA_P_MD	0,	D1000, L1000

For explanation of the devices and instructions used please see the above example for the ladder diagram.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

The ladder diagram consists of a single rungs with three normally open contacts labeled X20, X0, and X0F. The rung is connected to a series of ten MELSEC instructions, each enclosed in square brackets and separated by a vertical line. The instructions are: MOV P H1 D1000, MOV P H1 D1001, MOV P H2101 D1002, MOV P H80 D1003, MOV P H80 D1004, MOV P H600 D1005, LE DB RLPA, SUB K0, LE DC D1000, LE DC L1000, and LE DR.

---

MELSEC Instruction List

	LD	X20	
	ANI	X0	
	AND	X0F	
MELSEC	MOV P	H0001	D1000
	MOV P	H0001	D1001
	MOV P	H2101	D1002
	MOV P	H0080	D1003
	MOV P	H0080	D1004
	MOV P	H0600	D1005
	LE DB	RLPA	
	SUB	K0	
	LE DC	D1000	
	LE DC	L1000	
	LE DR		

**11.5.2 RLPASET (System Q)**

**CPU**

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
					●

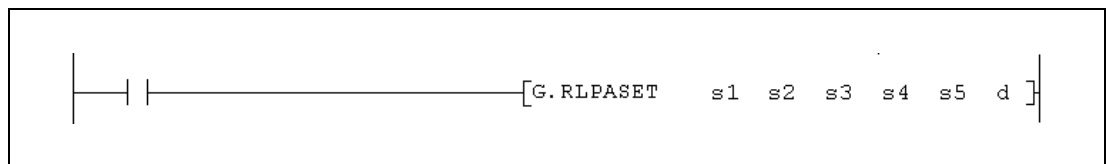
**Devices  
MELSEC Q**

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	
s2	—	●	●	—	—	—	—	—	—		
s3	—	●	●	—	—	—	—	—	—		
s4	—	●	●	—	—	—	—	—	—		
s5	—	●	●	—	—	—	—	—	—		
d	●	●	●	—	—	—	—	—	—		

**GX IEC Developer**

<p>MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">MELSEC</div> <p>G.RLPASET    Un                   s1                   s2                   s3                   s4                   s5                   d</p>	<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">RLPASET_MD Un, s1, s2, s3, s4, d</div>
---	-----------------------	--

**GX Developer**



Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
Un	Head I/O address of the CC-Link master module (Only the upper two digits of an address expressed as a 3-digit number are entered, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
Head number of the devices where control data for the execution of this instruction is stored.					
	<b>Set Data</b>	<b>Meaning</b>	<b>Description</b>	<b>Range</b>	<b>Contents is stored by</b>
s1	(s1)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : Error code	—	System
	(s1)+1	Validation of the settings	The first four bits are used to specify whether the settings made in s2 to s5 are valid or invalid: Bit 0 = 1:Slave station settings (s2) Bit 1 = 1:Reserved station specifications (s3) Bit 2 = 1:Error invalid station specifications (s4) Bit 3 = 1:Send, receive and automatic refresh buffer assignment data (s5) For the settings marked as invalid the default parameters will be applied.	0 to F	User
	(s1)+2	Number of connected modules	Set the number of connected slave stations (including the reserved stations)	1 to 64	
	(s1)+3	Number of retries	Set the number of retries to a communication faulty station.	1 to 7	
	(s1)+4	Number of automatic return modules	Set the number of slave modules which after a failure can be returned automatically to the link within one link scan.	1 to 10	
	(s1)+5	Operation specification when the PLC CPU has stopped	Specifies the data link status when a master station PLC CPU error occurs. 0: Stop 1: Continue	0 or 1	
	(s1)+6	Scan mode specification	Choose between the synchronous and the asynchronous mode 0: The data link is synchronous to the scan of the sequence program 1: The data link is asynchronous to the scan of the sequence program.	0 or 1	
	(s1)+7	Delay time setting	Link scan interval (Unit: 50 μs)	0 to 100	

**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
s2	Head device of the area where slave station settings are stored.				
	Set Data	Meaning	Description	Contents is stored by	
	(s2)+0	Settings for station No. 1	See the table at page 93 Make the settings for as much modules as are specified in (s1)+2 as number of connected modules.	User	
	(s2)+1	Settings for station No. 2			
	•	•			
	•	•			
	(s2)+62	Settings for station No. 63			
(s2)+63	Settings for station No. 64				
s3	Head device of the area where specifications for reserved stations are stored. Perform the setting for all stations up to the largest station number set in s2.				
	Set Data	Meaning	Description	Contents is stored by	
	(s3)+0	Setting for station No.'s 1 – 16	Specify a reserved station by setting the bit for the corresponding station number (see the table at page 93). Specify only the head station number of a module that occupies 2 or more stations.	User	
	(s3)+1	Setting for station No.'s 17 – 32			
	(s3)+2	Setting for station No.'s 33 – 48	No station is reserved in the default parameter setting.		
	(s3)+3	Setting for station No.'s 49 – 64			
s4	Head device of the area where specifications for error invalid stations are stored. Perform the setting for all stations up to the largest station number set in s2.				
	Set Data	Meaning	Description		Contents is stored by
	(s4)+0	Setting for station No.'s 1 – 16	When the error of a station should be ignored, set the bit for the corresponding station number (see the table at page 93). Specify only the head station number of a module that occupies 2 or more stations.	User	
	(s4)+1	Setting for station No.'s 17 – 32			
	(s4)+2	Setting for station No.'s 33 – 48	The reserved station number is given the higher priority if both error invalid station and reserved station specifications are made for the same station. No error invalid station is set in the default parameter setting.		
	(s4)+3	Setting for station No.'s 49 – 64			

Variables

Set Data	Meaning		Range	Contents is stored by	Data Type	
s5	Head device of the area where settings for the buffer memory size are stored. Perform the settings for stations specified in s2 as local stations and intelligent device stations. Start with the smallest station number.					
	Set Data	Meaning	Description	Range	Contents is stored by	
	(s5)+0	Send buffer size	Specify the buffer size needed for communication between the master station and local stations or intelligent device stations. The maximum size of the send and receive buffer together is 4096 words (1000 <sub>H</sub> ).	0 <sub>H</sub> : No buffer 40 <sub>H</sub> to 1000 <sub>H</sub> (64 to 4096 words) Default setting: 40 <sub>H</sub>	User	Address
	(s5)+1	Receive buffer size	For the sending/receiving buffer size, specify a number 7 words larger than the size actually required for communication.	0 <sub>H</sub> : No buffer 40 <sub>H</sub> to 1000 <sub>H</sub> (64 to 4096 words) Default setting: 40 <sub>H</sub>		
	(s5)+2	Automatic refresh buffer size	Number of points of the automatic refresh buffer used for communication between the master station and local stations or intelligent device stations. The size of the automatic updating buffer must be equal to the size necessary for the individual intelligent device station.	0 <sub>H</sub> : No buffer 80 <sub>H</sub> to 1000 <sub>H</sub> (128 to 4096 words) Default setting: 80 <sub>H</sub>		
	•	•	•	•		
	•	•	•	•		
(s5)+75	T w e n t y s i x t  m o d u l e	Send buffer size	The same as for the 1st module.			
(s5)+76		Receive buffer size				
(s5)+77		Automatic refresh buffer size				
d	Bit device which is set for one scan after completion of the RLPASET instruction. (d)+1 indicates that an error has occurred during execution of the instruction.					
	Set Data	Meaning		Description	Range	Contents is stored by
	(d)+0	Instruction completed		Indicates the completion of the RLPASET instruction ON: Instruction completed OFF: Instruction not completed	0 or 1	System
(d)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RLPASET instruction ON: Abnormal completion OFF: Normal completion		0 or 1		



**Slave station settings**

For each station a word device ((s2)+0 to (s2)+63) is reserved which contains settings for this station:

Meaning	Description	Range
Settings for 1 to 64 modules	<div style="text-align: center;"> </div> <p>0: Remote I/O station                      1: Remote device station                      2: Intelligent device station (including local stations and standby master station)</p> <p>Number of stations occupied by the slave station:                      1: 1 station are occupied                      2: 2 stations are occupied                      3: 3 stations are occupied                      4: 4 stations are occupied</p> <p style="text-align: right;">Set the number of the station in the range from 1 to 64.</p>	b0 to b7: 1 – 64 (01 <sub>H</sub> – 40 <sub>H</sub> ) b8 to b11: 1 – 4 b12 to b15: 0 – 2

The default parameter settings for (s2)+0 to (s2)+63 are „0101<sub>H</sub>“ bis „0140<sub>H</sub>“. (Station number 1 to 64, one station occupied, remote I/O station)

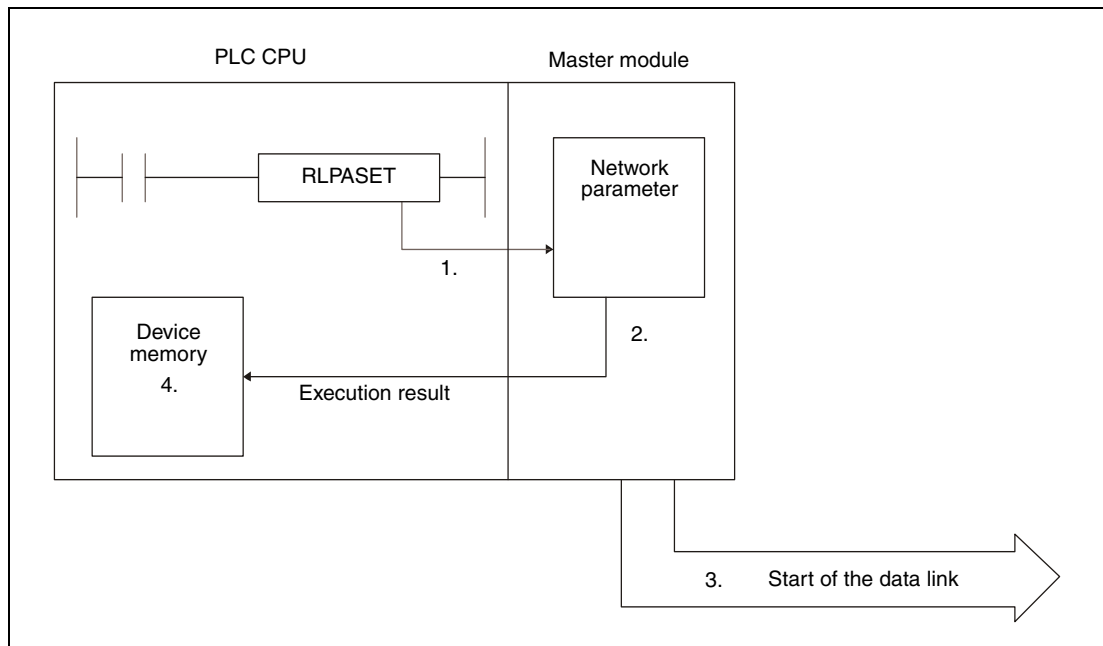
**Designation of the station number in s3 and s4**

Each bit of the four word devices used for s3 and s4 represents one station:

Set Data	Bit															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(s3)+0 (s4)+0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
(s3)+1 (s4)+1	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
(s3)+2 (s4)+2	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
(s3)+3 (s4)+3	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49

The numbers 1 to 64 in the table indicate a station number. When a bit is set the corresponding station is selected.

**Functions**     **Parameter setting for a CC-Link Network and start of the data link**  
**RLPASET**     **Parameter setting instruction**



1. The network parameters stored in (s1) to (s5) are send to master module of the CC-Link designated by Un using the RLPASET instruction.
2. The received settings are checked by the master module.
3. If the settings are correct, the data link is started.
4. The device specified by (d) is set.

It is only possible to execute one RLPASET instruction at a time.

### Number of required devices

The following numbers of devices are required for the RLPASET instruction:

- s1: 8 word devices
- s2: 64 word devices
- s3: 4 word devices
- s4: 4 word devices
- s5: 78 word devices

Please note the required areas for (s1) to (s5) during programming.

An example:

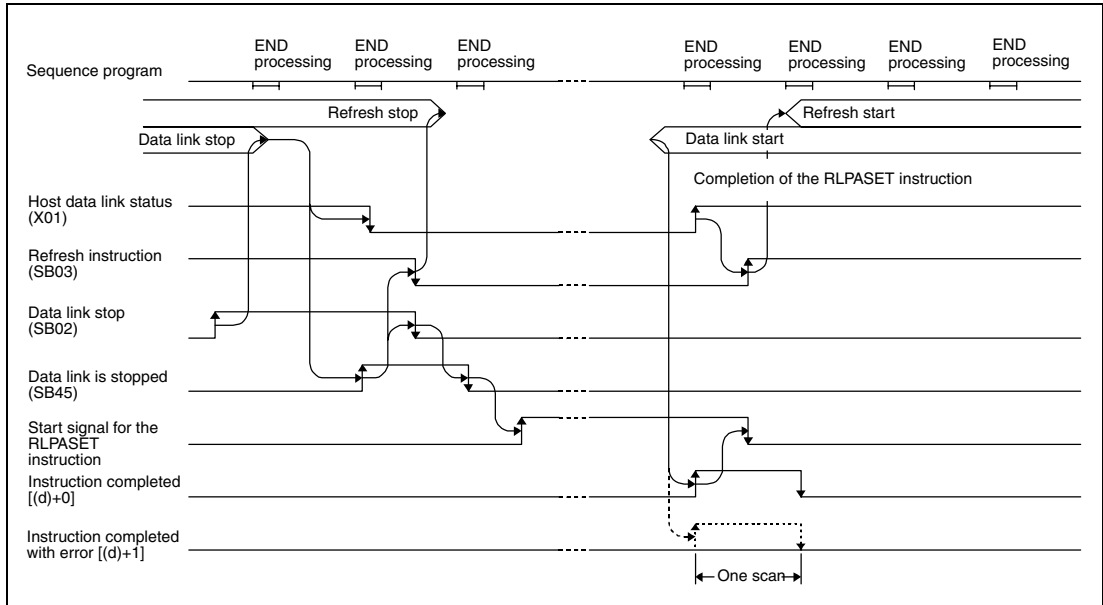
Four slave stations are connected to a master module. In the Q02CPU mounted in the PLC of the master station the data link registers D0 to D12287 are available. If D12284 is designated as head device for (s2) because there are only four slave stations, the execution of the RLPASET instruction will result in an error with the code 4101. This is because the PLC CPU always checks the range for 64 stations (D12284 to D12347 in this example) and in this case the available range is exceeded.

Whether the execution of the RLPASET instruction has been finished can be checked with the devices (d1)+0 and (d1)+1:

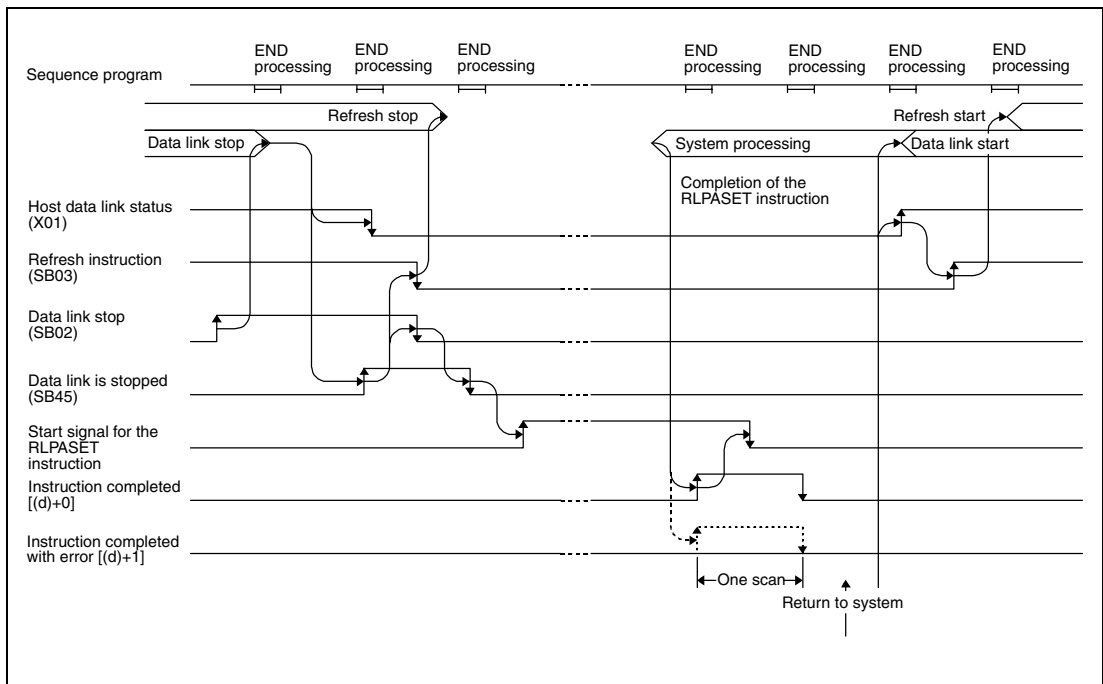
- The bit device (d1)+0 turns ON at the END processing of the scan in which the RLPASET instruction has been completed and turns OFF at the next END processing.
- The bit device (d1)+1 indicates an error during execution of the RLPASET instruction. When

the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the UINI instruction, (d1)+1 turns ON at the END processing of the scan in which the RLPASET instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RLPASET instruction is executed and all stations are normal:



The timing for the RLPASET instruction in the case of a faulty station is shown below:



**Operation Error**

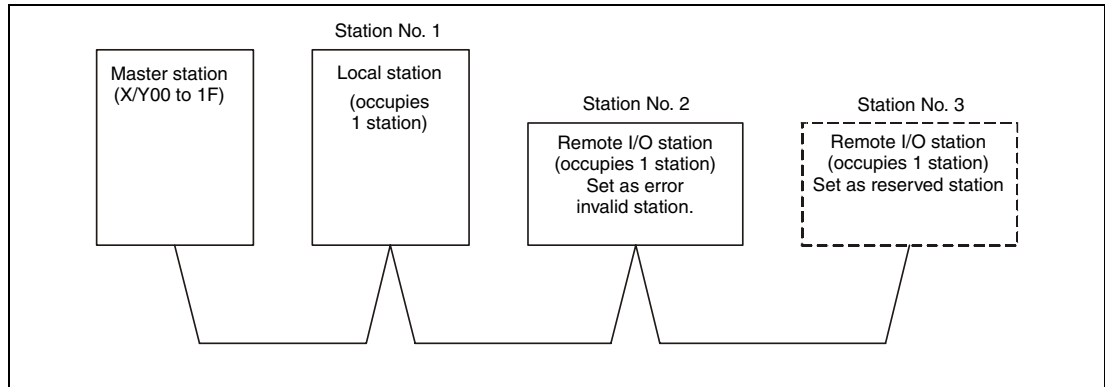
In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When the module designated by (Un) is not a special function module. (error code: 2112)
- When an attempt was made to execute an unsupported instruction. (error code: 4002)
- When the number of devices in the instruction is incorrect. (error code: 4003)
- When the instruction specifies a device that cannot be used. (error code: 4004)
- When the instruction contains data that cannot be used. (error code: 4100)
- When the number of points for data used in the instruction exceeds the available range, or storage data and constants of a device specified by the instruction exceeds the available range (including dummy devices). (error code: 4101)

**Program Example**

**RLPASET**

This program transfers the network parameter to the master station occupying the head I/O number X/Y000. The CC-Link network consists of three slave stations:

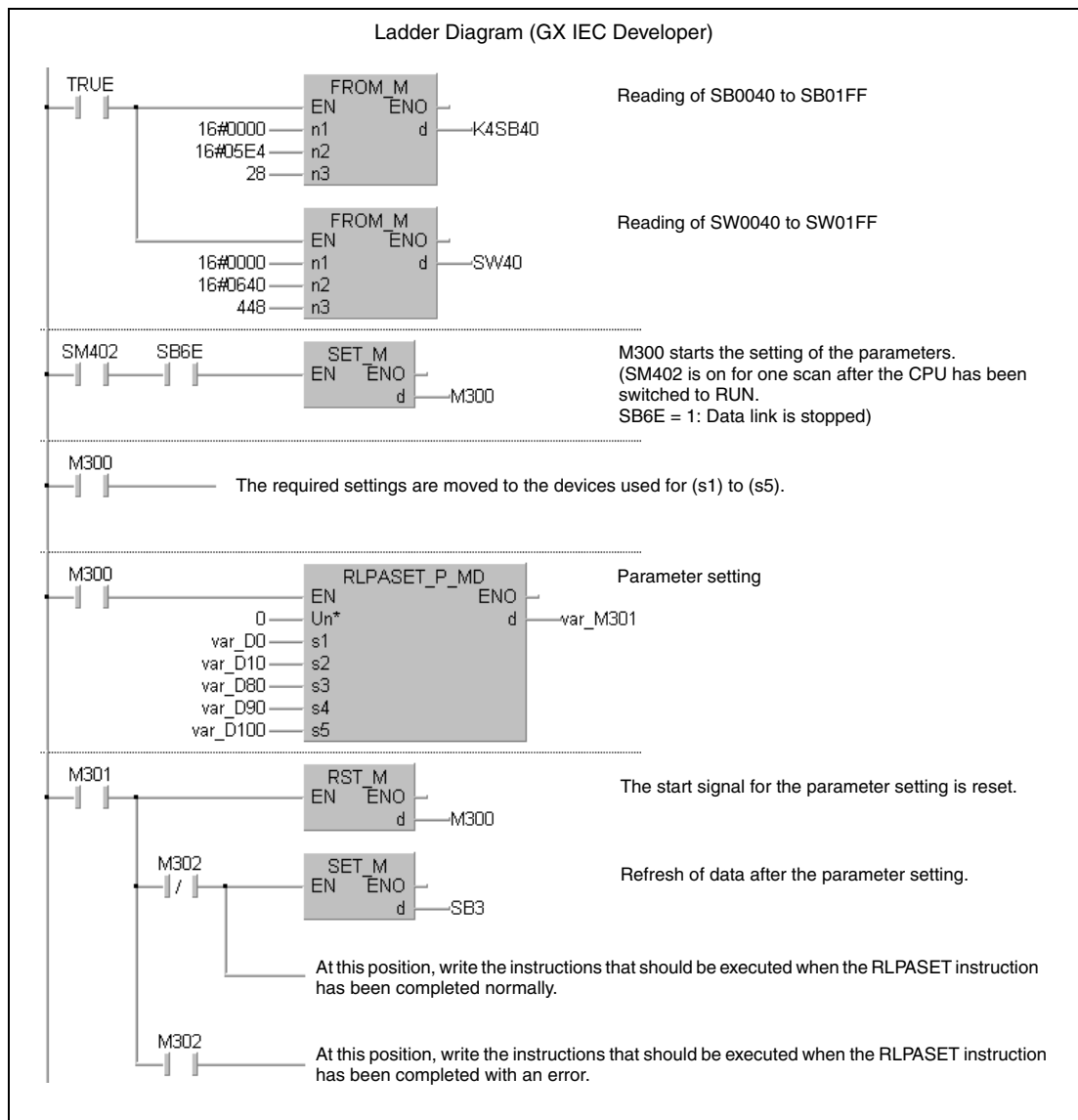


The devices designated by (s1) to (s5) are holding the following values:

Parameter		Setting	Set value	Allocated device	
Control data for the execution of the instruction	(s1)+1	Validation of the settings	All settings are valid.	15	D1
	(s1)+2	Number of connected modules	3 slave modules	3	D2
	(s1)+3	Number of retries	3 times	3	D3
	(s1)+4	Number of automatic return modules	1 module	1	D4
	(s1)+5	Operation specification when the PLC CPU has stopped	Stop	0	D5
	(s1)+6	Scan mode specification	Asynchronous	0	D6
	(s1)+7	Delay time setting	0 μs	0	D7
Settings for slave stations	(s2)+0	Settings for the first station	Local station, occupies 1 station, Station No. 1	2101 <sub>H</sub>	D10
	(s2)+1	Settings for the second station	Remote I/O station, occupies 1 station, Station No. 2	102 <sub>H</sub>	D11
	(s2)+2	Settings for the third station	Remote I/O station, occupies 1 station, Station No. 3	103 <sub>H</sub>	D12
Reserved stations	(s3)+0	Selection of reserved stations	Station No. 3 is reserved (bit 2 is set)	4	D80
	(s3)+1			0	D81
	(s3)+2			0	D82
	(s3)+3			0	D83
Error invalid stations	(s4)+0	Specification of error invalid stations	Station No. 2 (bit 1 is set)	2	D90
	(s4)+1			0	D91
	(s4)+2			0	D92
	(s4)+3			0	D93
Buffer sizes	(s5)+0	Send buffer of the first local station (Station No. 1)	100 words	64 <sub>H</sub>	D100
	(s5)+1	Receive buffer of the first local station (Station No. 1)	100 words	64 <sub>H</sub>	D101
	(s5)+2	Automatic refresh buffer of the first local station (Station No. 1)	Not used	0 <sub>H</sub>	D102

The contents of the data registers D1 to D102 must be set according to the above table before the RLPA instruction is called.

- IEC editors (On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

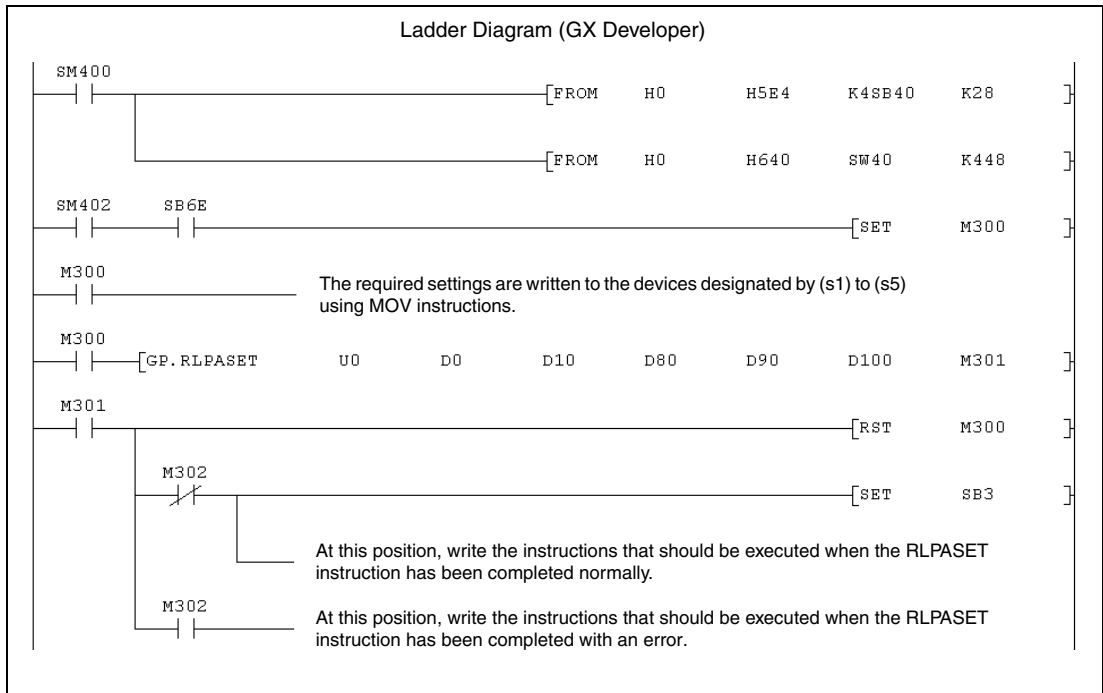


IEC Instruction List					
LD	TRUE				
FROM_M	16#0000,	16#05E4,	28,	K4SB40	
FROM_M	16#0000,	16#0640,	448,	SW40	
.....					
LD	SM402				
AND	SB6E				For an explanation of the devices and instructions used
SET_M	M300				please see the program example on the previous page.
.....					
LD	M300				
	The required settings are written to the devices designated by (s1) to (s5) using MOV instructions.				
.....					
LD	M300				
RLPASET_P_MD	0,	var_D0,	var_D10,	var_D80,	var_D90,
					var_D100,
					var_M301
.....					
LD	M301				
RST_M	M300				
LD	M301				
ANDN	M302				
SET_M	SB3				
	An instruction at this position will be executed when the RLPASET instruction has been completed normally.				
LD	M301				
AND	M302				
	An instruction at this position will be executed when the processing of the RLPASET instruction has been completed with an error.				

**NOTE**

*For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.*

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer.



MELSEC Instruction List

MELSEC	LD	SM400
	FROM	H0 H5E4 K4SB40 K28
MELSEC	FROM	H0 H640 SW40 K448
	LD	SM402
MELSEC	AND	SB6E
	SET	M300
MELSEC	LD	M300
		The required settings are written to the devices designated by (s1) to (s5) using MOV instructions.
MELSEC	LD	M300
	GP.RLPASET	U0 D0 D10 D80 D90 D100 M301
	LD	M301
	RST	M300
	MPS	
	ANI	M302
	SET	SB3
		An instruction at this position will be executed when the RLPASET instruction has been completed normally.
MELSEC	MPP	
	AND	M302
		An instruction at this position will be executed when the processing of the RLPASET instruction has been completed with an error.



11.5.3 RRPA (A series)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

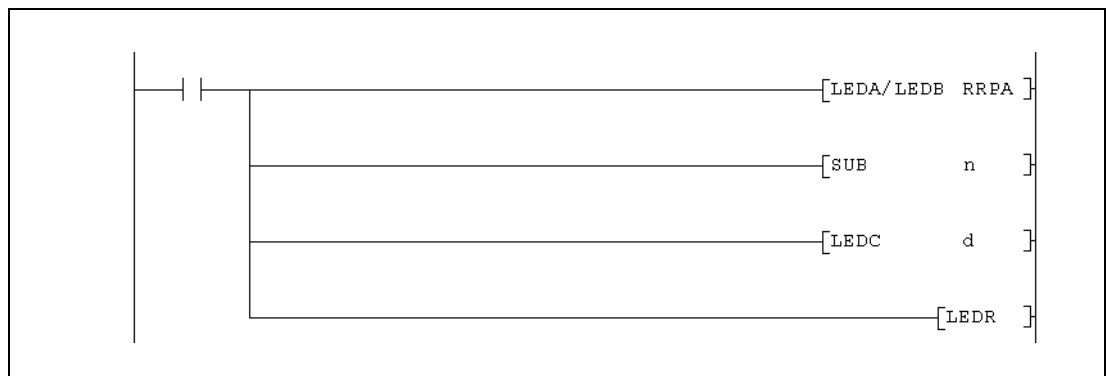
Devices  
MELSEC A

Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9011						
Bit Devices						Word Devices (16-bit)						Constant		Pointer	Level											
X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K	H (16#)	P	I	N						
n																●	●									
d							●	●	●	●	●											20				●

GX IEC  
Developer

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LEDA/LEDB</td> <td>RRPA</td> </tr> <tr> <td></td> <td>SUB</td> <td>n</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d</td> </tr> <tr> <td></td> <td>LEDR</td> <td></td> </tr> </table>	MELSEC	LEDA/LEDB	RRPA		SUB	n		LEDC	d		LEDR		<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">RRPA_MD</td> <td>n, d</td> </tr> </table>	RRPA_MD	n, d
MELSEC	LEDA/LEDB	RRPA														
	SUB	n														
	LEDC	d														
	LEDR															
RRPA_MD	n, d															

GX  
Developer



For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
n	Head I/O address of the CC-Link master module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
d	Head device of the area storing refresh parameter				
	Set Data	Meaning	Description	Contents is stored by	
	d+0	R X	Head number	Head address of the remote inputs (RX) in the master or local module	System
	d+1		Device of the PLC CPU	Set the automatic refreshed device of the CPU with the device code given in the table below.	User
	d+2		Head number of the CPU side device	Set the head device number on the CPU side*	
	d+3		Number of refresh points	Number of devices which are exchanged between CPU and CC-Link module*	
	d+4	R Y	Head number	Head address of the remote outputs (RY) in the master or local module	User
	d+5		Device of the PLC CPU	Set the automatic refreshed device of the CPU with the device code given in the table below.	
	d+6		Head number of the CPU side device	Set the head device number on the CPU side*	
	d+7		Number of refresh points	Number of devices which are exchanged between CPU and CC-Link module*	
	d+8	R W	Head number	Head address of the remote register (RW) in the master or local module	RW: System
	d+9		Device of the PLC CPU	Set the automatic refreshed device of the CPU with the device code given in the table below.	RWw: User
	d+10		Head number of the CPU side device	Set the head device number on the CPU side*	User
	d+11		Number of refresh points	Number of devices which are exchanged between CPU and CC-Link module*	
	d+12	S B	Head number	Head address of the link special relays (SB) in the master or local module	System
	d+13		Device of the PLC CPU	Set the automatic refreshed device of the CPU with the device code given in the table below.	User
	d+14		Head number of the CPU side device	Set the head device number on the CPU side*	
	d+15		Number of refresh points	Number of devices which are exchanged between CPU and CC-Link module*	
	d+16	S W	Head number	Head address of the link special register (SW) in the master or local module	System
	d+17		Device of the PLC CPU	Set the automatic refreshed device of the CPU with the device code given in the table below.	User
d+18	Head number of the CPU side device		Set the head device number on the CPU side*		
d+19	Number of refresh points		Number of devices which are exchanged between CPU and CC-Link module*		

\* Set „0“ or a multiple of „16“ for the device number of bit devices (X, Y, M, B) and the number

of automatic refresh points. Otherwise an error will occur. (However, when „0“ is set as number of refresh points the corresponding device will not be refreshed.)

In d+5, d+13 etc. the device of the PLC CPU which corresponds to a device of the CC-Link module is set. For example, internal relays (M) can be used to indicate the status of remote inputs (RX).

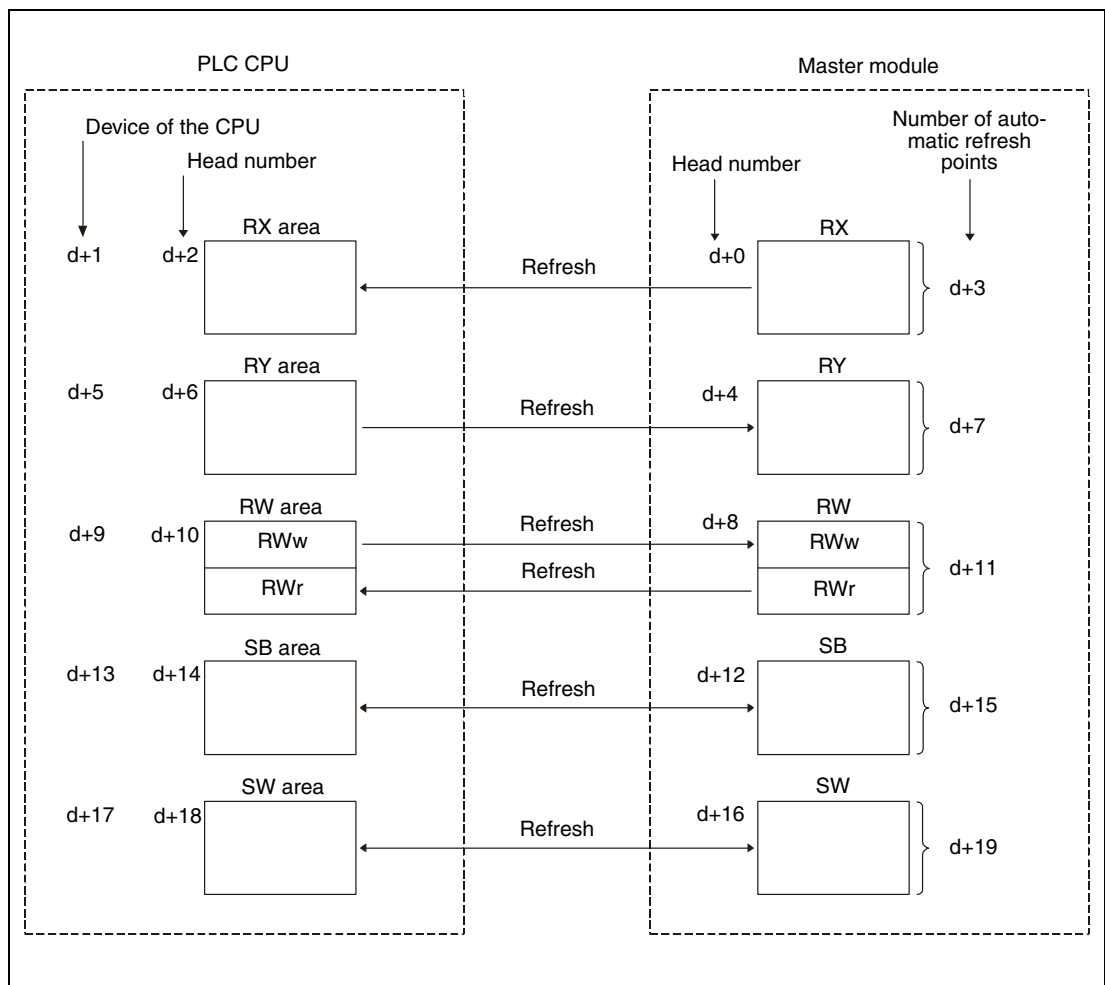
Code	Device	Code	Device
0	—	5	T
1	X	6	C
2	Y	7	D
3	M	8	W
4	B	9	R

**Functions Setting of automatic refresh parameter**

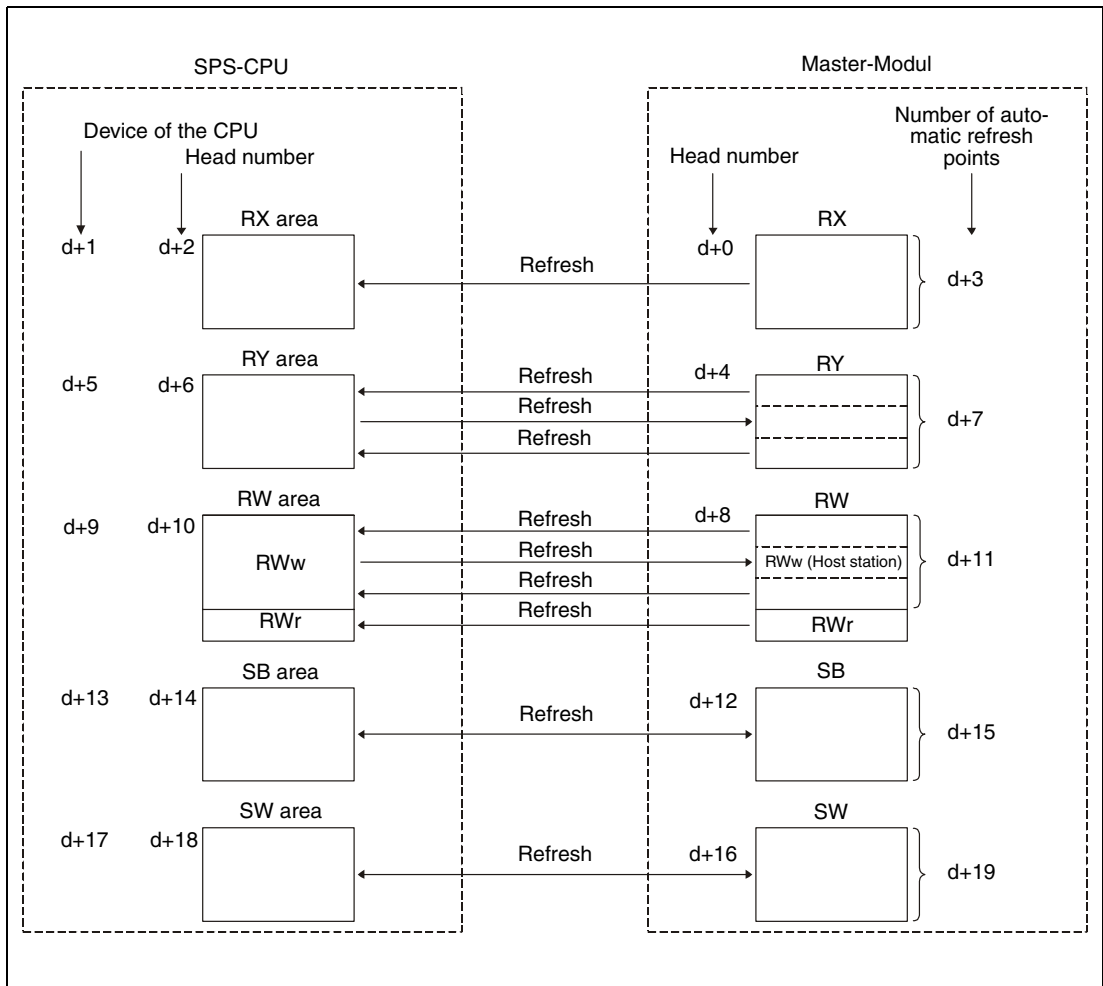
**RRPA Parameter setting instruction**

The RRPA instruction sets the devices and numbers of points on which automatic refresh will be performed between the CPU and master/local module. When FROM/TO instructions are used to read or write data from and to the master/local module, the execution of the RRPA instruction is not needed.

- Data exchange between PLC CPU and the master station



● Data exchange between PLC CPU and a local station



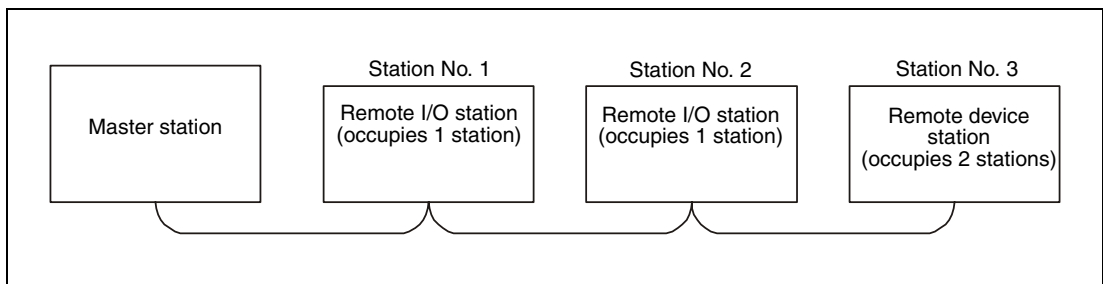
When the RRPA instruction is executed, the automatic refresh settings are registered to the CPU and automatic refresh is performed between the CPU and master/local module.

The RRPA instruction is executed only once after the RUN mode was entered. If several RRPA instructions are set for the same module, the settings done with the first instruction are valid. If you want to change the settings, perform the RRPA instruction with the new parameters. After the CPU has been switched to STOP/PAUSE and then to RUN again, the new automatic refresh parameters are used for refreshing.

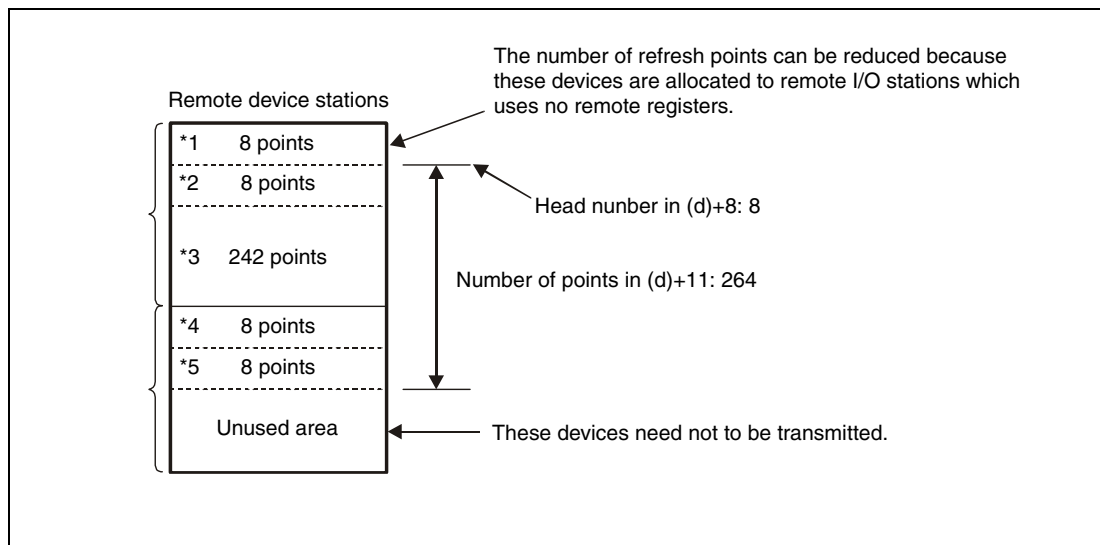
To refresh all the areas of the remote registers (RWw and RWr) write „0“ as the head number to d+8 and „512“ as the number of points to d+11.

**NOTE**

*The following system configuration is used to explain the refreshing of the remote registers:*



All 256 words (for 64 station) of RWw within the RW area are occupied even if the total number of stations is less than 64. The head of RWr therefore comes after those 256 RWw points.



- \*1: RWw area (8 points) of the station No.1 and 2 (Remote I/O station)
- \*2: RWw area (8 points) of the station No.3 (Remote device station)
- \*3: 242 points of the RWw area are occupied automatically by the system
- \*4: RWr area (8 points) of the station No.1 and 2 (Remote I/O station)
- \*5: RWr area (8 points) of the station No.3 (Remote device station)

#### Setting of refreshed devices SB and SW:

- Allocate refreshed devices of the PLC CPU to the special relays (SB) and special registers (SW). Please note the direction of the refreshing: SB0000 to SB003F are refreshed from the CPU to the master module, and SB0040 to SB00FF are refreshed from the master module to the CPU.
- File register (R) cannot be specified as refreshed devices for SB and SW. If file registers are set for SB or SW and written to the CPU, an instruction code error occurs and the CPU is inoperative.
- The device range set for refreshed devices in SB or SW should not be specified as a latch range. If the device range set for refreshed devices in SB or SW is specified as a latch range, normal operation may not be performed due to undefined data at power-on/reset.
- The SB and SW refresh ranges set with the RRPA instruction during power-on cannot be changed.

#### Execution Conditions

When the LEDA instruction is used, the RRPA instruction is executed every scan while the write command is ON.

When the LEDB instruction is used, the RRPA instruction is executed only one scan on the leading edge (OFF -> ON) of the write command.

#### Operation Error

In the following cases an operation error occurs, the error flag M9011 is set, and the error code „50“ is stored in D908. (The error code „503“ is written to D9001 when using an AnUCPU and to D9092 when using an AnSHCPU.)

- The device code is „0“ or other than 1 to 9.
- The head number of a bit device is not „0“ or is not a multiple of 16.
- The number of refresh points is not a multiple of 16.

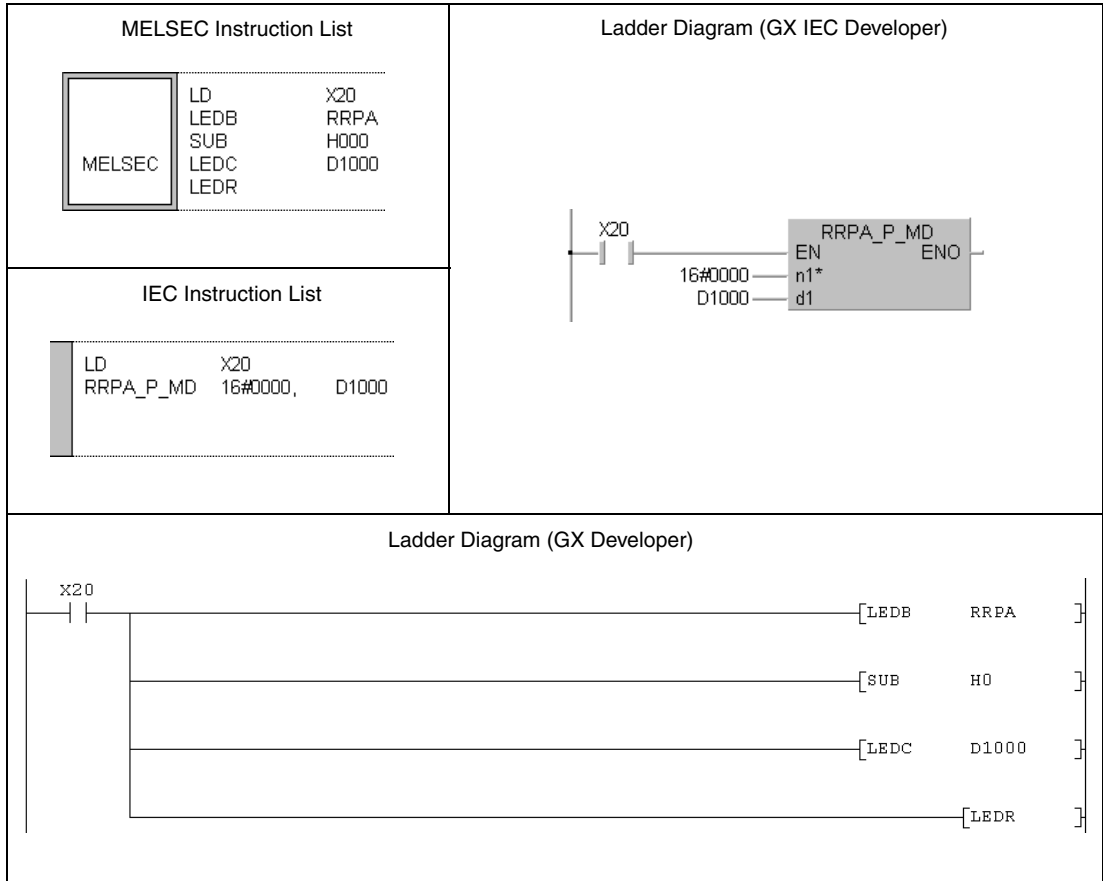
**Program  
Example**

## RRPA

The following program sets the automatic refresh parameters to the master module allocated to the I/O numbers X/Y000 to X/Y01F. The settings are stored from file register D1000 onward:

Parameter		Setting	Set data	Data storage device
RX	Head number	0	0 <sub>H</sub>	D1000
	Device of the PLC CPU	X (Code: 1)	1 <sub>H</sub>	D1001
	Head number of the CPU side device	XA0	A0 <sub>H</sub>	D1002
	Number of refresh points	32	32	D1003
RY	Head number	0	0 <sub>H</sub>	D1004
	Device of the PLC CPU	Y (Code: 2)	2 <sub>H</sub>	D1005
	Head number of the CPU side device	YA0	A0 <sub>H</sub>	D1006
	Number of refresh points	48	48	D1007
RW	Head number	0	0 <sub>H</sub>	D1008
	Device of the PLC CPU	D (Code: 7)	7 <sub>H</sub>	D1009
	Head number of the CPU side device	D160	160	D1010
	Number of refresh points	272	272	D1011
SB	Head number	0	0 <sub>H</sub>	D1012
	Device of the PLC CPU	M (Code: 3)	3 <sub>H</sub>	D1013
	Head number of the CPU side device	M160	160	D1014
	Number of refresh points	256	256	D1015
SW	Head number	0	0 <sub>H</sub>	D1016
	Device of the PLC CPU	W (Code: 8)	8 <sub>H</sub>	D1017
	Head number of the CPU side device	WA0	A0 <sub>H</sub>	D1018
	Number of refresh points	256	256	D1019

The contents of the data registers D1000 to D1019 must be set according to the above table before the RRPA instruction is called.



11.5.4 RIRD (A series)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

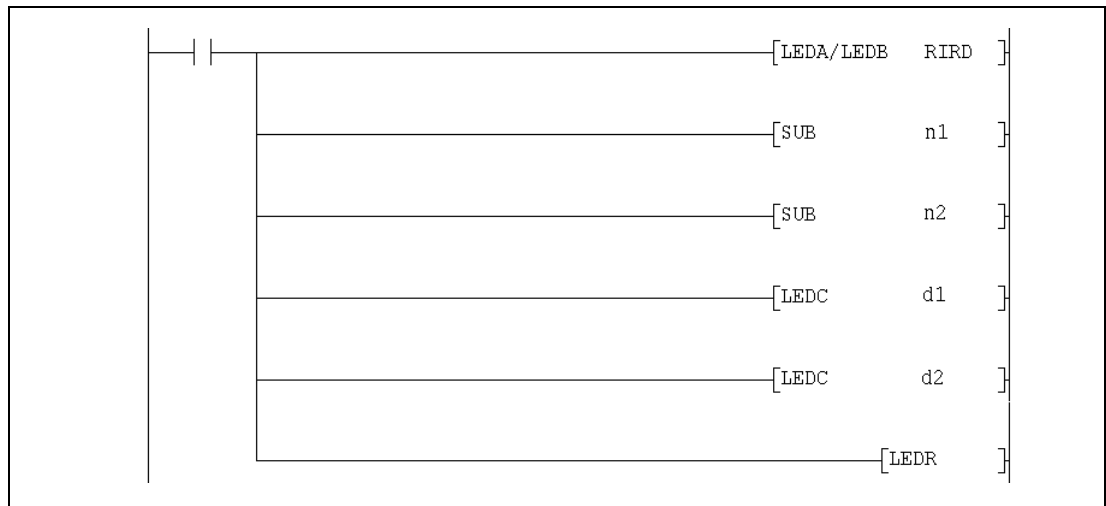
Devices  
MELSEC A

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9011
	Bit Devices						Word Devices (16-bit)						Constant		Pointer	Level					
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V					
n1																	●	●			
n2																	●	●			
d1							●	●	●	●	●										
d2	●	●	●	●	●																

GX IEC  
Developer

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LEDA/LEDB</td> <td>RIRD</td> </tr> <tr> <td></td> <td>SUB</td> <td>n1</td> </tr> <tr> <td></td> <td>SUB</td> <td>n2</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d1</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d2</td> </tr> <tr> <td></td> <td>LEDR</td> <td></td> </tr> </table>	MELSEC	LEDA/LEDB	RIRD		SUB	n1		SUB	n2		LEDC	d1		LEDC	d2		LEDR		<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td>RIRD_MD</td> <td>n1, n2, d1, d2</td> </tr> </table>	RIRD_MD	n1, n2, d1, d2
MELSEC	LEDA/LEDB	RIRD																				
	SUB	n1																				
	SUB	n2																				
	LEDC	d1																				
	LEDC	d2																				
	LEDR																					
RIRD_MD	n1, n2, d1, d2																					

GX  
Developer





Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
n1	Head I/O address of the CC-Link master/local module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 bis FE <sub>H</sub>	User	BIN 16-bit	
n2	Station number of the remote station, where data is read from Range: When the RIRD instruction is executed in the master station: 1 to 64 When the RIRD instruction is executed in a local station: 0 to 64		User	BIN 16-bit	
d1	Head number of the devices where control data for the execution of this instruction and read data is stored.				
	Operand	Meaning	Description	Range	Contents is stored by
	(d1)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—	System
	(d1)+1	Number of points to read	Specify the number of data (unit:words) to read-out. This number depends on the type of CPU module mounted in the station where the data is read from: AnU, QnA series, System Q: max. 480 words All other CPUs: max. 32 words	1 to 480 1 to 32	User
	(d1)+2	Access code	<ul style="list-style-type: none"> <li>For a master module with software version A to H Set „0004<sub>H</sub>“ to access the buffer memory of an intelligent device station. Set „2004<sub>H</sub>“ to access the random access buffer memory of a local station.</li> </ul>	0004 <sub>H</sub> or 2004 <sub>H</sub>	
		Device code and access code	<ul style="list-style-type: none"> <li>For a master module with software version J or higher A device code is stored in the upper 8 bits of this device. The access code which, specifies whether to access the buffer memory of a CC-Link module (04<sub>H</sub>) or a CPU device (05<sub>H</sub>), is entered in the lower 8 bits.</li> </ul>	Higher byte: see the table below  Lower byte: 04 <sub>H</sub> or 05 <sub>H</sub>	
	(d1)+3	Head address	<ul style="list-style-type: none"> <li>For a master module with software version A to H Head address of the buffer memory</li> </ul>	Depends on the accessed station	
<ul style="list-style-type: none"> <li>For a master module with software version J or higher Head address of the buffer memory or first device number</li> </ul>					
(d1)+4 bis (d1)+n	Storage area for the read data	The size of this area is determined by the number of points to read stored in (d1)+1.	—	System	

Variables

Set Data	Meaning			Range	Contents is stored by	Data Type
d2	Bit device which is set for one scan after completion of the RIRD instruction. (d2)+1 indicates that an error has occurred during execution of the instruction.					Bit
	Operand	Meaning	Description	Range	Contents is stored by	
	(d2)+0	Instruction completed	Indicates the completion of the RIRD instruction ON: Instruction completed OFF: Instruction not completed	—	System	
(d2)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RIRD instruction ON: Abnormal completion OFF: Normal completion	—			

From software version J of the master module two codes (both stored in (d1)+2) are used to specify the data to read: The **access code** selects whether access is made to the buffer memory of a CC-Link module or the device memory in the CPU module. With the **device code** the area of the buffer memory or the device is designated:

- Access to the buffer memory of a CC-Link module (Access code: 04<sub>H</sub>)

Access to		Device code
Buffer memory in an intelligent device station		00 <sub>H</sub>
Buffer memory in a master or local station	Random access buffer	20 <sub>H</sub>
	Remote inputs	21 <sub>H</sub>
	Remote outputs	22 <sub>H</sub>
	Remote register	24 <sub>H</sub>
	Special link relays	63 <sub>H</sub>
	Special link register	64 <sub>H</sub>

- Access to the device memory of a CPU module (Access code: 05<sub>H</sub>)

Devices not indicated in the following table are not accessible. To access a bit device, specify „0“ or a multiple of „16“ as head device. Otherwise an error will occur.

Devices		Device type		Device code
Name	Symbol	Bit	Word	
Inputs	X	●		00 <sub>H</sub>
Outputs	Y	●		02 <sub>H</sub>
Internal relays	M	●		03 <sub>H</sub>
Latch relays	L	●		83 <sub>H</sub>
Link relays	B	●		23 <sub>H</sub>
Timer (contact)	T	●		09 <sub>H</sub>
Timer (coil)		●		0A <sub>H</sub>
Timer (present value)			●	
Counter (contact)	C	●		11 <sub>H</sub>
Counter (coil)		●		12 <sub>H</sub>
Counter (present value)			●	
Data register	D		●	04 <sub>H</sub>
Link register	W		●	24 <sub>H</sub>
File register	R		●	84 <sub>H</sub>

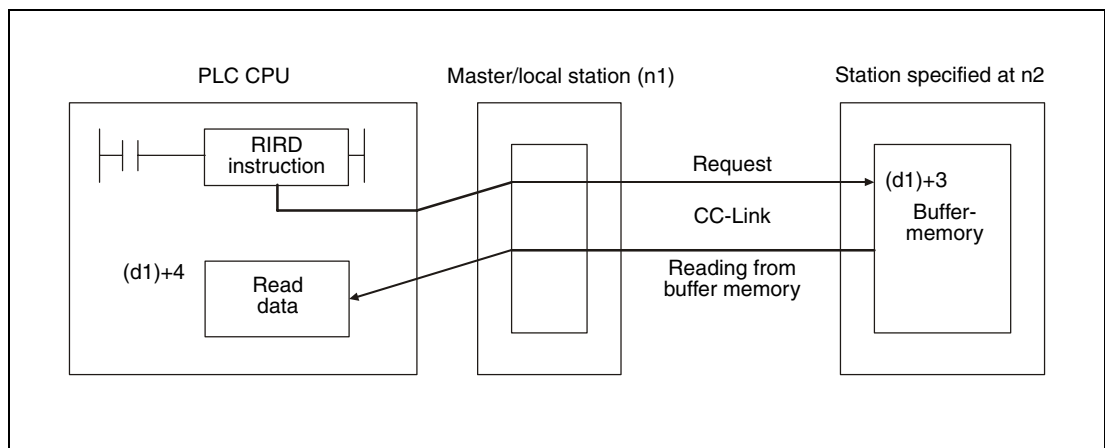
**Functions Read from buffer memory of intelligent device station or from device memory of PLC CPU**

**RIRD Data read**

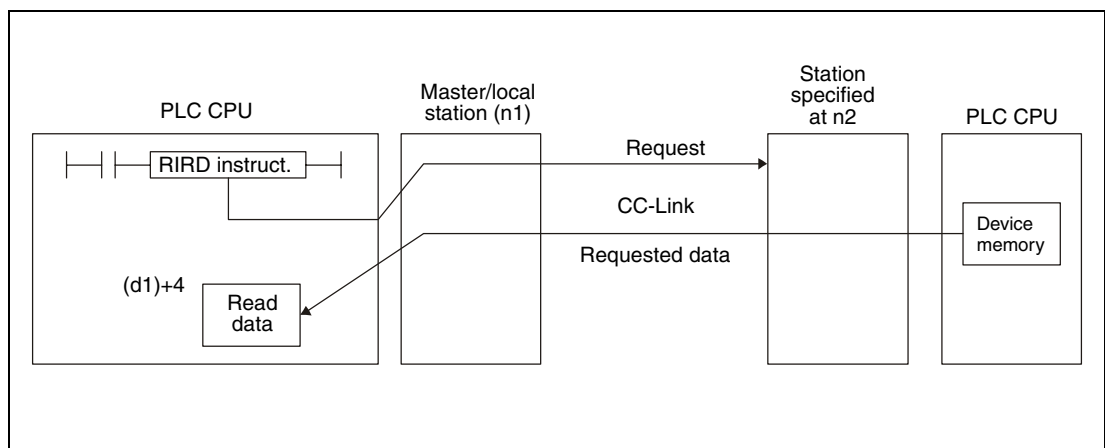
The RIRD instruction reads data from the buffer memory on an intelligent device station connected to the CC-Link. When a master module with a software version from J onward is used, it is also possible to access the device memory of the PLC CPU mounted in the other station.

The head address of the buffer memory or the head device is designated by (d1)+3. The station number of the other station is designated by n2. This station is connected to the master/local station specified at n1. The read data is stored in the CPU, which executes the RIRD instruction, to the devices starting from (d1)+4. The number of data to read is designated by (d1)+1.

- Function with software version A to H:



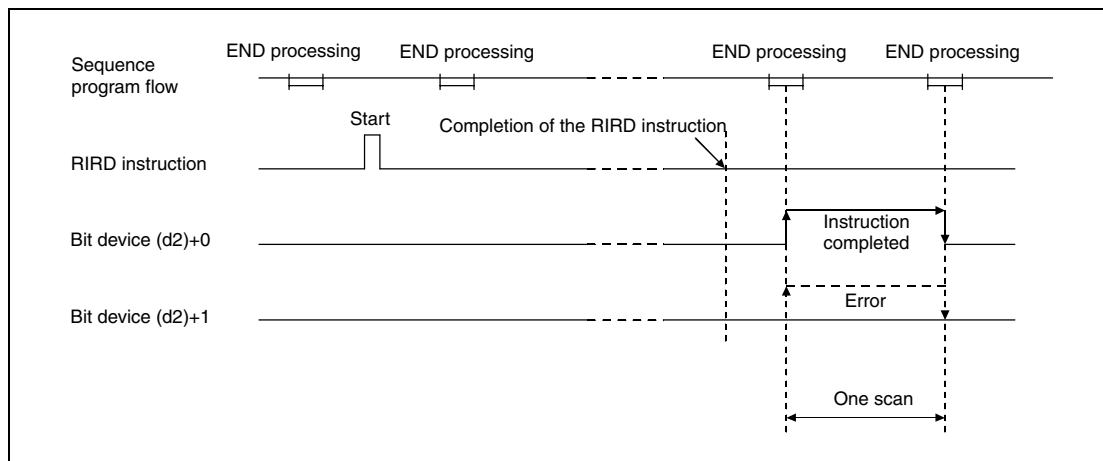
- Additional function with software version J and later:



Whether the execution of the RIRD instruction has been finished can be checked with the devices (d2)+0 and (d2)+1:

- The bit device (d2)+0 turns ON at the END processing of the scan in which the RIRD instruction has been completed and turns OFF at the next END processing.
- The bit device (d2)+1 indicates an error during execution of the RIRD instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the RIRD instruction, (d2)+1 turns ON at the END processing of the scan in which the RIRD instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RIRD instruction is being executed:



It is possible to execute RIRD instructions for multiple stations at the same time, but the same intelligent device station or local station cannot be accessed simultaneously from more than one station.

Set the network parameters by executing the RLPA instruction before executing an RIRD instruction.

When „0“ or a value outside the range from 1 to 480 is entered as number of data to read in (d1)+1, the device (d2)+1 is set at the completion of the RIRD instruction, thereby indicating an error.

#### Execution Conditions

When the LEDA instruction is used, the RIRD instruction is executed every scan while the read command is ON.

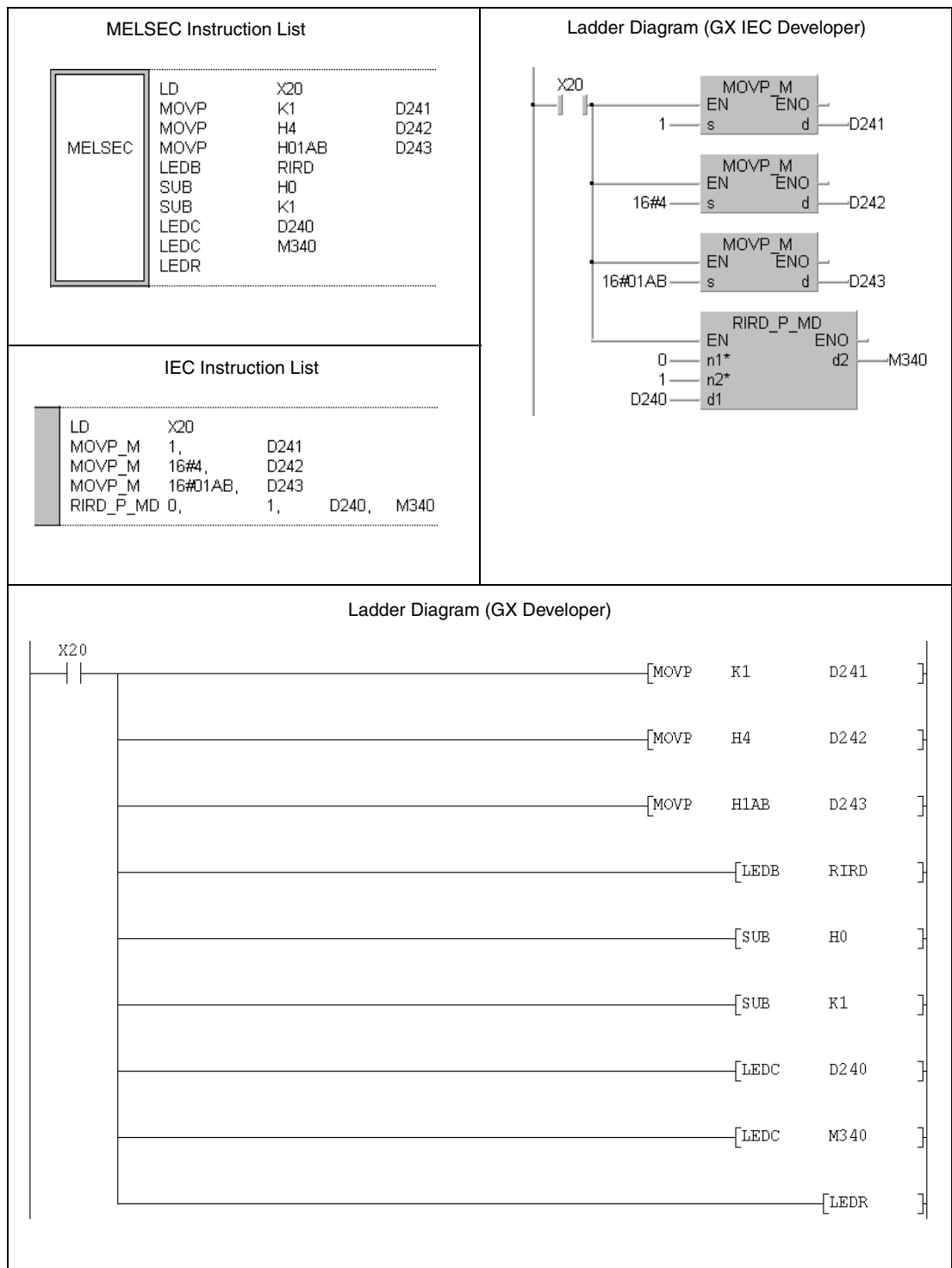
When the LEDB instruction is used, the RIRD instruction is executed only one scan on the leading edge (OFF -> ON) of the read command.

Note that the read processing executed by the RIRD instruction will take time for several scans before the processing is completed. Therefore, execute the next RIRD instruction only after the completion device (d2)+0 has been switched on.

**Program Example**

RIRD

The following program is executed in the PLC CPU of the master station. It reads the contents of the buffer memory address 1A8<sub>H</sub> from an intelligent device station with the station No. 1. The master module of the CC-Link occupies the I/O numbers from X/Y000 to X/Y01F.



For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.

11.5.5 RIRD (QnA series and System Q)

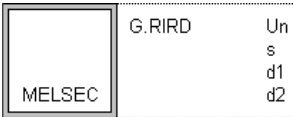
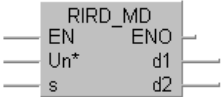
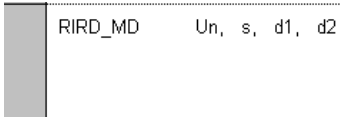
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

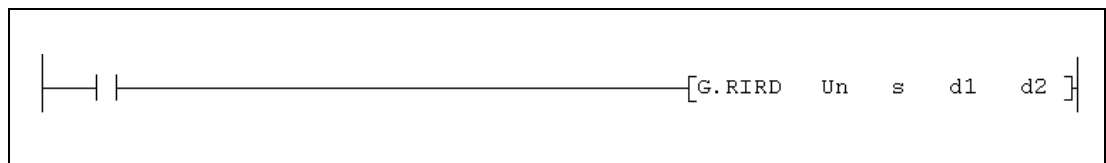
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	8
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	---	--

GX Developer



Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
Un	Head I/O address of the CC-Link master/local module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
s	Head number of the devices where control data for the execution of this instruction is stored				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—	System
	(s)+1	Station number	Station number of the remote station, where data is read from	0 to 64	User
	(s)+2	Access code	<ul style="list-style-type: none"> <li>For a A/Q series master module with software version A to H Set „0004<sub>H</sub>“ to access the buffer memory of an intelligent device station. Set „2004<sub>H</sub>“ to access the buffer memory of a local station.</li> </ul>	0004 <sub>H</sub> or 2004 <sub>H</sub>	
		Device code and access code	<ul style="list-style-type: none"> <li>For a A/Q series master module with software version J or higher or a module of System Q A device code is stored in the upper 8 bits of this device. The access code which, specifies whether to access the buffer memory of a CC-Link module (04<sub>H</sub>) or a CPU device (05<sub>H</sub>), is entered in the lower 8 bits.</li> </ul>	Higher byte: see the table below  Lower byte: 04 <sub>H</sub> or 05 <sub>H</sub>	
	(s)+3	Head address	<ul style="list-style-type: none"> <li>For a A/Q series master module with software version A to H Head address of the buffer memory</li> </ul>	Depends on the accessed station	
<ul style="list-style-type: none"> <li>For a A/Q series master module with software version J or higher or a module of System Q Head address of the buffer memory or first device number</li> </ul>					
(s)+4	Number of points to read	Specify the number of data (unit:words) to read-out. This number depends on the type of CPU module mounted in the station where the data is read from: AnU, QnA series, System Q: max. 480 words All other CPUs: max. 32 words	1 to 480 1 to 32		
d1	Head address of the area where the read data is stored		User	BIN 16-bit	

Variables

Set Data	Meaning			Range	Contents is stored by	Data Type
d2	Bit device which is set for one scan after completion of the RIRD instruction. (d2)+1 indicates that an error has occurred during execution of the instruction.					
	Set Data	Meaning	Description	Range	Contents is stored by	Bit
	(d2)+0	Instruction completed	Indicates the completion of the RIRD instruction ON: Instruction completed OFF: Instruction not completed	—	System	
(d2)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RIRD instruction ON: Abnormal completion OFF: Normal completion	—			

From software version J of the master module two codes (both stored in s+2) are used to specify the data to read: The **access code** selects whether access is made to the buffer memory of a CC-Link module or the device memory in the CPU module. With the **device code** the area of the buffer memory or the device is designated:

- Access to the buffer memory of a CC-Link module (Access code: 04<sub>H</sub>)

Access to		Device code
Buffer memory in an intelligent device station		00 <sub>H</sub>
Buffer memory in a master or local station	Random access buffer	20 <sub>H</sub>
	Remote inputs	21 <sub>H</sub>
	Remote outputs	22 <sub>H</sub>
	Remote register	24 <sub>H</sub>
	Link special relays	63 <sub>H</sub>
	Link special register	64 <sub>H</sub>

- Access to the device memory of a CPU module (Access code: 05<sub>H</sub>)

Devices not indicated in the following table are not accessible. To access a bit device, specify „0“ or a multiple of „16“ as head device. Otherwise an error will occur.

Device		Device type		Unit	Device code
Name	Symbol	Bit	Word		
Inputs	X	●		Hexadecimal	00 <sub>H</sub>
Outputs	Y	●			02 <sub>H</sub>
Internal relays	M	●		Decimal	03 <sub>H</sub>
Latch relays	L	●			83 <sub>H</sub>
Link relays	B	●		Hex.	23 <sub>H</sub>
Timer (contact)	T	●		Decimal	09 <sub>H</sub>
Timer (coil)		●			0A <sub>H</sub>
Timer (present value)			●		0C <sub>H</sub>
Retentive Timer (contact)	ST	●			89 <sub>H</sub>
Retentive Timer (coil)		●			8A <sub>H</sub>
Retentive Timer (present value)			●		8C <sub>H</sub>
Counter (contact)	C	●		Decimal	11 <sub>H</sub>
Counter (coil)		●			12 <sub>H</sub>
Counter (present value)			●		14 <sub>H</sub>
Data register	D		●		04 <sub>H</sub>
Link register	W		●	Hex.	24 <sub>H</sub>



Device		Device type		Unit	Device code
Name	Symbol	Bit	Word		
File register	R		●	Decimal	84 <sub>H</sub>
Special link relay	SB	●		Hexadecimal	63 <sub>H</sub>
Special link register	SW		●		64 <sub>H</sub>
Special relay	SM	●		Decimal	43 <sub>H</sub>
Special register	SD		●		44 <sub>H</sub>

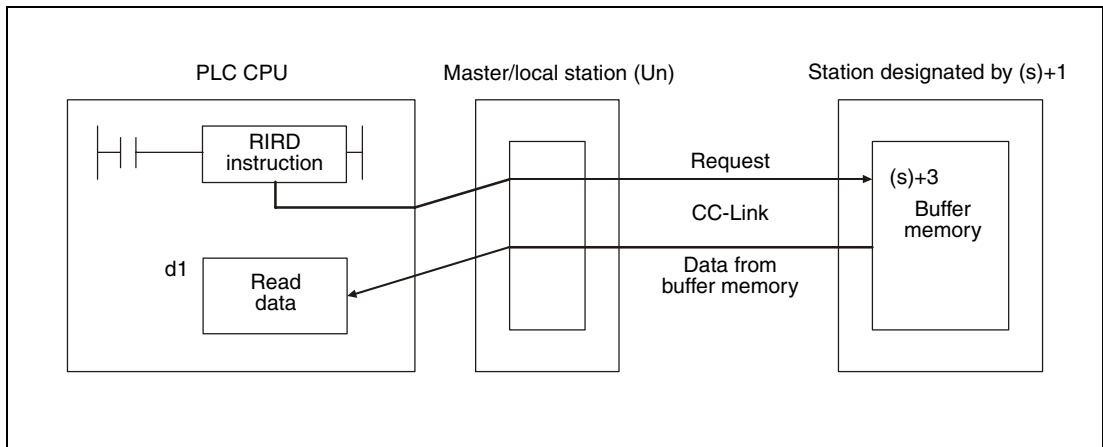
**Functions Read from buffer memory of intelligent device station or from device memory of PLC CPU**

**RIRD Data read**

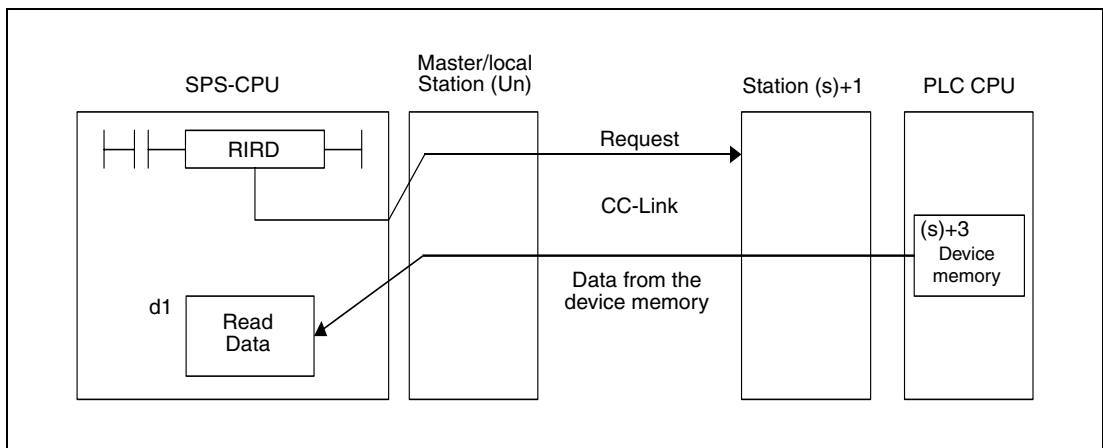
The RIRD instruction reads data from the buffer memory of an intelligent device connected to the CC-Link. When a master module with a software version from J onward or a CC-Link module of the MELSEC System Q is used, it is also possible to access the PLC CPU device memory of another station connected to the CC-Link network.

The head address of the buffer memory or the head device is designated by (s)+3. The station number of the other station is designated by (s)+1. This station is connected to the master/local station specified at Un. The read data is stored in the CPU which executes the RIRD instruction to the devices starting from d1. The number of data to read is designated by (s)+4.

- Accessing the buffer memory of an CC-Link module



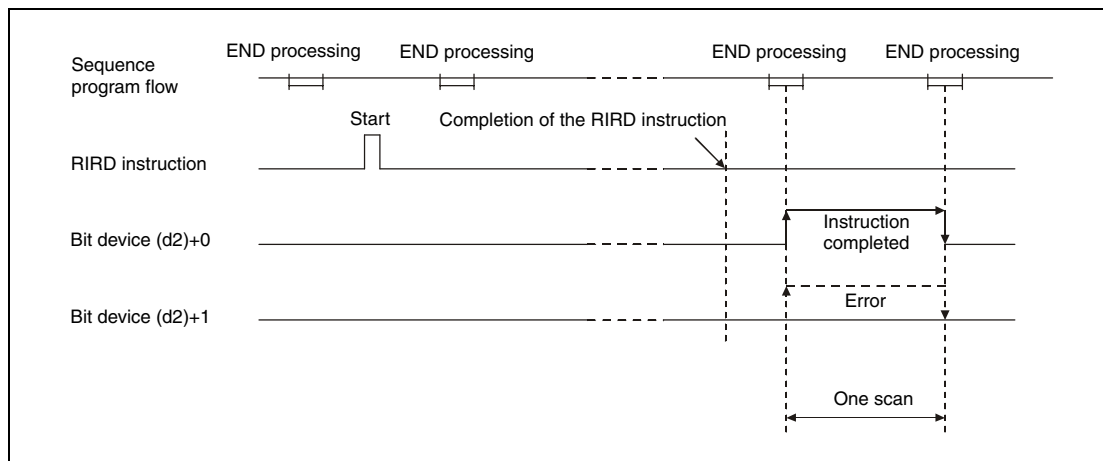
- Accessing the device memory in the PLC CPU of another station on CC-Link



Whether the execution of the RIRD instruction has been finished can be checked with the devices (d2)+0 and (d2)+1:

- The bit device (d2)+0 turns ON at the END processing of the scan in which the RIRD instruction has been completed and turns OFF at the next END processing.
- The bit device (d2)+1 indicates an error during execution of the RIRD instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the RIRD instruction, (d2)+1 turns ON at the END processing of the scan in which the RIRD instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RIRD instruction is being executed:



It is possible to execute RIRD instructions for multiple stations at the same time, but it is not possible to access the same intelligent device station or local station simultaneously from more than one station.

### Operation Error

In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When the module specified by Un is not an intelligent function module or a special function module. (error code: QnA series 2110, System Q 2112)
- When an attempt was made to execute an unsupported instruction (error code: 4002)
- When the number of devices in the instruction is incorrect. (error code: 4003)
- When the instruction specifies a device that cannot be used. (error code: 4004)
- When the area designated by s contains data that cannot be used. (error code: 4100)
- When the number of data set to be used exceeds the allowable range. (error code: 4101)
- When the storage data or constants of the device specified with the instruction exceeds the allowable range (error code: 4101)
- For QnA series only: Too many CC-Link related dedicated instructions are used. (error code: 4107).
- For QnA series only: The parameters for CC-Link are not set. (error code: 4108)

**Program Example**

**RIRD**

The following program is executed in the PLC CPU of the master station. When the input X0 is set the contents of 10 buffer memory addresses is read from the intelligent device station with the station number , starting with the buffer memory address 100<sub>H</sub>. The read data is stored in the PLC CPU from data register D0 onward. The head I/O number of the master module of CC-Link is X/Y40.

- IEC editors (On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

Ladder Diagram (GX IEC Developer)

The station number is written to (s)+1

To access the buffer memory of an intelligent device station the code 4<sub>H</sub> is written to (s)+2.

The head address of the buffer memory is stored in (s)+3.

Number of words to read

Read instruction

M100 is set to indicate that data is being read.

At this position, write the instructions that should be executed when the RIRD instruction has been completed normally.

At this position, write the instructions that should be executed when the RIRD instruction has been completed with an error.

When the reading of data has been completed M100 is reset.

---

IEC Instruction List

LD	X0		
ANDN	M100		
MOVP_M	1,	D101	
MOVP_M	16#4,	D102	
MOVP_M	16#100,	D103	
MOVP_M	10,	D104	
RIRD_P_MD	4,	var_D100, D0,	var_MD
SET_M	M100		

For an explanation of the devices and instructions used please see the above program example.

LD	M0		
ANDN	M1		
An instruction at this position will be executed when the RIRD instruction has been completed normally.			
LD	M0		
AND	M1		
An instruction at this position will be executed when the RIRD instruction has been completed anomalously.			
LD	M0		
RST_M	M100		

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

MELSEC Instruction List

MELSEC	LD	X0	
	ANI	M100	
	MOV P	K1	D101
	MOV P	H4	D102
	MOV P	H100	D103
	MOV P	K10	D104
	GP.RIRD	U4	D100
		D0	M0
	SET	M100	
		RST	M100
MELSEC	LD	M0	
	MPS		
	ANI	M1	
	An instruction at this position will be executed when the RIRD instruction has been completed normally.		
	MRD		
	AND M1		
	An instruction at this position will be executed when the RIRD instruction has been completed annormally.		
	RST	M100	

11.5.6 RIWT (A series)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

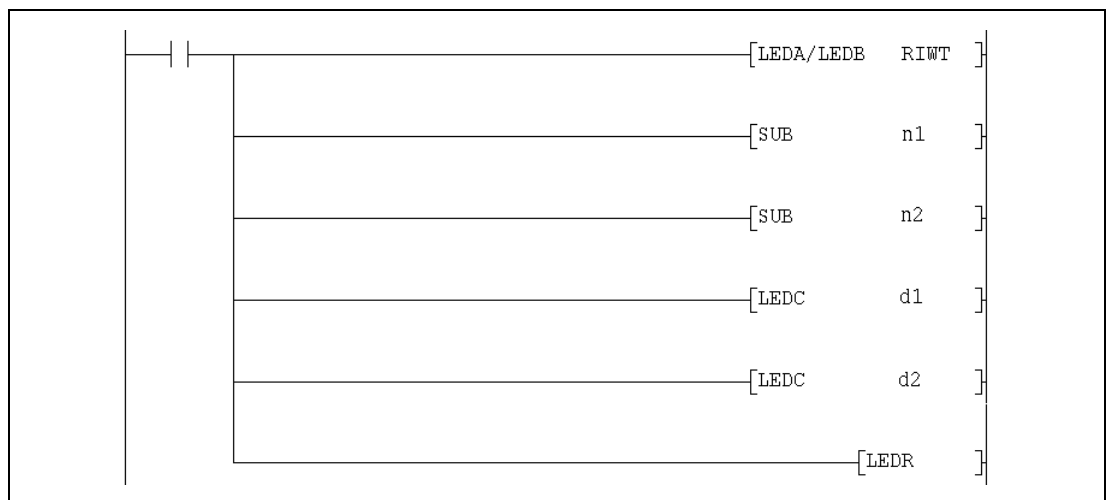
Devices  
MELSEC A

	Usable Devices																Digit designation	Number of steps	Index	Carry Flag	Error Flag
	Bit Devices						Word Devices (16-bit)						Constant	Pointer	Level						
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V					
n1																●	●				
n2																●	●				
d1							●	●	●	●	●										
d2	●	●	●	●	●																

GX IEC  
Developer

<p>MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center; width: 100px;">MELSEC</td> <td>LEDA/LEDB</td> <td>RIWT</td> </tr> <tr> <td></td> <td>SUB</td> <td>n1</td> </tr> <tr> <td></td> <td>SUB</td> <td>n2</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d1</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d2</td> </tr> <tr> <td></td> <td>LEDR</td> <td></td> </tr> </table>	MELSEC	LEDA/LEDB	RIWT		SUB	n1		SUB	n2		LEDC	d1		LEDC	d2		LEDR		<p>Ladder Diagram</p>	<p>IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 100px;">RIWT_MD</td> <td>n1, n2, d1, d2</td> </tr> </table>	RIWT_MD	n1, n2, d1, d2
MELSEC	LEDA/LEDB	RIWT																				
	SUB	n1																				
	SUB	n2																				
	LEDC	d1																				
	LEDC	d2																				
	LEDR																					
RIWT_MD	n1, n2, d1, d2																					

GX  
Developer



Variables

Set Data	Meaning	Range	Contents is stored by	Data Type
n1	Head I/O address of the CC-Link master/local module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit
n2	Station number of the remote station, where data is written to Range: When the RIRD instruction is executed in the master station: 1 to 64 When the RIRD instruction is executed in a local station: 0 to 64		User	BIN 16-bit

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type		
d1	Head number of the devices where control data for the execution of this instruction and read data is stored.				BIN 16-bit	
	Set Data	Meaning	Description	Range		Contents is stored by
	(d1)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—		System
	(d1)+1	Number of points to write	Specify the number of data (unit:words) to write. This number depends on the type of CPU module mounted in the station where the data is written to: AnU, QnA series, System Q: max. 480 words All other CPUs: max. 10 words	1 to 480 1 to 10		User
	(d1)+2	Access code	<ul style="list-style-type: none"> <li>For a master module with software version A to H Set „0004<sub>H</sub>“ to write to the buffer memory of an intelligent device station.</li> <li>Set „2004<sub>H</sub>“ to write to the buffer memory of a local station.</li> </ul>	0004 <sub>H</sub> or 2004 <sub>H</sub>		
		Device code and access code	<ul style="list-style-type: none"> <li>For a master module with software version J or higher A device code is stored in the upper 8 bits of this device. The access code, which specifies whether to access the buffer memory of a CC-Link module (04<sub>H</sub>) or a CPU device (05<sub>H</sub>), is entered in the lower 8 bits.</li> </ul>	Higher byte: see the table below  Lower byte: 04 <sub>H</sub> or 05 <sub>H</sub>		
	(d1)+3	Head address	<ul style="list-style-type: none"> <li>For a master module with software version A to H Head address of the buffer memory</li> <li>For a master module with software version J or higher Head address of the buffer memory or first device number</li> </ul>	Depends on the accessed station		
(d1)+4 bis (d1)+n	Storage area for the data to write	Specify the size of this area in (d1)+1	—	User		
d2	Bit device which is set for one scan after completion of the RIWT instruction. (d2)+1 indicates that an error has occurred during execution of the instruction.				Bit	
	Set Data	Meaning	Description	Range		Contents is stored by
	(d2)+0	Instruction completed	Indicates the completion of the RIWT instruction ON: Instruction completed OFF: Instruction not completed	—		System
(d2)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RIWT instruction ON: Abnormal completion OFF: Normal completion	—			

From software version J of the master module two codes (both stored in (d1)+2) are used to specify the target for the data: The **access code** selects whether data is written to the buffer memory of a CC-Link module or the device memory in the CPU module. With the **device code** the area of the buffer memory or the devices, which will be overwritten, is designated:

- Writing to the buffer memory of a CC-Link module (Access code: 04<sub>H</sub>)

Access to		Device code
Buffer memory in an intelligent device station		00 <sub>H</sub>
Buffer memory in a master or local station	Random access buffer	20 <sub>H</sub>
	Remote inputs	21 <sub>H</sub>
	Remote outputs	22 <sub>H</sub>
	Remote register	24 <sub>H</sub>
	Special link relays	63 <sub>H</sub>
	Special link register	64 <sub>H</sub>

- Access to the device memory of a CPU module (Access code: 05<sub>H</sub>)  
 Devices not indicated in the following table are not accessible. To access a bit device, specify „0“ or a multiple of „16“ as head device. Otherwise an error will occur.

Devices		Device type		Device code
Name	Symbol	Bit	Word	
Inputs	X	●		00 <sub>H</sub>
Outputs	Y	●		02 <sub>H</sub>
Internal relays	M	●		03 <sub>H</sub>
Latch relays	L	●		83 <sub>H</sub>
Link relays	B	●		23 <sub>H</sub>
Timer (contact)	T	●		09 <sub>H</sub>
Timer (coil)		●		0A <sub>H</sub>
Timer (present value)			●	
Counter (contact)	C	●		11 <sub>H</sub>
Counter (coil)		●		12 <sub>H</sub>
Counter (present value)			●	
Data register	D		●	04 <sub>H</sub>
Link register	W		●	24 <sub>H</sub>
File register	R		●	84 <sub>H</sub>



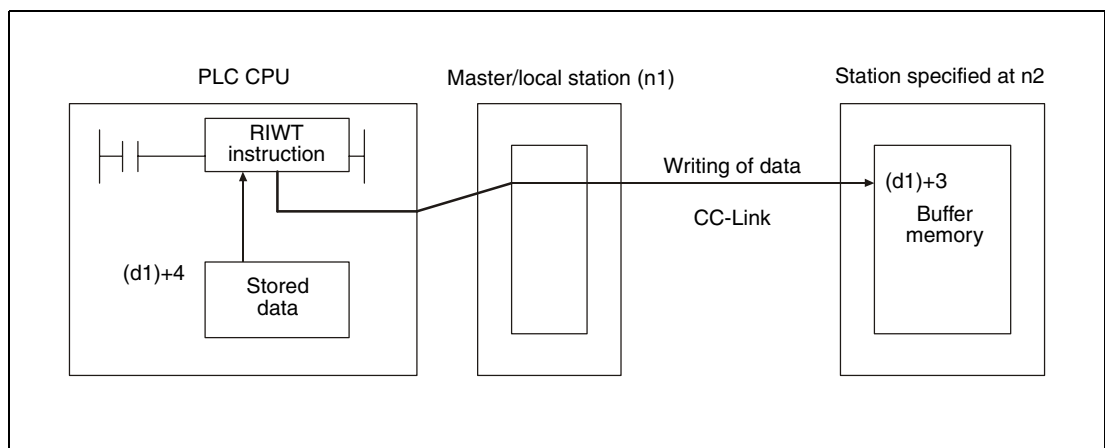
**Functions Write to buffer memory of intelligent device station or to device memory of PLC CPU**

**RIWT Data write**

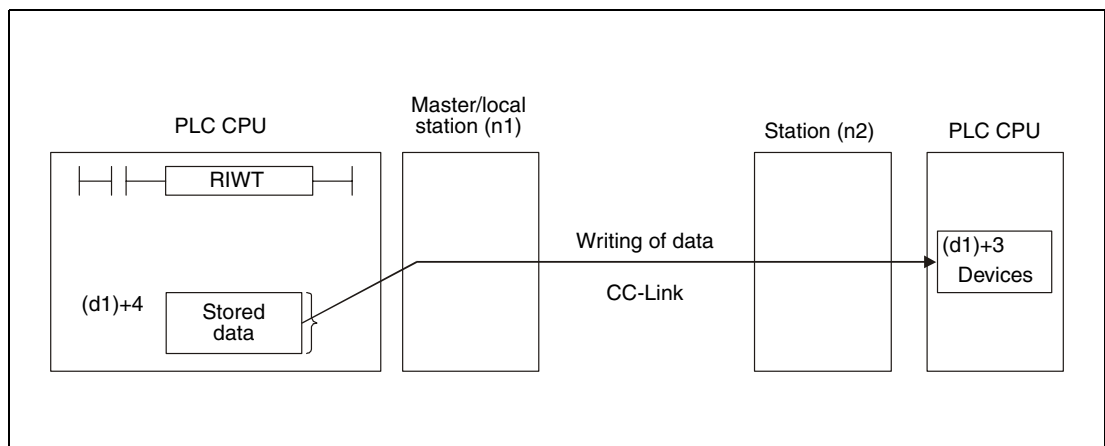
The RIRD instruction writes data to the buffer memory of an intelligent device station connected to the CC-Link. When a master module with a software version from J onward is used, it is also possible to write to the device memory of the PLC CPU mounted in the other station.

The station number of the other station is designated by n2. This station is connected to the master/local station specified at n1. The data for this station is stored in the CPU, which executes the RIWT instruction, in the devices starting from (d1)+4. The number of data to write is designated by (d1)+1. The head address of the buffer memory or the head number of the devices is designated by (d1)+3.

- Function with software version A to H:



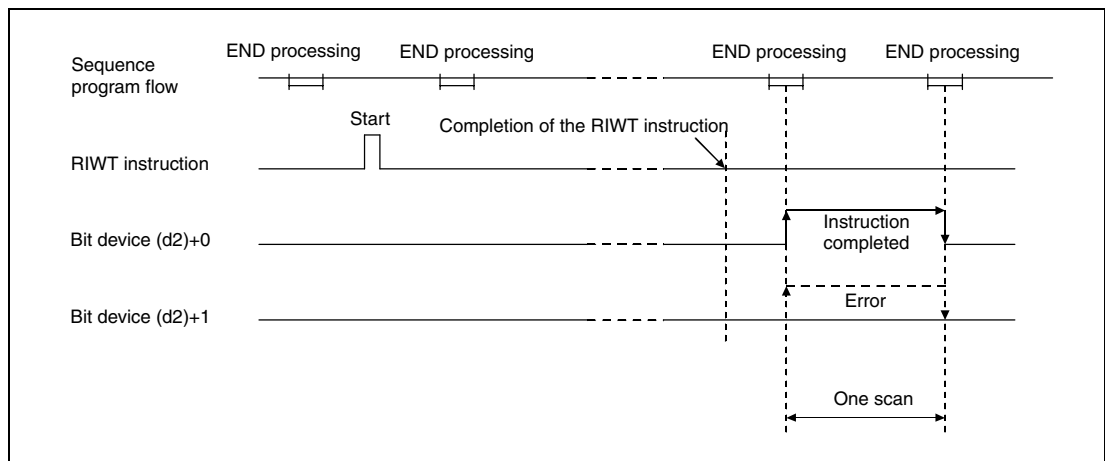
- Additional function with software version J and later:



Whether the execution of the RIWT instruction has been finished can be checked with the devices (d2)+0 and (d2)+1:

- The bit device (d2)+0 turns ON at the END processing of the scan in which the RIWT instruction has been completed and turns OFF at the next END processing.
- The bit device (d2)+1 indicates an error during execution of the RIWT instruction. When the instruction has been completed normal, this device stays OFF, but when an error occurs during execution of the RIWT instruction, (d2)+1 turns ON at the END processing of the scan in which the RIWT instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RIWT instruction is being executed:



It is possible to execute RIWT instructions for multiple stations at the same time, but it is not possible to access the same intelligent device station or local station simultaneously from more than one station.

Set the network parameters by executing the RLPA instruction before executing an RIWT instruction.

When „0“ or a value outside the range from 1 to 480 is entered as number of data to write in (d1)+1, the device (d2)+1 is set at the completion of the RIRD instruction, thereby indicating an error.

**Execution Conditions**

When the LEDA instruction is used, the RIWT instruction is executed every scan while the read command is ON.

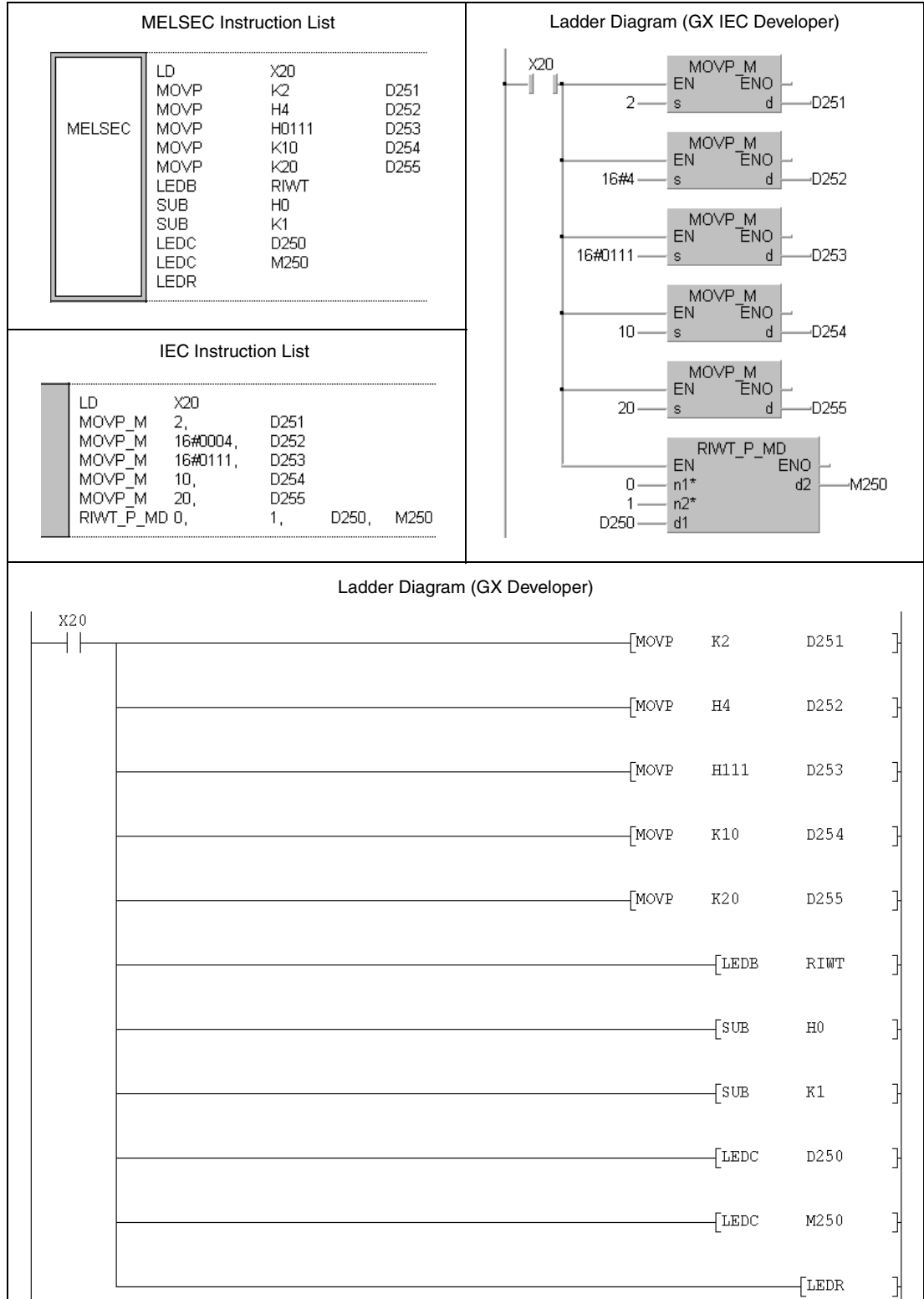
When the LEDB instruction is used, the RIWT instruction is executed only one scan on the leading edge (OFF -> ON) of the read command.

Note that the write processing executed by the RIWT instruction will take time for several scans before the processing is completed. Therefore, execute the next RIWT instruction only after the completion device (d2)+0 has been switched on.

**Program Example**

**RIWT**

The following program, which is executed by the PLC CPU of the master station, writes the value „10“ to the address 111<sub>H</sub> and the value „20“ to the address 112<sub>H</sub> of the buffer memory of an intelligent device station bearing the station number 1. The master module of the CC-Link occupies the I/O numbers from X/Y000 to X/Y01F.



For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.

11.5.7 RIWT (QnA series and System Q)

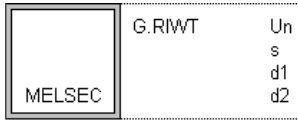
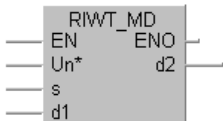
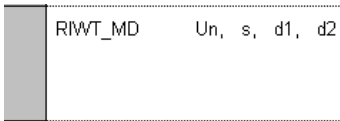
CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

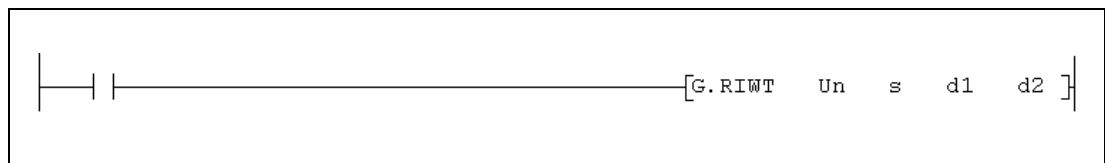
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s	—	●	●	—	—	—	—	—	—	SM0	8
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p>MELSEC Instruction List</p> 	<p>Ladder Diagram</p> 	<p>IEC Instruction List</p> 
---	---	--

GX Developer



Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
Un	Head I/O address of the CC-Link master/local module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
s	Head number of the devices where control data for the execution of this instruction is stored				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—	System
	(s)+1	Station number	Station number of the remote station, where data is written to.	0 to 64	User
	(s)+2	Access code	<ul style="list-style-type: none"> <li>For a A/Q series master module with software version A to H Set „0004<sub>H</sub>“ to write to the buffer memory of an intelligent device station. Set „2004<sub>H</sub>“ to write to the buffer memory of a local station.</li> </ul>	0004 <sub>H</sub> or 2004 <sub>H</sub>	
		Device code and access code	<ul style="list-style-type: none"> <li>For a A/Q series master module with software version J or higher or a module of System Q A device code is stored in the upper 8 bits of this device. The access code, which specifies whether to access the buffer memory of a CC-Link module (04<sub>H</sub>) or a CPU device (05<sub>H</sub>), is entered in the lower 8 bits.</li> </ul>	Higher byte: see the table below  Lower byte: 04 <sub>H</sub> or 05 <sub>H</sub>	
	(s)+3	Head address	<ul style="list-style-type: none"> <li>For a A/Q series master module with software version A to H Head address of the buffer memory</li> </ul>	Depends on the accessed station	
<ul style="list-style-type: none"> <li>For a A/Q series master module with software version J or higher or a module of System Q Head address of the buffer memory or head device</li> </ul>					
(s)+4	Datenlänge	Specify the number of data (unit:words) to write. This number depends on the type of CPU module mounted in the station where the data is written to: AnU, QnA series, System Q: max. 480 words All other CPUs: max. 32 words	1 to 480 1 to 10		
d1	Head address of the area where the write data is stored		User	BIN 16-bit	

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type			
d2	Bit device which is set for one scan after completion of the RIWT instruction. (d2)+1 indicates that an error has occurred during execution of the instruction.			System	Bit		
	Set Data	Meaning	Description			Range	Contents is stored by
	(d2)+0	Instruction completed	Indicates the completion of the RIWT instruction ON: Instruction completed OFF: Instruction not completed			—	
(d2)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RIWT instruction ON: Abnormal completion OFF: Normal completion	—				

From software version J of the master module two codes (both stored in (d1)+2) are used to specify the target for the data: The **access code** selects whether data is written to the buffer memory of a CC-Link module or the device memory in the CPU module. With the **device code** the area of the buffer memory or the devices, which will be overwritten, is designated:

- Access to the buffer memory of a CC-Link module (Access code: 04<sub>H</sub>)

Access to	Device code	
Buffer memory in an intelligent device station	00 <sub>H</sub>	
Buffer memory in a master or local station	Random access buffer	20 <sub>H</sub>
	Remote inputs	21 <sub>H</sub>
	Remote outputs	22 <sub>H</sub>
	Remote register	24 <sub>H</sub>
	Link special relays	63 <sub>H</sub>
	Link special register	64 <sub>H</sub>

- Access to the device memory of a CPU module (Access code: 05<sub>H</sub>)

Devices not indicated in the following table are not accessible. To access a bit device, specify „0“ or a multiple of „16“ as head device. Otherwise an error will occur.

Device		Device type		Unit	Device code
Name	Symbol	Bit	Word		
Inputs	X	●		Hexadecimal	00 <sub>H</sub>
Outputs	Y	●			02 <sub>H</sub>
Internal relays	M	●		Decimal	03 <sub>H</sub>
Latch relays	L	●			83 <sub>H</sub>
Link relays	B	●		Hex.	23 <sub>H</sub>
Timer (contact)	T	●		Decimal	09 <sub>H</sub>
Timer (coil)		●			0A <sub>H</sub>
Timer (present value)			●		0C <sub>H</sub>
Retentive Timer (contact)	ST	●			89 <sub>H</sub>
Retentive Timer (coil)		●			8A <sub>H</sub>
Retentive Timer (present value)			●		8C <sub>H</sub>
Counter (contact)	C	●		Decimal	11 <sub>H</sub>
Counter (coil)		●			12 <sub>H</sub>
Counter (present value)			●		14 <sub>H</sub>
Data register	D		●		04 <sub>H</sub>

Device		Device type		Unit	Device code
Name	Symbol	Bit	Word		
Link register	W		●	Hex.	24 <sub>H</sub>
File register	R		●	Decimal	84 <sub>H</sub>
Special link relay	SB	●		Hexadecimal	63 <sub>H</sub>
Special link register	SW		●		64 <sub>H</sub>
Special relay	SM	●		Decimal	43 <sub>H</sub>
Special register	SD		●		44 <sub>H</sub>

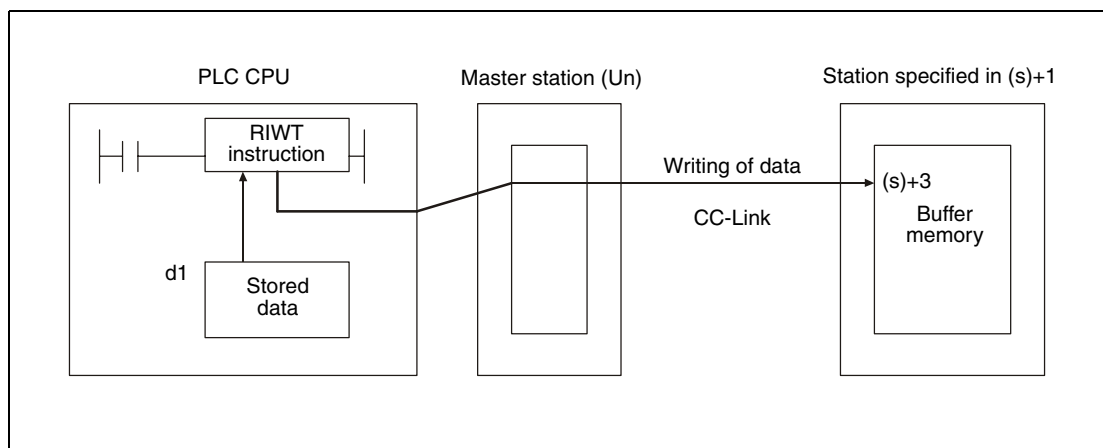
**Functions Write to buffer memory of intelligent device station or to device memory of PLC CPU**

**RIWT Data write**

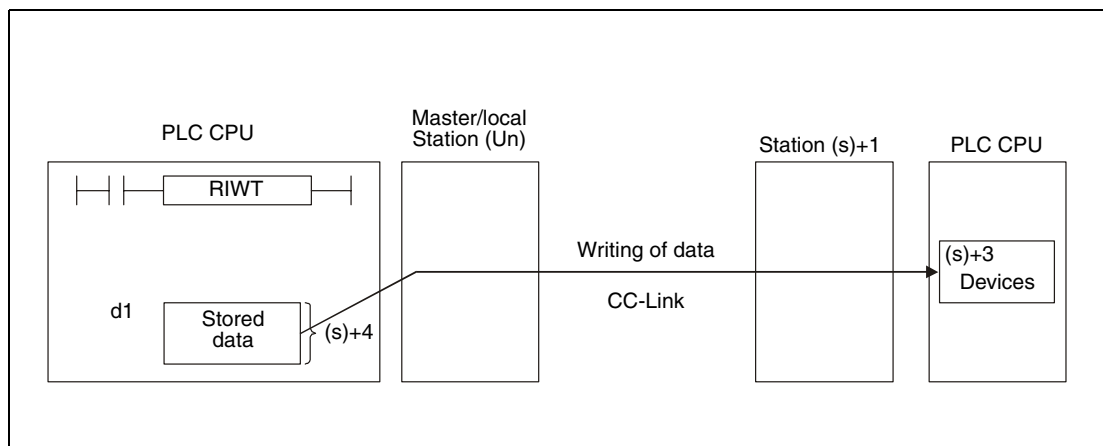
The RIWT instruction writes data to the buffer memory of an intelligent device connected to the CC-Link. When a master module with a software version from J onward or a CC-Link module of the MELSEC System Q is used, it is also possible to write to the PLC CPU device memory of another station connected to the CC-Link network.

The station number of the other station is designated by (s)+1. This station is connected to the master/local station specified at Un. Where the write data are is stored is designated by d1. At (s)+2 a code is stored which specifies whether to write to a buffer memory or to the device memory of a CPU module. The head address of the buffer memory or the head device is designated by (s)+3. The number of data to write is designated by (s)+4.

- Accessing the buffer memory of an CC-Link module



- Accessing the device memory in the PLC CPU of another station on CC-Link

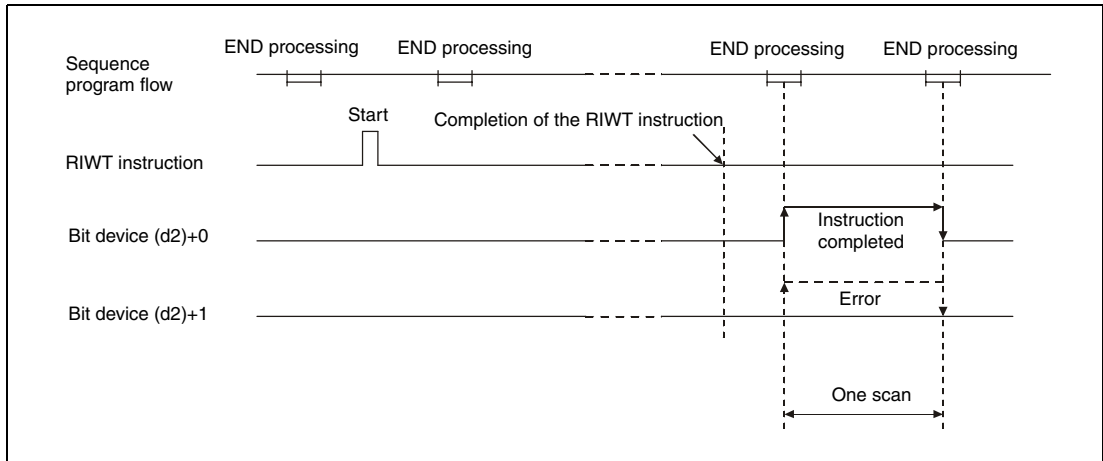


Whether the execution of the RIWT instruction has been finished can be checked with the devices (d2)+0 and (d2)+1:

- The bit device (d2)+0 turns ON at the END processing of the scan in which the RIWT instruction has been completed and turns OFF at the next END processing.
- The bit device (d2)+1 indicates an error during execution of the RIWT instruction. When the instruction has been completed normal, this device stays OFF, but when an error occurs during execution of the RIWT instruction, (d2)+1 turns ON at the END processing of the scan in which the RIWT instruction has been completed and turns OFF at the next END processing.



The following figure shows the timing when the RIWT instruction is being executed:



Please note, that it's possible to execute RIWT instructions for multiple stations at the same time, but the same intelligent device station or local station cannot be accessed simultaneously from more than one station.

**Operation Error**

In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When the module specified by Un is not an intelligent function module or a special function module. (error code: 2112)
- When an attempt was made to execute an unsupported instruction (error code: 4002)
- When the number of devices in the instruction is incorrect. (error code: 4003)
- When the instruction specifies a device that cannot be used. (error code: 4004)
- When the area designated by s contains data that cannot be used. (error code: 4100)
- When the number of data set to be used exceeds the allowable range. (error code: 4101)
- When the storage data or constants of the device specified with the instruction exceeds the allowable range (error code: 4101)

**Program Example**

**RIWT**

The following program is processed in the PLC CPU of the master station. When the input X0 is set, the contents of the data registers D0 to D9 is moved to the intelligent device station number 1 and stored to the buffer memory addresses 100<sub>H</sub> to 109<sub>H</sub>. The head I/O number of the master module of CC-Link is X/Y40.

● IEC editors

(On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

Ladder Diagram (GX IEC Developer)

The station number is written to (s)+1

To access the buffer memory of an intelligent device station the code 4<sub>H</sub> is written to (s)+2.

The head address of the buffer memory is stored in (s)+3.

Number of points to write

Writing to the buffer memory

M100 is set to indicate that data is being written.

At this position, write the instructions that should be executed when the RIWT instruction has been completed normally.

At this position, write the instructions that should be executed when the RIWT instruction has been completed with an error.

When the writing has been completed M100 is reset.

---

IEC Instruction List

LD	X0		
ANDN	M100		
MOV P M	1,	D101	
MOV P M	16#4,	D102	
MOV P M	16#100,	D103	
MOV P M	10,	D104	
RIWT P MD	4,	var_D100, D0,	var_MD
SET M	M100		

For an explanation of the devices and instructions used please see the above program example.

LD	MD		
ANDN	M1		
An instruction at this position will be executed when the RIWT instruction has been completed normally.			
LD	MD		
AND	M1		
An instruction at this position will be executed when the RIWT instruction has been completed anomalously.			
LD	MD		
RST M	M100		

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page

Ladder Diagram (GX Developer)

At this position, write the instructions that should be executed when the RIWT instruction has been completed normally.

At this position, write the instructions that should be executed when the RIWT instruction has been completed with an error.

---

MELSEC Instruction List

MELSEC	LD	X0		
	ANI	M100		
	MOV P	K1	D101	
	MOV P	H4	D102	
	MOV P	H100	D103	
	MOV P	K10	D104	
	GP.RIWT	U4	D100	D0
			D0	M0
	SET	M100		
MELSEC	LD	M0		
	MPS			
	ANI	M1		
	MRD			
	AND	M1		
	MPP			
	RST	M100		

11.5.8 RIRCV (A series)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

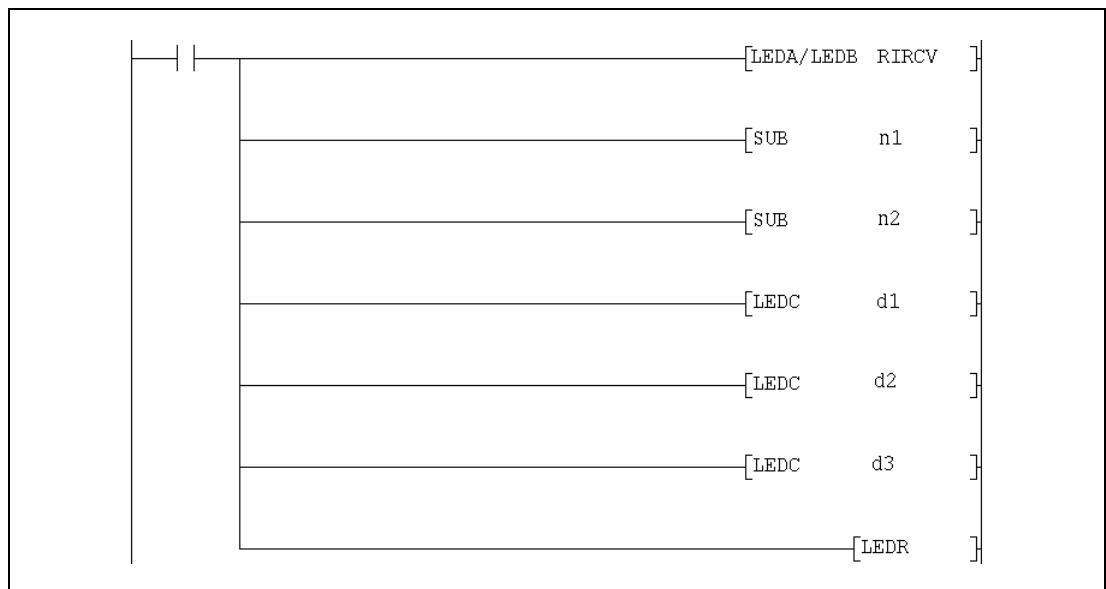
Devices  
MELSEC A

	Usable Devices																	Digit designation	Number of steps	Index	Carry Flag M9012	Error Flag M9011			
	Bit Devices							Word Devices (16-bit)							Constant		Pointer						Level		
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V	K						H (16#)	P	I
n1																	●	●				29			
n2																	●	●							
d1							●	●	●	●	●														●
d2							●	●	●	●	●														
d3	●	●	●	●	●																				

GX IEC  
Developer

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LEDA/LEDB</td> <td>RIRCV</td> </tr> <tr> <td></td> <td>SUB</td> <td>n1</td> </tr> <tr> <td></td> <td>SUB</td> <td>n2</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d1</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d2</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d3</td> </tr> <tr> <td></td> <td>LEDR</td> <td></td> </tr> </table>	MELSEC	LEDA/LEDB	RIRCV		SUB	n1		SUB	n2		LEDC	d1		LEDC	d2		LEDC	d3		LEDR		<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">RIRCV_MD</td> <td>n1, n2, d1, d2, d3</td> </tr> </table>	RIRCV_MD	n1, n2, d1, d2, d3
MELSEC	LEDA/LEDB	RIRCV																							
	SUB	n1																							
	SUB	n2																							
	LEDC	d1																							
	LEDC	d2																							
	LEDC	d3																							
	LEDR																								
RIRCV_MD	n1, n2, d1, d2, d3																								

GX  
Developer



Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
n1	Head I/O address of the CC-Link master module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
n2	Station number of the intelligent device station where data is read from	1 to 64	User	BIN 16-bit	
d1	Head number of the devices where control data for the execution of this instruction and read data is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—	System
	(d1)+1	Number of points to read	Specify how much data (in the unit „words“) should be read from the intelligent device station. Set a value within the intelligent device station buffer memory capacity and the parameter-set receiving buffer area of the master station.	1 to 480	User
	(d1)+2	Access code	Enter the value „0004 <sub>H</sub> “ (Read from the buffer memory of an intelligent device station.)	0004 <sub>H</sub>	
	(d1)+3	Error check	Specify the device which will indicate that an error has occurred during execution of the RIRCV instruction: 0: Device d1 1: Device RX+1	0 or 1	
	(d1)+4	Head address	Head address in the buffer memory (Address of the first data to read)	Depends on the accessed station	
(d1)+5 to (d1)+n	Storage area for the read data	The size of this area is determined by the number of points to read stored in (d1)+1.	—	System	
d2	Link devices used for handshaking				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d2)+0	Remote input (RX) and remote output (RY)	<ul style="list-style-type: none"> <li>Higher byte Set a remote input (RX) of the intelligent device station</li> </ul>	0 to 124	User (The RX, RY and RW <sub>r</sub> numbers are specified by the user but setting and resetting is performed by the system.)
<ul style="list-style-type: none"> <li>Lower byte Set a remote output (RY) of the intelligent device station</li> </ul>			0 to 125		
(d2)+1	Remote register (RW <sub>r</sub> )	Specify a remote register (RW <sub>r</sub> ) of the intelligent device station	0 to 15 or FF (When FF is set, no number is specified.)		

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
d3	Bit device which is set for one scan after completion of the RIRCV instruction. (d3)+1 indicates that an error has occurred during execution of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d3)+0	Instruction completed	Indicates the completion of the RIRCV instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d3)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RIRCV instruction ON: Abnormal completion OFF: Normal completion	—		

Functions

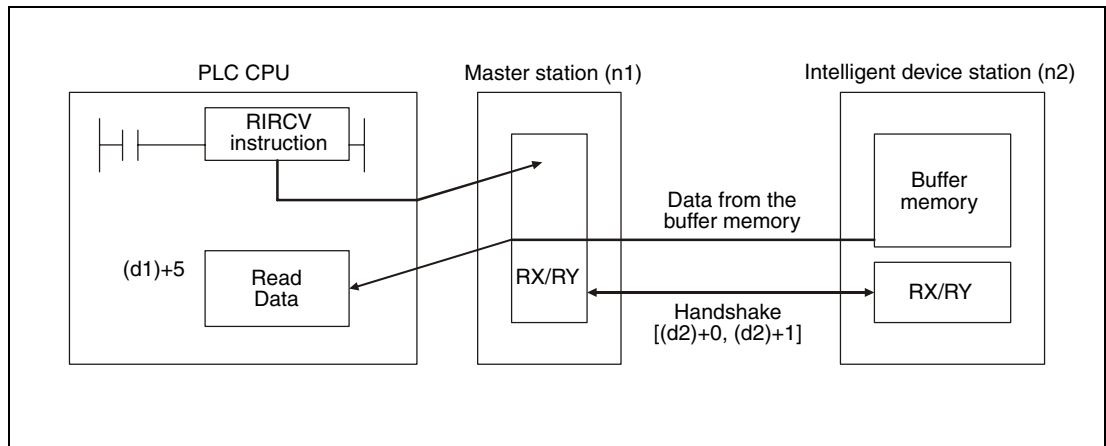
Reading of data from the buffer memory of an intelligent device station (with handshake)

RIRCV Data read (with handshake)

The execution of a RIRCV instruction is only possible in the PLC CPU of the master station. This instruction is used to read data from the buffer memory on an intelligent device station. The data exchange is controlled by a handshaking device.

The number of points to read is stored in (d1)+1. The head buffer memory address, which is specified in (d1)+3, is the first address to read from. The station number of the intelligent device station is designated by n2. This station is connected to the master station specified at n1. The read data is stored in the CPU, which executes the RIRCV instruction, to the devices starting from (d1)+5.

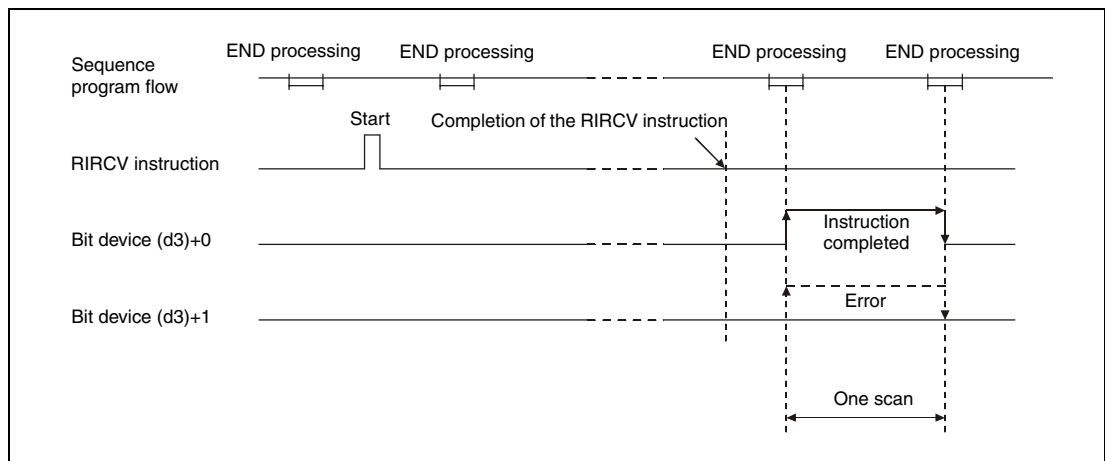
Function of the RIRCV instruction:



Whether the execution of the RIRCV instruction has been finished can be checked with the devices (d3)+0 and (d3)+1:

- The bit device (d3)+0 turns ON at the END processing of the scan in which the RIRCV instruction has been completed and turns OFF at the next END processing.
- The bit device (d3)+1 indicates an error during execution of the RIRCV instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the RIRCV instruction, (d3)+1 turns ON at the END processing of the scan in which the RIRCV instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RIRCV instruction is being executed:



Although it's possible to execute RIRCV instructions for multiple intelligent device stations at the same time, it's not possible to access the same intelligent device station simultaneously from more than one station.

**Execution Conditions**

When the LEDA instruction is used, the RIRCV instruction is executed every scan while the read command is ON.

When the LEDB instruction is used, the RIRCV instruction is executed only one scan on the leading edge (OFF -> ON) of the read command.

Note that the read processing executed by the RIRCV instruction will take time for several scans before the processing is completed. Therefore, execute the next RIRCV instruction only after the completion device (d3)+0 has been switched on. (A RIRCV instruction will not be processed if the execution is started while another RIRCV instruction is being executed.)

**Operation Error**

Before executing the RIRCV instruction, set the network parameters using the RLPA instruction. If the RIRCV instruction is executed without the parameters set, the device (d3)+1 will be set after completion of the instruction and the device designated by (d1)+0 will hold the error code 4B00<sub>H</sub>.

When „0“ or a value outside the range from 1 to 480 is entered as number of data to read in (d1)+1, the device (d3)+1 will be set and the error code BB42<sub>H</sub> will be stored in (d1)+0 at the completion of the RIRCV instruction.

**Program Example**

**RIRCV**

The following program, which is executed in the PLC CPU of the master station, reads the contents of the buffer memory addresses 400<sub>H</sub> to 405<sub>H</sub> from an intelligent device station (station number 1) when X20 is ON. The devices RX2, RY2 and RWr2 are used for the handshake. An error is indicated by the device designated by (d1)+0. To the master module of CC-Link, the head I/O number X/Y000 is assigned.

● IEC editors

(On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

Ladder Diagram (GX IEC Developer)

The number of points to read (6 words) is stored in (d1)+1.

The access code 4<sub>H</sub> for access to the buffer memory of an intelligent device station is stored in (d1)+2.

„0“ is stored to (d1)+3, therefore an error will be indicated by the device designated by (d1)+0.

The head address in the buffer memory is specified in (d1)+4.

RX2 and RY2 are stored in (d2)+0.

In (d2)+1, RWr2 is specified as handshake device.

Reading of the buffer memory

IEC Instruction List

LD	X20			
MOV_P_M	6,	D221		
MOV_P_M	16#0004,	D222		
MOV_P_M	0,	D223		
MOV_P_M	16#0400,	D224		
MOV_P_M	16#0202,	D320		
MOV_P_M	2,	D321		
RIRCV_P_MD	0,	1,	D220,	D320,
				M320

For an explanation of the devices and instructions used please see the above program example.



- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

```

graph TD
    X20((X20)) --- I1[MOV P K6 D221]
    I1 --- I2[MOV P H4 D222]
    I2 --- I3[MOV P K0 D223]
    I3 --- I4[MOV P H400 D224]
    I4 --- I5[MOV P H202 D320]
    I5 --- I6[MOV P K2 D321]
    I6 --- I7[LE DB RIRCV]
    I7 --- I8[SUB H0]
    I8 --- I9[SUB K1]
    I9 --- I10[LE DC D220]
    I10 --- I11[LE DC D320]
    I11 --- I12[LE DC M320]
    I12 --- I13[LE DR]
    
```

---

MELSEC Instruction List

	LD	X20	
MELSEC	MOV P	K6	D221
	MOV P	H0004	D222
	MOV P	K0	D223
	MOV P	H0400	D224
	MOV P	H0202	D320
	MOV P	K2	D321
	LE DB		RIRCV
	SUB	H0	
	SUB	K1	
	LE DC		D220
	LE DC		D320
	LE DC		M320
	LE DR		

For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.

11.5.9 RIRCV (QnA series and System Q)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
s1	—	●	●	—	—	—	—	—	—	SM0	10
s1	—	●	●	—	—	—	—	—	—		
d1	—	●	●	—	—	—	—	—	—		
d2	●	●	●	—	—	—	—	—	—		

GX IEC Developer

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center; width: 50px; height: 50px;">MELSEC</td> <td style="padding-left: 10px;">G.RIRCV</td> <td style="padding-left: 10px;">Un s1 d1 s2 d2</td> </tr> </table>	MELSEC	G.RIRCV	Un s1 d1 s2 d2	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 50px; height: 50px;"></td> <td style="padding-left: 10px;">RIRCV_MD</td> <td style="padding-left: 10px;">Un, s1, s2, d1, d2</td> </tr> </table>		RIRCV_MD	Un, s1, s2, d1, d2
MELSEC	G.RIRCV	Un s1 d1 s2 d2						
	RIRCV_MD	Un, s1, s2, d1, d2						

GX Developer

--

Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
Un	Head I/O address of the CC-Link master module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Head number of the devices where control data for the execution of this instruction is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s1)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—	System
	(s1)+1	Station number	Station number of the intelligent device station where data is read from	0 to 64	User
	(s1)+2	Access code	Enter the value „0004 <sub>H</sub> “ (Read from the buffer memory of an intelligent device station.)	0004 <sub>H</sub>	
	(s1)+3	Head address	Head address in the buffer memory (Address of the first data to read)	Depends on the accessed station	
(s1)+4	Number of points to read	Specify how much data (in the unit „words“) should be read from the intelligent device station. Set a value within the intelligent device station buffer memory capacity and the parameter-set receiving buffer area of the master station.	1 to 480		
s2	Link devices used for handshaking				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s2)+0	Remote output (RY) for data request	• Higher byte Set the upper 8 bits to „0“.	0	User
			• Lower byte Specify a remote output (RY) of the intelligent device station	0 to 127	
	(s2)+1	Remote register (RW <sub>r</sub> ) used as error code storage device Remote input (RX) used as completion device.	• Higher byte Specify a remote register (RW <sub>r</sub> ) of the intelligent device station, in which the same error code as in (s1)+0 will be stored.	0 to 15 or FF (When FF is set, no number is specified.)	
• Lower byte Specify a remote input (RX) of the intelligent device station			0 to 127		
(s2)+2	Completion mode	Specify, how the completion of the reading process should be indicated: 0: Using 1 device (RX <sub>n</sub> ) 1: Using 2 devices (RX <sub>n</sub> , RX <sub>n+1</sub> ) (RX <sub>n+1</sub> will be set at abnormal completion.)	0 or 1		
d1	Head address of the devices where the read data is to be stored.		User	BIN 16-bit	

Variables

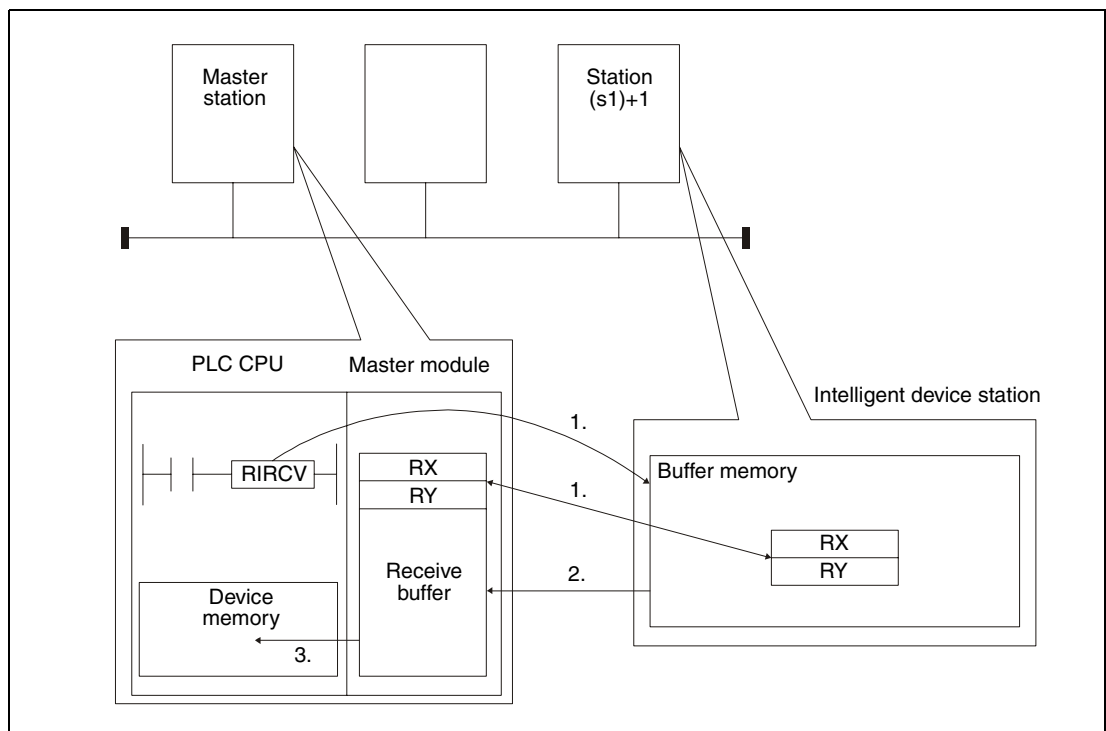
Set Data	Meaning	Range	Contents is stored by	Data Type	
d2	Bit device which is set for one scan after completion of the RIRCV instruction. (d2)+1 indicates that an error has occurred during execution of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d2)+0	Instruction completed	Indicates the completion of the RIRCV instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d2)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RIRCV instruction ON: Abnormal completion OFF: Normal completion	—		

Functions

Reading of data from the buffer memory of an intelligent device station (with handshake)

RIRCV Data read (with handshake)

The execution of a RIRCV instruction is only possible in the PLC CPU of the master station. This instruction is used to read data from the buffer memory on an intelligent device station. The data exchange is controlled by handshaking devices:



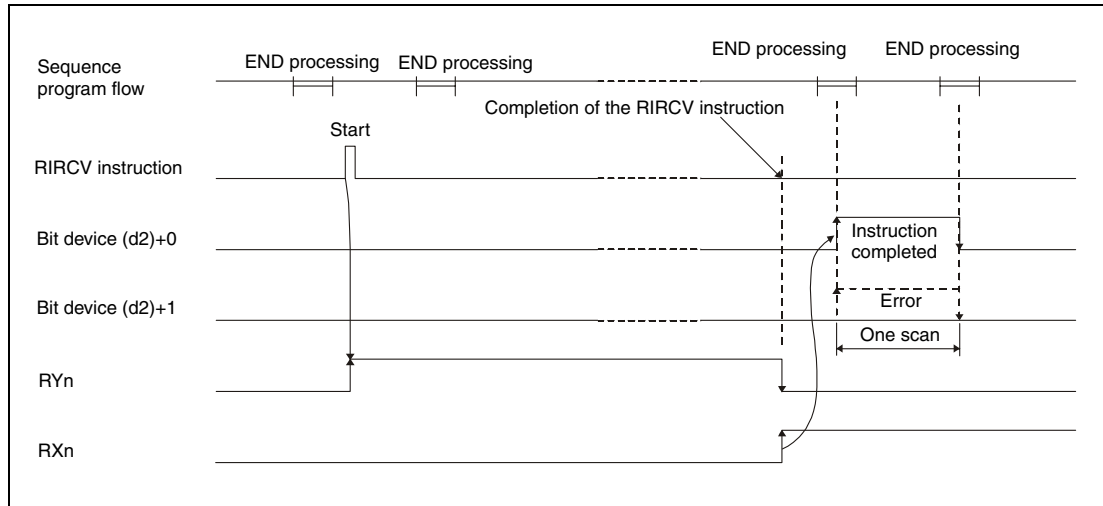
1. The buffer memory address specified by (s1)+3 of the station specified by (s1)+1 is accessed. The devices specified in s2 are used for the handshake.
2. The contents of the number of buffer memory addresses specified in (s1)+4 is read to the receive buffer of the master module.
3. The read data is stored in the PLC CPU to the devices starting with the one specified in d1. After that, the bit device specified in (d2)+0 is set for one scan.

Whether the execution of the RIRCV instruction has been finished can be checked with the devices (d2)+0 and (d2)+1:

- The bit device (d2)+0 turns ON at the END processing of the scan in which the RIRCV instruction has been completed and turns OFF at the next END processing.

- The bit device (d2)+1 indicates an error during execution of the RIRCV instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the RIRCV instruction, (d2)+1 turns ON at the END processing of the scan in which the RIRCV instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RIRCV instruction is being executed:



Although it's possible to execute RIRCV instructions for multiple intelligent device stations at the same time, it's not possible to access the same intelligent device station simultaneously from more than one station.

### Operation Error

In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When the module specified by Un is not an intelligent function module or a special function module. (error code: 2112)
- When an attempt was made to execute an unsupported instruction (error code: 4002)
- When the number of devices in the instruction is incorrect. (error code: 4003)
- When the instruction specifies a device that cannot be used. (error code: 4004)
- When the area designated by s contains data that cannot be used. (error code: 4100)
- When the number of data set to be used exceeds the allowable range. (error code: 4101)
- When the storage data or constants of the device specified with the instruction exceeds the allowable range (error code: 4101)

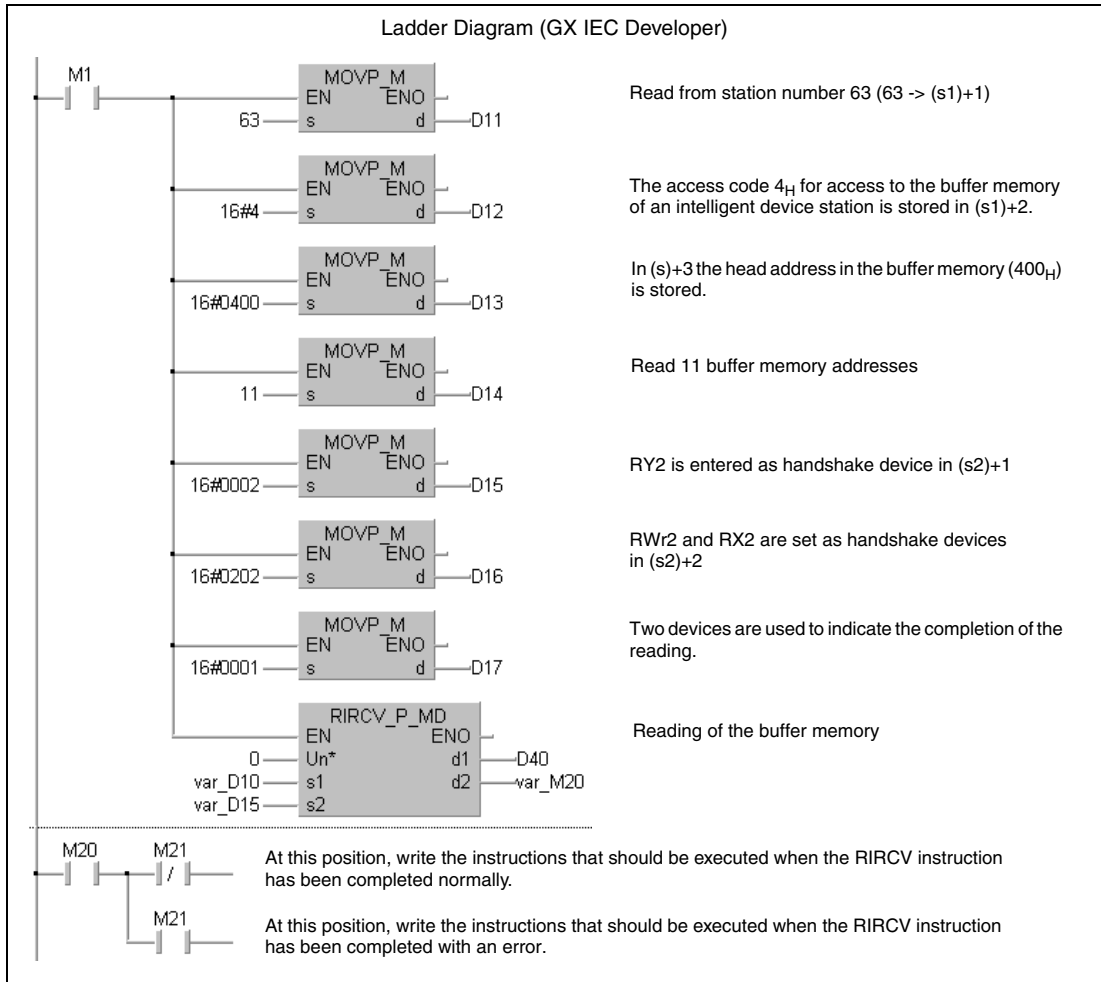
**Program Example**

**RIRCV**

The following program is executed in the PLC CPU of the master station. When M1 is set, the contents of 11 buffer memory addresses is read from the intelligent device station with the station number 63. Reading starts at the buffer memory address 400<sub>H</sub>. The data will be stored in the CPU module from data register D40 onward. To the master module of CC-Link the head I/O number X/Y00 is assigned. The remote devices RX2, RY2 and RWr2 are used for handshake. The completion of the reading is indicated by two devices. ((s2)+2 is set to „1“.)

● IEC editors

(On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)



IEC Instruction List

LD	M1				
MOV_P_M	63,	D11			
MOV_P_M	16#0004,	D12			
MOV_P_M	16#0400,	D13			
MOV_P_M	11,	D14			
MOV_P_M	16#0002,	D15			
MOV_P_M	16#0202,	D16			
MOV_P_M	16#0001,	D17			
RIRCV_P_MD	0,	var_D10,	var_d15,	D40,	var_M20

For an explanation of the devices and instructions used please see the above program example.

LD	M20	
ANDN	M21	
An instruction at this position will be executed when the RIRCV instruction has been completed normally.		
LD	M20	
AND	M21	
An instruction at this position will be executed when the RIRCV instruction has been completed annormally		

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.

Ladder Diagram (GX Developer)

MELSEC Instruction List

MELSEC	LD	M1	
	MOV	K63	D11
	MOV	H0004	D12
	MOV	H0400	D13
	MOV	K11	D14
	MOV	H0002	D15
	MOV	H0202	D16
	MOV	H0001	D17
	GP.RIRCV	U0	D10
		D40	D15
	M20		
MELSEC	LD	M20	
	MPS		
	ANI	M21	
	MRD		
	AND	M21	
		An instruction at this position will be executed when the RIRCV instruction has been completed normally.	





Variables

Set Data	Meaning	Range	Contents is stored by	Data Type	
n1	Head I/O address of the CC-Link master module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
n2	Station number of the intelligent device station where the data is written to	1 to 64	User	BIN 16-bit	
d1	Head number of the devices where control data for the execution of this instruction and the write data is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d1)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—	System
	(d1)+1	Number of points to write	Specify the number of data (unit: words) that should be written to the intelligent device station.	1 to 480	User
	(d1)+2	Access code	Enter the value „0004 <sub>H</sub> “ (Write to the buffer memory of an intelligent device station.)	0004 <sub>H</sub>	
	(d1)+3	Error check	Specify the device which will indicate that an error has occurred during execution of the RISEND instruction: 0: Device d1 1: Device RX+1	0 or 1	
	(d1)+4	Head address	Head address in the buffer memory (First address where data will be written to.)	Depends on the accessed station	—
(d1)+5 to (d1)+n	Storage area for the data to write	The size of this area is determined by the number of points to write specified at (d1)+1			
d2	Link devices used for handshaking				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d2)+0	Remote input (RX) and remote output (RY)	<ul style="list-style-type: none"> <li>Higher byte Set a remote input (RX) of the intelligent device station</li> </ul>	0 to 127	User (The RX, RY and RWr numbers are specified by the user but setting and resetting is performed by the system.)
<ul style="list-style-type: none"> <li>Lower byte Set a remote output (RY) of the intelligent device station</li> </ul>			0 to 127		
(d2)+1	Remote register (RWr)	Specify a remote register (RWr) of the intelligent device station	0 to 15 or FF (When FF is set, no number is specified.)		

**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
d3	Bit device which is set for one scan after completion of the RISEND instruction. (d3)+1 indicates that an error has occurred during execution of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d3)+0	Instruction completed	Indicates the completion of the RISEND instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d3)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RISEND instruction ON: Abnormal completion OFF: Normal completion	—		

**Functions**

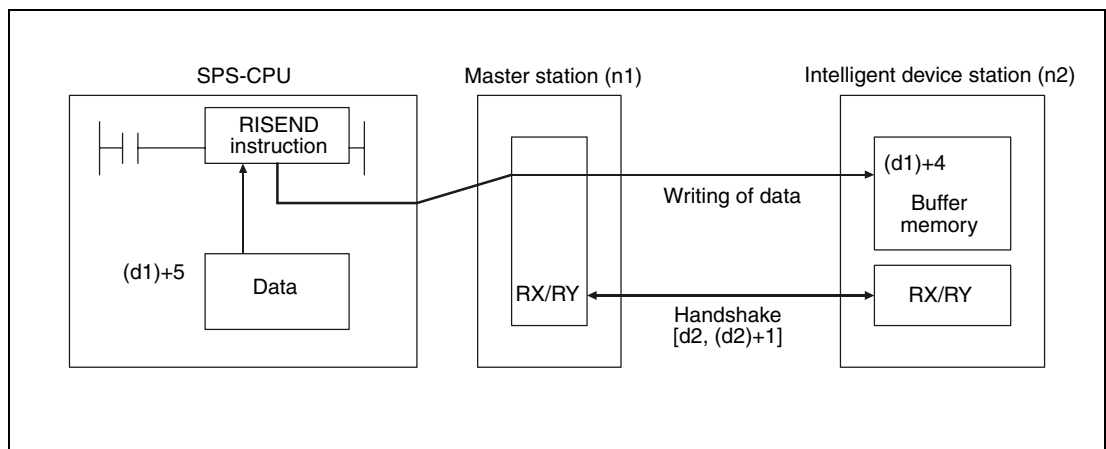
**Write (with handshake) to the buffer memory of an intelligent device station**

**RISEND Sending of data (with handshake)**

The execution of a RIRCV instruction can only be performed in the PLC CPU of the master station. This instruction is used to write data to the buffer memory on an intelligent device station. The data exchange is controlled by a handshaking device.

The number of points to write is stored in (d1)+1. The head buffer memory address specified in (d1)+3 is the first address to write to. The station number of the intelligent device station is designated by n2. This station is connected to the master station specified at n1. The data to write is stored in the CPU, which executes the RISEND instruction, in the devices starting with the one specified in (d1)+5.

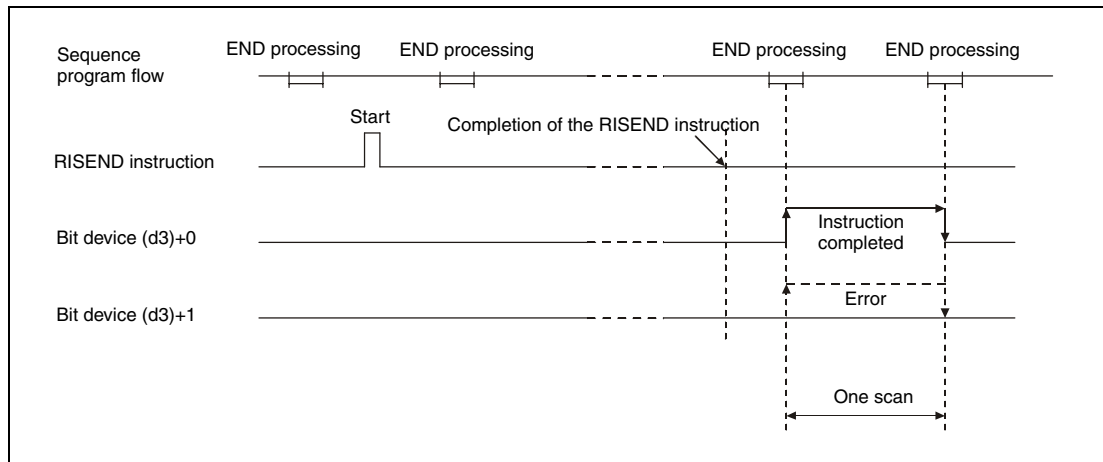
Function of the RISEND instruction:



Whether the execution of the RISEND instruction has been finished can be checked with the devices (d3)+0 and (d3)+1:

- The bit device (d3)+0 turns ON at the END processing of the scan in which the RISEND instruction has been completed and turns OFF at the next END processing.
- The bit device (d3)+1 indicates an error during execution of the RISEND instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the RISEND instruction, (d3)+1 turns ON at the END processing of the scan in which the RISEND instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RISEND instruction is being executed:



Although it's possible to execute RISEND instructions for multiple intelligent device stations at the same time, it's not possible to access the same intelligent device station simultaneously from more than one station.

**Execution Conditions**

When the LEDA instruction is used, the RISEND instruction is executed every scan while the read command is ON.

When the LEDB instruction is used, the RISEND instruction is executed only one scan on the leading edge (OFF -> ON) of the read command.

Note that the read processing executed by the RISEND instruction will take time for several scans before the processing is completed. Therefore, execute the next RISEND instruction only after the completion device (d3)+0 has been switched on. (A RISEND instruction will not be processed if the execution is started while another RISEND instruction is being executed.)

Before executing the RISEND instruction, set the network parameters using the RLPA instruction.

**Operation Error**

When „0“ or a value outside the range from 1 to 480 is entered as number of data to write in (d1)+1, the device (d3)+1 will be set and the error code BB42<sub>H</sub> will be stored in (d1)+0 at the completion of the RISEND instruction.

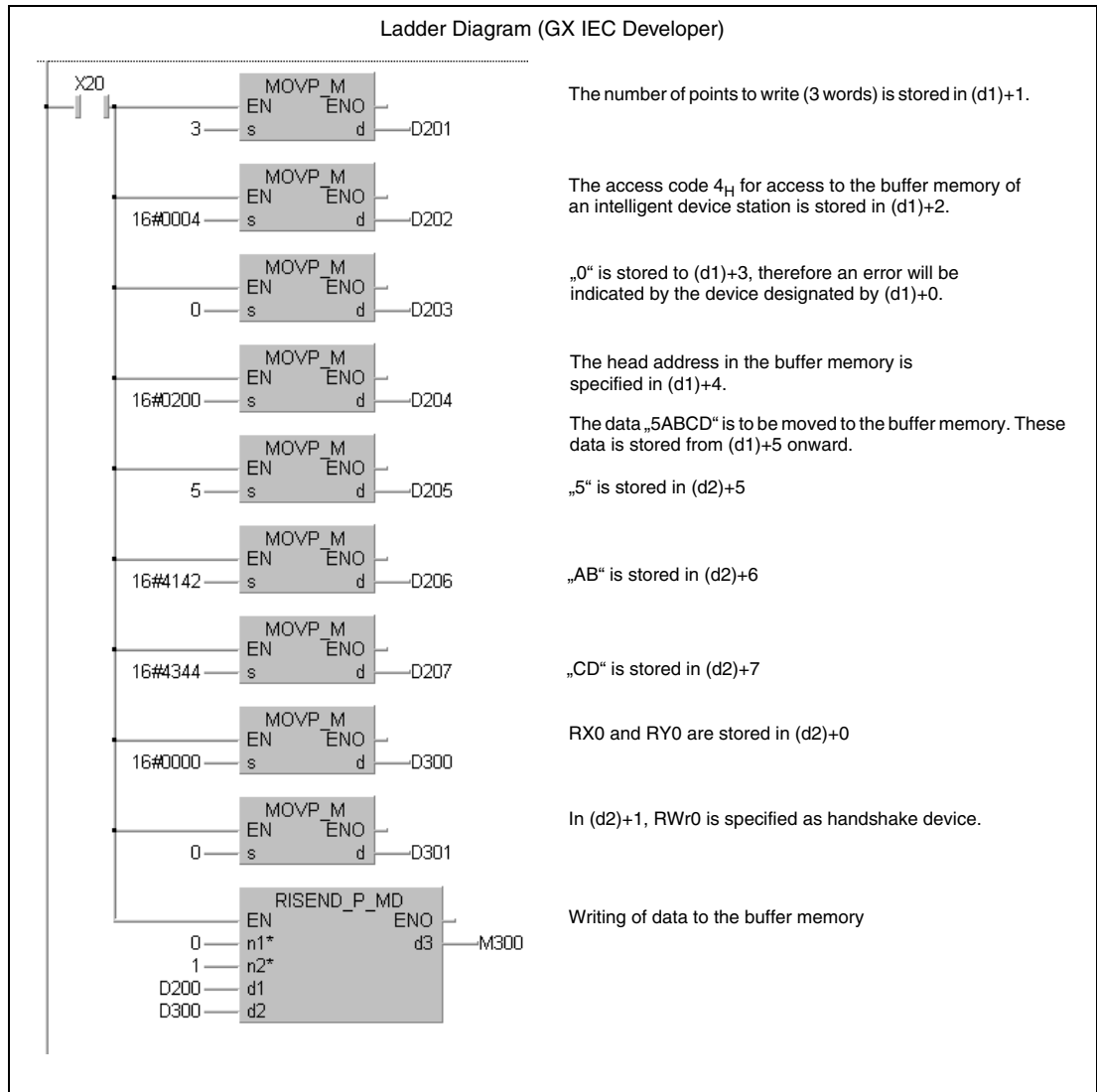
**Program Example**

**RISEND**

The following program, which is executed in the PLC CPU of the master station, writes to the buffer memory addresses from 200<sub>H</sub> to 202<sub>H</sub> of the intelligent device station with the station number 1. The devices RX0, RY0 and RWr0 are used for handshaking. An error is indicated by the device designated by (d1)+0. To the master module of CC-Link, the head I/O number X/Y000 is assigned.

● IEC editors

(On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

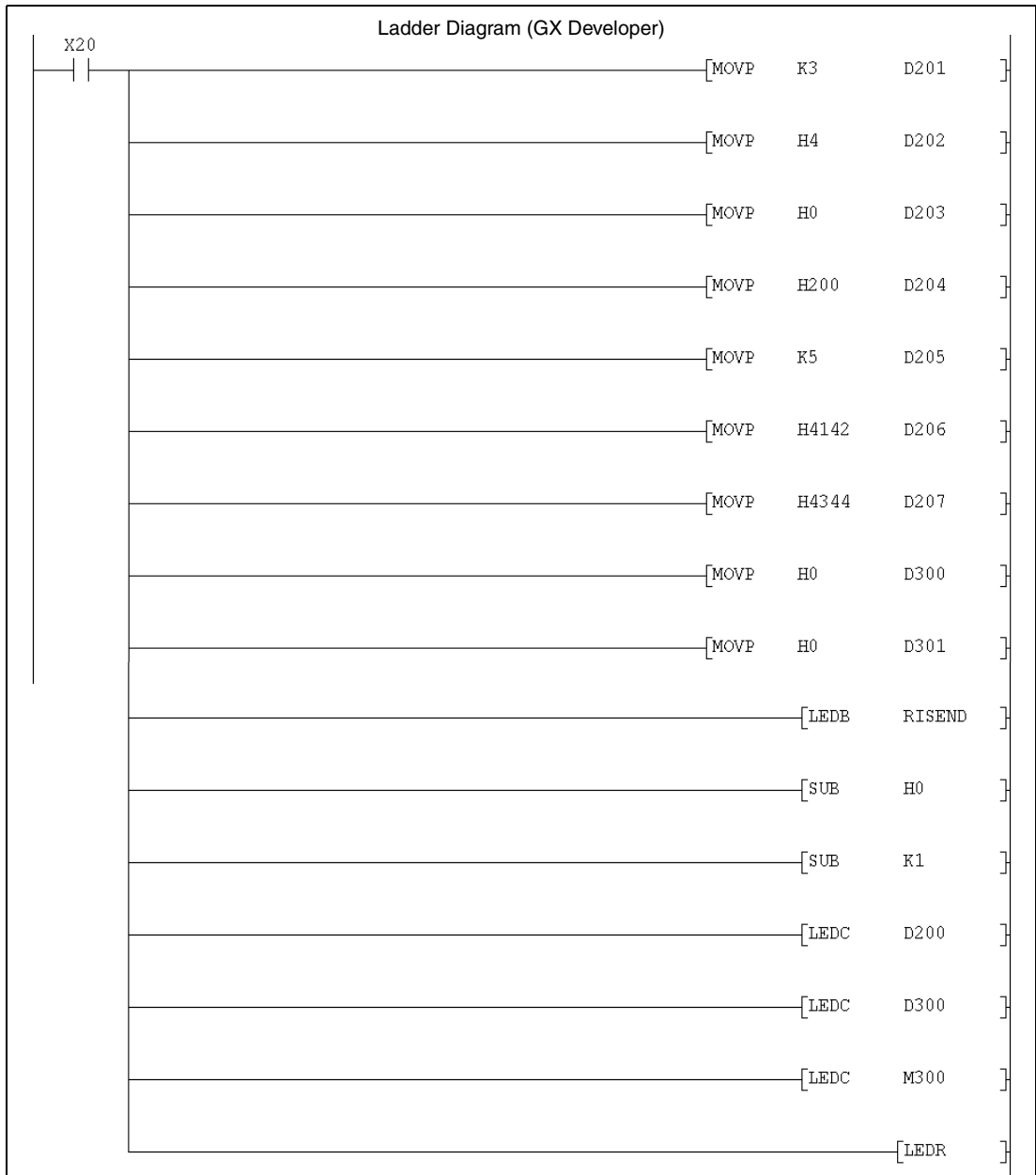


IEC Instruction List

LD	X20			
MOV_P_M	2,	D201		
MOV_P_M	16#0004,	D202		
MOV_P_M	16#0000,	D203		
MOV_P_M	16#0200,	D204		
MOV_P_M	5,	D205		
MOV_P_M	16#4142,	D206		
MOV_P_M	16#4344,	D207		
MOV_P_M	16#0000,	D300		
MOV_P_M	0,	D301		
RISEND_P_MD	0,	1,	D200,	D300,
			M300	

For an explanation of the devices and instructions used please see the above program example.

- MELSEC instruction list and ladder diagram of the GX Developer  
 For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.



MELSEC Instruction List

MELSEC	LD	X20	
	MOV P	K3	D201
	MOV P	H0004	D202
	MOV P	H0	D203
	MOV P	H0200	D204
	MOV P	K5	D205
	MOV P	H4142	D206
	MOV P	H4344	D207
	MOV P	H0	D300
	MOV P	H0	D301
	LE DB	RISEND	
	SUB	H0	
	SUB	K1	
	LE DC	D200	
	LE DC	D300	
	LE DC	M300	
	LE DR		

For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.



**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type	
Un	Head I/O address of the CC-Link master module (The upper two digits of an address expressed as a 3-digit number, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit	
s1	Head number of the devices where control data for the execution of this instruction is stored.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s1)+0	Execution result of the instruction	Indicates whether an error has occurred during execution of the instruction: 0000 <sub>H</sub> : No error Any value other than 0000 <sub>H</sub> : An error has occurred. The stored value is an error code which is explained in the user's manual of the CC-Link module.	—	System
	(s1)+1	Station number	Station number of the intelligent device station where the data is send to	0 to 64	User
	(s1)+2	Access code	Enter the value „0004 <sub>H</sub> “ (Write to the buffer memory of an intelligent device station.)	0004 <sub>H</sub>	
	(s1)+3	Head address	Head address in the buffer memory (First address where data is written to)	Depends on the accessed station	
(s1)+4	Number of points to write	Specify how much data (in the unit „words“) should be written to the intelligent device station.	1 to 480		
s2	Link devices used for handshaking				
	Set Data	Meaning	Description	Range	Contents is stored by
	(s2)+0	Remote output (RY) to request the sending of data	• Higher byte Set the upper 8 bits to „0“.	0	User
			• Lower byte Specify a remote output (RY) of the intelligent device station	0 to 127	
	(s2)+1	Remote register (RW <sub>r</sub> ) used as error code storage device Remote input (RX) used as completion device.	• Higher byte Specify a remote register (RW <sub>r</sub> ) of the intelligent device station, in which the same error code as in (s1)+0 will be stored.	0 to 15 or FF (When FF is set, no number is specified.)	
• Lower byte Specify a remote input (RX) of the intelligent device station			0 to 127		
(s2)+2	Completion mode	Specify, how the completion of the reading process should be indicated: 0: Using 1 device (RX <sub>n</sub> ) 1: Using 2 devices (RX <sub>n</sub> , RX <sub>n+1</sub> ) (RX <sub>n+1</sub> will be set at abnormal completion.)	0 or 1		
d1	First address of the area where the data for the intelligent device station is stored		User	BIN 16-bit	

Variables

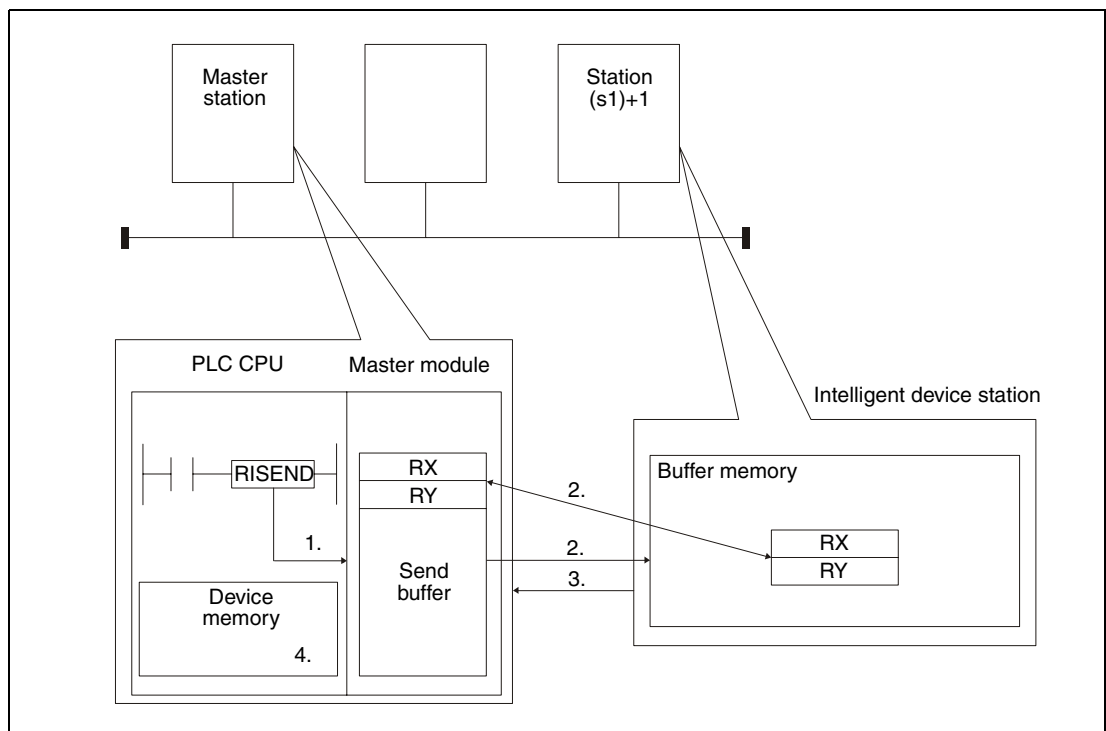
Set Data	Meaning	Range	Contents is stored by	Data Type	
d2	Bit device which is set for one scan after completion of the RIRCV instruction. (d2)+1 indicates that an error has occurred during execution of the instruction.				
	Set Data	Meaning	Description	Range	Contents is stored by
	(d2)+0	Instruction completed	Indicates the completion of the RISEND instruction ON: Instruction completed OFF: Instruction not completed	—	System
(d2)+1	Instruction completed with error	Indicates that an error has occurred during the processing of the RIRCV instruction ON: Abnormal completion OFF: Normal completion	—		

Functions

**Write (with handshake) to the buffer memory of an intelligent device station**

**RISEND Sending of data (with handshake)**

The RIRCV instruction can only be performed in the PLC CPU of the master station and is used to write data to the buffer memory on an intelligent device station. The data exchange is controlled by handshaking devices:



1. The data for the intelligent device station is moved to the send buffer of the master station.
2. The data is written to the buffer memory address specified by (s1)+3 of the station specified by (s1)+1. The devices specified in s2 are used for the handshake.
3. A write complete response is send to the master station.
4. The device specified in (d2)+0 is set.

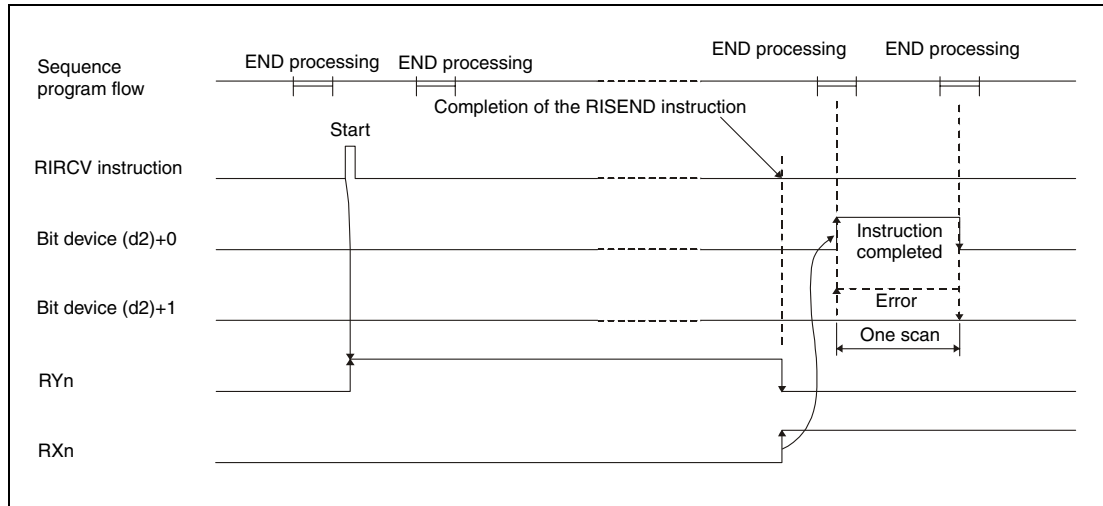
Whether the execution of the RISEND instruction has been finished can be checked with the devices (d2)+0 and (d2)+1:

- The bit device (d2)+0 turns ON at the END processing of the scan in which the RISEND instruction has been completed and turns OFF at the next END processing.



- The bit device (d2)+1 indicates an error during execution of the RISEND instruction. When the instruction has been completed normal, this device stays OFF. When an error occurs during execution of the RISEND instruction, (d2)+1 turns ON at the END processing of the scan in which the RISEND instruction has been completed and turns OFF at the next END processing.

The following figure shows the timing when the RIRCV instruction is being executed:



Although it's possible to execute RISEND instructions for multiple intelligent device stations at the same time, it's not possible to access the same intelligent device station simultaneously from more than one station.

### Operation Error

In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When the module specified by Un is not an intelligent function module or a special function module. (error code: 2112)
- When an attempt was made to execute an unsupported instruction (error code: 4002)
- When the number of devices in the instruction is incorrect. (error code: 4003)
- When the instruction specifies a device that cannot be used. (error code: 4004)
- When the area designated by s contains data that cannot be used. (error code: 4100)
- When the number of data set to be used exceeds the allowable range. (error code: 4101)
- When the storage data or constants of the device specified with the instruction exceeds the allowable range (error code: 4101)

**Program Example**

**RISEND**

The following program, which is executed in the PLC CPU of the master station, writes 1 word of data to the buffer memory address 111<sub>H</sub> of the intelligent device station with the station number 63. To the master module of CC-Link, the head I/O number X/Y000 is assigned. The devices RX4, RY4 and RWr4 are used for handshaking. The completion of the reading is indicated by two devices. ((s2)+2 is set to „1“.)

- IEC editors (On the next page this program example is shown for the MELSEC instruction list and the ladder diagram of the GX Developer.)

Ladder Diagram (GX IEC Developer)

Write to station number 63 (63 -> (s1)+1)

The access code 4<sub>H</sub> for access to the buffer memory of an intelligent device station is stored in (s1)+2

The head address in the buffer memory (111<sub>H</sub>) is stored to (s)+3.

Write to only buffer memory address

RY4 is entered as handshake device in (s2)+1

RWr1 and RX4 are set as handshake devices in (s2)+2

Two devices are used to indicate the completion of the reading.

The value „11“ is stored in D10. The contents of D10 is moved to the buffer memory address 111<sub>H</sub>.

Write to buffer memory

At this position, write the instructions that should be executed when the RISEND instruction has been completed normally.

At this position, write the instructions that should be executed when the RISEND instruction has been completed with an error.

IEC Instruction List

LD	M6	
MOV_P_M	63,	D1
MOV_P_M	16#0004,	D2
MOV_P_M	16#0111,	D3
MOV_P_M	1,	D4
MOV_P_M	16#0004,	D5
MOV_P_M	16#0104,	D6
MOV_P_M	16#0001,	D7
MOV_P_M	11,	D10
RISEND_P_MD	0,	var_D0, D10, var_D5, var_M40

For an explanation of the devices and instructions used please see the above program example.

LD	M40
ANDN	M41

An instruction at this position will be executed when the RISEND instruction has been completed normally.

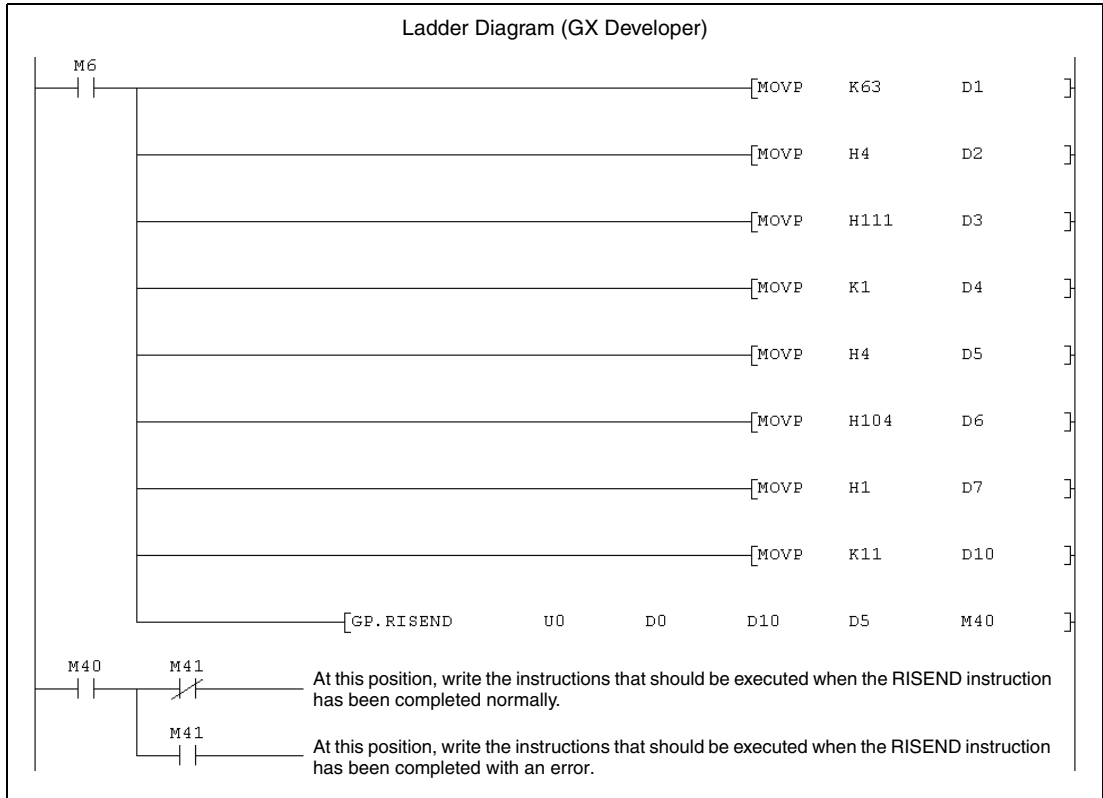
LD	M40
AND	M41

An instruction at this position will be executed when the RISEND instruction has been completed anomalously

**NOTE**

For the IEC editors it is necessary to define the variables in the header of the program organization unit (POU). Without variable definition it would cause compiler or checker error messages. For details see chapter 3.5.2 "Addressing of arrays and registers in the GX IEC Developer" of this manual.

- MELSEC instruction list and ladder diagram of the GX Developer  
For explanation of the devices and instructions used please see the program example for the ladder diagram of the GX IEC Developer on the previous page.



MELSEC Instruction List

MELSEC	LD	M6			
	MOV	K63	D1		
	MOV	H0004	D2		
	MOV	H0111	D3		
	MOV	K1	D4		
	MOV	H0004	D5		
	MOV	H0104	D6		
	MOV	H0001	D7		
	MOV	K11	D10		
	GP.RISEND	U0	D0	D10	D5
MELSEC	LD	M40			
	MPS				
	ANI	M41			
	MPP				
	AND	M41			

An instruction at this position will be executed when the RISEND instruction has been completed normally.

An instruction at this position will be executed when the RISEND instruction has been completed anomalously.

11.5.12 RITO (A series)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

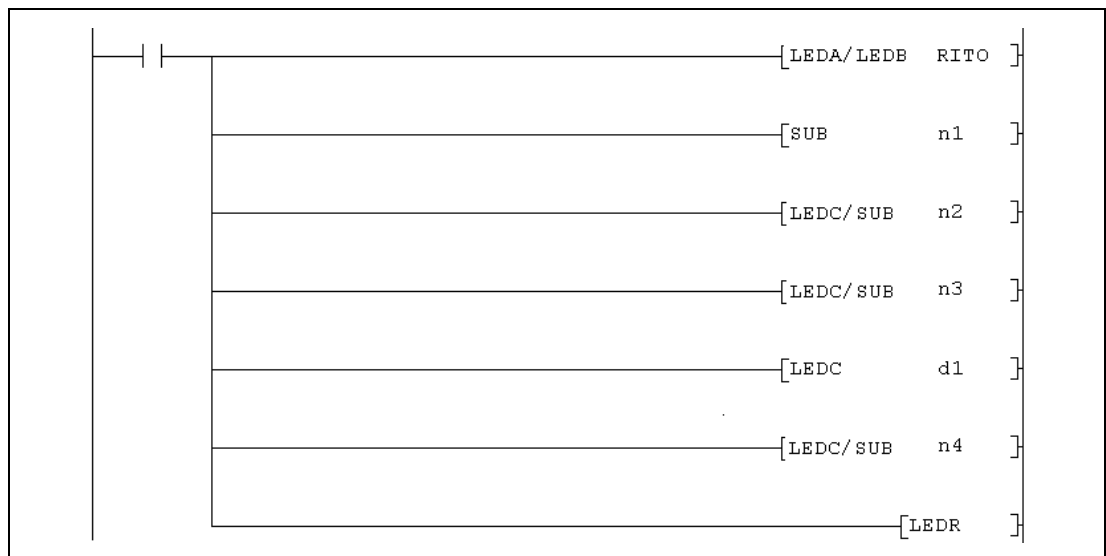
Devices  
MELSEC A

	Usable Devices																Blockänge	Number of steps	Index	Carry Flag M9012	Error Flag M9011	
	Bit Devices							Word Devices (16-bit)							Constant	Pointer						Level
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K
n1																	●	●				
n2							●	●	●	●	●						●	●				
n3							●	●	●	●	●						●	●				
d1							●	●	●	●	●											
n4							●	●	●	●	●						●	●				

GX IEC  
Developer

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center; width: 10%;">MELSEC</td> <td style="width: 50%;">LEDA/LEDB RITO</td> <td style="width: 40%;">n1</td> </tr> <tr> <td></td> <td>SUB</td> <td>n1</td> </tr> <tr> <td></td> <td>LEDC/SUB</td> <td>n2</td> </tr> <tr> <td></td> <td>LEDC/SUB</td> <td>n3</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d1</td> </tr> <tr> <td></td> <td>LEDC/SUB</td> <td>n4</td> </tr> <tr> <td></td> <td>LEDR</td> <td></td> </tr> </table>	MELSEC	LEDA/LEDB RITO	n1		SUB	n1		LEDC/SUB	n2		LEDC/SUB	n3		LEDC	d1		LEDC/SUB	n4		LEDR		<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="width: 10%;">RITO_MD</td> <td style="width: 90%;">n1, n2, n3, d1, n4</td> </tr> </table>	RITO_MD	n1, n2, n3, d1, n4
MELSEC	LEDA/LEDB RITO	n1																							
	SUB	n1																							
	LEDC/SUB	n2																							
	LEDC/SUB	n3																							
	LEDC	d1																							
	LEDC/SUB	n4																							
	LEDR																								
RITO_MD	n1, n2, n3, d1, n4																								

GX  
Developer



## Variables

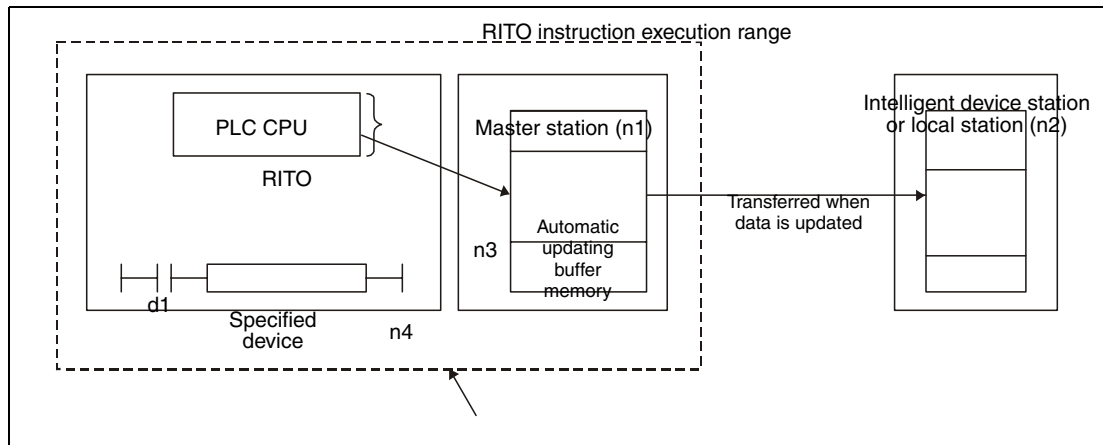
Set Data	Meaning	Range	Contents is stored by	Data Type
n1	Head I/O address of the CC-Link master module (Only the upper two digits of an address expressed as a 3-digit number are specified, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit
n2	Write destination Specify the station number of the intelligent device station where data is written to. (Only when the station which executes the RITO instruction is the master station.) When data is to be moved to the random access buffer, specify „FF <sub>H</sub> “.	1 to 64 or FF <sub>H</sub>		
n3	<ul style="list-style-type: none"> <li>• Sending/receiving buffer address of the intelligent device station specified in master station</li> <li>• Offset for random access buffer</li> </ul>	Between 0 and the max. value set in the parameters		
d1	First address of the area where the write data is stored.	Within the range of the specified device		Address
n4	Number of points to write (unit: words)	1 to 4096		BIN 16-bit

**Functions Write to automatic updating buffer memory****RITO Data write**

The RITO instruction moves data from the device memory of the PLC CPU to the automatic updating buffer memory in the master station. The data is then transferred to another station on CC-Link.

The data is specified by the head address (d1) and the number of words (n4). The destination in the master station is designated by n2 (equals the station number of the station where the data is finally sent to) and n3 (head address of the automatic updating buffer memory in the master station). The head I/O number of the master station is specified in n1.

The function of the RITO instruction is explained in the following figure:



Up to 4096 words may be written by the RITO instruction.

The size of the automatic updating buffer can be set using a RLPA instruction.

**Execution Conditions**

When the LEDA instruction is used, the RITO instruction is executed every scan while the write command is ON.

When the LEDB instruction is used, the RITO instruction is executed only one scan on the leading edge (OFF -> ON) of the write command.

**Operation Error**

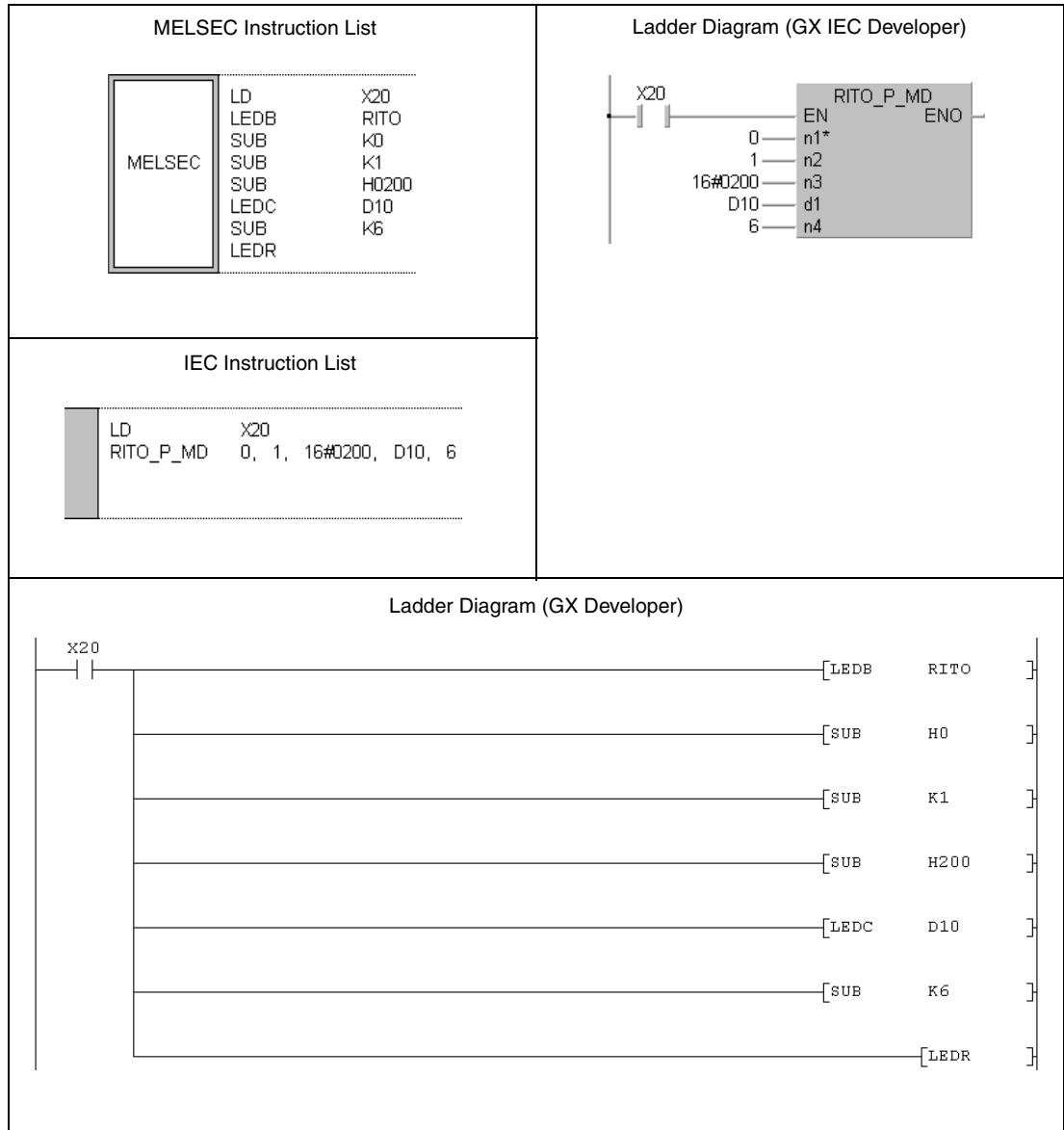
Either of the following conditions will result in an operation error. In this case the error flag M9011 is set and an error code is issued:

- The buffer memory address specified is outside the allowable range.  
(error code in D9008: 50, error code in D9091 (AnUCPU) or D9092 (AnSHCPU): 503)
- The number of data to write is larger than 4096.  
(error code in D9008: 50, error code in D9091 (AnUCPU) or D9092 (AnSHCPU): 503)

**Program Example**

**RITO**

When input X20 is set, the contents of the six data registers D10 to D15 is moved to the automatic updated buffer memory for the station set to station number 1 in the master module. There the data is stored from the address 200<sub>H</sub> onward. The master module of CC-Link is allocated to the I/O numbers X/Y000 to X/Y01F.



For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.

11.5.13 RITO (QnA series and System Q)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

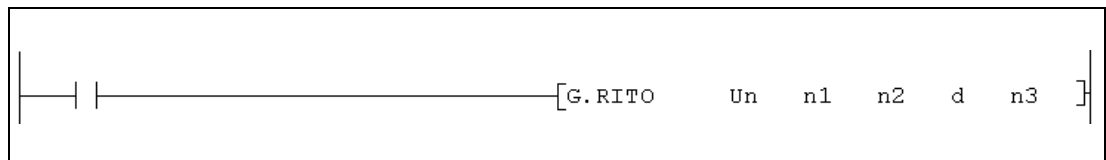
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
n1	●	●	●	—	—	—	—	●	—	SM0	9
n2	●	●	●	—	—	—	—	●	—		
d	—	●	●	—	—	—	—	—	—		
n3	●	●	●	—	—	—	—	●	—		

GX IEC Developer

<p style="text-align: center;">MELSEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p style="margin: 0;">MELSEC</p> </div> <p style="margin-left: 20px;">G.RITO    Un                   n1                   n2                   d                   n3</p>	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p style="margin: 0;">RITO_MD    Un, n1, n2, d, n3</p> </div>
---	---	--

GX Developer





**Variables**

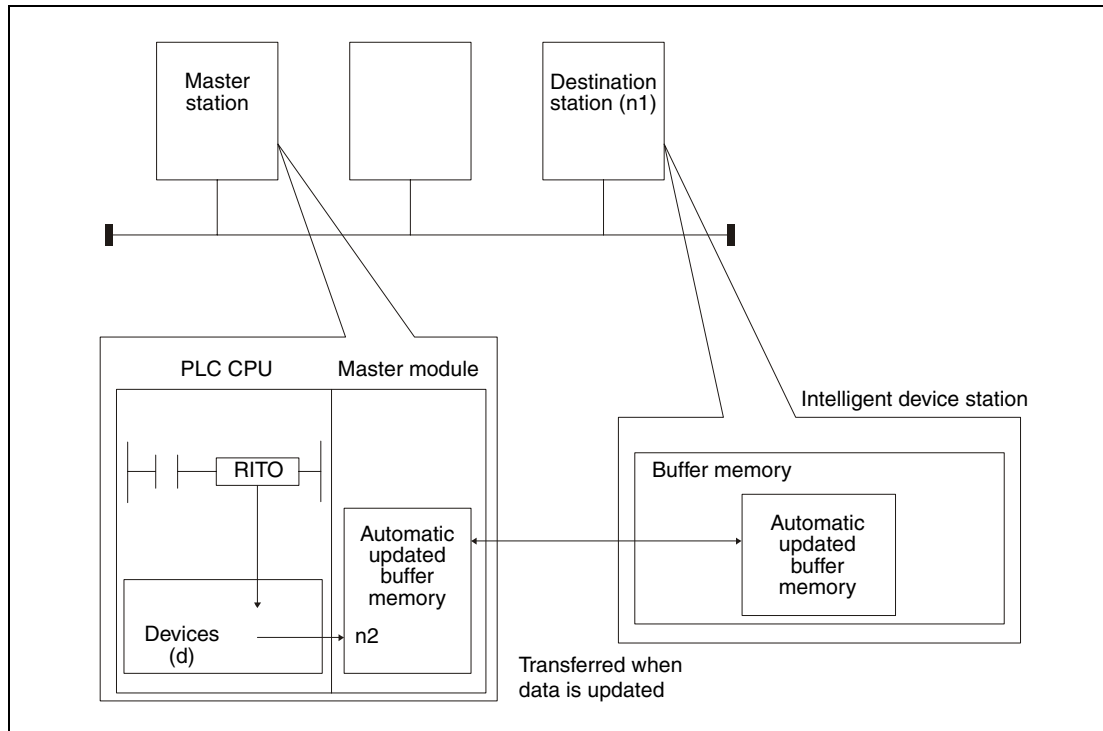
Set Data	Meaning	Range	Contents is stored by	Data Type
Un	Head I/O address of the CC-Link master module (Only the upper two digits of an address expressed as a 3-digit number are specified, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit
n1	Write destination <ul style="list-style-type: none"> <li>Specify the station number of the intelligent device station where data is written to.</li> <li>Specify „FF<sub>H</sub>“ when data is to be moved to the random access buffer.</li> </ul>	1 to 64 or FF <sub>H</sub>		
n2	The offset value of the automatic updated buffer of the intelligent device station specified by the master station or the random access buffer. The head address to write to is designated relative to the head address of the automatic updated buffer. An example: To write data to the address 356 <sub>H</sub> of the buffer memory, which starts at address 350 <sub>H</sub> , the value 6 <sub>H</sub> must be specified at n2.	Between 0 and the max. value set in the parameters.		
d	First address of the area where the write data is stored.	Within the range of the specified device		Address
n3	Number of points to write (unit: words)	1 to 4096		BIN 16-bit

**Functions Write to automatic updating buffer memory****RITO Data write**

The RITO instruction moves data from the device memory of the PLC CPU to the automatic updating buffer memory in the master station. The data is then transferred to another station on CC-Link.

The data is specified by the head address (d) and the number of words (n3). The destination in the master station is designated by n1 (equals the station number of the station where the data is finally sent to) and n2 (head address of the automatic updating buffer memory in the master station). The head I/O number of the master station is specified in Un.

The function of the RITO instruction is explained in the following figure:



The RITO instruction cannot be executed at more than one station for the same intelligent device station.

Up to 4096 words may be written by the RITO instruction.

The assignment of the automatic updated buffers is performed using the „station information settings“ of the network parameters of the GX Developer or GX IEC Developer.

**Operation Error**

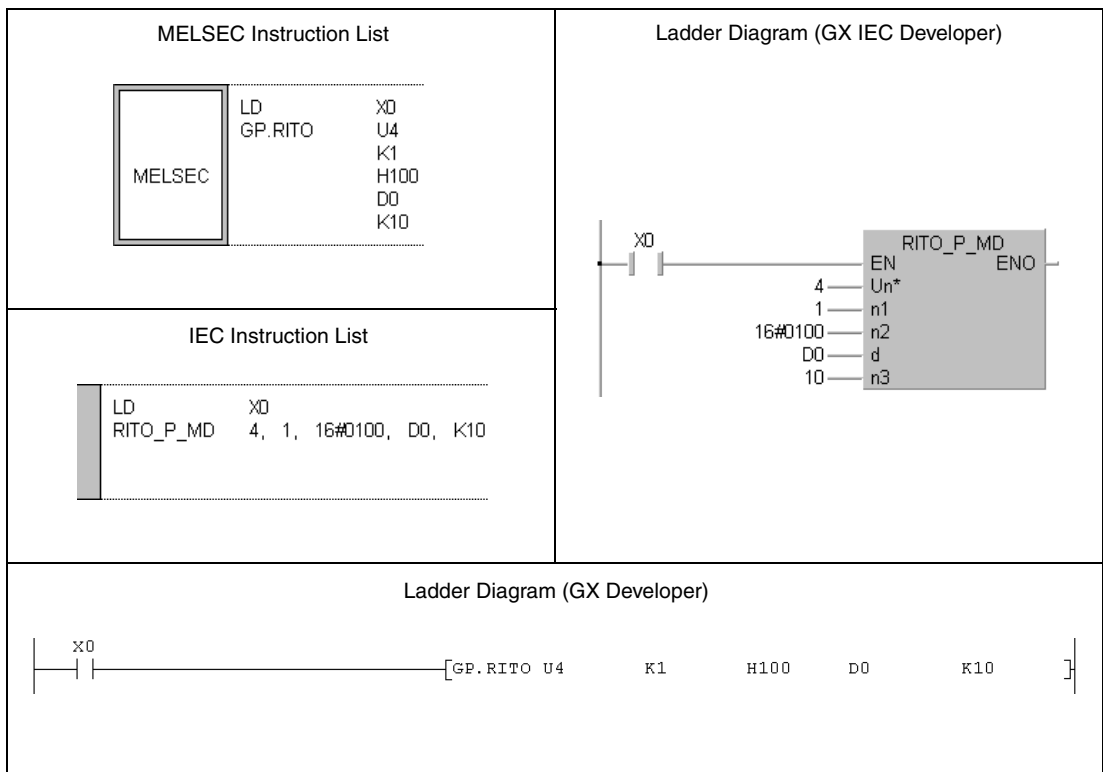
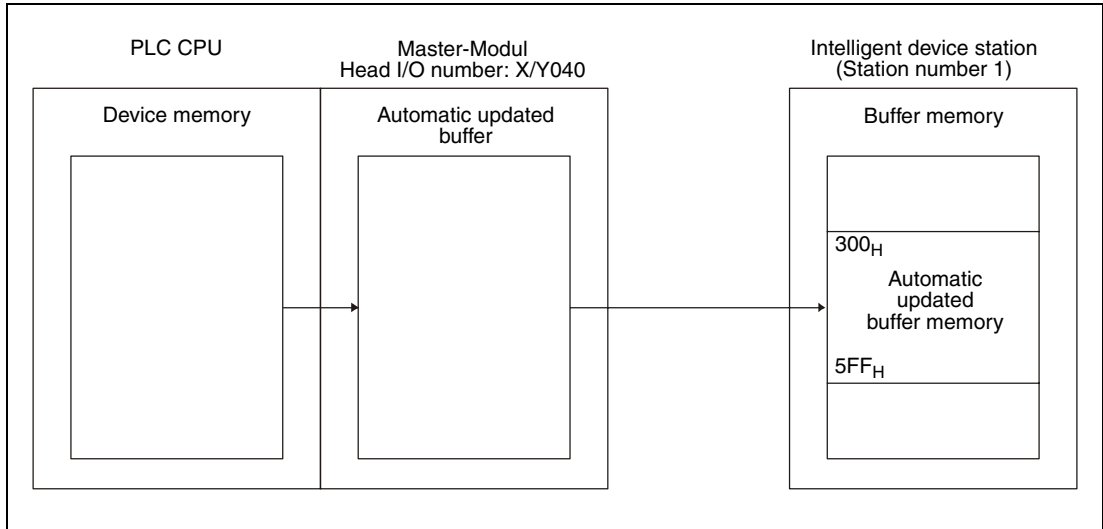
In the following cases an operation error occurs, the error flag SM0 is set, and an error code is stored in SD0:

- When the module specified by Un is not an intelligent function module or a special function module. (error code: 2112)
- When an attempt was made to execute an unsupported instruction (error code: 4002)
- When the number of devices in the instruction is incorrect. (error code: 4003)
- When the instruction specifies a device that cannot be used. (error code: 4004)
- When the station number specified at n1 does not exist. (error code: 4100)
- When the number of words to write specified in n3 is outside of the setting range. (error code: 4100)

**Program Example**

**RITO**

When the input X0 is set, the contents of 10 data registers (D0 to D10) is moved to the automatic updated buffer memory for the station set to station number 1 in the master module. This buffer begins at the address 300<sub>H</sub>. The data is stored from address 400<sub>H</sub> onward (offset = 100).



11.5.14 RIFR (A series)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
●	●	●	●		

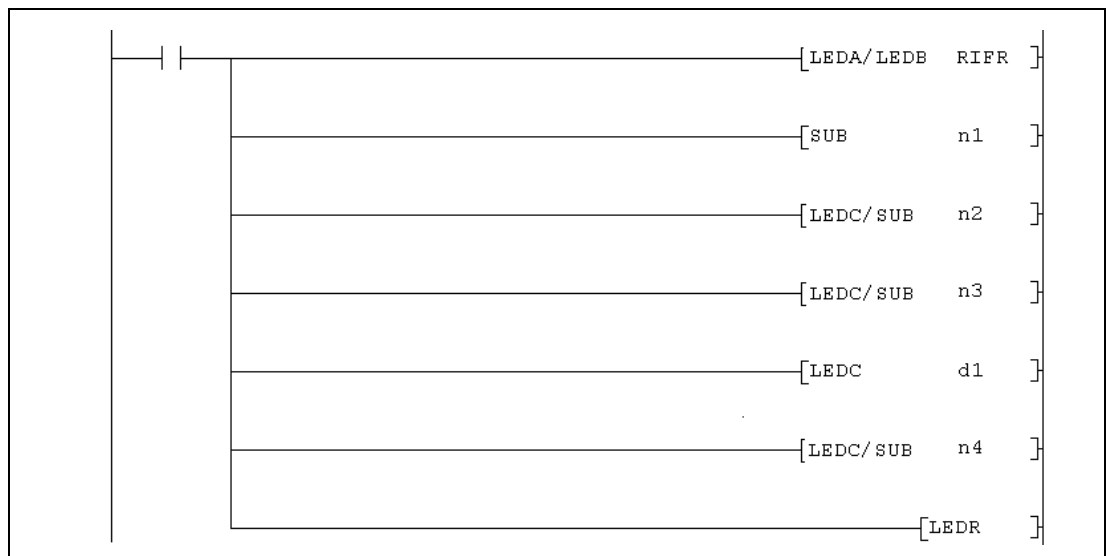
Devices  
MELSEC A

	Usable Devices																Block/ange	Number of steps	Index	Carry Flag M9012	Error Flag M9011			
	Bit Devices							Word Devices (16-bit)							Constant	Pointer						Level		
	X	Y	M	L	S	B	F	T	C	D	W	R	A0	A1	Z	V						K	H (16#)	P
n1																	●	●				29		
n2							●	●	●	●	●						●	●						
n3							●	●	●	●	●						●	●						
n4							●	●	●	●	●						●	●						
d1							●	●	●	●	●													

GX IEC  
Developer

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td style="text-align: center;">MELSEC</td> <td>LEDA/LEDB</td> <td>RIFR</td> </tr> <tr> <td></td> <td>SUB</td> <td>n1</td> </tr> <tr> <td></td> <td>LEDC/SUB</td> <td>n2</td> </tr> <tr> <td></td> <td>LEDC/SUB</td> <td>n3</td> </tr> <tr> <td></td> <td>LEDC</td> <td>d1</td> </tr> <tr> <td></td> <td>LEDC/SUB</td> <td>n4</td> </tr> <tr> <td></td> <td>LEDR</td> <td></td> </tr> </table>	MELSEC	LEDA/LEDB	RIFR		SUB	n1		LEDC/SUB	n2		LEDC/SUB	n3		LEDC	d1		LEDC/SUB	n4		LEDR		<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%;"> <tr> <td>RIFR_MD</td> <td>n1, n2, n3, n4, d1</td> </tr> </table>	RIFR_MD	n1, n2, n3, n4, d1
MELSEC	LEDA/LEDB	RIFR																							
	SUB	n1																							
	LEDC/SUB	n2																							
	LEDC/SUB	n3																							
	LEDC	d1																							
	LEDC/SUB	n4																							
	LEDR																								
RIFR_MD	n1, n2, n3, n4, d1																								

GX  
Developer



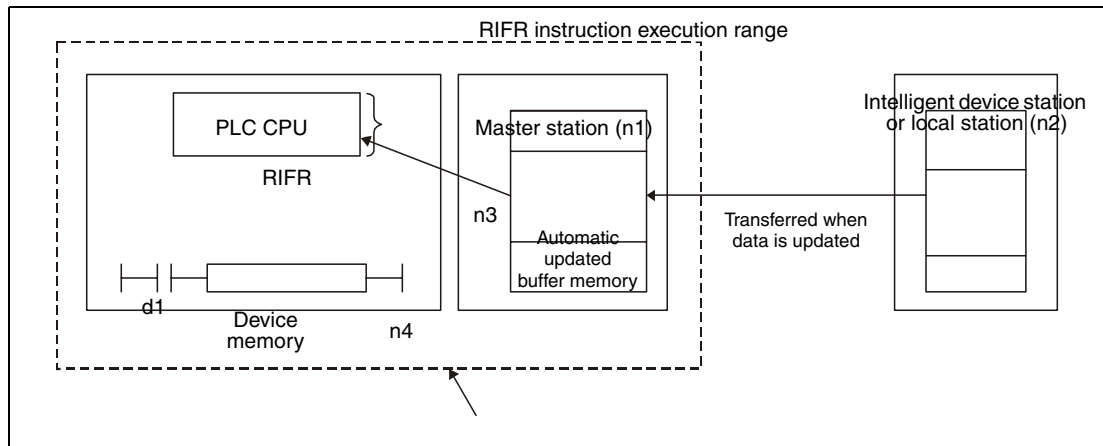
**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type
n1	Head I/O address of the CC-Link master module (Only the upper two digits of an address expressed as a 3-digit number are specified, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit
n2	Source of the data Specify the station number of the intelligent device station where data is read from. (Only when the station which executes the RIFR instruction is the master station.) When data is to be read from the random access buffer, specify „FF <sub>H</sub> “.	1 to 64 or FF <sub>H</sub>		
n3	<ul style="list-style-type: none"> <li>• Sending/receiving buffer address of the intelligent device station specified in master station</li> <li>• Offset for random access buffer</li> </ul>	Between 0 and the max. value set in the parameters		
n4	Number of points to read (unit: words)	1 to 4096		
d1	First address of the area where the read data will be stored.	Within the range of the specified device		Address

**Functions**    **Read from to automatic updating buffer memory****RIFR**    **Data read**

The RIFR instruction moves data from the automatic updating buffer memory in the master station to the device memory of the PLC CPU. The storage area for this data is specified by the head address (d1) and the number of words (n4). The source of the data is designated by the station number entered in n2 and the head address in the automatic updating buffer memory of the master station (n3). The head I/O number of the master station is specified in n1.

The function of the RIFR instruction is explained in the following figure:



Up to 4096 words may be read by the RIFR instruction.

The size of the automatic updating buffer can be set using a RLPA instruction.

**Execution Conditions**

When the LEDA instruction is used, the RIFR instruction is executed every scan while the read command is ON.

When the LEDB instruction is used, the RIFR instruction is executed only one scan on the leading edge (OFF -> ON) of the read command.

**Operation Error**

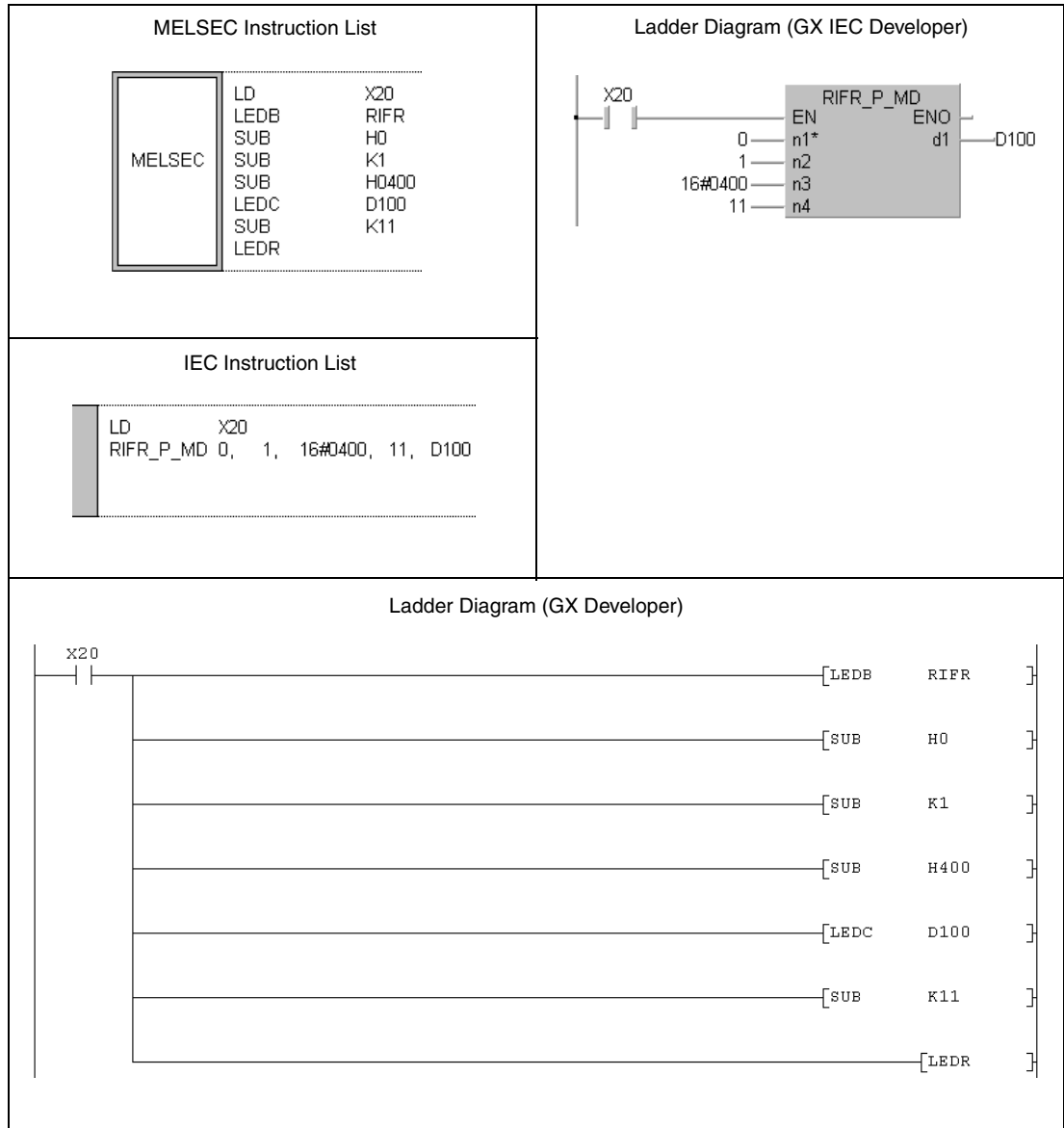
Either of the following conditions will result in an operation error. In this case the error flag M9011 is set and an error code is issued:

- The buffer memory address specified is outside the allowable range.  
(error code in D9008: 50, error code in D9091 (AnUCPU) or D9092 (AnSHCPU): 503)
- The number of data to write is larger than 4096.  
(error code in D9008: 50, error code in D9091 (AnUCPU) or D9092 (AnSHCPU): 503)

**Program Example**

RIFR

When the input X20 is set, the following program reads the contents of 11 points of the automatic updated buffer set to station number 1 in the master module, starting with address 400<sub>H</sub>. This data is then stored in the PLC CPU to D100 and the successive registers. The master module of CC-Link is allocated to the I/O numbers X/Y000 to X/Y01F.



For further information about the programming of dedicated instructions using the MELSEC editors please refer to chapter 3.3 of this manual.

11.5.15 RIFR (QnA series and System Q)

CPU

AnS	AnN	AnA(S)	AnU	QnA(S), Q4AR	System Q
				●	●

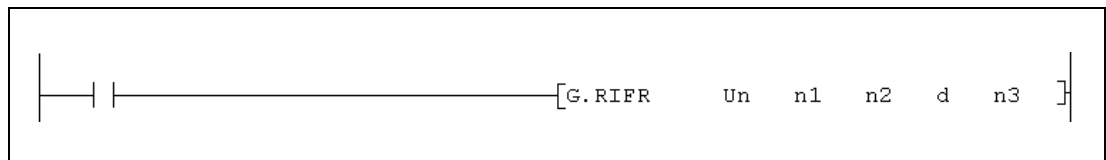
Devices  
MELSEC Q

	Usable Devices									Error Flag	Number of steps
	Internal Devices (System, User)		File-Register	MELSECNET/10 Direct J□□		Special Function Module U□NG□	Index Register Zn	Constants K, H (16#)	Other		
	Bit	Word		Bit	Word						
n1	●	●	●	—	—	—	—	●	—	SM0	9
n2	●	●	●	—	—	—	—	●	—		
n3	●	●	●	—	—	—	—	●	—		
d	—	●	●	—	—	—	—	—	—		

GX IEC Developer

<p style="text-align: center;">MELSEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: center;">MELSEC</td> <td style="width: 15%; border: none;">G.RIFR</td> <td style="width: 70%; border: none;">Un n1 n2 d n3</td> </tr> </table>	MELSEC	G.RIFR	Un n1 n2 d n3	<p style="text-align: center;">Ladder Diagram</p>	<p style="text-align: center;">IEC Instruction List</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; border: none;">RIFR_MD</td> <td style="width: 85%; border: none;">Un, n1, n2, n3, d</td> </tr> </table>	RIFR_MD	Un, n1, n2, n3, d
MELSEC	G.RIFR	Un n1 n2 d n3					
RIFR_MD	Un, n1, n2, n3, d						

GX Developer





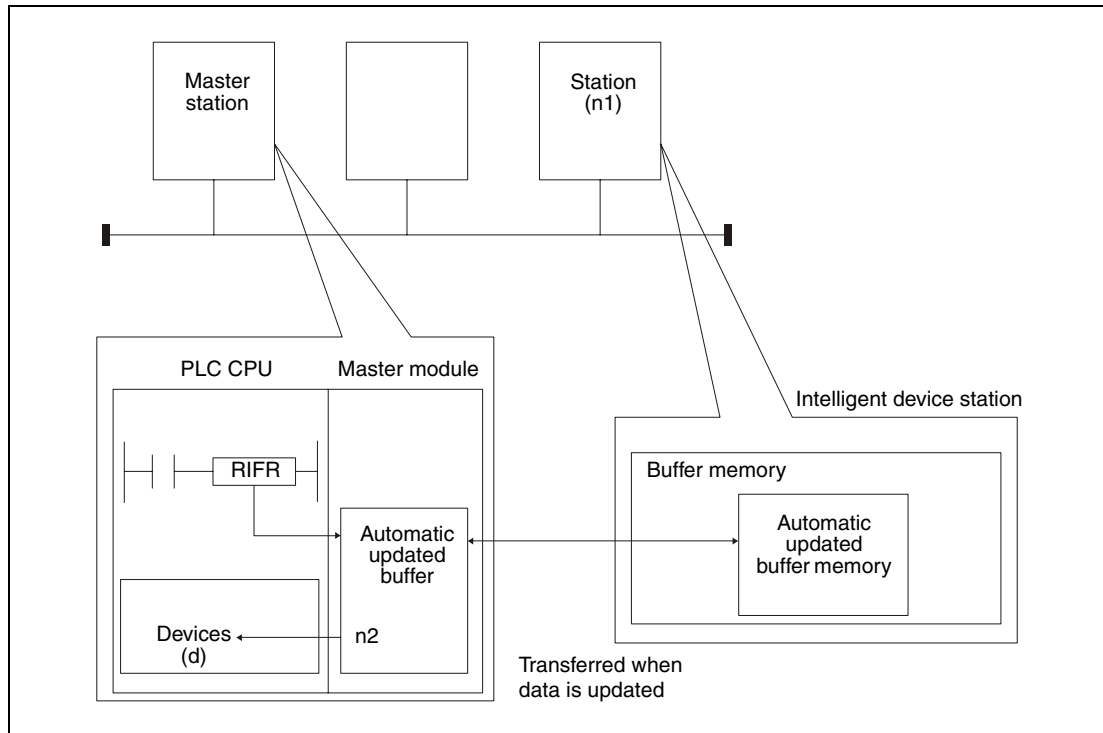
**Variables**

Set Data	Meaning	Range	Contents is stored by	Data Type
Un	Head I/O address of the CC-Link master module (Only the upper two digits of an address expressed as a 3-digit number are specified, e. g. the head address X/Y100 is set as 10 <sub>H</sub> )	0 to FE <sub>H</sub>	User	BIN 16-bit
n1	Source of the data <ul style="list-style-type: none"> <li>• Specify the station number of the intelligent device station where data is read from.</li> <li>• Specify „FF<sub>H</sub>“, when data is to be read from the random access buffer.</li> </ul>	1 to 64 or FF <sub>H</sub>		
n2	The offset value of the automatic updated buffer of the intelligent device station specified by the master station or the random access buffer. The head address for the data to read is designated relative to the head address of the automatic updated buffer. An example: When reading should start at the address 356 <sub>H</sub> of the buffer memory, which starts at address 350 <sub>H</sub> , the value 6 <sub>H</sub> must be specified at n2.	Between 0 and the max. value set in the parameters		
n3	Number of points to read (unit: words)	1 to 4096		
d	First address of the area where the read data will be stored.	Within the range of the specified device		Address

**Functions**    **Read from to automatic updating buffer memory****RIFR**    **Data read**

The RIFR instruction moves data from the automatic updating buffer memory in the master station to the device memory of the PLC CPU. The storage area for this data is specified by the head address (d) and the number of words (n3). The source of the data is designated by the station number entered in n1 and the offset for the automatic updating buffer memory of the master station (n2). The head I/O number of the master station is specified in Un.

The function of the RIFR instruction is explained in the following figure:



The RIFR instruction cannot be executed at more than one station for the same intelligent device station.

Up to 4096 words may be read by the RIFR instruction.

The assignment of the automatic updated buffers is performed using the „station information settings“ of the network parameters of the GX Developer or GX IEC Developer.

**Operation Error**

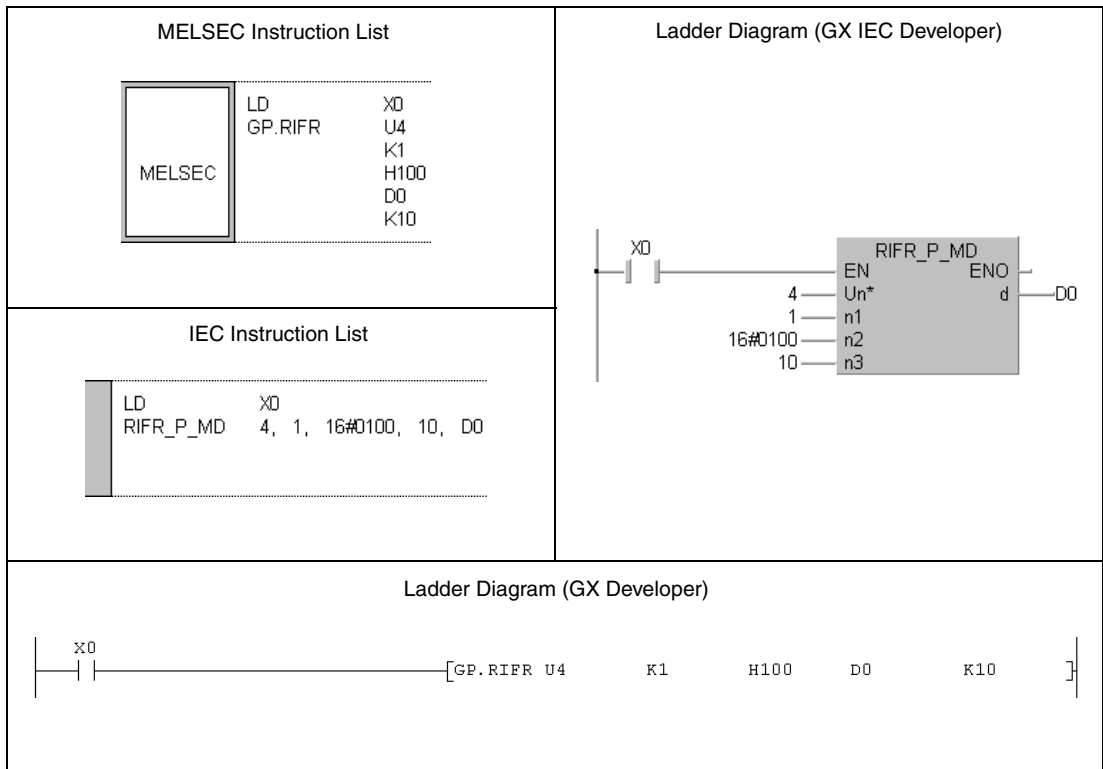
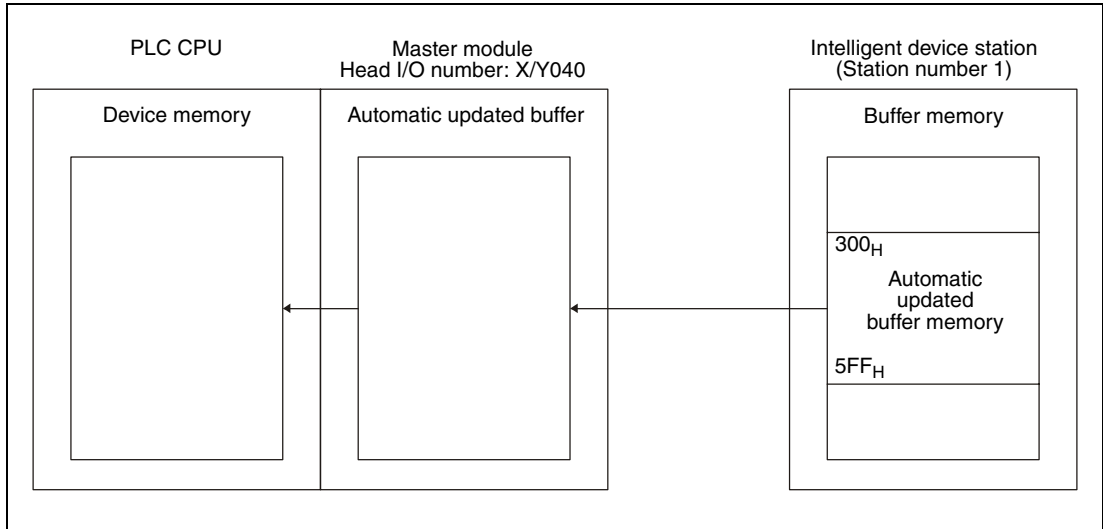
In the following cases an operation error occurs, the error flag SMO is set, and an error code is stored in SD0:

- When the module specified by Un is not an intelligent function module or a special function module. (error code: 2112)
- When an attempt was made to execute an unsupported instruction (error code: 4002)
- When the number of devices in the instruction is incorrect. (error code: 4003)
- When the instruction specifies a device that cannot be used. (error code: 4004)
- When the station number specified at n1 does not exist. (error code: 4100)
- When the number of words to read specified in n3 is outside of the setting range. (error code: 4100)

**Program Example**

RIFR

When the input X0 is set, the following program reads the contents of 10 points of the automatic updated buffer set to station number 1 in the master module and stores this data in the PLC CPU to D0 and the successive registers. The automatic updated buffer begins at the address 300<sub>H</sub>. Reading starts at the address 400<sub>H</sub> (offset = 100). The master module of CC-Link is allocated to the I/O numbers X/Y040 to X/Y41F.





# 12 Microcomputer Mode (AnN(S))

The MELSEC A series (except for AnA, AnAS, and AnU) supports the combined execution of sequence program and microcomputer program. The microcomputer program allows the execution of program sequences leaving the macro range (main and sub program). A microcomputer program is invoked via a SUB(P) instruction.

MELSEC AnA, AnAS, AnUS, QnA, QnAS and System Q CPUs do not process microcomputer programs.

## 12.1 Storage capacities and memory areas

The following table gives a CPU related overview of capacities and memory areas for microcomputer programs:

CPU	Processor	Microcomputer Program Area	Work Area	Stack Area	Instructions not supported	
A1	8086 (8 MHz)	0 – 10 kBytes	A100H – A1FFH (256 Bytes)	User area: 128 Bytes	INT, INTO, IRET, IN, OUT, HLT, WAIT, LOCK, ESC	
A2		0 – 26 kBytes				
A3		0 – 58 kBytes (MAIN) 0 – 58 kBytes (SUB)				
A1N	8086 (10 MHz)	0 – 10 kBytes				
A2N-S1		0 – 26 kBytes				
A2S		0 – 26 kBytes				
A3N		0 – 58 kBytes (MAIN) 0 – 58 kBytes (SUB)				
A1S	8086 (8 MHz)	0 – 14 kBytes				
A1S-S1		0 – 14 kBytes				
A2C		0 – 26 kBytes				
A3H	80286 (8 MHz)	0 – 58 kBytes (MAIN) 0 – 58 kBytes (SUB)				INT, INTO, IRET, IN, OUT, HLT, WAIT, LOCK, ESC, CLI, STI
A3M		0 – 58 kBytes (MAIN) 0 – 58 kBytes (SUB)				

**NOTE**

*The microcomputer program area is specified in multiples of 2 kBytes. The memory areas for the individual program parts (microcomputer and sequence program in the MAIN and SUB range) must not overlap.*

*In order to avoid malfunction never use instructions not supported by microcomputer programs (refer to table above).*

## 12.2 Applying user-created microcomputer programs

The source code written by the user in 8086 assembly language is compiled (converted) into a machine language comprehensible to the PLC by assembler programs under CP/M<sup>®</sup> or MS-DOS<sup>®</sup>. The compiled program is called the "object program" and is to be stored in the microcomputer memory area of the CPU. A C-compiler transfers the OBJ file to the PLC via a programming terminal.

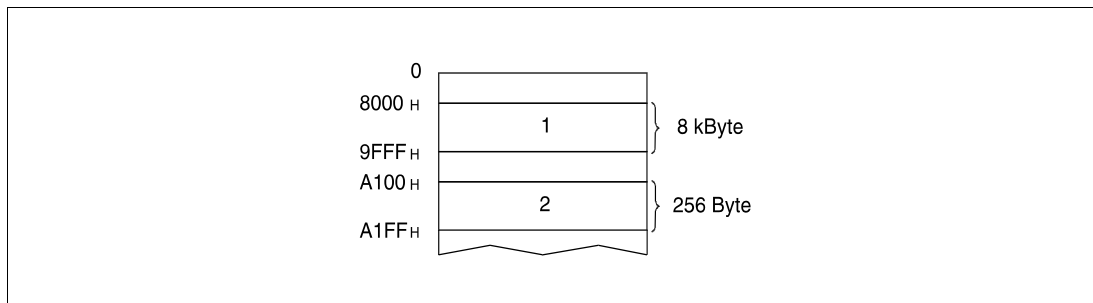
**NOTE** *Please check, whether these functions are available and supported by your version of the programming software.*

### Precaution on preparing the microcomputer program

- Provide the PUSH instruction at the start of the microcomputer program so that the contents of registers used during execution are saved in the stack areas. Also, provide the POP instruction at the end of the program so that the contents of registers saved in the stack areas are returned.
- Initialize the registers to be used in the microcomputer program at the start of the microcomputer program. The contents of the registers are not defined once the microcomputer program is called from the sequence program.
- Since the microcomputer program is executed only if it is called from the sequence program with the SUB(P) instruction, the sequence program is always required.
- To return from the microcomputer program to the sequence program, use the RETF instruction.

### 12.2.1 Memory map

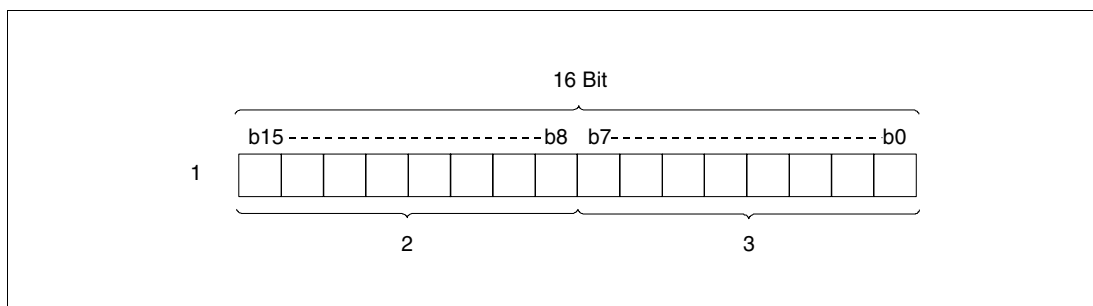
The microcomputer program is stored in two different memory areas of the CPU. The area from 8000H through 9FFFH with a capacity of 8 kBytes is used for data storage and the area from A100H through A1FFH is used as work area for the microcomputer program.



- 1 Data storage area
- 2 Work area for microcomputer program

### 12.2.2 Address configuration of the data storage area

One address of the data storage area consists of 16 bits and is subdivided into an even and an odd area of 8 bits each. The following figure shows the configuration pattern of one address.



- 1 One address 8000H
- 2 Odd 8-bit area (8001H)
- 3 Even 8-bit area (8000H)

### 12.2.3 Configuration of data memory area

The data memory area from 8000H through 9FFFH is used by the CPU to store the device data. The following tables show the device related configuration of this area:

Device	CPU Type	Address	Configuration
Input (X)	A1 A1N	8000H – 803FH	X0 - FF
	A2 A2C A2N A2S	8000H – 803FH	X0 - 1FF
	A2N-S1	8000H – 80FFH	X0 - 3FF
	A3 A3N	8000H – 81FFH	X0 - 7FF
Output (Y)	A1 A1N	8200H – 823FH	Y0 - FF
	A2 A2C A2N A2S	8000H – 827FH	Y0 - 1FF
	A2N-S1	8200H – 827FH	X0 - 3FF
	A3 A3N	8200H – 83FFH	Y0 - 7FF

1

2

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
XIM7	XIM6	XIM5	XIM4	XIM3	XIM2	XIM1	XIM0	X7	X6	X5	X4	X3	X2	X1	X0
XIMF	XIME	XIMD	XIMC	XIMB	XIMA	XIM9	XIM8	XF	XE	XD	XC	XB	XA	X9	X8
XIM17	XIM16	XIM15	XIM14	XIM13	XIM12	XIM11	XIM10	X17	X16	X15	X14	X13	X12	X11	X10

3

4

1

2

b15	-----							b8	b7	b6	b5	b4	b3	b2	b1	b0
8200H								Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	
8202H								YF	YE	YD	YC	YB	YA	Y9	Y8	
8204H								Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	

5

In refresh mode operation the data is written to and read from the output module via the output memory.

In direct mode operation the data is written directly to the output module but read from the output module via the output memory.

- 1 Odd addresses
- 2 Even addresses
- 3 Area storing ON/OFF data status of a remote station (read and write), 0 = OFF, 1 = ON. Obtain actual input by the following expression: (X)=(XIM) ∨ (X̄)
- 4 Area storing ON/OFF data status of an input module (read only), 0 = OFF, 1 = ON.
- 5 Area storing operation results of the PLC (read and write), 0 = OFF, 1 = ON.



Device	CPU Type	Address	Configuration
Internal relay (M) Latch relay (L) Step relay (S)	A1 A2 A3 A1N A2N-S1 A3N A1S A1S-S1 A2C	8400H – 85FFH	M/L/S 0 - 2047
Link relay (B)		8600H – 86FFH	F0 - 255
Annunciator (F)		8700H – 873FH	F0 - 255
Special relay (M)		8740H – 877FH	M 9000-9255
Timer contact (T)		8780H – 87BFH	T0 - 255
Counter contact (C)		87C0H – 87FFH	C0 - 255
Timer coil (T)		9C00H – 9C3FH	T0 - 255
Counter coil (C)		9C40H – 9C7FH	C0 - 255

- 1 Odd area
- 2 Even area
- 3 Area storing operation results of the PLC (read and write).

Device	CPU Type	Address	Configuration
Data register (D)	A1 A2 A3 A1N A2N-S1 A3N A1S1 A1S-S1 A2C	8800H – 8FFFH	
Link register (W)		9000H – 97FFH	
Current value of timer (T)		9800H – 99FFH	
Current value of counter (C)		9A00H – 9BFFH	
Special register (D)		9D00H – 9EFFH	
Accumulator (A0, A1)		9FF8H – 9FFAH	
Index register (Z, V)		9FFCH – 9FFEH	

Device	CPU Type	Address	Configuration																																																																																																				
Input (X)	A3H A3M	8000H — 80FFH	<div style="text-align: center;"> <table border="1" style="margin: auto;"> <tr> <td colspan="8" style="text-align: center;">1</td> <td colspan="8" style="text-align: center;">2</td> </tr> <tr> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>8000H</td><td>XF</td><td>XE</td><td>XD</td><td>XC</td><td>XB</td><td>XA</td><td>X9</td> <td>X8</td><td>X7</td><td>X6</td><td>X5</td><td>X4</td><td>X3</td><td>X2</td><td>X1</td><td>X0</td> </tr> <tr> <td>8002H</td><td>X1F</td><td>X1E</td><td>X1D</td><td>X1C</td><td>X1B</td><td>X1A</td><td>X19</td> <td>X18</td><td>X17</td><td>X16</td><td>X15</td><td>X14</td><td>X13</td><td>X12</td><td>X11</td><td>X10</td> </tr> <tr> <td>8004H</td><td>X2F</td><td>X2E</td><td>X2D</td><td>X2C</td><td>X2B</td><td>X2A</td><td>X29</td> <td>X28</td><td>X27</td><td>X26</td><td>X25</td><td>X24</td><td>X23</td><td>X22</td><td>X21</td><td>X20</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table> <p style="text-align: center;">↓ 3</p> </div>	1								2								b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	8000H	XF	XE	XD	XC	XB	XA	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0	8002H	X1F	X1E	X1D	X1C	X1B	X1A	X19	X18	X17	X16	X15	X14	X13	X12	X11	X10	8004H	X2F	X2E	X2D	X2C	X2B	X2A	X29	X28	X27	X26	X25	X24	X23	X22	X21	X20																	
1								2																																																																																															
b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																																							
8000H		XF	XE	XD	XC	XB	XA	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0																																																																																						
8002H		X1F	X1E	X1D	X1C	X1B	X1A	X19	X18	X17	X16	X15	X14	X13	X12	X11	X10																																																																																						
8004H	X2F	X2E	X2D	X2C	X2B	X2A	X29	X28	X27	X26	X25	X24	X23	X22	X21	X20																																																																																							
Output (Y)	8200H — 82FFH	<div style="text-align: center;"> <table border="1" style="margin: auto;"> <tr> <td colspan="8" style="text-align: center;">1</td> <td colspan="8" style="text-align: center;">2</td> </tr> <tr> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>8000H</td><td>YF</td><td>YE</td><td>YD</td><td>YC</td><td>YB</td><td>YA</td><td>Y9</td> <td>Y8</td><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td> </tr> <tr> <td>8002H</td><td>Y1F</td><td>Y1E</td><td>Y1D</td><td>Y1C</td><td>Y1B</td><td>Y1A</td><td>Y19</td> <td>Y18</td><td>Y17</td><td>Y16</td><td>Y15</td><td>Y14</td><td>Y13</td><td>Y12</td><td>Y11</td><td>Y10</td> </tr> <tr> <td>8004H</td><td>Y2F</td><td>Y2E</td><td>Y2D</td><td>Y2C</td><td>Y2B</td><td>Y2A</td><td>Y29</td> <td>Y28</td><td>Y27</td><td>Y26</td><td>Y25</td><td>Y24</td><td>Y23</td><td>Y22</td><td>Y21</td><td>Y20</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table> <p style="text-align: center;">↓ 4</p> <p>In refresh mode operation the data is written to and read from the output module via the output memory.</p> <p>In direct mode operation the data is written directly to the output module but read from the output module via the output memory.</p> </div>	1								2								b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	8000H	YF	YE	YD	YC	YB	YA	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	8002H	Y1F	Y1E	Y1D	Y1C	Y1B	Y1A	Y19	Y18	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	8004H	Y2F	Y2E	Y2D	Y2C	Y2B	Y2A	Y29	Y28	Y27	Y26	Y25	Y24	Y23	Y22	Y21	Y20																		
1								2																																																																																															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																																								
8000H	YF	YE	YD	YC	YB	YA	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0																																																																																							
8002H	Y1F	Y1E	Y1D	Y1C	Y1B	Y1A	Y19	Y18	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10																																																																																							
8004H	Y2F	Y2E	Y2D	Y2C	Y2B	Y2A	Y29	Y28	Y27	Y26	Y25	Y24	Y23	Y22	Y21	Y20																																																																																							
Internal relay (M) Latch relay (L) Step relay (S)	8400H — 84FFH	M/L/S 0 - 2047																																																																																																					
Link relay (B)	8600H — 867FH	B0 - 3FF	<div style="text-align: center;"> <table border="1" style="margin: auto;"> <tr> <td colspan="8" style="text-align: center;">1</td> <td colspan="8" style="text-align: center;">2</td> </tr> <tr> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td> <td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>8400H</td><td>M15</td><td>M14</td><td>M13</td><td>M12</td><td>M11</td><td>M10</td><td>M9</td> <td>M8</td><td>M7</td><td>M6</td><td>M5</td><td>M4</td><td>M3</td><td>M2</td><td>M1</td><td>M0</td> </tr> <tr> <td>8402H</td><td>M31</td><td>M30</td><td>M29</td><td>M28</td><td>M27</td><td>M26</td><td>M25</td> <td>M24</td><td>M23</td><td>M22</td><td>M21</td><td>M20</td><td>M19</td><td>M18</td><td>M17</td><td>M16</td> </tr> <tr> <td>8404H</td><td>M47</td><td>M46</td><td>M45</td><td>M44</td><td>M43</td><td>M42</td><td>M41</td> <td>M40</td><td>M39</td><td>M38</td><td>M37</td><td>M36</td><td>M35</td><td>M34</td><td>M33</td><td>M32</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table> <p style="text-align: center;">↓</p> </div>	1								2								b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	8400H	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	8402H	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M18	M17	M16	8404H	M47	M46	M45	M44	M43	M42	M41	M40	M39	M38	M37	M36	M35	M34	M33	M32																	
1								2																																																																																															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																																								
8400H	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0																																																																																							
8402H	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M18	M17	M16																																																																																							
8404H	M47	M46	M45	M44	M43	M42	M41	M40	M39	M38	M37	M36	M35	M34	M33	M32																																																																																							
Annunciator (F)	8700H — 871FH	F0 - 255																																																																																																					

- 1 Odd addresses
- 2 Even addresses
- 3 Area storing ON/OFF data status of an input module (read only)
- 4 Area storing operation results of the PLC (read and write)

Device	CPU Type	Address	Configuration
Special relay (M)	A3H A3M	8740H – 875FH	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">M 9000-9255</div> </div> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">T0 - 255</div> </div> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">C0 - 255</div> </div> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">T0 - 255</div> </div> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;">C0 - 255</div> </div> <div style="text-align: center;"> <p>8400 H    b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0</p> <p>8402 H    M15 M14 M13 M12 M11 M10 M9 M8 M7 M6 M5 M4 M3 M2 M1 M0</p> <p>8404 H    M47 M46 M45 M44 M43 M42 M41 M40 M39 M38 M37 M36 M35 M34 M33 M32</p> <p style="text-align: center;">↓ 3</p> </div>
Timer contact (T)		8780H – 879FH	
Counter contact (C)		87C0H – 87DFH	
Timer coil (T)		9C00H – 9C1FH	
Counter coil (C)		9C40H – 9C5FH	

- <sup>1</sup> Odd addresses
- <sup>2</sup> Even addresses
- <sup>3</sup> Area storing operation results of the PLC (read and write)

Device	CPU Type	Address	Configuration
Data register (D)	A3H A3M	8800H – 8FFFH	<div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 10px;">8800H</div> <div style="border: 1px solid black; width: 150px; height: 15px; margin-right: 5px;"></div> <div style="margin-right: 10px;">8801H</div> <div style="border: 1px solid black; width: 150px; height: 15px; margin-right: 5px;"></div> <div style="margin-left: 10px;">             b7-----b0              b15-----b8           </div> </div>
Link register (B)		9000H – 97FFH	
Current value of timer (T)		9800H – 99FFH	
Current value of counter (C)		9A00H – 9BFFH	
Special register (D)		9D00H – 9EFFH	
Accumulator (A0, A1)		9FF8H – 9FFAH	
Index register (Z, V)		9FFCH – 9FFEH	

Device	CPU Type	Address
<p>File Register (R)</p> <p>Block no. 0</p>	<p>A2</p> <p>A3</p> <p>A2N</p> <p>A2N-S1</p> <p>A2S</p> <p>A3N</p> <p>A3H</p> <p>A3M</p> <p>A1S</p> <p>A1S-S1</p> <p>A2C</p>	<p>Head address of file register</p> <p style="padding-left: 40px;">= 20000H + (capacity of RAM cartridge) - (capacity of file register)</p> <p>Capacity of RAM cartridges (Value for calculation)</p> <p style="padding-left: 40px;">A3(N)MCA-0 = 16 kByte A3(N)MCA-2 = 16 kByte A3(N)MCA-4 = 32 kByte A3(N)MCA-8 = 64 kByte A3MCA-12 = 96 kByte A3MCA-16 = 144 kByte (actual capacity: 128k) A3MCA-18 = 144 kByte A3MCA-24 = 144 kByte (actual capacity: 192 k) A3NMCA-40 = 144 kByte (actual capacity: 320 k) A3NMCA-56 = 144 kByte (actual capacity: 448 k)</p> <p style="text-align: right;">Value for calculation</p> <p>Comment data capacity: (Number of comments) x 16 Byte + 1 kByte</p> <p>File register capacity: (Number of file registers) x 2 Byte</p> <p>Note: For the calculation of capacities, note that 1 kByte equals 1024 Bytes and not 1000 Bytes.</p>
<p>Extension file register (R)</p> <p>Block no. 1 – 9</p>		<p>Head address of file register by each block number</p> <p style="padding-left: 40px;">= 20000H + (Capacity of RAM cartridge) - (comment data capacity) -(file register capacity) - (status latch capacity) -(sampling trace capacity) - 4000H x n</p> <p>Comment data capacity: (Number of comments) x 16 Byte + 1KByte</p> <p>File register capacity: (Number of file registers) x 2 Byte</p> <p>Status latch capacity: Number of set bytes</p> <p>Sampling trace capacity: 8 kByte (if setting is provided)</p> <p>n: Block number</p>

Device	CPU Type	Address			
Extension file register (R)  Block no. 10 - 28	A2 A3 A2N A2N-S1 A2S A3N A3H A3M A1S A1S-S1 A2C	Memory cartridges			
		1		2	
		3	4	3	4
		11	38000 H	28	A0000 H
		10	3C000 H	27	A4000 H
				26	A8000 H
				25	AC000 H
				24	B0000 H
				23	B4000 H
				22	B8000 H
				21	BC000 H
				20	C0000 H
				19	C4000 H
				18	C8000 H
				17	CC000 H
				16	D0000 H
				15	D4000 H
				14	D8000 H
				13	DC000 H
				12	E4000 H
				11	E8000 H
				10	EC000 H

- <sup>1</sup> A3NMCA-16
- <sup>2</sup> A3NMCA-24, 40 oder 56
- <sup>3</sup> Block no.
- <sup>4</sup> Head address





# 13 Error Codes

If an error occurs when the PLC is turned ON, set into RUN mode, or during operation, the self diagnostic functions of the CPU returns an error (LED indication or message on LED display) and store the error information in special relays (M) or diagnostic relays (SM) and in the special register (D9008) or diagnostic registers (SD).

### 13.1 Table of error codes; Q00J, Q00 and Q01CPU

The following table contains an overview of all possible errors and the corresponding error messages, possible causes, and remedial advice. Only the error messages of the Q00JCPU, Q00CPU and the Q01CPU are included.

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
1000	MAIN CPU DOWN	—	—	OFF	Flashes/ON	Stop	Always
1010	END NOT EXECUTE	—	—	OFF	Flashes	Stop	When an END instruction is executed.
1011							
1012							
1101	RAM ERROR	—	—	OFF	Flashes	Stop	At power ON/At reset
1102							
1103							
1104							
1200	OPE. CIRCUIT ERR.	—	—	OFF	Flashes	Stop	At power ON/At reset
1201							
1202							
1300	FUSE BREAK OFF	Unit/Module no.	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When an END instruction is executed.
1310	I/O INT ERROR	Unit/Module no.	—	OFF	Flashes	Stop	During interrupt
1401	SP. UNIT DOWN	Unit/Module no.	—	OFF	Flashes	Stop/ Continue <sup>3</sup>	At power ON/At reset When an intelligent function module is accessed.
1402			Program error location				When an intelligent function module is accessed.
1403			—				When an END instruction is executed.

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

<sup>3</sup> Stop/continue operation is selectable for each module by setting parameters..

Error description and cause	Remedy
RUN mode suspended or failure of main CPU 1.) Malfunction due to noise or other reasons 2.) Hardware fault	1.) Measure noise level. 2.) Reset CPU and switch back to RUN mode. If the same error is indicated again this hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.
Entire program was executed without execution of an END instruction. 1.) When END instruction is executed it is read as a different instruction code. 2.) The END instruction was altered to another instruction code.	
Error in internal RAM where CPU sequence program is stored. Error in RAM used as CPU work area. Internal CPU error RAM address error in CPU	This hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.
Malfunction of operation circuit performing CPU-internal index qualification.	
Malfunction of CPU hardware (Logic) Malfunction of circuit performing sequence processing.	
Blown fuse in output module.	1.) Check ERR LEDs of output modules and replace the module whose LED is lit. 2.) The module with a blown fuse can also be checked with a programming terminal. Monitor the special registers SD130 to SD137 on the display of a programming terminal and check if there is a bit set (1), which corresponds to the module with the blown fuse.
An interrupt occurred although there is no interrupt module in the system.	One of the connected modules has a hardware fault. Check the connected modules. Please contact your nearest MITSUBISHI service.
1.) There was no response from an intelligent function module during initial communications. 2.) The size of the buffer memory of the intelligent function module is abnormal.	The CPU module has a hardware fault. Please contact your nearest MITSUBISHI service.
A special function module was accessed in the program, but there was no response.	The CPU module has a hardware fault. Please contact your nearest MITSUBISHI service.
1.) There was no response from the intelligent function module when the END instruction is executed. 2.) An error is detected at the intelligent function module.	The accessed special function module has a hardware fault. Please contact your nearest MITSUBISHI service.

Error codes 1411 to 2112

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
1411	CONTROL-BUS ERR.	Unit/Module no.	Program error location	OFF	Flashes	Stop	At power ON/At reset
1412							During execution of FROM/TO instruction set.
1413		—	—				Always
1414		—	—				When an END instruction is executed.
1415		Base no.	—				
1500	AC DOWN	—	—	ON	OFF	Continue	Always
1600	BATTERY ERROR	Drive name	—	ON	OFF	Continue	Always
2000	UNIT VERIFY ERR.	Unit/Module no	—	OFF/ON	Flashes/ON	Stop/Continue <sup>2</sup>	When an END instruction is executed.
2100	SP. UNIT LAY ERR.	Unit/Module no.	—	OFF	Flashes	Stop	At power ON/At reset
2103							
2106							
2107							
2110	SP UNIT ERROR	Unit/Module no	Program error location	OFF/ON	Flashes/ON	Stop/Continue <sup>2</sup>	When instruction is executed.
2111							When instruction is executed. STOP → RUN
2112							

<sup>1</sup> Specifications in parentheses ( ) indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

Error description and cause	Remedy
After performing a parameter I/O allocation a special function module cannot be accessed at initial communications. On occurrence of this error the initial I/O number of the according module is stored	Either a special function module or the CPU module or a base unit has a hardware fault. Please contact your nearest MITSUBISHI service.
The FROM/TO instruction set could not be executed due to a control pulse error with a special function module. On occurrence of this error the program error location is stored.	
An error is detected on the system bus (Wait-length time-out, arbitration time-out).	
An error is detected on the systembus.	
Fault of the main or extension base unit was detected.	
A momentary interruption of the power supply occurred.	Check the power supply.
1.) Low voltage of CPU module battery 2.) The battery of the CPU battery is not connected.	1.) Replace battery. 2.) Install a lead connector, if the battery is intended for the internal RAM or backup.
I/O module information changed at power ON I/O module (or special function module) not installed properly on the base unit	Read the common error information on the display of a programming terminal and check the installation of the according module on the base unit. Alternatively, monitor the special registers SD150 to SD157 on the display of a programming terminal and check the installation of the modules of which the according bit is set „1,“.
In the parameter I/O allocation settings, an intelligent function module was allocated to a location reserved for an I/O module or vice versa. In the parameter I/O allocation settings, a module other than CPU was allocated to a location reserved for a CPU module or vice versa. No CPU module was allocated to a location for a CPU module. A general-purpose switch was set to the module with no general purpose switches.	Correct the parameter I/O allocation settings accordingly.  Reset the general-purpose switch settings.
More than one QI60 interrupt module is installed on the base unit.	Install one QI60 module only
1.) More than one MELSECNET/H module is installed. 2.) More than one Ethernet module is installed. 3.) More than two CC-Link modules are installed. 4.) There are identical network or station numbers in a MELSECNET/H network.	1.) Run max. 1 module. 2.) Run max. 1 module. 3.) Run max. 2 modules. 4.) Check network and station numbers.
The head X/Y set in the parameter I/O allocation settings is also the head X/Y for another module.	Reset the parameter I/O allocation settings and adopt it to the actual status.
The module addressed by a FROM/TO instruction set is not a special function module. The special function module being accessed is faulty.	Read individual error information then check and edit the FROM/TO instruction set that corresponds to the numeric value there (program error location). In case of a faulty module, please contact your nearest MITSUBISHI service.
The location designated by link direct device is not a network module.	
The designated special function module is not a special function module or not the relevant one.	

Error codes 2120 to 3004

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
2120	SP. UNIT LAY ERR.	—	—	OFF	Flashes	Stop	At power ON/At reset
2122							
2124							
2125							
2200	MISSING PARA.	Drive Name	—	OFF	Flashes	Stop	At power ON/At reset
2400	FILE SET ERROR	File-Name	Parameter number	OFF	Flashes	Stop	At power ON/At reset
2401							
2500	CAN'T EXE. PRG.	File-Name	—	OFF	Flashes	Stop	At power ON/At reset
2501							
2502							
2503							
3000	PARAMETER ERROR	File-Name	Parameter number	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
3001							
3003							
3004							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

Error description and cause	Remedy
A QA[]B or QA1S[]B is used as the base unit.	Use a Q[]B as the base unit.
A QA1S[]B is used as the main base unit.	Use a Q[]B as the main base unit.
<p>A module is installed at the 25th or later slot (17th or later for Q00JCPU).  A module is installed at the slot later than the number of slots specified with base allocation setting.  A module is installed at the I/O points later than the actual I/O points.  A module installed at the last I/O point occupies more points.  More than 4 extension bases are connected (more than 2 extension bases for Q00JCPU).</p>	<p>Remove the module installed at the 25th or later slot (17th or later for Q00JCPU).  Remove the module installed at the slot later than the number of slots specified with base allocation setting.  Remove the module installed at the I/O points later than the actual I/O points.  Change the last module for a module whose occupying points do not exceed the actual I/O points.  Connect max. 4 extension bases (2 for Q00JCPU).</p>
<p>A module which the CPU cannot recognize has been installed.  There was no response from an intelligent function module.</p>	<p>Install a module, which can be used with the Q-CPU.  The intelligent function module has a hardware fault. Please contact your nearest MITSUBISHI service.</p>
There is no parameter file at the program memory.	<p>Check and correct the setting of the parameter enabled drive switch.  Provide a parameter file on the designated drive.</p>
The file designated by the PC file settings in the parameters cannot be found.	<p>Read the individual error information on the display of a programming terminal and ensure that the drive and file destination correspond to the numerical values there.  Create the designated file.</p>
The file designated by the parameter PC RAS settings fault history area was not created.	<p>Read the individual error information on the display of a programming terminal and ensure that the drive and file destination correspond to the numerical values there.  Check the remaining free memory on the memory card.</p>
There is a program file that uses a device that exceeds the device allocation range designated by the parameter device settings.	<p>Read the individual error information on the display of a programming terminal and ensure that the parameter device settings and the program file device allocation correspond to the numerical values there (file name).</p>
There are multiple program files although „none“ is set in the parameter program settings.	<p>Edit the parameter program settings to 'yes'. Delete unneeded programs.</p>
<p>The program file is not correct.  Alternatively, the file contents are not those of a sequence program.</p>	<p>Check whether the file format is *.QPG and whether the file contents are intended for a sequence program.</p>
There are no program files at all.	<p>Check the program configuration and the parameters.</p>
<p>The parameter settings for timer time limit setting, the RUN-PAUSE contact, the common pointer number, the general data processing, number of vacant slots, or system interrupt settings exceed the relevant CPU range.</p>	<p>1.) Read the detailed error information on the display of the programming terminal, check the parameter items corresponding to the numerical values (parameter numbers) there, and correct if necessary.</p>
Parameter settings were destroyed.	<p>2.) If the error is still generated, this hints to a memory error either in the internal CPU RAM or on the memory card.</p>
The number of devices set at the parameter device settings exceeds the relevant CPU range.	<p>Please contact your nearest MITSUBISHI service.</p>
The parameter file is incorrect. Alternatively, the contents of the file are not parameters.	<p>Check whether the file format is *.QPA and whether the file actually contains parameters.</p>

Error codes 3100 to 4030

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
3100	LINK PARA. ERROR	File-Name	Parameter number	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
3101							
3102							
3103							
3104							
3105							
3106							
3107							
3300	SP. PARA. ERROR	File-Name	Parameter number	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
3301							
3302							
4000	INSTRCT CODE. ERR.	Program error location	—	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
4002							
4003							
4004							
4010	MISSING END INS.	Program error location	—	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
4021	CAN'T SET (P)						
4030	CAN'T SET (I)						

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.



Error description and cause	Remedy
1.) The number of actually installed modules is different from that designated in the number of modules setting parameter of MELSECNET/H 2.) The head I/O number of actually installed modules is different from that designated in the network setting parameter of MELSECNET/H 3.) Some of the data in the parameter cannot be handled. 4.) The station type of MELSECNET/H has been changed, while the power is on. (A change from RESET to RUN is required to change the station type).	1.) Write after correcting the network parameters. 2.) If the error is still generated, please contact your nearest MITSUBISHI service
The network no. specified by a parameter is different from that of the actually mounted network. The head I/O number specified by a parameter is different from that of the actually mounted I/O unit. The network class specified by a parameter is different from that of the actually mounted network. The network refresh parameter of the MELSECNET/H is out of the specified area.	Match the data specified by the parameters with those of the actually mounted network and units.
An error was discovered when the network parameter check was made at the network module.	1.) Write after correcting the network parameters. 2.) If the error is still generated, please contact your nearest MITSUBISHI service.
Though the number of units for the Ethernet unit quantity set parameter is set at one or more, the actually mounted number of units is zero. The head I/O number for the Ethernet set parameters is different from that of the actually mounted I/O unit.	
Ethernet and MELSECNET/H use the same network number. Network number, station number and group number set by the parameter is out of range. The I/O number is out of range of the CPU used. Ethernet-specific parameter setting is not normal. The parameter peculiar to Ethernet is not normal.	
Though the number of units for the CC-Link unit quantity set parameters is set at one or more, the actually mounted number of units is zero. The head I/O number for the common parameters is different from that of the actually mounted I/O unit. The station class for the CC-Link unit quantity set parameters is different from that of the actually mounted station.	
The network refresh parameter for CC-Link is out of range.	
The contents of the CC-Link parameter are incorrect.	
The first I/O number in the intelligent function module parameter set on GX Configurator differs from the actual I/O number.	Check the parameter setting.
The refresh range of the intelligent function module exceeds the file register capacity The intelligent function module's refresh parameter setting is outside the available range.	
The intelligent function module's refresh parameter are incorrect.	
The program contains an instruction code that cannot be decoded. An unusable instruction is included in the program.	Read the common error information on the display of the programming terminal and check the files corresponding to the numerical values there (program error location).
The instruction name is incorrect.	
The instruction designates an incorrect number of devices.	
The instruction designates a device that cannot be used.	
The program contains no END-(FEND-) instruction.	
The common pointer numbers used by individual files overlap. The allocation pointer numbers used by individual files overlap.	

Error codes 4100 to 9000

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED-Status		CPU Status	Diagnostic timing
				RUN	ERROR		
4100	OPERATION ERROR	Program error location	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When an instruction is executed.
4101							
4102							
4108							
4200	FOR NEXT ERROR	Program error location	—	OFF	Flashes	Stop	When an instruction is executed.
4201							
4202							
4203							
4210	CAN'T EXECUTE ( P )	Program error location	—	OFF	Flashes	Stop	When an instruction is executed.
4211							
4212							
4213							
4220	CAN'T EXECUTE ( I )	Program error location	—	OFF	Flashes	Stop	When an instruction is executed.
4221							
4223							
4231	INST. FORMAT ERR	Program error location	—	OFF	Flashes	Stop	When an instruction is executed.
5001	WDT ERROR	Time (Set value)	Time (value actually measured)	OFF	Flashes	Stop	Always
5010	PRG. TIME OVER	Time (Set value)	Time (value actually measured)	ON	ON	Continue	Always
9000	F**** <sup>3</sup>	Program error location	Annunciator number	ON	OFF	Continue	When an instruction is executed.
				USER LED ON			

<sup>1</sup> Specifications in parentheses ( ) indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

<sup>3</sup> \*\*\*\* indicates the annunciator number.

Ursache	Abhilfe
The instruction cannot process the contained data.	Read the common error information on the display of the programming terminal and check the indicated program step.
The designated device numbers for data processed by the instruction exceed the usable device range. Alternatively, the stored data or constants for the devices designated by the instruction exceed the usable range.	
The network number or station number designated by a dedicated network instruction is incorrect. The link direct device is not set correctly.	
The CC-Link parameter are not set when the CC-Link instruction is executed.	Execute the CC-Link instruction after setting the CC-Link parameter.
No NEXT instruction was executed following the execution of a FOR instruction. Alternatively, there are fewer NEXT instructions than FOR instructions.	Read the common error information on the display of the programming terminal and check the indicated program step.
A NEXT instruction is executed without a prior FOR instruction. Alternatively, there are more NEXT instruction than FOR instructions.	
More than 16 nesting levels are programmed.	Reduce the number of nesting levels to 16 max.
A BREAK instruction is executed without a prior FOR instruction.	Read the common error information on the display of the programming terminal and check the indicated program step.
The CALL instruction is executed but there is no subroutine at the specified pointer.	
The executed subroutine contains no RET instruction.	
The RET instruction is programmed prior to the FEND instruction.	
More than 16 nesting levels are programmed.	Reduce the number of nesting levels to 16 max.
An interrupt was generated but no corresponding interrupt pointer was found.	Read the common error information on the display of the programming terminal and check the indicated program step.
The executed subroutine contains no IRET instruction.	
The IRET instruction is programmed prior to the FEND instruction.	
The IX and IXEND instructions are not programmed in combination. The numbers of IX and IXEND instructions are equal.	Read the individual error information on the display of the programming terminal and check the numerical value (time) there, and shorten scan time if necessary.
The program scan time exceeds the WDT setting value designated by the parameter PC RAS setting.	
The run time of a low-speed execution type program that is set in the parameter PC RAS setting exceeds the margin time of constant scan.	Check and change the constant scan time and the run time for the low-speed execution type program.
The annunciator F was set ON.	Read the individual error information on the display of the programming terminal and check the program corresponding to the numerical value (annunciator number).

## Table of Error Codes; QnA CPUs and System Q

### 13.2 Table of Error Codes; QnA CPUs and System Q

The following table contains an overview of all possible errors and the corresponding error messages, possible causes, and remedial advice. Only the error messages of the Q02(H), Q06H, Q12H, Q25H, QnA, QnAS and Q4AR CPUs are listed. The sign „●“ in the corresponding CPU column indicates that the error is applied to all types of CPUs mentioned above. „Rem“ indicates compatibility with remote I/O modules. A CPU type name in this column indicates that the error is applied to the specific type of CPU only.

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
1000	MAIN CPU DOWN	—	—	OFF	Flashes/ON	Stop	Always
1001							
1002							
1003							
1004							
1005							
1006							
1007							
1008							
1009							
1010	END NOT EXECUTE	—	—	OFF	Flashes	Stop	When an END instruction is executed.
1011							
1012							
1101	RAM ERROR	—	—	OFF	Flashes	Stop	At power ON/At reset
1102							
1103							
1104							
1105							
1200	OPE. CIRCUIT ERR.	—	—	OFF	Flashes	Stop	At power ON/At reset
1201							
1202							
1203							
1204							
1205							
1206							
							When an instruction is executed.

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

## Table of Error Codes; QnA CPUs and System Q

	Error description and cause	Remedy	Valid for:
	RUN mode suspended or failure of main CPU 1.) Malfunction due to noise or other reasons 2.) Hardware fault	1.) Measure noise level. 2.) Reset CPU and switch back to RUN mode. If the same error is indicated again this hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.	QnA CPU
	RUN mode suspended or failure of main CPU 1.) Malfunction due to noise or other reasons 2.) Hardware fault	1.) Measure noise level. 2.) Reset CPU and switch back to RUN mode. If the same error is indicated again this hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU
			Q CPU/Rem
			Q CPU
	Entire program was executed without execution of an END instruction. 1.) When END instruction is executed it is read as a different instruction code. 2.) The END instruction was altered to another instruction code.	1.) Measure noise level. 2.) Reset CPU and switch back to RUN mode. If the same error is indicated again this hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.	
	Error in internal RAM where CPU sequence program is stored.	This hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.	●
	Error in RAM used as CPU work area.		
	Internal CPU error		
	RAM address error in CPU.		
	CPU shared memory fault	1.) Measure noise level. 2.) Reset CPU and switch back to RUN mode. If the same error is indicated again this hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU (Ver. B or later)
	Malfunction of operation circuit performing CPU-internal index qualification.	This hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.	●
	Malfunction of CPU hardware (Logic)		
	Malfunction of circuit performing sequence processing.		
	Malfunction of operation circuit performing CPU-internal index qualification.	This hints to a CPU hardware fault. Please contact your nearest MITSUBISHI service.	Q4AR CPU
	Malfunction of CPU hardware (Logic)		
	Malfunction of circuit performing sequence processing.		
	The DSP operation circuit in the CPU is not operating properly.		

# Table of Error Codes; QnA CPUs and System Q

## Error codes 1300 to 1413

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
1300	FUSE BREAK OFF	Unit/Module no..	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When an END instruction is executed.
1301	EX POWER OFF	Unit/Module no.	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When an END instruction is executed.
1310	I/O INT ERROR	Unit/Module no.	—	OFF	Flashes/ON	Stop	During an interrupt
1401	SP. UNIT DOWN	Unit/Module no.	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>3</sup>	At power ON/At reset When an intelligent function module is accessed.
1402			Program error location				When an intelligent function module is accessed.
1403			—				During execution of FROM/TO instruction set
1411	CONTROL-BUS ERR.	Unit/Module no.	Program error location	OFF	Flashes	Stop	At power ON/At reset
1412							During execution of FROM/TO instruction set
1413	CONTROL-BUS ERR.	—	—	OFF	Flashes	Stop	Always

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

<sup>3</sup> This error can only be detected in redundant systems. Detection is possible in either the control system or the standby system.

<sup>4</sup> Stop/continue operation is selectable for each module by setting parameters.

## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
Blown fuse in output module	<ol style="list-style-type: none"> <li>1.) Check ERR LEDs of output modules and replace the module whose LED is lit.</li> <li>2.) The module with a blown fuse can also be checked with a programming terminal. Monitor the special registers SD1300 to SD1331 and check if a bit is set („1“), which corresponds to the module with a blown fuse.</li> </ol>	Q CPU Rem
Blown fuse in output module	<ol style="list-style-type: none"> <li>1.) Check blown fuse indicator LEDs of output modules and replace according fuse.</li> <li>2.) Read common error information on display of programming terminal, and replace fuse of indicated output module. Alternatively, monitor the special registers SD1300 to SD1331 on the display of a programming terminal and replace the fuse of the output module of which the according bit is set („1“).</li> </ol>	QnA CPU, Q4AR CPU
Blown fuse in output module The external power supply for output load is turned off or is disconnected.	<ol style="list-style-type: none"> <li>1.) Check blown fuse indicator LEDs of output modules and replace according fuse.</li> <li>2.) Read common error information on display of programming terminal, and replace fuse of indicated output module. Alternatively, monitor the special registers SD1300 to SD1331 on the display of a programming terminal and replace the fuse of the output module of which the according bit is set („1“).</li> <li>3.) Check whether the external power supply for output load is off or disconnected.</li> </ol>	Q2AS CPU
The external power supply for output load is turned off or is disconnected. (For future use).	Check the external power supply for output loads.	Q CPU Rem
An interrupt occurred although there is no interrupt module in the system.	One of the connected modules has a hardware fault. Check the connected modules. Please contact your nearest MITSUBISHI service.	●
<ol style="list-style-type: none"> <li>1.) There was no response from an intelligent function module during initial communications.</li> <li>2.) The size of the buffer memory of the intelligent function module is abnormal.</li> </ol>	The CPU module has a hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU Rem
After performing a parameter I/O allocation there was no return signal from the special function module at initial communications. On occurrence of this error the initial I/O number of the according module is stored.	The accessed special function module has a hardware fault. Please contact your nearest MITSUBISHI service.	QnA CPU
A special function module was accessed in the program, but there was no response.	The CPU module has a hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU Rem
A special function module was accessed during the execution of a FROM/TO instruction set but there was no response. On occurrence of this error the program error location is stored.	The accessed special function module has a hardware fault. Please contact your nearest MITSUBISHI service.	QnA CPU
<ol style="list-style-type: none"> <li>1.) There was no response from the intelligent function module when the END instruction is executed.</li> <li>2.) An error is detected at the intelligent function module.</li> </ol>		Q CPU Rem
After performing a parameter I/O allocation a special function module cannot be accessed at initial communications. On occurrence of this error the initial I/O number of the according module is stored.	Either a special function module or the CPU module or a base unit has a hardware fault. Please contact your nearest MITSUBISHI service.	● Rem
The FROM/TO instruction set could not be executed due to a control pulse error with a special function module. On occurrence of this error the program error location is stored.		●
A System Q CPU of function version A is used in a multi-CPU system.	<ol style="list-style-type: none"> <li>1.) Change the CPU module to one of function version B or later.</li> <li>2.) A special function module or the CPU module or a base unit has a hardware fault. Please contact your nearest MITSUBISHI service.</li> </ol>	Q CPU (Ver. B or later)
An error is detected on the system bus (Wait-length time-out, arbitration time-out).	Either a special function module or the CPU module or a base unit has a hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU Rem

# Table of Error Codes; QnA CPUs and System Q

## Error codes 1414 to 2101

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
1414	CONTROL-BUS ERR.	Unit/Module no.	—	OFF	Flashes	Stop	When an END instruction is executed.
1415		—					
1416		Base no.					
1421	SYS. UNIT DOWN <sup>3</sup>	—	—	OFF	Flashes	Stop	At power ON/At reset
1500	AC DOWN	—	—	ON	OFF	Continue	Always
1510	DUAL DC DOWN 5V <sup>4</sup>	—	—	ON	ON	Continue	Always
1520	DC DOWN 5V <sup>5</sup>	—	—	OFF	Flashes	Stop	Always
1530	DC DOWN 24V <sup>3</sup>	—	—	ON	ON	Continue	Always
1600	BATTERY ERROR	Drive name	—	ON	OFF	Continue	Always
1601				BAT. ALM LED ON			
1602							
2000	UNIT VERIFY ERR.	Unit/Module no.	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When an END instruction is executed.
2100	SP. UNIT LAY ERR.	Unit/Module no.	—	OFF	Flashes	Stop	At power ON/At reset
2101							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

<sup>3</sup> This error can only be detected in redundant systems. Detection is possible in either the control system or the standby system.

<sup>4</sup> This error can only be detected in redundant systems.

<sup>5</sup> This error can be detected in either a standalone system or a in the control system of a redundant system.



## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
Malfunction in one the installed modules. A System Q CPU of function version A is used in a multi-CPU system.	1.) Change the CPU module to one of function version B or later. 2.) A special function module or the CPU module or a base unit has a hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU (Ver. B or later)
An error is detected on the system bus.	Either a special function module or the CPU module or a base unit has a hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU Rem
Fault of the main or extension base unit.	Either a special function module or the CPU module or a base unit has a hardware fault. Please contact your nearest MITSUBISHI service.	Q CPU (Ver. B or later)
A Bus fault was detected at power-on or reset.		
Hardware fault at the system management module AS92R.	Please contact your nearest MITSUBISHI service.	Q4AR CPU
A momentary interruption of the power supply occurred.	Check the power supply.	●/Rem
The 5 V DC voltage of one of the two power supplies in the redundant system extension bases has dropped below 85 % of the rated voltage.	Check the supply voltage of the power module. Replace the power module if the voltage is below the rated voltage.	Q4AR CPU
The 5 V DC supplied voltage of the power supply modules in the extension base has dropped below 80 % of the rated voltage.		
The 24 V DC supplied to the system management module AS92R has dropped below 85 % of the rated voltage.	Check the power supply.	
1.) Low voltage of CPU battery 2.) CPU module battery is not connected.	1.) Replace battery. 2.) Install a lead connector, if the battery is intended for the internal RAM or backup.	●
Low voltage of the battery in memory card 1.	Replace the battery.	●
Low voltage of the battery in memory card 2.		QnA CPU
I/O module information changed at power ON. I/O module (or special function module) not installed properly on the base unit.	Read the common error information on the display of a programming terminal and check the installation of the according module on the base unit. Alternatively, monitor the special registers SD1400 to SD1431 on the display of a programming terminal and check the installation of the modules of which the according bit is set „1“.	● Rem
A System Q CPU of function version A is used in a multi-CPU system.	Change the CPU module to one of function version B or later.	Q CPU (Ver. B or later)
A Slot, in which a QI60 interrupt module is installed, is set to other than intelligent function module or interrupt module.	Make settings to match the actual loading status.	Q CPU (Ver. B or later)
In the parameter I/O allocation settings, an intelligent function module was allocated to a location reserved for an I/O module or vice versa. In the parameter I/O allocation settings, a module other than CPU was allocated to a location reserved for a CPU module or vice versa. No CPU module was allocated to a location for a CPU module. A general-purpose switch was set to the module with no general purpose switches.	Correct the parameter I/O allocation settings accordingly.  Reset the general-purpose switch settings.	Q CPU Rem
In parameter I/O allocation settings a special function module was allocated to a location reserved for an I/O module or vice versa.	Correct the parameter I/O allocation settings accordingly.	QnA CPU
More than 12 special function modules (except A1SI61 and QI60) capable of sending an interrupt to the CPU are installed.	Reduce the number of special function modules (except A1SI61 and QI60) to 12 or less.	Q CPU
More than 12 special function modules (except A161) capable of sending an interrupt to the CPU are installed.	Reduce the number of special function modules (except A161) to 12 or less.	QnA CPU

# Table of Error Codes; QnA CPUs and System Q

## Error codes 2102 to 2109

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
2102	SP. UNIT LAY ERR.	Unit/Module no.	—	OFF	Flashes	Stop	At power ON/At reset
2103							
2104							
2105							
2106	SP. UNIT LAY ERR.	Unit/Module no.	—	OFF	Flashes	Stop	At power ON/At reset
2107							
2108							
2109 <sup>2</sup>							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> This error can only be detected in the standby system of a redundant systems.

## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
More than 6 modules A1SD51S are installed.	Reduce the number of A1SD51S to 6 or less.	Q CPU
More than 6 serial communications modules are installed (except A(1S)J71QC24).	Reduce the number of serial communications modules (except A(1S)J71QC24) to 6 or less.	QnA CPU Rem
1.) More than one QI60 or A1SI61 interrupt module is installed in a single-CPU system. 2.) More than one QI60/A1SI61 module is set to the same control CPU in a multi-CPU system. 3.) More than one A1SI61 module is installed in a multi-CPU system.	1.) Reduce the number of QI60 or A1SI61 in a single-CPU system to one. 2.) Each CPU can control one QI60 or A1SI61 only. 3.) Install only one A1SI61 in a multi-CPU system. Use the QI60 when each CPU of a multi-CPU system shall control an interrupt module. (A combination of one A1S61 plus 3 QI60 is possible. Or use the QI60 modules only.)	Q CPU (Ver. B or later)
More than one A1SI61 or QI60 interrupt module is installed on the base unit.	Install one A1SI61 or QI60 module only	Q CPU
More than one A1SI61 interrupt module is installed on the base unit.	Install one A1SI61 module only.	QnA CPU
At the MELSECNET/MINI auto refresh parameter settings the current module allocation does not correspond to actual module models at the station numbers in the link system.	Reset and correct the MELSECNET/MINI auto refresh parameter settings.	QnA CPU
The maximum number of special function modules executing dedicated instructions allocated to a CPU module is exceeded (max. number is 1344). (number of installed AD59 x 5) (number of installed AD57(S1)/AD58 x 8) (number of installed AJ71C24(S3/S6/S8) x 10) (number of installed AJ71UC24 x 10) (number of installed AJ71C21(S1) x 29) (number of installed AJ71PT32(S3) x 125)* (number of installed AJ71QC24 (R2, R4) x 29) (number of installed AJ71ID1 (2)-R4 x 18) (number of installed AD75 x 12)  Total must be equal to or below 1344 *: When the expansion mode is used.	Reduce the number of installed special function modules.	QnA CPU
1.) More than four MELSECNET/H modules are installed in a multi-CPU system. 2.) More than four System Q ETHERNET modules are installed in a multi-CPU system.	Run max. 4 modules	Q CPU (Ver. B or later)
1.) More than four MELSECNET/H modules are installed. 2.) More than four System Q ETHERNET modules are installed. 3.) Identical Network or station numbers exist in a MELSECNET/H network.	1.) Run max. 4 modules. 2.) Run max. 4 modules. 3.) Check the network and station numbers.	Q CPU Rem
1.) More than four AJ71QLP21 or AJ71QBR11 are installed in the system. 2.) More than two AJ71AP21/R21 or AJ71AT21B are installed in the system. 3.) More than four AJ71QLP21, AJ71QBR11, AJ71AP21/R21, or AJ71AT21B are installed in the system. 4.) There are identical network or station numbers in a MELSECNET/10 network. 5.) There is more than one master station or local station in a MELSECNET (II) or MELSECNET/B network.	1.) Run max. 4 modules 2.) Run max. 2 modules 3.) Run max. 4 modules  4.) Check network and station numbers 5.) Check station numbers	QnA CPU
The head X/Y set in the parameter I/O allocation settings is also the head X/Y for another module.	Reset the parameter I/O allocation settings and adopt it to the actual status.	● Rem
Network modules A1SJ71LP21, A1SJ71BR11, A1SJ71AP21, A1SJ71AR21 or A1SJ71AT2 intended for an A2US CPU network are installed. Network modules A1SJ71QLP21 or A1SJ71QBR11 intended for a Q2AS CPU network are installed.	Replace the modules by a QJ71LP21 or a QJ71BR11.	Q CPU
The modules A(1S)J71LP21 or A(1S)J71BR11 intended for an AnU CPU network are installed.	Replace the modules by an A(1S)J71QLP21 or an A(1S)J71QBR11.	QnA CPU
The control system and standby system module configurations are different when a redundant system is in the backup mode.	Check the module configuration of the standby system.	Q4AR CPU

# Table of Error Codes; QnA CPUs and System Q

## Error codes 2110 to 2150

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
2110	SP UNIT ERROR	Unit/Module no.	Program error location	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	During execution of FROM/TO instruction set.
2111							
2112	SP UNIT ERROR	Unit/Module no.	Program error location	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	During execution of FROM/TO instruction set.
2113		FFFFH (fest)					
2114	SP UNIT ERROR	Unit/Module no.	Program error location	Flashes/ ON	Flashes/ON	Stop/ Continue	When an instruction is executed
2115							
2116							
2117							
2120	SP. UNIT LAY ERR.	—	—	OFF	Flashes	Stop	At power ON/At reset
2121							
2122							
2124							
2125							
2126							
2150	SP.UNIT VER. ERR.	Unit/Module no.	—	OFF	Flashes	Stop	At power ON/At reset

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

## Table of Error Codes; QnA CPUs and System Q

	Error description and cause	Remedy	Valid for:
	A non existing CPU was specified in an instruction for accessing CPU shared memory.	Read individual error information then check the program that corresponds to the numerical value there (program error location).	Q CPU (Ver. B or later)
	The module addressed by a FROM/TO instruction set is not a special function module. The special function module being accessed is faulty.	Read individual error information then check and edit the FROM/TO instruction set that corresponds to the numeric value there (program error location).	●
	The location designated by link direct device is not a network module.	In case of a faulty module, please contact your nearest MITSUBISHI service.	●
	The designated special function module is not a special function module or the relevant one. The designated network number does not exist, or the network module is not the corresponding one.	Read individual error information then check and edit the special function module (network module) dedicated instruction that corresponds to the numerical value there (program error location).	● Rem
	The module specified in a network dedicated instruction is not a network module, or a relay target network does not exist.		●
	The CPU which is executing the instruction is specified in an instruction for which the specification of another CPU is necessary.	Read individual error information then check and edit the program that corresponds to the numerical value there (program error location).	Q CPU (Version B or later)
	Another CPU is specified in an instruction for which the specification of the CPU where this instruction is executed is necessary.		
	An intelligent function module under control of another station was designated in an instruction.		
	A CPU that cannot be specified in the instruction dedicated to the multi-CPU system was specified.		
	The location of Q[]B and QA1S[]B is not correct.	Check the location of the base unit.	Q CPU Rem
	The CPU mode is not installed at the CPU slot or slots 0 to 2.	Check the location of the CPU module.	
	A QA1S[] is installed as the main base unit.	Install the Q[]B as the main base unit.	
	A module is installed at the 65th or later slot. A module is installed at the slot later than the number of slots specified with base allocation setting. A module is installed at the I/O points later than the 4096th point. A module installed at the 4096th point occupies later points.	Remove the module installed at the 65th or later slot. Remove the module installed at the slot later than the number of slots specified with base allocation setting. Remove the module installed at the I/O points later than the 4096th point. Change the last module to a module which does not exceed the 4096th point.	
	A module which the System Q CPU cannot recognize has been installed. There was no response from the intelligent function module.	Install a module, which can be used with the System Q CPU. The intelligent function module has a hardware fault. Please contact your nearest MITSUBISHI service.	
	1.) In a multi-CPU system there is an empty slot between the CPU modules. 2.) A module other than a CPU module (i.e. an I/O-module or a motion controller) is installed between two CPU modules.	1.) No empty slot is allowed between CPU modules. On the right side of the CPU modules slots can be left vacant. 2.) Remove the module that is installed between the CPU modules. Install motion controller on the right side of System Q (PLC) CPUs.	
	An intelligent function module which is incompatible with the multi-CPU system is allocated to CPU 2, 3, or 4.	1.) Replace the module with one that is compatible with the multi-CPU system. 2.) Allocate the module which is incompatible with the multi-CPU system to CPU 1.	Q CPU (Version B or later)

# Table of Error Codes; QnA CPUs and System Q

## Error codes 2200 to 2413

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
2200	MISSING PARA.	Drive name	—	OFF	Flashes	Stop	At power ON/At reset
2210	BOOT ERROR	Drive name	—	OFF	Flashes	Stop	At power ON/At reset
2300	ICM. OPE. ERROR	Drive name	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When a memory card is inserted or removed.
2301							
2302							
2400	FILE SET ERROR	File name	Parameter number	OFF	Flashes	Stop	At power ON/At reset
2401							
2410	FILE OPE. ERROR	File name	Program error location	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When an instruction is executed.
2411							
2412							
2413							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
There is no parameter file at the drive designated by DIP switches as a valid drive switch.	Check and correct the setting of the parameter enabled drive switch. Provide a parameter file on the designated drive.	●
The contents of the boot file are incorrect.	Check the boot setting.	Q CPU
There is no boot file at the drive designated by DIP switches although the BOOT DIP switch is set ON.	Check and correct the setting of the parameter enabled drive switch. Provide a parameter file on the designated drive.	QnA CPU
1.) A memory card was removed without switching the memory card in/out switch OFF. 2.)The card insert switch is turned ON although a memory card is not actually installed.	Only remove the memory card after switching the memory card in/out switch OFF. Turn the card insert switch OFF when no card is installed.	●
1.) The memory card is not formatted. 2.) Memory card format status is incorrect.	1.) Format the memory card. 2.) Reformat the memory card.	
A memory card not intended for the Q/QnA CPU was inserted.	Check the memory card.	
Automatic write to standard ROM was performed on a System A CPU that is incompatible with that function. (Automatic write from memory card to standard ROM is selected in the boot file and the parameter enable drive was set to the memory card.)	1.) Execute automatic write to standard ROM only on a CPU which supports this function. 2.) Write parameters and programs to standard ROM using the programming software. 3.) Switch off the automatic writing in standard ROM and perform boot operation from the memory card.	Q CPU (Version B or later)
The file designated by the PC file settings in the parameters cannot be found.	Read the individual error information on the display of a programming terminal and ensure that the drive and file destination correspond to the numerical values there. Create the designated file.	●
The Ethernet parameter that was added for QnA CPU, with the function version „B“, has been set to QnA CPU without the function version „B“.	Change to QnA CPU with the function version „B“. Delete the Ethernet parameter.	QnA CPU
During boot operation or automatic write the program memory capacity of the standard ROM has been exceeded.	1.) Check and correct the parameters (boot settings). 2.) Delete unnecessary files in the program memory. 3.) Choose „Clear program memory“ in the parameter so that boot is started after the program memory is cleared.	Q CPU (Version B or later)
The file designated by the parameter PC RAS settings fault history area was not created.	Read the individual error information on the display of a programming terminal and ensure that the drive and file destination correspond to the numerical values there. Check the remaining free memory on the memory card.	●
The file designated by the sequence program cannot be found.	Read the individual error information on the display of a programming terminal and ensure that the program corresponds to the numerical values there. Create the specified file.	●
The sequence program cannot designate the file type (comment file, etc.).	Read the individual error information on the display of a programming terminal and ensure that the program corresponds to the numerical values there.	
The SFC program file is one that cannot be designated by the sequence program.		
TNo data was written to the file designated by the sequence program.	Read the individual error information on the display of a programming terminal and ensure that the program corresponds to the numerical values there. Ensure that the designated file is not write protected.	

# Table of Error Codes; QnA CPUs and System Q

## Error codes 2500 to 3013

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
2500	CAN'T EXE. PRG.	File name	—	OFF	Flashes	Stop	At power ON/At reset
2501							
2502							
2503							
2504							
3000	PARAMETER ERROR	File name	Parameter number	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
3001							
3002							
3003		File name/Drive name					When an END instruction is executed.
3004		File name					At power ON/At reset/ STOP → RUN
3009	PARAMETER ERROR	File name/Drive name	Parameter number	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
3010							
3012							
3013							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.



## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
There is a program file that uses a device that exceeds the device allocation range designated by the parameter device settings.	Read the individual error information on the display of a programming terminal and ensure that the parameter device settings and the program file device allocation correspond to the numerical values there (file name).	●
There are multiple program files although „none“ is set in the parameter program settings.	Edit the parameter program settings to „yes“. Delete unneeded programs.	
The program file is not a QnA CPU program file. Alternatively, the file contents are not those of a sequence program.	Check whether the file format is *.QPG and whether the file contents are intended for a sequence program.	
There are no program files at all.	Check the program configuration.	
Two or more SFC normal programs or control programs are designated.	Check parameters and program configuration.	
In a multi-CPU system, an intelligent function module under control of another CPU is specified in the interrupt pointer settings.	1.) Specify the head I/O number of a module which is not under control of another CPU. 2.) Delete the interrupt pointer setting in the parameters.	Q CPU (Ver. B or later)
The parameter settings for timer time limit setting, the RUN-PAUSE contact, the common pointer number, the general data processing, number of vacant slots, or system interrupt settings exceed the relevant CPU range.	1.) Read the detailed error information on the display of the programming terminal, check the parameter items corresponding to the numerical values (parameter numbers) there, and correct if necessary. 2.) If the error is still generated, this hints to a memory error either in the internal CPU RAM or on the memory card. Please contact your nearest MITSUBISHI service.	● Rem
Parameter settings were destroyed.		●
When „use the following files“ is selected for the file registers on the PLC system setting screen under the parameter, the file specified does not exist, though the file register size has been set.		
The automatic refresh range of the multi-CPU system exceeds the file register capacity.	Use a file register area for which automatic refresh is possible.	Q CPU (Ver. B or later)
The number of devices set at the parameter device settings exceeds the relevant CPU range.	1.) Read the detailed error information on the display of the programming terminal, check the parameter items corresponding to the numerical values (parameter numbers) there, and correct if necessary. 2.) If the error is still generated, this hints to a memory error either in the internal CPU RAM or on the memory card. Please contact your nearest MITSUBISHI service.	●
The parameter file is not applicable for the QnA CPU. Alternatively, the contents of the file are not parameters.	Check whether the file format is *.QPA and whether the file actually contains parameters.	
In a multi-CPU system one module is allocated to more than one CPU.	A module can be controlled by one CPU only. Change the settings off all CPUs in the multi-CPU system.	Q CPU (Version B or later)
The set number of CPU modules differs from the actual number of CPU modules.	Match the settings with the actual system configuration.	
The parameters for the multi-CPU system set in the individual CPU modules are different to the parameters in CPU 1.	Match the settings for the individual CPU modules with the settings in CPU 1.	
Incorrect settings for automatic refresh in a multi-CPU system: 1.) When a bit device is specified as a refresh device, a number other than a multiple of 16 or 0 is specified for the refresh-starting device. 2.) The specified device is incorrect. 3.) The number of send points is an odd number.	1.) Specify a multiple of 16 or 0 as refresh starting device for bit operands. 2.) Specify the correct device.. 3.) Specify an even number of devices.	

# Table of Error Codes; QnA CPUs and System Q

## Error codes 3100 to 3105

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
3100							
3101							
3102	LINK PARA. ERROR	File name	Parameter number	OFF	OFF	Stop	At power ON/At reset/ STOP → RUN
3103							
3104							
3105							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
The MELSECNET/H module with the parameter set head I/O number is controlled by another CPU of the multi-CPU system.	Delete the network parameter for that MELSECNET/H module, and specify the head I/O number of the correct module.	Q CPU (Version B or later)
The network parameter of a normal station of the MELSECNET/H were written to the control station or vice versa.	Reset the CPU.	
1.) The number of actually installed modules is different from that designated in the number of modules setting parameter of MELSECNET/H 2.) The head I/O number of actually installed modules is different from that designated in the network setting parameter of MELSECNET/H 3.) Some of the data in the parameter cannot be handled. 4.) The station type of MELSECNET/H has been changed, while the power is on. (A change from RESET to RUN is required to change the station type).	1.) Write after correcting the network parameters. 2.) If the error is still generated, please contact your nearest MITSUBISHI service.	Q CPU
The network parameter were not written although the QnA CPU is the control station or the master station respectively	1.) Write after correcting the network parameters. 2.) If the error is still generated, please contact your nearest MITSUBISHI service.	QnA CPU
1.) The inter-PLC network parameter setting has been made for a MELSECNET/H module with station number 0. 2.) The remote master parameter setting has been made for a MELSECNET/H module with a station number other than 0.	Change the type of station or the station number.	Q CPU (Version B or later)
The network no. specified by a parameter is different from that of the actually mounted network. The head I/O number specified by a parameter is different from that of the actually mounted I/O unit The network class specified by a parameter is different from that of the actually mounted network. The network refresh parameter of the MELSECNET/10(H) is out of the specified area.	Match the data specified by the parameters with those of the actually mounted network and units.	●
An error was discovered when the network parameter check was made at the network module.	1.) Write after correcting the network parameters. 2.) If the error is still generated, please contact your nearest MITSUBISHI service.	●
The ETHERNET module with the parameter set head I/O number is controlled by another CPU of the multi-CPU system.	Delete the network parameter for that ETHERNET module, and specify the head I/O number of the correct module.	Q CPU (Ver. B or later)
Though the number of units for the Ethernet unit quantity set parameter is set at one or more, no module is installed. The head I/O number for the Ethernet set parameters is different from that of the actually mounted I/O unit.	1.) Write after correcting the network parameters. 2.) If the error is still generated, please contact your nearest MITSUBISHI service.	● Rem
AJ71QE71 does not exist in the position of I/O number set by the parameter. The I/O number designation is overlapping. Numbers of the parameter and loaded AJ71QE71 are different. Ethernet (parameter + dedicated instruction) is set to more than 5.		QnA CPU
Ethernet and MELSECNET/10 use the same network number. Network number, station number and group number set by the parameter is out of range. The I/O number is out of range of the CPU used.		● Rem
The CC-Link module with the parameter set head I/O number is controlled by another CPU of the multi-CPU system.	Delete the network parameter for that CC-Link module, and specify the head I/O number of the correct module.	Q CPU (Ver. B or later)
Though the number of units for the CC-Link unit quantity set parameters is set at one or more, no module is installed. The head I/O number for the common parameters is different from that of the actually mounted I/O unit. The station class for the CC-Link unit quantity set parameters is different from that of the actually mounted station.	1.) Write after correcting the network parameters. 2.) If the error is still generated, please contact your nearest MITSUBISHI service.	● Rem
The contents of the parameter peculiar to Ethernet are incorrect.	Write after correcting the network parameters.	QnA CPU

# Table of Error Codes; QnA CPUs and System Q

## Error codes 3106 to 4004

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
3106	LINK PARA. ERROR	File name/Drive name	Parameter number	OFF	Flashes	Stop	When an END instruction is executed.
3107		File name					At power ON/At reset/ STOP → RUN
		File name					
3200	SFC PARA. ERROR	File name	Parameter number	OFF	Flashes	Stop	STOP → RUN
3201							
3202							
3203							
3300	SP. PARA. ERROR	File name	Parameter number	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
3301		File name	Parameter number				When an END instruction is executed.
3302		File name	Parameter number				At power ON/At reset/ STOP → RUN
3303		File name/Drive name	Parameter number				At power ON/At reset/ STOP → RUN
3400	REMOTE PASS. ERROR	—	—	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
3401							
4000	INSTRCT CODE. ERR.	Program error location	—	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
4001							
4002							
4003							
4004							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
The CC-Link refresh range exceeds the file register capacity.	Use a file register area for which refresh is possible.	Q CPU (Ver. B or later)
The network refresh parameter for CC-Link is out of range.	Check the parameter setting.	Q CPU/Rem
The contents of the CC-Link parameter are incorrect.	Check the parameter setting.	●
The parameter contents are incorrect.	Write after correcting the network parameters.	●
The contents of the SFC block attributes information are incorrect.		
The number of step relays designated by the parameters is less than the number used by the program.		
The execution type set for an SFC program in the parameters is not the scan execution type.		
The head I/O number in the intelligent function module parameter set with GX Configurator differs from the actual I/O number.	Check the parameter setting.	Q CPU/Rem
The refresh setting for the intelligent function module exceeds the file register capacity.	Use a file register area for which refresh is possible in the whole range.	Q CPU (Version B or later)
The intelligent function module's refresh parameter setting is outside the available range.	Check the parameter setting.	
The intelligent function module's refresh parameter setting is incorrect.		● Rem
Automatic refresh settings or similar parameter setting was made for an intelligent function module under control by another CPU of the multi-CPU system.	Change the settings to a module which is under control of the CPU where the instruction is executed.	●
The head I/O number of the target module in the remote password file is set to other than 0 <sub>H</sub> or 0FF0 <sub>H</sub> .	Change the head address of the accessed module to 0 <sub>H</sub> or 0FF0 <sub>H</sub> .	Q CPU (Version B or later)
The slot specified as the head I/O number in the remote password file is incorrect due to one of the following reasons: – The module is not installed. – The module is incompatible with the System Q. – The module is other than a QJ71C24(-R2) or an ETHERNET module of the System Q. – A function version A module (QJ71C24(-R2) or an ETHERNET module of the System Q) is installed.	Install a QJ71C24(-R2), function version B or a System Q ETHERNET module with function version B in the slot specified by the head I/O number.	
A QJ71C24(-R2) with function version B or a System Q ETHERNET module (function version B) under control of another CPU is specified in a multi-CPU system.	Specify the correct module. Delete the remote password setting.	
The program contains an instruction code that cannot be decoded. An unusable instruction is included in the program.	Read the common error information on the display of the programming terminal and check the error step corresponding to its numerical value (program error location).	●
The program contains a dedicated instruction for an SFC program although it is none.		
The instruction name is incorrect.		● Rem
The instruction designates an incorrect number of devices.		
The instruction designates a device that cannot be used.		●

# Table of Error Codes; QnA CPUs and System Q

## Error codes 4010 to 4213

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
4010	MISSING END INS.	Program error location	—	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
4020	CAN'T SET (P)	Program error location	—	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
4021							
4030	CAN'T SET (I)	Program error location	—	OFF	Flashes	Stop	At power ON/At reset/ STOP → RUN
4100	OPERATION ERROR	Program error location	—	OFF/ON	Flashes/ON	Stop/ Continue <sup>2</sup>	When an instruction is executed.
4101							
4102		Program	Program error location				
		Program error location	—				
4103		Program error location	—				
4107		Program	Program error location				
		Program error location	—				
4108		Program error location	—				
4200	FOR NEXT ERROR	Program error location	—	OFF	Flashes	STOPP	When an instruction is executed.
4201							
4202							
4203							
4210	CAN'T EXECUTE ( P )	Program error location	—	OFF	Flashes	STOPP	When an instruction is executed.
4211							
4212							
4213							

<sup>1</sup> Specifications in parentheses ( ) indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).

## Table of Error Codes; QnA CPUs and System Q

	Error description and cause	Remedy	Valid for:
	The program contains no END-(FEND-) instruction.	Read the common error information on the display of the programming terminal and check the files corresponding to the numerical values there (program error location).	●
	The total number of internal file pointers used by the program exceeds the number of internal file pointers set by the parameters.		●
	The common pointer numbers used by individual files overlap.		
	The allocation pointer numbers used by individual files overlap.		
	The instruction cannot process the contained data.	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	The designated device numbers for data processed by the instruction exceed the usable device range. Alternatively, the stored data or constants for the devices designated by the instruction exceed the usable range.		
	In a multi-CPU system, the link direct device (J[\G[]]) was specified for a network module under control of another CPU.	Delete the link direct device which caused the error. Specify a module which is under control of the CPU where the instruction is executed.	Q CPU (Version B or later)
	The network number or station number designated by a dedicated network instruction is incorrect. The link direct device is not set correctly.	Read the common error information on the display of the programming terminal and check the indicated program step.	● Rem
	The configuration of the dedicated PID instruction is incorrect.		●
	More than 32 multi-CPU dedicated instructions were executed by one CPU.	Use the bit device which indicates the completion of an instruction as interlock to prevent one CPU from executing more than 32 multi-CPU dedicated instructions.	Q CPU (Version B or later)
	The CC-Link instruction is executed more than 64 times.	Set the numbers of execution to the CC-Link instruction to 64 or less.	QnA CPU
	The CC-Link parameter are not set when the CC-Link instruction is executed.	Execute the CC-Link instruction after setting the CC-Link parameter.	
	No NEXT instruction was executed following the execution of a FOR instruction. Alternatively, there are fewer NEXT instructions than FOR instructions	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	A NEXT instruction is executed without a prior FOR instruction. Alternatively, there are more NEXT instruction than FOR instructions.		●
	More than 16 nesting levels are programmed.	Reduce the number of nesting levels to 16 max.	
	A BREAK instruction is executed without a prior FOR instruction.	Read the common error information on the display of the programming terminal and check the indicated program step.	
	The CALL instruction is executed but there is no subroutine at the specified pointer.	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	The executed subroutine contains no RET instruction.		
	The RET instruction is programmed prior to the FEND instruction.		
	More than 16 nesting levels are programmed.	Reduce the number of nesting levels to 16 max.	

# Table of Error Codes; QnA CPUs and System Q

## Error codes 4220 to 4611

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
4220	CAN'T EXECUTE ( I )	Program error location	—	OFF	Flashes	STOPP	When an instruction is executed.
4221							
4223							
4230	INST. FORMAT ERR	Program error location	—	OFF	Flashes	STOPP	When an instruction is executed.
4231							
4235							
4300	EXTEND INST. ERR.	Program error location	—	OFF/ON	Flashes/ON	STOPP/ Continue <sup>2</sup>	When an instruction is executed.
4301							
4400	SFCP. CODE ERROR	Program error location	—	OFF	Flashes	STOPP	STOP → RUN
4410	CAN'T SET ( BL )	Program error location	—	OFF	Flashes	STOPP	STOP → RUN
4411							
4420	CAN'T SET ( S )	Program error location	—	OFF	Flashes	STOPP	STOP → RUN
4421							
4422							
4500	SFCP. FORMAT ERR.	Program error location	—	OFF	Flashes	STOPP	STOP → RUN
4501							
4502							
4503							
4504							
4600	SFCP. OPE. ERROR	Program error location	—	OFF/ON	Flashes/ON	STOPP/ Continue <sup>2</sup>	When an instruction is executed.
4601							
4602							
4610	SFCP. EXE. ERROR	Program error location	—	ON	ON	Continue	STOP → RUN
4611							

<sup>1</sup> Specifications in parentheses ( ) indicate the special register numbers where individual information is stored.

<sup>2</sup> The CPU operation status on occurrence of an error can be set via parameters (LED display will change accordingly).



## Table of Error Codes; QnA CPUs and System Q

	Error description and cause	Remedy	Valid for:
	An interrupt was generated but no corresponding interrupt pointer was found.	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	The executed subroutine contains no IRET instruction.		
	The IRET instruction is programmed prior to the FEND instruction.		
	The CHKEND instruction is executed prior to the CHKCIR instruction. The numbers of CHK and CHKEND instructions are equal.	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	The IX and IXEND instructions are not programmed in combination. The numbers of IX and IXEND instructions are equal.		
	The configuration of the check conditions for the CHK instruction is incorrect. Alternatively, a CHK instruction was used in a low-speed program.		
	The designation of a MELSECNET/MINI-S3 master modul control instruction is wrong.	Read the common error information on the display of the programming terminal and check the indicated program step.	QnA CPU
	The designation of an AD57/AD58 control instruction is wrong.		
	There is no SFCP or SFCPEND instruction in an SFC program.		●
	The block number designated by the SFC program exceeds the maximum setting value.	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	Block number designations in the SFC program overlap.		
	The step number designated in the SFC program exceeds step 511.		
	The total number of steps in all SFC programs exceeds the maximum value.	Reduce the number of steps.	●
	Step number designations in SFC program overlap.		●
	The numbers of BLOCK and BEND instructions in the SFC program do not equal.	Read the common error information on the display of the programming terminal and check the indicated program step.	
	The configuration of the STEP* to TRAN* to TSET to SEND instructions in the SFC program is incorrect.		
	There was no STEPI* instruction in the SFC program block.		
	The step designated by the TSET instruction in the SFC program does not exist.	Read the common error information on the display of the programming terminal and check the indicated program step.	
	The step designated by the TAND instruction in the SFC program does not exist.		
	The SFC program contains data that cannot be processed.	Read the common error information on the display of the programming terminal and check the indicated program step. The program starts automatically at initialization.	●
	The SFC program exceeds the relevant device range.		
	The START instruction in the SFC program is preceded by an END instruction.		
	The active step information at a resumtive start of the SFC program in incorrect.	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	The RESET/L.CLR switch was reset during RUN when resumtive start was designated for the SFC program.		

# Table of Error Codes; QnA CPUs and System Q

## Error codes 4620 to 6222

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
4620	BLOCK EXE. ERROR	Program error location	—	OFF	Flashes	STOPP	When an instruction is executed
4621							
4630	STEP EXE. ERROR	Program error location	—	OFF	Flashes	STOPP	When an instruction is executed
4631							
4632							
4633							
5000	WDT ERROR	Time (Set value)	Time (value actually measured)	OFF	Flashes	STOPP	Always
5001							
5010	PRG. TIME OVER	Time (Set value)	Time (value actually measured)	ON	ON	Continue	Always
5011							
6000	PRG. VERIFY ERR. <sup>2</sup>	File name	—	OFF	Flashes	Stop	Always
6010	MODF VERIFY ERR. <sup>2</sup>	—	—	ON	ON	Continue	Always
6100	TRK. MEMORY ERR. <sup>3</sup>	—	—	ON	ON	Continue	At power ON/At reset/ STOP → RUN
6101							When an END instruction is executed.
6200	CONTROL EXE. <sup>4</sup>	Cause of switch	—	ON	OFF	Continue	Always
6210	CONTROL WAIT. <sup>2</sup>	Cause of switch	—	ON	OFF	Continue	Always
6220	CAN'T EXE CHANGE <sup>4</sup>	Cause of switch	—	ON	ON	Continue	Always
6221							
6222							

<sup>1</sup> Specifications in parentheses () indicate the special register numbers where individual information is stored.

<sup>2</sup> This error can only be detected in the standby system of a redundant system.

<sup>3</sup> This error can only be detected in redundant systems. Can be detected either in the control system or the standby system

<sup>4</sup> This error can be detected in the control system of a redundant system.

## Table of Error Codes; QnA CPUs and System Q

	Error description and cause	Remedy	Valid for:
	Startup was executed at a block in the SFC program that was already started up.	Read the common error information on the display of the programming terminal and check the indicated program step.	●
	Startup was attempted at a block that does not exist in the SFC program.		
	Startup was executed at a block in the SFC program that was already started up.		●
	Startup was attempted at a block that does not exist in the SFC program.		
	There were too many simultaneously active steps in blocks that can be designated by the SFC program.		
	There were too many simultaneously active steps in all blocks that can be designated.		
	The program scan time for initial execution type program exceeds the initial execution WDT time setting designated by the parameter PC RAS setting.	Read the individual error information on the display of the programming terminal and check the numerical value (time) there, and shorten scan time if necessary.	●
	The program scan time exceeds the WDT setting value designated by the parameter PC RAS setting.		
	1.) The scan time of the program exceeds the constant scan setting time specified in the PC RAS setting parameter. 2.) The run time of a low-speed execution type program that is set in the parameter PC RAS setting exceeds the margin time of constant scan.	1.) Check the constant scan setting time. 2.) Check and change the constant scan time and the run time for the low-speed execution type program.	●
	The low-speed scan type program scan time exceeds the low-speed execution WDT time setting designated in the parameter PC RAS settings.	Read the individual error information on the display of the programming terminal and check the numerical value (time) there, and shorten scan time if necessary.	
	The control system and the standby system in the redundant system do not have the same programs and parameters.	Synchronize the programs and parameters of the control system and the standby system.	Q4AR CPU
	The operational statuses of the control system and the standby system in the redundant system are not the same.	Run both the control system and the standby system with the same operational statuses.	
	During initialisation of the PLC a CPU module tracking memory error was detected.	The CPU module has a hardware fault. Please contact your nearest MITSUBISHI service. To replace the module, replace the CPU of the standby system first, then the control system CPU.	
	The CPU module detected an error during the handshake for tracking.	Check the condition of the other stations.	
	The standby system of a redundant system is switched as the control system.	Check the condition of the control system.	
	The control system of a redundant system is switched as the standby system.		
	The standby system of a redundant system could not be switched from the control system to the standby system because of an error status or other reason.	Check the condition of the standby system.	
	Switching is disabled because of a bus switching module error.	The switching module has a hardware fault. Please contact your nearest MITSUBISHI service.	
	Switching is disabled because a multiplexed master station of a remote I/O network was installed in the standby station during initialisation.	Check the remote I/O network setting.	

# Table of Error Codes; QnA CPUs and System Q

## Error codes 7000 to 10000

Error code (SD0) <sup>1</sup>	Error message	Common information (SD5 to 15) <sup>1</sup>	Individual information (SD13 to 20) <sup>1</sup>	LED Status		CPU Status	Diagnostic timing
				RUN	ERROR		
7000	MULT CPU DOWN	Unit/Module no.	—	OFF	Flashes	Stop	Always
7002							At power ON/At reset
7003							At power ON/At reset
7010	MULTI EXE. ERROR	Unit/Module no.	—	OFF	Flashes	Stop	At power ON/At reset
7020	MULTI CPU ERROR		—	ON	ON	Continue	Always
9000	F**** <sup>2</sup>	Program error location	Annunciator number	ON	OFF	Continue	When an instruction is executed
				USER LED ON			
9010	<CHK> ERR ***_*** <sup>3</sup>	Program error location	Failure No.	ON	OFF	Continue	When an instruction is executed
				USER LED ON			
9020	BOOT OK	—	—	OFF	Flashes	Stop	At power ON/At reset
10000	CONT.UNIT ERROR	—	—	—	—	—	—

<sup>1</sup> Specifications in parentheses ( ) indicate the special register numbers where individual information is stored.

<sup>2</sup> \*\*\*\* indicates detected annunciator number.

<sup>3</sup> \*\*\* indicates detected contact and coil number.

## Table of Error Codes; QnA CPUs and System Q

Error description and cause	Remedy	Valid for:
<p>1.) An error occurred in a CPU module where the stop of all other CPUs of the multi-CPU system on occurrence of an error has been set.                      einem Fehler dieser CPU Fehler gestoppt werden, ist ein Fehler aufgetreten.</p> <p>2.) A CPU of function version A is installed in a multi-CPU system.</p>	<p>1.) Continue the diagnosis at the CPU module that caused the stop of the multi-CPU system.</p> <p>2.) Only CPU modules of function version B can be used in a multi-CPU system.</p>	Q CPU (Version B or later)
<p>CPU 1 of a multi-CPU system was stopped due to an error during power-on. Therefore the other CPUs cannot start. (This error code will occur on CPU 2, 3 and 4.)</p>	<p>Continue the diagnosis at the CPU.</p>	
<p>1.) There is no response from the target CPU in a multi-CPU system at initial communication stage.</p> <p>2.) A CPU of function version A is installed in a multi-CPU system.</p>	<p>1.) Reset the CPU. If the same error is displayed again, a hardware fault of any CPU is likely. Please contact your nearest MITSUBISHI service.</p> <p>2.) Only CPU modules of function version B can be used in a multi-CPU system.</p>	
<p>There is no response from the target CPU in a multi-CPU system at initial communication stage.</p>	<p>Reset the CPU. If the same error is displayed again, a hardware fault of any CPU is likely. Please contact your nearest MITSUBISHI service.</p>	
<p>1.) One of the CPUs in a multi-CPU system has a hardware fault.</p> <p>2.) A CPU of function version A is installed in a multi-CPU system. (This error is detected at the CPUs of function version B.)</p> <p>3.) CPU2, 3 or 4 has been reset during power-on. (This error will occur at the CPU which was reset only.)</p>	<p>1.) Read the individual error information and replace the faulty CPU.</p> <p>2.) Only CPU modules of function version B can be used in a multi-CPU system. Replace the CPU (Ver. A) with one of function version B.</p> <p>3.) An individual reset at the CPUs 2 to 4 is not possible. Reset CPU 1 to reset the whole multi-CPU system.</p>	
<p>An error occurred in a CPU module where the stop of all other CPUs of the multi-CPU system on occurrence of an error has not been set. (This error is detected in the other CPUs but not in the CPU module where the fault has occurred).</p>	<p>Continue the diagnosis at the CPU module where the fault has occurred.</p>	
<p>An annunciator F was set ON.</p>	<p>Read the individual error information on the display of the programming terminal and check the program corresponding to the numerical value (annunciator number; error number).</p>	●
<p>Error detected by the CHK instruction.</p>		
<p>Automatic storage of data onto standard ROM was completed normally. (The BOOT LED also flickers.)</p>	<p>Set the parameter enable drive to standard ROM, switch power on again, and perform boot operation from standard-ROM zu laden.</p>	Q CPU (Version B or later)
<p>In a multi-CPU system, an error occurred in a CPU other than a PLC CPU (e.g. a motion controller).</p>	<p>Continue the diagnosis at the CPU module where the fault has occurred.</p>	



## Table of error codes; A series (except AnA and AnAS)

### 13.3 Table of error codes; A series (except AnA and AnAS)

The following table contains an overview of all possible errors and the corresponding error messages, possible causes, and remedial advice. The error codes are written into the special register D9008 and the corresponding step number in which the error occurred is written into the special registers D9010 and D9011. This table only includes the error messages of the AnN, AnU, AnS, A3M, and A2C CPUs.

Error message	Error code in D9008	CPU Status	Error description and cause	Remedy
INSTRCT. CODE ERR (Checked at program execution)	10	Stop	Instruction code that cannot be decoded by CPU is included in program. An EPROM chip with faulty program was inserted. The memory contents have been changed. A PR or IRET instruction was programmed.	Read the error step by use of a programming terminal and correct the program at that step. Correct the EPROM program or replace EPROM chip.
PARAMETER ERROR (Checked at Power ON/Reset, Stop → RUN, and PAUSE → RUN)	11	Stop	Capacity larger than the memory capacity of CPU was set and then data were written to the CPU. Contents of parameters of CPU memory changed due to noise or improper loading of memory. RAM not loaded (into A1 or A1N CPUs).	Check and correct parameters, and write them to the CPU by use of a programming terminal. Check whether the RAM chip is installed properly to its socket.
MISSING END INS. (Checked after setting M9056 or M9057 or at Stop → RUN, and PAUSE → RUN)	12	Stop	There is no END (FEND) instruction in the program. There is no END instruction in a subprogram set by parameters.	Add END instruction at program end.
CAN'T EXECUTE (P) (Checked at execution of one of the following instructions: CJ, SCJ, JMP, CALLP, FOR/NEXT and at Stop → RUN, and PAUSE → RUN)	13	Stop	There is no jump destination or multiple destinations specified by the CJ, SCJ, CALL, CALLP or JMP instruction. There is a CHG instruction but no subprogram. There is a RET instruction without a CALL instruction in the program. The CJ, SCJ, CALL, CALLP or JMP instruction with its jump destination beyond the END instruction. The number of FOR and the number of NEXT instructions do not equal. A JMP is given within a FOR to NEXT loop causing the processing to exit the loop. Processing exited subroutine by JMP instruction before execution of the RET instruction. Processing jumped into a step in a FOR to NEXT loop or into a subroutine by the JMP instruction. The STOP instruction is given in an interrupt program, a subroutine program or in a FOR to NEXT loop.	Read the error step by use of a programming terminal and correct the program at that step.

## Table of error codes; A series (except AnA and AnAS)

### Error codes 14 to 21

Error message	Error code in D9008	CPU Status	Error description and cause	Remedy
<p>CHK FORMAT ERR</p> <p>(Checked at Power ON/Reset, Stop → RUN, and PAUSE → RUN)</p>	14	Stop	<p>Instructions (NOP incl.) except LDX, LDIX, ANDX, and ANIX are included in the CHK instruction circuit block.</p> <p>Multiple CHK instructions are given.</p> <p>The number of contact points in the CHK instruction circuit block exceeds 150.</p> <p>The number of an input instruction X in the CHK instruction circuit block exceeds the maximum value.</p> <p>Prior to the CHK instruction circuit block there is no CJP instruction with input condition.</p> <p>The device number of D1 of the CHK D1 D2 instruction is different from that of the contact point before the CJP instruction.</p> <p>Pointer P254 is not given to the head of the CHK instruction circuit block.</p>	<p>Check the program in the CHK instruction circuit block. Correct the problem using a programming terminal (eg. add a jump instruction) and perform operation again.</p>
<p>CAN'T EXECUTE (I)</p> <p>(Checked at execution of interrupt)</p>	15	Stop	<p>Although the interrupt module is used there is no number of interrupt pointer I which corresponds to that module in the program or there are multiple numbers.</p> <p>There is no IRET instruction in the program.</p> <p>The IRET instruction is programmed in another program part than the interrupt program.</p>	<p>Check for the presence of the interrupt program which corresponds to the interrupt unit and reduce the same numbers of I.</p> <p>Check if there is an IRET instruction in the interrupt program and enter the IRET instruction.</p> <p>Check whether there is an IRET instruction in another program than the interrupt program and delete the IRET instruction.</p>
<p>CASSETTE ERROR</p> <p>(Checked at Power ON/Reset)</p> <p>Memory cassette cannot be accessed</p>	16	Stop	<p>The memory cassette is not loaded.</p>	<p>Turn off the power, insert the memory cassette and turn the power on again.</p>
<p>ROM-ERROR</p> <p>(Checked at Power ON/Reset)</p>	17	Stop	<p>Parameters and sequence program are not stored correctly in the inserted EPROM.</p> <p>The EPROM is defective.</p>	<p>Rewrite the program and parameters to the EPROM.</p> <p>Replace the defective EPROM.</p>
<p>MEMORY PROTECT ERROR</p> <p>(Checked at Power ON/Reset)</p>	18	Stop	<p>The EPROM was write protected when the CPU attempted to access the program stored in the EPROM (DIP switch in ON position).</p>	<p>Set the write protection switch OFF.</p>
<p>RAM ERROR</p> <p>(Checked at Power ON/Reset and after setting M9084 during Stop)</p>	20	Stop	<p>The CPU has checked if write and read operations can be performed properly to the data memory area of CPU, and as a result, either or both has not been performed.</p>	<p>Since this is a CPU hardware error, consult Mitsubishi representative.</p>
<p>OPE. CIRCUIT ERR</p> <p>(Checked at Power ON/Reset)</p>	21	Stop	<p>The operation circuit, which performs the sequence processing in the CPU, does not operate properly.</p>	<p>Since this is a CPU hardware error, consult Mitsubishi representative.</p>



## Table of error codes; A series (except AnA and AnAS)

### Error codes 22 to 41

Error message	Error code in D9008	CPU Status	Error description and cause	Remedy
WDT ERROR (1) (Checked after execution of an END instruction)	22	Stop	Scan time exceeds watch dog error monitor time. 1. Scan time of user program has been exceeded. 2. Scan time has lengthened due to instantaneous power failure which occurred during scan.	1. Calculate and check the scan time of user program and reduce the scan time. 2. Monitor the contents of special register D9005 by use of peripheral equipment. When the contents are other than 0, line voltage is insufficient. Therefore, check the power and reduce the fluctuation of voltage.
SUB-CPU ERROR	23	Stop	Sub-CPU is locked-up or defective.	Since this is a CPU hardware error, consult Mitsubishi representative.
END NOT EXECUTE (Checked during execution of an END instruction)	24	Stop	The END instruction has changed to another instruction code for some reason.	Perform reset and run. If the same error is displayed again, it is the CPU hardware error. Therefore, consult nearby service center, representative, or branch.
WDT ERROR (2) (Checked continuously)	25	Stop	1. The CPU is executing an endless loop. 2. Main-CPU is malfunctioning.	1. Since the program is in an endless loop due to the JMP and CJ instructions, check the program. 2. Since this is a CPU hardware error, consult Mitsubishi representative.
MAIN CPU DOWN (1) (Checked continuously)	26	Stop	Main-CPU is defective.	Since this is a CPU hardware error, consult Mitsubishi representative.
UNIT VERIFY ERR. (Checked continuously)	31	RUN (Stop)	I/O module data are different from those at power-on. The I/O module (including the special function module) is incorrectly loaded or has been removed, or a different unit has been loaded.	1. Among special registers D9116 to D9123, the bit corresponding to the module of verify error is „1“. Therefore, monitor the registers and check for the module with „1“ and make replacement. 2. When the present unit arrangement is OK, perform reset with the reset switch.
FUSE BREAK OFF (Checked continuously)	32	RUN (Stop)	A fuse is blown in an output module.	1. Check the fuse blown indicator LED of output module and change the fuse of module of which LED is on. 2. Among special registers D9100 to D9107, the bit corresponding to the unit of fuse break is „1“. Replace the fuse of a corresponding module.
CONTROL BUS ERR. (Checked during execution of a FROM / TO instruction)	40	Stop	The FROM and TO instructions cannot be executed. Error of control bus with special function module.	This error can be caused by a special function module, CPU module or base unit hardware, consult Mitsubishi representative.
SPUNIT DOWN (Checked during execution of a FROM / TO instruction)	41	Stop	When the FROM or TO instruction is executed, access has been made to the special function module but the answer is not given. The accessed special function module is defective.	Since this is an accessed special function module error, consult Mitsubishi representative.

## Table of error codes; A series (except AnA and AnAS)

### Error codes 42 to 70

Error message	Error code in D9008	CPU Status	Error description and cause	Remedy
LINK UNIT ERROR	42	Stop	AJ71(A)R22 or AJ71(A)P22 is loaded in the master station.	Remove the AJ71(A)R22 or AJ71(A)P22 from the master station.
I/O INT. ERROR	43	Stop	Although the interrupt module is not loaded, interruption has occurred.	This is a specific module hardware error, consult Mitsubishi representative.
SPUNIT LAY ERROR	44	Stop	<ol style="list-style-type: none"> <li>Three or more computer link modules are loaded with respect to one CPU module.</li> <li>Two or more modules of AJ71(A)P22, AJ71(A)R22, AJ71AP21 or AJ71AR21 are loaded.</li> <li>Two or more interrupt modules are loaded.</li> <li>A special function module is assigned in place of an I/O module, or vice versa, at I/O assignment of parameters on peripheral devices.</li> </ol>	<ol style="list-style-type: none"> <li>Reduce the computer link modules to two or less.</li> <li>Reduce the AJ71(A)P22, AJ71(A)R22, AJ71AP21 or AJ71AR21 to one or less.</li> <li>Reduce the interrupt module to one.</li> <li>Reset the I/O assignment of parameter setting by use of peripheral devices according to the actually loaded special function module.</li> </ol>
SP. UNIT ERROR (Checked during execution of a FROM / TO instruction)	46	Stop	Access (execution of FROM to TO instruction) has been made to a location where there is no special function module.	Read the error step by use of peripheral equipment, and check and correct the FROM or TO instruction at that step.
LINK PARA. ERROR	47	Continue	<ol style="list-style-type: none"> <li>If a data link CPU is used to set a master station (station number „00“): The link parameters written by the link CPU do not correspond to the link parameters read by the master station. Or else, link parameters are not written.</li> <li>The setting of the total number of slave stations is 0.</li> </ol>	<ol style="list-style-type: none"> <li>Write parameters again and check.</li> <li>Check setting of station numbers.</li> <li>When the error is displayed again, it is a hardware error. Therefore, consult your Mitsubishi representative.</li> </ol>
OPERATION ERROR (Checked during execution of an instruction)	50	Continue	<ol style="list-style-type: none"> <li>The result of BCD conversion has exceeded the specified range (9999 or 99999999).</li> <li>Operation impossible because specified device range has been exceeded.</li> <li>File registers used in program without capacity setting.</li> <li>Operation error occurred during execution of the RTOP, RFRP, LWTP or LRDP instruction.</li> </ol>	Read the error step using peripheral devices and check the program at the error step, and correct it.
MAIN CPU DOWN (2) (Error on interrupt)	60	Stop	<ol style="list-style-type: none"> <li>INT instruction processed in microcomputer program area.</li> <li>CPU malfunction due to noise.</li> <li>Hardware fault.</li> </ol>	<ol style="list-style-type: none"> <li>Remove INT.</li> <li>Eliminate noise.</li> <li>Consult Mitsubishi representative.</li> </ol>
BATTERY ERROR (Checked continuously)	70	RUN	<ol style="list-style-type: none"> <li>Battery voltage low.</li> <li>Battery not connected.</li> </ol>	<ol style="list-style-type: none"> <li>Replace battery.</li> <li>Connect battery if RAM memory or power failure compensation function is used.</li> </ol>

## 13.4 Table of error codes; AnA and AnAS CPUs

The following table contains an overview of all possible errors and the corresponding error messages, possible causes, and remedial advice. The error codes are written into the special register D9008, the detailed error code is written into the special register D9091, and the corresponding step number in which the error occurred is written into the special registers D9010 and D9011. This table only includes the error messages of the AnA and AnAS CPUs.

Error message	Error code in D9008	Detailed error code in D9091	Error description and cause	Remedy
INSTRCT CODE ERR (Checked at Stop → RUN or execution of instruction)	10	101	Instruction codes which the CPU cannot decode are included in the program.	1. Read the error step using a peripheral device and correct the program of the step. 2. Check the ROM if it contains instruction codes which cannot be decoded. If it does, replace it with a correct ROM.
		102	Index qualification is specified for a 32-bit constant.	Read the error step using a peripheral device and correct the program of the step.
		103	Device specified by an extended application instruction is not correct.	
		104	An extended application instruction has incorrect program structure.	
		105	An extended application instruction has incorrect command name.	
		106	Index qualification using Z or V is included in the program between LEDA/B IX and LEDA/B IXEND.	
		107	1. Index qualification is specified for the device numbers and set values in the OUT instruction of timers and counters. 2. Index qualification is specified at the label number of the pointer (P) provided to the head of destination of the CJ, SCJ, CALL, CALLP, JMP, LEDA/B FCALL and LEDA/B BREAK instructions or at the label number of the interrupt pointer (I) provided to the head of an interrupt program.	
		108	Errors other than 101 to 107 mentioned above.	

## Table of error codes; AnA and AnAS CPUs

### Error codes 11 to 13

Error message	Error code in D9008	Detailed error code in D9091	Error description and cause	Remedy
PARAMETER ERROR (Checked at Power ON/Reset, Stop → RUN, and PAUSE → RUN)	11	111	Capacity settings of the main and sub programs, microcomputer program, file register comments, status latch, sampling trace and extension file registers are not within the usable range of the CPU.	Read parameters in the CPU memory, check the contents, make necessary corrections and write them again to the memory.
		112	Total of the set capacity of the main and sub programs, file register comments, status latch, sampling trace and extension file registers exceeds capacity of the memory cassette.	
		113	Latch range set by parameters or setting of M, L or S is incorrect.	
		114	Sum check error	
		115	Either of settings of the remote RUN/PAUSE contact point by parameters, operation mode at occurrence of error, annunciator indication mode, or STOP → RUN indication mode is incorrect.	
		116	The MNET-MINI automatic refresh setting by parameters is incorrect.	
		117	Timer setting by parameters is incorrect.	
		118	Counter setting by parameters is incorrect.	
MISSING END INS. (Checked at Stop → RUN)	12	121	The END (FEND) instruction is not given in the main program.	Write the END instruction at the end of the main program.
		122	The END (FEND) instruction is not given in the sub program if the sub program is set by parameters.	Write the END instruction at the end of the sub program.
CAN'T EXECUTE (P) (Checked at execution of instruction)	13	131	The same device number is used at two or more steps for the pointers (P) and interrupt pointers (I) used as labels to be specified at the head of jump destination.	Eliminate the same pointer numbers provided at the head of jump destination.
		132	Label of the pointer (P) specified in the CJ, SCJ, CALL, CALLP, JMP, LEDA/B FCALL oder LEDA/B BREAK instruction is not provided before the END instruction.	Read the error step using a peripheral device, check contents and insert a jump destination pointer (P).

## Table of error codes; AnA and AnAS CPUs

### Error codes 13 to 14

Error message	Error code in D9008	Detailed error code in D9091	Error description and cause	Remedy
CAN'T EXECUTE (P) (Checked at execution of instruction)	13	133	1. The RET instruction was included in the program and executed though the CALL instruction was not given. 2. The NEXT and LEDA/B BREAK instructions were included in the program and executed though the FOR instruction was not given. 3. Nesting level of the CALL, CALLP and FOR instructions is 6 levels or deeper, and the 6th level was executed. 4. There is no RET or NEXT instruction at execution of the CALL or FOR instruction.	1. Read the error step using a peripheral device, check contents and correct program of the step. 2. Reduce the number of nesting levels of the CALL, CALLP and FOR instructions to 5 or less.
		134	The CHG instruction was included in the program and executed though no sub program was provided.	Read the error step using a peripheral device and delete the CHG instruction circuit block.
		135	1. LEDA/B IX and LEDA IXEND instructions are not paired. 2. There are 33 or more sets of LEDA/B IX and LEDA IXEND instructions.	1. Read the error step using a peripheral device, check contents and correct program of the step. 2. Reduce the number of sets of LEDA/B IX and LEDA IXEND instructions to 32 or less.
CHK FORMAT ERR (Checked at Stop → RUN, and PAUSE → RUN)	14	141	Instructions (including NOP) other than LDX, LDIX, ANDX and ANIX are included in the CHK instruction circuit block.	Check the program of the CHK instruction and correct it referring to contents of detailed error codes.
		142	Multiple CHK instructions are given.	
		143	The number of contact points in the CHK instruction circuit block exceeds 150.	
		144	The LEDA/CHK instructions are not paired with the LEDA/CHKEND instructions, or 2 or more pairs of them are given.	
		145	There is no CJ instruction with input condition programmed prior to the CHK instruction block.	
		146	Device number of D1 in the CHK D1 D2 instruction is different from that of the contact point before the CJ instruction.	
		147	Index qualification is used in the check pattern circuit.	
148	1. Multiple check pattern circuits of the LEDA/CHK - LEDA/CHKEND instructions are given. 2. There are 7 or more check condition circuits in the LEDA/CHK - LEDA/CHKEND instructions. 3. The check condition circuits in the LEDA/CHK - LEDA/CHKEND instructions are written without using X and Y contact instructions or compare instructions. 4. The check pattern circuits of the LEDA/CHK - LEDA/CHKEND instructions are written with 257 or more steps.			

## Table of error codes; AnA and AnAS CPUs

### Error codes 15 to 24

Error message	Error code in D9008	Detailed error code in D9091	Error description and cause	Remedy
CAN'T EXECUTE (1) (Checked at execution of interrupt)	15	151	The IRET instruction was given outside of the interrupt program and was executed.	Read the error step using a peripheral device and delete the IRET instruction.
		152	There is no IRET instruction in the interrupt program.	Check the interrupt program if the IRET instruction is given in it. Write the IRET instruction if it is not given.
		153	Though an interrupt module is used, no interrupt pointer (I) which corresponds to the module is given in the program. Upon occurrence of error, the problem pointer (I) number is stored at D9011.	Monitor special register D9011 using a peripheral device, and check if the interrupt program that corresponds to the stored data is provided or if two or more interrupt pointers (I) of the same number are given. Make necessary corrections.
CASSETTE ERROR	16		Memory cassette is not loaded.	Turn off the PC power and load the memory cassette.
RAM ERROR (Checked at Power ON)	20	201	The sequence program storage RAM in the CPU module caused an error.	Since this is a CPU hardware error, consult your Mitsubishi representative.
		202	The work area RAM in the CPU module caused an error.	
		203	The device memory in the CPU module caused an error.	
		204	The address RAM in the CPU module caused an error.	
OPE. CIRCUIT ERR (Checked at Power ON/Reset)	21	211	The operation circuit for index qualification in the CPU does not work correctly.	Since this is a CPU hardware error, consult your Mitsubishi representative.
		212	Hardware (logic) in the CPU does not operate correctly.	
		213	The operation circuit for sequential processing in the CPU does not operate correctly.	
WDT ERROR (Checked during execution of an END instruction)	22	-	Scan time is longer than the WDT time. 1. Scan time of the user's program has been extended due to certain conditions. 2. Scan time has been extended due to momentary power failure occurred during scanning.	1. Check the scan time of the user program and shorten it using the CJ instructions. 2. Monitor contents of special register D9005 using a peripheral device. If the contents are other than 0, power supply voltage may not be stable. Check power supply and reduce variation in voltage.
END NOT EXECUTE (Checked during execution of an END instruction)	24	241	Whole program of specified program capacity was executed without executing the END instructions. 1. When the END instruction was to be executed, the instruction was read as other instruction code due to noise. 2. The END instruction changed to other instruction code due to unknown cause.	1. Reset and run the CPU again. If the same error recurs. Since this is a CPU hardware error, consult your Mitsubishi representative.

## Table of error codes; AnA and AnAS CPUs

### Error codes 26 to 43

Error message	Error code in D9008	Detailed error code in D9091	Error description and cause	Remedy
MAIN CPU DOWN (1) (Checked continuously)	26		The main CPU is malfunctioning or faulty.	Since this is a CPU hardware error, consult your Mitsubishi representative.
UNIT VERIFY ERR. (Checked continuously)	31		Current I/O module information is different from that recognized when the power was turned on. 1. The I/O module (including special function modules) connection became loose or the module was disconnected during operation, or wrong module was connected.	Read detailed error code using a peripheral device and check or replace the module which corresponds to the data (I/O head number). Or, monitor special registers D9116 to D9123 using a peripheral device and check or replace the modules if corresponding data bit is „1“.
'FUSE BREAK OFF (Checked continuously)	32		There is an output module of which fuse is blown.	1. Check the FUSE BLOWN indicator LED on the output module and replace the fuse. 2. Read detailed error code using a peripheral device and replace the fuse of the output module which corresponds to the date (I/O head number). Or, monitor special registers D9100 to D9107 using a peripheral device and replace the fuse of the output module of which corresponding data bit is „1“.
CONTROL BUS ERR.	40	401	Due to the error of the control bus which connects to special function modules, the FROM/TO instruction cannot be executed.	Since it is a hardware error of special function module, CPU module or base module, replace and check defective module(s). Consult Mitsubishi representative for defective modules.
		402	If parameter I/O assignment is being executed, special function modules are not accessible at initial communication. At error occurrence, the head I/O number (upper 2 digits of 3 digits) of the special function module that caused error is stored at D9011.	
SPUNIT DOWN	41	411	Though an access was made to a special function module at execution of the FROM/TO instruction, no response is received.	Since it is a hardware error of special function module to which an access was made, consult Mitsubishi representative.
		412	If parameter I/O assignment is being executed, no response is received from a special function module at initial communication. At error occurrence, the head I/O number (upper 2 digits of 3 digits) of the special function module that caused error is stored at D9011.	
LINK UNIT ERROR	42	-	1. Either AJ71(A)R22, AJ71(A)P22 is loaded to the master station. 2. There are 2 link modules which are set to the master station (station 0).	1. Remove either AJ71(A)R22 or AJ71(A)P22 from the master station. 2. Reduce the number of master stations to 1. Reduce the link modules to 1 when the 3-tier system is not used.
I/O INT. ERROR	43	-	Though the interrupt module is not loaded, an interrupt occurred.	Since it is hardware error of a module, replace and check a defective module. For defective modules, consult Mitsubishi representative.

# Table of error codes; AnA and AnAS CPUs

## TError codes 44 to 70

Error message	Error code in D9008	Detailed error code in D9091	Error description and cause	Remedy
SPUNIT LAY ERROR	44	441	A special function module is assigned as an I/O module, or vice versa, in the I/O assignment using parameters from the peripheral device.	Execute I/O assignment again using parameters from the peripheral device according to the loading status of special function modules.
		442	There are 9 or more special function modules (except AI61(S1)) which can execute interruption to the CPU module loaded.	Reduce the special function modules (except AI61(S1)) which can execute interrupt start to 8 or less.
		443	There are 2 or more AJ71AP21, AJ71AR21, AJ71P22 or AJ71R22 modules loaded.	Reduce the AJ71AP21, AJ71AR21, AJ71P22 or AJ71R22 modules to 1 or less.
		444	There are 7 or more modules such as a computer link module loaded to one CPU module.	Reduce the computer link modules to 6 or less.
		445	There are 2 or more AI61(S1) modules loaded.	Reduce the AI61 module to 1.
		446	Modules assigned by parameters for MNT/MINI automatic refresh from the peripheral device do not conform with the types of station modules actually linked.	Perform again module assignment for MNT/MINI automatic refresh with parameters according to actually linked station modules.
		447	The number of modules of I/O assignment registration (number of loaded modules) per one CPU module for the special function modules which can use dedicated instructions is larger than the specified limit. (Total of the number of computers shown below is larger than 1344.)  <div style="text-align: right;">                     (AD59 X 5)                      (AD57(S1)/AD58 X 8)                      (AJ71C24(S3/S6/S8) X 10)                      (AJ71UC24 X 10)                      AJ71C21(S1) X 29                      + (AJ71PT32(S3) X 125)  <hr style="width: 10%; margin-left: auto; margin-right: 0;"/>                     Total &gt; 1344                 </div>	Reduce the number of loaded special function modules.
SP. UNIT ERROR (Checked during execution of a FROM / TO instruction)	46	461	Module specified by the FROM/TO instruction is not a special function module.	Read the error step using a peripheral device and check and correct contents of the FROM/TO instruction of the step.
		462	Module specified by the dedicated instruction for special function module is not a special function module or not the correct special function module.	Read the error step using a peripheral device and check and correct contents of the dedicated instruction for special function modules of the step.
LINK PARA. ERROR	47	-	1. If a data link CPU is used to set a master station (station number „0“): The link parameters written by the link CPU do not correspond to the link parameters read by the master station. Or else, link parameters are not written.  2. The setting of the total number of local stations is 0.	1. Write in parameters again and check. 2. Check setting of station numbers. 3. If the same error indication is given again, it is a hardware failure. Consult Mitsubishi representative.
BATTERY ERROR (Checked continuously)	70	-	1. Battery voltage has lowered below specified level.  2. Battery lead connector is not connected.	1. Replace battery.  2. If a RAM memory or power failure compensation function is used, connect the lead connector.



## Table of error codes; AnA and AnAS CPUs

### Error codes 50 to 60

Error message	Error code in D9008	Detailed error code in D9091	Error description and cause	Remedy
OPERATION ERROR (Checked at execution of instruction)	50	501	<ol style="list-style-type: none"> <li>1. When file registers (R) are used, operation is executed outside of specified ranges of device numbers and block numbers of file registers (R).</li> <li>2. File registers are used in the program without setting capacity of file registers.</li> </ol>	Read the error step using a peripheral device and check and correct program of the step.
		502	Combination of the devices specified by instruction is incorrect.	
		503	Stored data or constant of specified device is not in the usable range.	
		504	Set number of data to be handled is out of the usable range.	
		505	<ol style="list-style-type: none"> <li>1. Station number specified by the LEDA/B LRDP, LCDA/B LWTP, LRDP, LWTP instructions is not a local station.</li> <li>2. Head I/O number specified by the LEDA/B RFRP, LEDA/B RTOP, RFRP, RTOP instructions is not of a remote station.</li> </ol>	
		506	Head I/O number specified by the LEDA/B RFRP, LEDA/B RTOP, RFRP, RTOP instructions is not of a special function module.	
		507	<ol style="list-style-type: none"> <li>1. When the AD57(S1) or AD58 was executing instructions in divided processing mode, other instructions were executed to either of them.</li> <li>2. When an AD57(S1) or AD58 was executing instructions in divided processing mode, other instructions were executed in divided mode to another AD57(S1) or AD58.</li> </ol>	
509	<ol style="list-style-type: none"> <li>1. An instruction which cannot be executed by remote terminal modules connected to the MNET/ MINI-S3 was executed to the modules.</li> <li>2. When the PRC instruction was executed to a remote terminal, the communication request registration areas overflowed.</li> <li>3. The PIDCONT instruction was executed without executing the PIDINIT instruction. The PID57 instruction was executed without executing the PIDINIT or PIDCONT instruction.</li> </ol>	<ol style="list-style-type: none"> <li>1. Read the error step using a peripheral device and correct the program, meeting loaded conditions of remote terminal modules.</li> <li>2. Provide interlock using M9081 (communication request registration areas BUSY signal) or D9081 (number of vacant areas in the communication request registration areas) when the PRC instruction is executed to a remote terminal.</li> <li>3. Execute the PIDCONT instruction after execution of the PIDINIT instruction. Execute the PID57 instruction after execution of the PIDINIT and PIDCONT instructions.</li> </ol>		
MAIN CPU DOWN (2)  (Error during interrupt)	60	-	<ol style="list-style-type: none"> <li>1. The CPU malfunctioned due to noise.</li> <li>2. Hardware failure.</li> </ol>	<ol style="list-style-type: none"> <li>1. Take proper countermeasures for noise.</li> <li>2. Hardware failure.</li> </ol>



# A Appendix A

## A.1 Definition of the Processing Times

The operation processing time is the total of the following:

- Total of each instruction processing time.
- The END processing time. This time consists of the time to execute the END instruction, the MELSECNET related refresh time, the processing time for the communication with peripheral devices, and the time for serial communication.
- The I/O refresh time can be calculated with the following formula:

$$\text{I/O refresh time} = \frac{\text{Number of input points}}{16} \times N1 + \frac{\text{Number of output points}}{16} \times N2$$

The following table indicates the two times N1 and N2 for System Q and QnA CPUs:

Type of CPU	N1 (μs)			N2 (μs)		
	System Q Main Base	System Q Extension Base	QnA Extension Base	System Q Main Base	System Q Extension Base	QnA Extension Base
Q00JCPU	2,5	3,3	—	1,3	2,3	—
Q00CPU	2,4	3,2	—		2,3	—
Q01CPU	2,3	3,1	—		2,3	—
Q02CPU)	2,2	2,9	4,3		2,1	3,5
Q02HCPU Q06HCPU Q12HCPU Q12PHCPU Q25HCPU Q25PHCPU	1,7	2,4	3,7		2,1	3,5
Q2ASCPU (S1) Q2ACPU	5,2			5,0		
Q3ACPU	4,8			4,65		
Q2ASHCPU (S1) Q4ACPU Q4ARCPU	4,34			4,26		

## A.2 Processing times

The table on the following pages contains the processing times of all instructions. The according processing times depend on the values of source and destination data. The time values serves as calculation aid for the total processing time of a program.

The processing time for the instruction does not include the time for index qualification.

When the instruction is not executed the processing time is calculated as follows:

Type of CPU	Processing time when the instruction is not executed (μs)
Q00JCPU	0.20 x (Number of steps for each instruction +1)
Q00CPU	0.16 x (Number of steps for each instruction +1)
Q01CPU	0.10 x (Number of steps for each instruction +1)
Q02CPU)	0.079 x (Number of steps for each instruction +1)
Q02HCPU Q06HCPU Q12HCPU Q12PHCPU Q25HCPU Q25PHCPU	0.034 x (Number of steps for each instruction +1)
Q2ASCPU (S1) Q2ACPU	0.20 x (Number of steps for each instruction +1)
Q3ACPU	0.15 x (Number of steps for each instruction +1)
Q2ASHCPU (S1) Q4ACPU Q4ARCPU	0.075 x (Number of steps for each instruction +1)

**A.2.1 Table of Processing Times (QnA series and System Q)**

Instruction	Processing (Device)		Processing time (μs)									
			QnA series CPU modules				System Q CPU modules					
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
LD	e.g. X0		0.20	0.15	0.075	0.075	0.20	0.16	0.10	0.079	0.034	
LDI												
AND												
ANI												
OR												
ORI												
LDP	e.g. D0.0		6.6	5.0	2.5	2.5	0.30	0.24	0.15	0.158	0.068	
LDF												
ANDP												
ANDF												
ORP												
ORF												
ANB	0.20		0.20	0.15	0.075	0.075	0.20	0.16	0.10	0.079	0.034	
ORB												
MPS												
MRD												
MPP												
INV												not executed
	executed											
MEP	not executed	2.0	1.5	0.75	0.75	0.30	0.24	0.15	0.173	0.073		
MEF	executed											
EGP	not changed	0.6	0.3	0.15	0.15	0.20	0.16	0.10	0.158	0.068		
	changed (OFF/ON or ON/OFF)											
EGF	not changed	0.6	0.3	0.15	0.15	17	9.5	9.4	0.158	0.068		
	changed (OFF/ON or ON/OFF)					18	14	14				
OUT	excl. F, T and C	not changed	0.40	0.30	0.15	0.15	0.20	0.16	0.10	0.158	0.068	
		changed (OFF/ON or ON/OFF)										
	D0.0	not changed	0.40	0.30	0.15	0.15	0.40	0.32	0.20	0.158	0.068	
		changed (OFF/ON or ON/OFF)										
	F	not executed	7.0	5.3	2.7	2.7	24	20	19	2.8	1.2	
		executed	displayed	167	126	63	63	260	210	200	162	69.7
			display completed	166	125	62	62	205	165	155	126	54
	T	not executed	1.6	1.2	0.6	0.6	1.1	0.88	0.55	0.63	0.27	
		after time out										
		executed										added
	D											
	C	not executed	1.6	1.2	0.6	0.6	1.1	0.88	0.55	0.63	0.27	
after time out												
executed		added										K
	D											
OUTH	not executed	1.6	1.2	0.6	0.6	1.1	0.88	0.55	0.63	0.27		
	after time out											
	executed										added	K
D												
SET	all devices except F and D0.0	not executed	0.40	0.30	0.15	0.15	0.20	0.16	0.10	0.158	0.068	
		executed										not changed
		changed										
	D0.0	not executed	0.40	0.30	0.15	0.15	0.40	0.32	0.20	0.158	0.068	
		executed										not changed
		changed										
	F	not executed	1.2	0.90	0.45	0.45	0.50	0.44	0.25	0.47	0.20	
		executed	displayed	277	208	104	104	255	205	195	161	69
display completed			1.2	0.90	0.45	0.45	195	160	150	0.47	0.20	

Instruction	Processing (Device)		Processing time (μs)									
			QnA series CPU modules				System Q CPU modules					
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
RST	All devices except the ones listed below	not executed	0.40	0.30	0.15	0.15	0.20	0.16	0.10	0.158	0.068	
		executed										not changed
												changed
	D0.0	not executed	0.40	0.30	0.15	0.15	0.40	0.32	0.20	0.158	0.068	
		executed										not changed
												changed
	SM	not executed	0.40	0.30	0.15	0.15	0.20	0.16	0.10	0.158	0.068	
		executed										
	F	not executed	1.2	0.90	0.45	0.45	0.48	0.44	0.25	0.47	0.20	
		executed										displayed
												display completed
	T, C	not executed	1.4	1.1	0.6	0.6	0.80	0.64	0.40	0.63	0.27	
		executed					1.0	0.80	0.50			
	D	not executed	0.60	0.45	0.23	0.23	0.40	0.32	0.20	0.24	0.10	
executed		0.60					0.48	0.30				
Z	not executed	1.2	0.90	0.45	0.45	0.50	0.40	0.25	0.47	0.20		
	executed										10.8	8.1
R	not executed	1.0	0.75	0.38	0.38	-	0.32	0.20	0.40	0.17		
	executed					-	0.48	0.30				
PLS			2.6	2.0	0.98	0.98	12	9.5	9.2	1.0	0.44	
PLF			2.6	2.0	0.98	0.98	11	9.5	8.9	1.0	0.44	
FF	Y	not executed	1.2	0.90	0.45	0.45	0.68	0.40	0.25	0.47	0.20	
		executed					7.5	6.2	5.7			
DELTA	DY0	not executed	1.2	0.90	0.45	0.45	0.50	0.40	0.25	0.47	0.20	
		executed										16.8
DELTAP	DY0	not executed	1.2	0.90	0.45	0.45	0.48	0.40	0.25	0.47	0.20	
		executed										16.8
SFT SFTP	not executed		1.2	0.90	0.45	0.45	0.50	0.34	0.25	0.47	0.20	
	executed		4.2	3.2	1.6	1.6	12	8.7	8.3	1.66	0.71	
MC	M0.0		0.60	0.45	0.23	0.23	0.40	0.32	0.20	0.24	0.10	
	D0.0						3.3	2.9	2.8			
MCR			0.20	0.15	0.075	0.075	0.20	0.16	0.10	0.079	0.034	
FEND END	error check executed		1643	1236	618	618	660	530	480	348	150	
	without error check: -Battery check -Blown fuse check -Verification of I/O module		1106	832	416	416	660	530	480	359	150	
NOP			0.2	0.15	0.075	0.075	0.20	0.16	0.10	0.079	0.034	
NOPLF PAGE			0.2	0.15	0.075	0.075	0.20	0.16	0.10	0.79	0.034	
LD=	continuity		3.8	2.9	1.5	1.5	0.80	0.64	0.40	0.24	0.10	
	no continuity		3.6	2.7	1.4	1.4						
AND=	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10	
	executed	continuity	2.8	2.1	1.1	1.1	0.80	0.64	0.40			
		no continuity	3.2	2.4	1.2	1.2						
OR=	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10	
	executed	continuity	3.8	2.9	1.5	1.5	0.80	0.64	0.40			
		no continuity	2.8	2.1	1.1	1.1						
LD<>	continuity		4.4	3.3	1.7	1.7	0.80	0.64	0.40	0.24	0.10	
	no continuity		3.6	2.7	1.4	1.4						
AND<>	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10	
	executed	continuity	2.8	2.1	1.1	1.1	0.80	0.64	0.40			
		no continuity	3.2	2.4	1.2	1.2						

Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
OR<>	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	3.8	2.9	1.5	1.5	0.80	0.64	0.40		
		no continuity	2.8	2.1	1.1	1.1					
LD>	continuity		4.4	3.3	1.7	1.7	0.80	0.64	0.40	0.24	0.10
	no continuity		3.6	2.7	1.4	1.4					
AND>	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	2.8	2.1	1.1	1.1	0.80	0.64	0.40		
		no continuity	3.2	2.4	1.2	1.2					
OR>	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	3.8	2.9	1.5	1.5	0.80	0.64	0.40		
		no continuity	2.8	2.1	1.1	1.1					
LD<=	continuity		4.4	3.3	1.7	1.7	0.80	0.64	0.40	0.24	0.10
	no continuity		3.6	2.7	1.4	1.4					
AND<=	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	2.8	2.1	1.1	1.1	0.80	0.64	0.40		
		no continuity	3.2	2.4	1.2	1.2					
OR<=	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	3.8	2.9	1.5	1.5	0.80	0.64	0.40		
		no continuity	2.8	2.1	1.1	1.1					
LD<	continuity		4.4	3.3	1.7	1.7	0.80	0.64	0.40	0.24	0.10
	no continuity		3.6	2.7	1.4	1.4					
AND<	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	2.8	2.1	1.1	1.1	0.80	0.64	0.40		
		no continuity	3.2	2.4	1.2	1.2					
OR<	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	3.8	2.9	1.5	1.5	0.80	0.64	0.40		
		no continuity	2.8	2.1	1.1	1.1					
LD>=	continuity		4.4	3.3	1.7	1.7	0.80	0.64	0.40	0.24	0.10
	no continuity		3.6	2.7	1.4	1.4					
AND>=	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	2.8	2.1	1.1	1.1	0.80	0.64	0.40		
		no continuity	3.2	2.4	1.2	1.2					
OR>=	not executed		1.4	1.1	0.55	0.55	0.70	0.56	0.35	0.24	0.10
	executed	continuity	3.8	2.9	1.5	1.5	0.80	0.64	0.40		
		no continuity	2.8	2.1	1.1	1.1					
LDD=	continuity		5.0	3.8	1.9	1.9	1.0	0.80	0.50	0.55	0.24
	no continuity		4.2	3.2	1.6	1.6				0.39	0.17
ANDD=	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17
	executed	continuity	3.4	2.6	1.3	1.3	1.0	0.80	0.50	0.55	0.24
		no continuity	3.8	2.9	1.5	1.5				0.39	0.17
ORD=	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17
	executed	continuity	4.4	3.3	1.7	1.7	1.0	0.80	0.50	0.55	0.24
		no continuity	3.4	2.6	1.3	1.3					
LDD<>	continuity		5.0	3.8	1.9	1.9	1.0	0.80	0.50	0.55	0.24
	no continuity		4.2	3.2	1.6	1.6					
ANDD<>	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17
	executed	continuity	3.4	2.6	1.3	1.3	1.0	0.80	0.50	0.55	0.24
		no continuity	3.8	2.9	1.5	1.5					
ORD<>	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17
	executed	continuity	4.4	3.3	1.7	1.7	1.0	0.80	0.50	0.55	0.24
		no continuity	3.4	2.6	1.3	1.3					
LDD>	continuity		3.8	2.9	1.5	1.5	1.0	0.80	0.50	0.55	0.24
	no continuity		4.2	3.2	1.6	1.6					

Instruction	Processing (Device)		Processing time (μs)									
			QnA series CPU modules				System Q CPU modules					
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
ANDD>	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	2.8	2.1	1.1	1.1	1.0	0.80	0.50	0.55	0.24	
		no continuity	3.8	2.9	1.5	1.5						
ORD>	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	3.8	2.9	1.5	1.5	1.0	0.80	0.50	0.55	0.24	
		no continuity	3.4	2.6	1.3	1.3						
LDD<=	continuity		4.4	3.3	1.7	1.7	1.0	0.80	0.50	0.55	0.24	
	no continuity		3.6	2.7	1.4	1.4						
ANDD<=	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	3.4	2.6	1.3	1.3	1.0	0.80	0.50	0.55	0.24	
		no continuity	3.2	2.4	1.2	1.2						
ORD<=	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	4.4	3.3	1.7	1.7	1.0	0.80	0.50	0.55	0.24	
		no continuity	2.8	2.1	1.1	1.1						
LDD<	continuity		3.8	2.9	1.5	1.5	1.0	0.80	0.50	0.55	0.24	
	no continuity		4.2	3.2	1.6	1.6						
ANDD<	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	2.8	2.1	1.1	1.1	1.0	0.80	0.50	0.55	0.24	
		no continuity	3.8	2.9	1.5	1.5						
ORD<	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	3.8	2.9	1.5	1.5	1.0	0.80	0.50	0.55	0.24	
		no continuity	3.4	2.6	1.3	1.3						
LDD>=	continuity		4.4	3.3	1.7	1.7	1.0	0.80	0.50	0.55	0.24	
	no continuity		3.6	2.7	1.4	1.4						
ANDD>=	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	3.4	2.6	1.3	1.3	1.0	0.80	0.50	0.55	0.24	
		no continuity	3.2	2.4	1.2	1.2						
ORD>=	not executed		1.4	1.1	0.55	0.55	0.80	0.64	0.40	0.39	0.17	
	executed	continuity	4.4	3.3	1.7	1.7	1.0	0.80	0.50	0.55	0.24	
		no continuity	2.8	2.1	1.1	1.1						
LDE=	Single precision	continuity		235	177	89	35	—	—	—	93	40
		no continuity		231	174	87	87	—	—	—	92	
	Double precision	continuity		—	—	—	—	—	—	—	93	40
		no continuity		—	—	—	—	—	—	—	92	
ANDE=	Single precision	not executed		1.4	1.1	0.55	0.55	—	—	—	0.55	0.24
		executed	continuity	234	176	88	35	—	—	—	93	
			no continuity	230	172	86	86	—	—	—	92	
	Double precision	not executed		—	—	—	—	—	—	—	—	—
		executed	continuity	—	—	—	—	—	—	—	93	
			no continuity	—	—	—	—	—	—	—	92	
ORE=	Single precision	not executed		1.4	1.1	0.55	0.55	—	—	—	0.55	0.24
		executed	continuity	234	176	88	35	—	—	—	93	
			no continuity	230	172	86	86	—	—	—	92	
	Double precision	not executed		—	—	—	—	—	—	—	—	—
		executed	continuity	—	—	—	—	—	—	—	93	
			no continuity	—	—	—	—	—	—	—	92	
LDE<>	Single precision	continuity		231	174	87	35	—	—	—	92	40
		no continuity		234	176	88	88	—	—	—		
	Double precision	continuity		—	—	—	—	—	—	—	92	40
		no continuity		—	—	—	—	—	—	—		



Instruction	Processing (Device)		Processing time (μs)									
			QnA series CPU modules				System Q CPU modules					
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
ANDE<>	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	230	172	86	34	—	—	—	92	40
			no continuity	234	176	88	88	—	—	—	93	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	92	40
			no continuity	—	—	—	—	—	—	—	—	
ORE<>	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	231	174	87	35	—	—	—	93	40
			no continuity	234	176	88	88	—	—	—	92	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	93	40
			no continuity	—	—	—	—	—	—	—	92	
LDE>	Single precision	continuity	231	174	87	35	—	—	—	92	40	
		no continuity	234	176	88	88	—	—	—	—		
	Double precision	continuity	—	—	—	—	—	—	—	92	40	
		no continuity	—	—	—	—	—	—	—	—		
ANDE>	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	230	172	86	34	—	—	—	92	40
			no continuity	234	176	88	88	—	—	—	93	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	92	40
			no continuity	—	—	—	—	—	—	—	—	
ORE>	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	231	174	87	34	—	—	—	93	40
			no continuity	234	176	88	88	—	—	—	92	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	93	40
			no continuity	—	—	—	—	—	—	—	92	
LDE<=	Single precision	continuity	235	177	89	34	—	—	—	93	40	
		no continuity	231	174	87	88	—	—	—	92		
	Double precision	continuity	—	—	—	—	—	—	—	93	40	
		no continuity	—	—	—	—	—	—	—	92		
ANDE<=	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	234	176	88	34	—	—	—	92	40
			no continuity	230	172	86	86	—	—	—	—	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	92	40
			no continuity	—	—	—	—	—	—	—	—	
ORE<=	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	234	176	88	34	—	—	—	92	40
			no continuity	230	172	86	86	—	—	—	—	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	92	40
			no continuity	—	—	—	—	—	—	—	—	
LDE<	Single precision	continuity	231	174	87	35	—	—	—	92	40	
		no continuity	234	176	88	88	—	—	—	—		
	Double precision	continuity	—	—	—	—	—	—	—	92	92	
		no continuity	—	—	—	—	—	—	—	—		
ANDE<	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	230	172	86	34	—	—	—	92	40
			no continuity	234	176	88	88	—	—	—	—	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	92	40
			no continuity	—	—	—	—	—	—	—	—	

Instruction	Processing (Device)		Processing time (μs)									
			QnA series CPU modules				System Q CPU modules					
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
ORE<	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	231	174	87	34	—	—	—	93	40
			no continuity	234	176	88	88	—	—	—	92	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	93	40
			no continuity	—	—	—	—	—	—	—	92	
LDE>=	Single precision	continuity	235	177	89	35	—	—	—	93	40	
		no continuity	231	174	87	87	—	—	—	92		
	Double precision	continuity	—	—	—	—	—	—	—	93	40	
		no continuity	—	—	—	—	—	—	—	92		
ANDE>=	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	234	176	88	34	—	—	—	92	40
			no continuity	231	174	87	87	—	—	—	92	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	92	40
			no continuity	—	—	—	—	—	—	—	92	
ORE>=	Single precision	not executed	1.4	1.1	0.55	0.55	—	—	—	0.55	0.24	
		executed	continuity	234	176	88	34	—	—	—	92	40
			no continuity	230	172	86	86	—	—	—	92	
	Double precision	not executed	—	—	—	—	—	—	—	—	—	
		executed	continuity	—	—	—	—	—	—	—	92	40
			no continuity	—	—	—	—	—	—	—	92	
LD\$=	continuity		97	73	37	37	—	—	—	38	16	
	no continuity		81	61	31	31	—	—	—	34	15	
AND\$=	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.23	
	executed	continuity	96	72	36	36	—	—	—	39	17	
		no continuity	81	61	31	31	—	—	—	32	14	
OR\$=	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.24	
	executed	continuity	97	73	37	37	—	—	—	40	17	
		no continuity	80	60	30	30	—	—	—	33	14	
LD\$<>	continuity		83	62	31	31	—	—	—	32	14	
	no continuity		97	73	37	37	—	—	—	40	17	
AND\$<>	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.23	
	executed	continuity	80	60	30	30	—	—	—	33	14	
		no continuity	96	72	36	36	—	—	—	39	17	
OR\$<>	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.24	
	executed	continuity	81	61	31	31	—	—	—	32	14	
		no continuity	96	72	36	36	—	—	—	39	17	
LD\$>	continuity		83	62	31	31	—	—	—	32	14	
	no continuity		97	73	37	37	—	—	—	40	17	
AND\$>	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.23	
	executed	continuity	80	60	30	30	—	—	—	33	14	
		no continuity	96	72	36	36	—	—	—	39	17	
OR\$>	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.24	
	executed	continuity	81	61	31	31	—	—	—	32	14	
		no continuity	96	72	36	36	—	—	—	39	17	
LD\$<=	continuity		97	73	37	37	—	—	—	40	17	
	no continuity		81	61	31	31	—	—	—	32	14	
AND\$<=	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.23	
	executed	continuity	96	72	36	36	—	—	—	39	17	
		no continuity	81	61	31	31	—	—	—	32	14	
OR\$<=	not executed		1.4	1.1	0.55	0.55	—	—	—	0.56	0.24	
	executed	continuity	97	73	37	37	—	—	—	40	17	
		no continuity	80	60	30	30	—	—	—	33	14	

Instruction	Processing (Device)	Processing time (μs)									
		QnA series CPU modules				System Q CPU modules					
		Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
LD\$<	continuity	81	61	31	31	—	—	—	32	14	
	no continuity	97	73	37	37	—	—	—	40	17	
AND\$<	not executed	1.4	1.1	0.55	0.55	—	—	—	0.56	0.23	
	executed	continuity	80	60	30	30	—	—	—	32	14
		no continuity	96	72	36	36	—	—	—	39	16
OR\$<	not executed	1.4	1.1	0.55	0.55	—	—	—	0.56	0.24	
	executed	continuity	81	61	31	31	—	—	—	32	14
		no continuity	96	72	36	36	—	—	—	39	16
LD\$>=	continuity	97	73	37	37	—	—	—	40	17	
	no continuity	81	61	31	31	—	—	—	32	14	
AND\$>=	not executed	1.4	1.1	0.55	0.55	—	—	—	0.56	0.23	
	executed	continuity	96	72	36	36	—	—	—	39	16
		no continuity	81	61	31	31	—	—	—	32	14
OR\$>=	not executed	1.4	1.1	0.55	0.55	—	—	—	0.56	0.24	
	executed	continuity	97	73	37	37	—	—	—	39	17
		no continuity	80	60	30	30	—	—	—	32	14
BKCM=	n = 1	120	90	45	45	130	105	97	48	21	
BKCM=P	n = 96	367	276	138	138	205	175	165	142	61	
BKCM<>	n = 1	123	92	46	46	130	105	98	48	21	
BKCM<>P	n = 96	346	260	130	130	210	180	165	150	65	
BKCM>	n = 1	123	92	96	96	130	105	97	48	21	
BKCM>P	n = 96	366	275	138	138	210	180	165	142	61	
BKCM>=	n = 1	121	91	46	46	130	105	98	48	21	
BKCM>=P	n = 96	386	290	145	145	205	175	165	150	65	
BKCM<	n = 1	121	91	96	96	130	105	98	48	21	
BKCM<P	n = 96	366	275	138	138	210	180	165	158	68	
BKCM<=	n = 1	121	91	46	46	130	105	97	48	21	
BKCM<=P	n = 96	348	261	131	131	205	175	165	150	65	
+ (s,d) +P (s,d)		2.4	1.8	0.9	0.9	1.0	0.80	0.50	0.39	0.17	
+ (s1,s2,d) +P (s1,s2,d)		2.7	2.0	1.0	1.0	1.2	0.96	0.60	0.47	0.20	
-(s,d) -P (s,d)		2.4	1.8	0.9	0.9	1.0	0.80	0.50	0.39	0.17	
-(s1,s2,d) -P (s1,s2,d)		2.6	2.0	1.0	1.0	1.2	0.96	0.60	0.47	0.20	
D+ (s,d) D+P (s,d)		2.8	2.1	1.1	1.1	1.3	1.04	0.65	0.71	.031	
D+ (s1,s2,d) D+P (s1,s2,d)		3.2	2.4	1.2	1.2	1.5	1.2	0.75	0.79	0.34	
D-(s,d) D-P(s,d)		2.8	2.1	1.1	1.1	1.3	1.04	0.65	0.71	0.30	
D- (s1,s2,d) D-P (s1,s2,d)		3.2	2.4	1.2	1.2	1.5	1.2	0.75	0.79	0.34	
x (s1,s2,d) xP (s1,s2,d)		2.8	2.1	1.1	1.1	1.1	0.88	0.55	0.47	0.20	
/ (s1,s2,d) /P (s1,s2,d)		6.8	5.1	2.6	2.6	19	16	15	2.7	1.2	

Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
Dx (s1,s2,d) DxP (s1,s2,d)			20	15	7.5	7.5	41	34	31	7.9	3.4
D/ (s1,s2,d) D/P (s1,s2,d)			36	27	13.5	13.5	28	23	21	14	6.1
B+ (s,d) B+P (s,d)			5.5	4.1	2.1	2.1	34	28	26	2.2	1.0
B+ (s1,s2,d) B+P (s1,s2,d)			13	9.6	4.8	4.8	47	39	37	5.0	2.2
B- (s,d) B-P (s,d)			5.2	3.9	2.0	2.0	34	28	26	2.0	0.9
B- (s1,s2,d) B-P (s1,s2,d)			13	9.4	4.7	4.7	48	40	38	4.9	2.1
DB+ (s,d) DB+P (s,d)			29	22	11	11	58	48	44	12	5.0
DB+ (s1,s2,d) DB+P (s1,s2,d)			32	24	12	12	60	49	46	12	5.3
DB- (s,d) DB-P (s,d)			29	22	11	11	59	48	45	11	4.8
DB- (s1,s2,d) DB-P (s1,s2,d)			32	24	12	12	60	51	45	12	5.2
Bx (s1, s2, d) BxP (s1, s2, d)			9.4	7.1	3.6	3.6	42	35	33	3.7	1.6
B/ (s1, s2, d) B/P (s1, s2, d)			9.4	7.1	3.6	3.6	48	40	37	3.8	1.6
DBx (s1, s2, d) DBxP (s1, s2, d)			62	46	23	23	140	120	110	24	10
DB/ (s1, s2, d) DB/P (s1,s2,d)			69	52	26	26	83	69	65	27	12
E+ (s, d) E+P (s, d)	Single precision	s = 0, d = 0	54	40	20	35	—	—	—	1.8	0.78
		s = 2 <sup>127</sup> , d = 2 <sup>127</sup>	524	394	197						
E+ (s, d)	Double precision	s = 0, d = 0	—	—	—	—	—	—	—	203	87
		s = 2 <sup>127</sup> , d = 2 <sup>127</sup>	—	—	—						
E+ (s1, s2, d) E+P (s1, s2, d)	Single precision	s1 = 0, s2 = 0	54	40	20	35	—	—	—	2.4	1.1
		s1 = 2 <sup>127</sup> , s2 = 2 <sup>127</sup>	524	394	197						
E+ (s1, s2, d)	Double precision	s = 0, d = 0	—	—	—	—	—	—	—	209	90
		s = 2 <sup>127</sup> , d = 2 <sup>127</sup>	—	—	—						
E- (s, d) E-P (s, d)	Single precision	s = 0, d = 0	54	40	20	35	—	—	—	1.8	0.78
		s = 2 <sup>127</sup> , d = 2 <sup>127</sup>	515	387	194						
E- (s, d)	Double precision	s = 0, d = 0	—	—	—	—	—	—	—	202	87
		s = 2 <sup>127</sup> , d = 2 <sup>127</sup>	—	—	—						

Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
E- (s1, s2, d) E-P (s1, s2, d)	Single precision	s1 = 0, s2 = 0	55	41	21	36	—	—	—	2.4	1.1
		$s1 = 2^{127}, s2 = 2^{127}$	520	391	146						
	Double precision	s = 0, d = 0	—	—	—	—	—	—	—	210	90
		$s = 2^{127}, d = 2^{127}$	—	—	—						
Ex (s1, s2, d) Exp (s1, s2, d)	Single precision	s1 = 0, s2 = 0	55	41	21	36	—	—	—	2.4	1.1
		$s1 = 2^{127}, s2 = 2^{127}$	567	426	218						
	Double precision	s = 0, d = 0	—	—	—	—	—	—	—	222	96
		$s = 2^{127}, d = 2^{127}$	—	—	—						
E/ (s1, s2, d) E/P (s1, s2, d)	Single precision	s1 = 0, s2 = 1	149	112	56	37	—	—	—	12	5.2
		$s1 = 2^{127}, s2 = -2^{126}$	1109	834	417						
	Double precision	s = 0, d = 0	—	—	—	—	—	—	—	369	159
		$s1 = 2^{127}, s2 = -2^{126}$	—	—	—						
\$+ (s, d) \$+P (s, d)			179	134	67	67	—	—	—	68	29
\$+ (s1, s2, d) \$+P (s1, s2, d)			206	155	78	78	—	—	—	81	35
INC INCP			1.9	1.4	0.7	0.7	0.70	0.56	0.35	0.32	0.14
DINC DINCP			2.3	1.7	0.9	0.9	0.90	0.72	0.45	0.47	0.20
DEC DECP			1.9	1.4	0.7	0.7	0.70	0.56	0.35	0.32	0.14
DDEC DDECP			2.3	1.7	0.9	0.9	0.90	0.72	0.45	0.47	0.20
BCD BCDP			2.7	2.0	1.0	1.0	20	16	15	1.1	0.48
DBCD DBCDP			7.9	5.9	3.0	3.0	26	21	20	3.2	1.4
BIN BINP			2.7	2.0	1.0	1.0	19	16	15	1.0	0.44
DBIN DBINP			4.8	3.6	1.8	1.8	22	18	17	1.9	0.82
INT INTP	Single precision	s = 0	20	15	7.5	7.5	—	—	—	3.2	1.4
		s = 32766.5	54	40	20	20					
	Double precision	s = 0	—	—	—	—	—	—	—	22	9.3
		s = 32766.5	—	—	—	—					
DINT DINTP	Single precision	s = 0	20	15	7.5	7.5	—	—	—	2.5	1.1
		s = 1234567890.3	59	44	22	22					
	Double precision	s = 0	—	—	—	—	—	—	—	24	10
		s = 1234567890.3	—	—	—	—					
FLT FLTP	Single precision	s = 0	27	20	10	10	—	—	—	2.1	0.92
		s = 7FFF <sub>H</sub>	55	41	21	21					
	Double precision	s = 0	—	—	—	—	—	—	—	22	9.6
		s = 7FFF <sub>H</sub>	—	—	—	—					
DFLT DFLTP	Single precision	s = 0	28	21	11	11	—	—	—	2.1	0.88
		s = 7FFFFFFF <sub>H</sub>	56	42	21	21					
	Double precision	s = 0	—	—	—	—	—	—	—	26	11
		s = 7FFFFFFF <sub>H</sub>	—	—	—	—					
DBL DBLP			12	8.6	4.3	4.3	19	16	15	4.5	1.9
WORD WORDP			12	9.0	4.5	4.5	23	19	17	4.7	2.0

Instruction	Processing (Device)	Processing time (μs)								
		QnA series CPU modules				System Q CPU modules				
		Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
GRY GRYP		12	9.0	4.5	4.5	19	16	15	4.7	2.0
DGRY DGRYP		14	10	5.0	5.0	23	19	17	5.3	2.3
GBIN GBINP		46	34	17	17	52	42	40	18	7.7
DGBIN DGBINP		83	62	31	31	110	88	84	32	14
NEG NEGP		9.3	7	3.5	3.5	16	13	12	3.6	1.6
DNEG DNEGP		11	8.2	4.1	4.1	19	17	15	4.3	1.8
ENEG ENEGP		9.8	7.4	3.7	3.7	—	—	—	3.9	1.7
BKBCD (s, d, n)	n = 1	102	76	38	38	78	63	57	38	17
BKBCDP (s, d, n)	n = 96	272	204	102	102	315	275	250	99	43
BKBIN (s, d, n)	n = 1	102	76	38	38	74	61	57	38	17
BKBINP (s, d, n)	n = 96	272	204	102	102	285	255	230	99	43
MOV MOVVP	s = D0, d = D1	0.7	0.5	0.3	0.3	0.70	0.56	0.35	0.24	0.10
	s = D0, d = J1/W1	392 <sup>1</sup> 391 <sup>2</sup>	305 <sup>1</sup> 299 <sup>2</sup>	176 <sup>1</sup> 165 <sup>2</sup>	176 <sup>1</sup> 165 <sup>2</sup>	155	130	120	140	160
DMOV DMOVVP	s = K4X0, d = D1	2.4	1.8	0.9	0.9	0.90	0.72	0.45	0.47	0.20
	s = K4X0, d = J1/W1	400 <sup>1</sup> 395 <sup>2</sup>	313 <sup>1</sup> 301 <sup>2</sup>	183 <sup>1</sup> 167 <sup>2</sup>	183 <sup>1</sup> 167 <sup>2</sup>	165	135	120	147	64
EMOV EMOVVP		12	8.6	4.3	4.3	—	—	—	0.63	0.27
\$MOV \$MOVVP		100	75	38	38	46 98	38 80	35 73	40	17
CML CMLP		2.0	1.5	0.8	0.8	0.70	0.56	0.35	0.40	0.17
DCML DCMLP		2.4	1.8	0.9	0.9	0.90	0.72	0.45	0.55	0.24
BMOV (s, d, n)	n = 1	43	32	16	16	27	21	20	17	7.1
BMOVVP (s, d, n)	n = 96	81	61	31	31	72	62	53	32	14
FMOV (s, d, n)	n = 1	18	13	6.5	6.5	23	19	17	6.7	2.9
FMOVVP (s, d, n)	n = 96	36	27	14	14	48	41	36	14	6.1
XCH XCHP		3.1	2.3	1.2	1.2	7.6	6.3	5.7	1.3	0.54
DXCH DXCHP		3.1	2.3	1.2	1.2	9.5	8.0	7.1	1.3	0.54
BXCH (d1, d2, n)	n = 1	77	58	29	29	62	51	48	31	13
BXCHP (d1, d2, n)	n = 96	213	160	80	80	165	140	125	84	36
SWAP SWAPP		9.2	6.9	3.5	3.5	17	14	13	3.7	1.6

<sup>1</sup> These are the processing times when a A38B/A1S38B main base is used in combination with an extension base.

<sup>2</sup> These processing times are for a A38HB/A1S38HB main base.

Instruction	Processing (Device)	Processing time (μs)								
		QnA series CPU modules				System Q CPU modules				
		Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
CJ		7.8	5.8	2.9	2.9	10	8.5	8.1	3.2	1.4
SCJ		7.8	5.8	2.9	2.9	10	8.5	8.1	3.2	1.4
JMP		8.0	6.0	3.0	3.0	11	8.5	8.1	3.2	1.4
GOEND		2.0	1.5	0.75	0.75	3.3	2.9	2.8	0.39	0.34
EI		3.1	2.3	1.2	1.2	14	11	11	1.3	0.54
DI		2.3	1.7	0.9	0.9	13	12	11	0.95	0.41
IMASK		8.1	6.5	3.3	3.3	41	34	35	11	4.6
IRET		4.0	3.0	1.5	1.5	205	170	155	1.6	0.68
RFS RFSP	s = X, n = 1	31.3	23.4	11.7	11.7	55	46	43	6.7	4.7
	s = Y, n = 1					54	45	41		
	s = X, n = 96	97.6	72.8	36.4	36.4	79	64	59	19	13
	s = Y, n = 96					73	61	56		
UDCNT1		42.6	31.8	15.9	15.9	—	—	—	15	6.5
UDCNT2		44.6	33.3	16.7	16	—	—	—	16	6.8
TTMR		25.9	19.3	9.7	9.7	—	—	—	10	4.4
STMR		41.7	31.1	15.6	15.6	—	—	—	20	7.1
ROTC		66.1	49.3	24.7	24.7	—	—	—	26	11
RAMP		45.4	33.9	17.0	17.0	—	—	—	18	7.7
SPD		48.9	36.5	18.3	18.3	—	—	—	19	8.3
PLSY		26.9	20.1	10.1	10.1	—	—	—	10	4.5
PWM		32.8	24.5	12.3	12.3	—	—	—	9.1	3.9
MTR		29.2	21.8	10.9	10.9	—	—	—	11	4.9
WAND (s, d) WANDP (s, d)		2.4	1.8	0.9	0.9	1.0	0.80	0.50	0.39	0.17
WAND (s1, s2, d) WANDP (s1, s2, d)		9.5	7.1	3.6	3.6	1.2	0.96	0.60	0.47	0.20
DAND (s, d) DANDP (s, d)		3.0	2.3	1.2	1.2	1.3	1.04	0.65	0.71	0.31
DAND (s1, s2, d) DANDP (s1, s2, d)		19	14	7.0	7.0	1.5	1.2	0.75	0.79	0.34
BKAND (s1, s2, d, n)	n = 1	89	67	34	34	110	87	79	36	16
BKANDP (s1, s2, d, n)	n = 96	184	138	69	69	185	155	140	74	32
WOR (s, d) WORP (s, d)		2.4	1.8	0.9	0.9	1.0	0.80	0.50	0.40	0.17
WOR (s1, s2, d) WORP (s1, s2, d)		9.5	7.1	3.6	3.6	1.2	0.96	0.60	0.47	0.20
DOR (s, d) DORP (s, d)		3.0	2.3	1.2	1.2	1.3	1.04	0.65	0.71	0.31
DOR (s1, s2, d) DORP (s1, s2, d)		19	14	7.0	7.0	1.5	1.2	0.75	0.79	0.34
BKOR (s1, s2, d, n)	n = 1	89	67	34	34	110	87	81	36	16
BKORP (s1, s2, d, n)	n = 96	184	138	69	69	185	155	140	74	32

Instruction	Processing (Device)	Processing time (μs)								
		QnA series CPU modules				System Q CPU modules				
		Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
WXOR (s, d) WXORP (s, d)		2.4	1.8	0.9	0.9	1.0	0.80	0.50	0.39	0.17
WXOR (s1, s2, d) WXORP (s1, s2, d)		17.2	7.1	3.6	3.6	1.2	0.96	0.60	0.47	0.20
DXOR (s, d) DXORP (s, d)		3.0	2.3	1.2	1.2	1.3	1.04	0.65	0.71	0.31
DXOR (s1, s2, d) DXORP (s1, s2, d)		19	14	7.0	7.0	1.5	1.2	0.75	0.79	0.34
BKXOR (s1, s2, d, n)	n = 1	89	67	34	34	110	87	81	36	16
BKXORP (s1, s2, d, n)	n = 96	184	138	69	69	183	155	140	74	32
WXNR (s, d) WXNRP (s, d)		2.4	1.8	0.9	0.9	1.0	0.80	0.50	0.40	0.17
WXNR (s1, s2, d) WXNRP (s1, s2, d)		9.5	7.1	3.6	3.6	1.2	0.96	0.60	0.47	0.20
DXNR (s,d) DXNRP (s,d)		3.0	2.3	1.2	1.2	1.3	1.04	0.65	0.71	0.31
DXNR (s1,s2,d) DXNRP (s1,s2,d)		24	18	9	9	1.5	1.2	0.75	0.79	0.34
BKXNR (s1, s2, d, n)	n = 1	89	67	34	34	110	87	82	36	16
BKXNR (s1, s2, d, n)	n = 96	184	138	69	69	185	155	140	74	32
ROR (d, n)	n = 1	5.0	3.8	1.9	1.9	13	11	9.7	2.0	0.85
RORP (d, n)	n = 15	5.0	3.8	1.9	1.9	13	11	9.7	2.0	0.85
RCR (d, n)	n = 1	4.0	3.0	1.5	1.5	15	12	12	1.6	0.68
RCRP (d, n)	n = 15	4.0	3.0	1.5	1.5	15	13	12	1.6	0.68
ROL (d, n)	n = 1	5.0	3.8	1.9	1.9	13	11	10	2.0	0.85
ROLP (d, n)	n = 15	5.0	3.8	1.9	1.9	13	11	10	2.0	0.85
RCL (d, n)	n = 1	4.0	3.0	1.5	1.5	15	13	12	1.6	0.68
RCLP (d, n)	n = 15	4.0	3.0	1.5	1.5	16	13	12	1.6	0.68
DROR (d, n)	n = 1	9.8	7.4	3.7	3.7	15	12	12	3.9	1.7
DRORP (d, n)	n = 31	10	7.8	3.9	3.9	15	13	12	4.0	1.7
DRCR (d, n)	n = 1	11	8.1	4.1	4.1	17	14	14	4.3	1.8
DRCRP (d, n)	n = 31	11	8.3	4.2	4.2	18	16	15	4.3	1.9



Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
DROL (d, n)	n = 1		9.8	7.4	3.7	3.7	14	13	12	3.9	1.7
DROLP (d, n)	n = 31		10	7.8	3.9	3.9	14	13	12	4.0	1.7
DRCL (d, n)	n = 1		11	8.1	4.1	4.1	18	15	14	4.3	1.8
DRCLP (d, n)	n = 31		11	8.3	4.2	4.2	20	17	16	4.3	1.9
SFR (d, n)	n = 1		4.4	3.3	1.7	1.7	13	10	9.7	1.7	0.75
SFRP (d, n)	n = 15		5.0	3.8	1.9	1.9	13	11	9.5	2.0	0.85
SFL (d, n)	n = 1		4.4	3.3	1.7	1.7	12	10	9.5	1.7	0.75
SFLP (d, n)	n = 15		5.0	3.8	1.9	1.9	12	9.8	9.5	2.0	0.85
BSFLR (d, n)	n = 1		51	38	19	19	42	35	33	20	8.6
BSFLRP (d, n)	n = 96		60	45	23	23	69	58	54	24	10
BSFL (d, n)	n = 1		49	37	19	19	41	34	32	20	8.5
BSFLP (d, n)	n = 96		58	44	22	22	63	53	50	23	10
DSFR (d, n)	n = 1		3.6	2.6	1.3	1.3	19	16	15	1.3	0.58
DSFRP (d, n)	n = 96		63	47	24	24	71	61	53	25	11
DSFL (d, n)	n = 1		3.6	2.6	1.3	1.3	19	16	15	1.3	0.58
DSFLP (d, n)	n = 96		65	49	25	25	70	60	52	26	11
BSET (d, n)	n = 1		20	15	7.5	7.5	27	22	20	7.6	3.3
BSETP (d, n)	n = 15		20	15	7.5	7.5	27	22	20	7.6	3.3
BRST (d, n)	n = 1		20	15	7.5	7.5	27	22	21	7.6	3.3
BRSTP (d, n)	n = 15		20	15	7.5	7.5	27	22	21	7.6	3.3
TEST (s1, s2, d)			21	16	8.0	8.0	35	30	27	8.2	3.5
TESTP (s1, s2, d)											
DTEST (s1, s2, d)			24	18	9.0	9.0	37	31	28	9.2	3.9
DTESTP (s1, s2, d)											
BKRST (s, n)	n = 1		45	34	17	17	49	41	38	18	7.8
BKRST (s, n)	n = 96		49	37	19	19	64	54	50	19	8.2
SER (s1, s2, d, n)	n = 1	match	58	44	22	22	56	54	42	22	9.6
SER (s1, s2, d, n)		no match	57	43	21	21	56	54	42	21	8.9
SERP (s1, s2, d, n)	n = 96	match	293	220	110	110	280	240	220	115	49
SERP (s1, s2, d, n)		no match	340	256	128	128	280	240	220	133	57
DSER (s1, s2, d, n)	n = 1	match	61	46	23	23	71	67	53	23	9.9
DSER (s1, s2, d, n)		no match	58	44	22	22	71	67	54	23	9.7
DSERP (s1, s2, d, n)	n = 96	match	354	266	133	133	495	415	375	142	61
DSERP (s1, s2, d, n)		no match	354	266	133	133	500	415	375	132	57
SUM	s = 0		9.8	7.4	3.7	3.7	32	26	25	3.9	1.7
SUMP	s = FFFF <sub>H</sub>						27	22	21		
DSUM	s = 0		12	9.0	4.5	4.5	54	44	42	4.7	2.0
DSUMP	s = FFFFFFFF <sub>H</sub>		31	23	12	12	54	44	42	12	5.0

Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
DECO (s, d, n)	n = 2		48	36	18	18	60	50	46	20	8.6
DECOP (s, d, n)	n = 8		62	47	24	24	80	65	61	27	12
ENCO (s, d, n)	n = 2	M1 = ON	52	39	20	20	66	55	51	21	9.1
		M4 = ON	52	39	20	20	66	54	51	21	9.1
	n = 8	M1 = ON	65	49	25	25	90	76	71	28	12
		M256 = ON	65	49	25	25	76	74	71	26	11
SEG SEGP			3.2	2.4	1.2	1.2	8.0	6.8	6.1	1.3	0.54
DIS (s, d, n)	n = 1		46	34	17	17	47	39	36	18	7.7
DISP (s, d, n)	n = 4		51	38	19	19	53	43	40	19	8.3
UNI (s, d, n)	n = 1		53	40	20	20	54	44	41	21	8.9
UNIP (s, d, n)	n = 4		57	43	22	22	60	49	46	23	9.7
NDIS (s1, d, s2)			104	78	39	39	92	76	38	41	18
NDISP (s1, d, s2)			104	78	39	39	92	76	38	41	18
NUNI (s1, d, s2)			105	79	40	40	47	39	36	42	18
NUNIP (s1, d, s2)			105	79	40	40	47	39	36	42	18
WTOB (s, d, n)	n = 1		125	94	47	47	56	46	42	47	20
WTOBP (s, d, n)	n = 96		257	193	97	97	190	155	145	99	43
BTOW (s, d, n)	n = 1		121	91	46	46	56	46	42	45	19
BTOWP (s, d, n)	n = 96		233	175	88	88	190	155	145	89	38
MAX (s, d, n)	n = 1		43	32	16	16	48	40	36	17	7.1
MAXP (s, d, n)	n = 96		318	239	120	120	300	240	235	136	59
MIN (s, d, n)	n = 1		43	32	16	16	48	40	36	17	7.1
MINP (s, d, n)	n = 96		436	326	163	163	300	240	235	159	69
DMAX (s, d, n)	n = 1		71	53	27	27	52	43	39	27	12
DMAXP (s, d, n)	n = 96		427	321	161	161	600	490	460	181	78
DMIN (s, d, n)	n = 1		71	53	27	27	52	43	39	27	12
DMINP (s, d, n)	n = 96		268	201	101	101	585	475	445	112	48
SORT (s1, n, s2, d1, d2)	n = 1		43	32	16	16	66	55	50	16	7.1
	n = 96		40*	30*	15*	15*	105	86	80	14	6.2
DSORT (s1, n, s2, d1, d2)	n = 1		44	33	17	17	98	57	52	17	7.1
	n = 96		43*	32*	16*	16*	115	96	88	16	6.8
* Indicates extension of scan time when instruction is completed											
WSUM (s, d, n)	n = 1		41.5	31.1	15.6	15.6	52	43	40	16.4	7.1
	n = 96		173.2	129.9	65	65	175	140	135	68.4	29.5
DWSUM (s, d, n)	n = 1		47.9	35.9	18	18	61	51	46	18.9	8.2
	n = 96		330	247.5	123.8	123.8	515	420	395	130.4	56.1
FOR	n = 0		5.2	3.9	2.0	2.0	11	8.9	8.1	2.3	1.0
NEXT			8.0	6.0	3.0	3.0	8.8	7.3	6.8	3.3	1.4

Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
BREAK BREAKP			26	19	9.5	9.5	37	30	28	11	4.6
CALL (pn) CALLP (pn)	internal file pointer		5.1	3.8	1.9	1.9	17	14	13	2.1	0.88
	common file pointer		85	64	32	32				33	14
CALL (pn s1 bis s5) CALLP (pn s1 bis s5)			348	261	131	131	245	200	190	135	58
RET	Return to origin program		7.5	5.6	2.8	2.8	16	13	12	2.9	1.3
	Return to other program		51	38	19	19	—	—	—	20	8.5
FCALL (pn) FCALLP (pn)	internal file pointer		8.8	6.6	3.3	3.3	29	24	22	3.6	1.6
	Common file pointer		48	36	18	18				20	8.7
FCALL (pn S1 bis S5) FCALLP (pn S1 bis S5)			388	254	127	127	250	205	190	134	57
ECALL (pn) ECALLP (pn)			187	140	70	70	—	—	—	77	33
ECALL (pn S1 bis S5) ECALLP (pn S1 bis S5)			515	387	144	144	—	—	—	162	70
EFCALL (pn) EFCALLP (pn)			188	141	71	71	—	—	—	78	34
EFCALL (pn S1 bis S5) EFCALLP (pn S1 bis S5)			516	388	194	194	—	—	—	200	86
COM			137	103	52	52	110	77	72	55	16
IX			31	23	12	12	65	54	51	12	5.2
IXEND			12	8.9	4.5	4.5	30	26	25	4.7	2.0
IXDEV	number of contacts: 1		127	95	46	46	145	120	110	48	21
	number of contacts: 14		238	179	85	85	770	630	585	93	40
IXSET	number of contacts: 1		127	95	46	46	145	120	110	48	21
	number of contacts: 14		238	179	85	85	770	630	585	93	40
FIFW FIFWP	number of data points: 1		27	20	10	10	36	32	28	11	4.5
	number of data points: 96										
FIFR FIFRP	number of data points: 1		34	25	13	13	45	41	36	13	5.6
	number of data points: 96		79	59	30	30	93	82	70	32	14
FPOP FPOPP	number of data points: 1		46	34	17	17	40	37	32	16	7.0
	number of data points: 96										
FINS FINSP	number of data points: 1		48	36	18	18	53	44	38	20	8.4
	number of data points: 96		96	72	36	36	100	89	76	36	15
FDEL FDELP	number of data points: 1		47	35	18	18	60	50	43	19	7.5
	number of data points: 96		97	73	37	37	110	95	82	39	15
PR	SM701 ON	1 character	83	62	31	31	—	—	—	33	11
		32 character	123	92	46	46				48	18
	SM701 OFF	54	40	20	20	21				7.8	
PRC			400	301	151	151	—	—	—	181	16
LED	displayed		223	167	84	84	—	—	—	—	—
	display completed		79	59	30	30					
LEDC	displayed		559	420	210	210	—	—	—	—	—
	display completed		413	310	155	155					

Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
LEDR	no display → no display		18	13	6.5	6.5	—	—	—	0.40	0.17
	Execute LED instruction → no display		205	154	77	77				103	44
CHKST			15	11	5.5	5.5	—	—	—	5.8	2.5
CHK (Fehler- kontrolle)	1 input contact	no error at contact 1	61	46	23	23	—	—	—	24	10
		150 input contacts	no error at contact 150	4232	3182	1591				1591	1676
	no error at contact 1		224	168	84	84				88	38
CHKCIR	10 failure check circuits		15	11	5.5	5.5	—	—	—	5.8	2.5
SLT	all internal devices		2399	1804	902	902	—	—	—	—	—
	file registers 8 kByte		7254	5454	2727	2727					
	completion of SLT instruction		15	11	5.5	5.5					
SLTR			1.1	0.8	0.4	0.4	—	—	—	—	—
STRA	Start		47	35	18	18	—	—	—	—	—
	completion of STRA instruction		15	11	5.5	5.5					
STRAR			1.1	0.8	0.4	0.4	—	—	—	—	—
PTRA			15	11	5.5	5.5	—	—	—	—	—
PTRAR			15	11	5.5	5.5	—	—	—	—	—
PTRAEXE	instruction execution		1.6	1.2	0.6	0.6	—	—	—	—	—
PTRAEXEP	program trace		169	127	64	64					
BINDA	BINDAP	s = 1	40	30	15	15	—	—	—	15	6.7
		s = -32768	60	45	23	23				24	10
DBINDA	DBINDAP	s = 1	63	47	44	44	—	—	—	43	18
		s = -2147483648	217	163	82	82				86	37
BINHA	BINHAP	s = 1	46	34	17	17	—	—	—	18	7.7
		s = FFFF <sub>H</sub>	48	36	18	18				19	8.2
DBINHA	DBINHAP	s = 1	59	44	22	22	—	—	—	23	10
		s = FFFFFFFF <sub>H</sub>	62	46	23	23				24	10
BCDDA	BCDDAP	s = 1	58	43	22	22	—	—	—	23	9.8
		s = 9999	54	40	20	20				21	8.9
DBCDDA	DBCDDAP	s = 1	61	46	23	23	—	—	—	22	9.5
		s = 99999999	75	56	28	28				29	13
DABIN	DABINP	s = 1	133	100	50	50	—	—	—	57	25
		s = -32768	145	109	55	55				58	28
DDABIN	DDABINP	s = 1	241	181	91	91	—	—	—	92	40
		s = -2147483648	268	201	101	101				106	46
HABIN	HABINP	s = 1	32	24	12	12	—	—	—	13	5.8
		s = FFFF <sub>H</sub>	38	28	14	14				15	6.4
DHABIN	DHABINP	s = 1	54	40	20	20	—	—	—	22	9.5
		s = FFFFFFFF <sub>H</sub>	63	47	24	24				25	11
DABCD	DABCDP	s = 1	36	27	14	14	—	—	—	16	6.9
		s = 9999	42	31	16	16				17	7.2
DDABCD	DDABCDP	s = 1	63	47	24	24	—	—	—	25	11
		s = 99999999	75	56	28	28				29	13
COMRD COMRDP			36	27	14	14	—	—	—	40	16
LEN LENP	1 character		48	36	18	18	—	—	—	18	8.0
	96 characters		229	172	86	86				86	37
STR STRP			132	99	50	50	—	—	—	53	23
DSTR DSTRP			285	214	107	107	—	—	—	123	53
VAL VALP			258	194	97	97	—	—	—	95	41
DVAL DVALP			402	302	151	151	—	—	—	166	72

Instruction	Processing (Device)		Processing time (μs)									
			QnA series CPU modules				System Q CPU modules					
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
ESTR ESTRP			1337	1005	503	503	—	—	—	564	243	
EVAL EVALP		floating point format	242	182	91	91	—	—	—	100	43	
		exponential format	306	230	115	115				127	55	
ASC (s, d, n) ASCP (s, d, n)		n = 1	164	123	62	62	—	—	—	64	28	
		n = 96	780	586	293	293				289	125	
HEX (s, d, n) HEXP (s, d, n)		n = 1	161	121	61	61	—	—	—	60	26	
		n = 96	826	621	311	311				343	148	
RIGHT (s, d, n) RIGHTP (s, d, n)		n = 1	131	98	49	49	—	—	—	49	21	
		n = 96	354	266	133	133				131	56	
LEFT (s, d, n) LEFTP (s, d, n)		n = 1	129	97	49	49	—	—	—	50	21	
		n = 96	354	266	133	133				131	56	
MIDR MIDRP			141	106	53	53	—	—	—	53	23	
MIDW MIDWP			341	256	128	128	—	—	—	128	55	
INSTR INSTRP		no match	156	117	59	59	—	—	—	58	25	
		match	first	141	106	53				53	55	24
			finals	155	116	58				58	58	25
EMOD EMODP			1313	987	494	494	—	—	—	527	227	
EREXP EREXP			4423	3325	1663	1663	—	—	—	1656	713	
SIN SINP		single precision	4921	3700	1850	35	—	—	—	115	50	
		double precision	—	—	—	—				1945	837	
COS COSP		single precision	6462	4858	2429	35	—	—	—	122	53	
		double precision	—	—	—	—				2618	1127	
TAN TANP		single precision	6515	4898	2449	38	—	—	—	123	53	
		double precision	—	—	—	—				2618	1127	
ASIN ASINP		single precision	890	669	335	44	—	—	—	111	48	
		double precision	—	—	—	—				2491	1072	
ACOS ACOSP		single precision	801	602	301	44	—	—	—	115	49	
		double precision	—	—	—	—				2367	1019	
ATAN ATANP		single precision	7818	5878	2939	39	—	—	—	157	68	
		double precision	—	—	—	—				3140	1352	
RAD RADP		single precision	465	349	175	31	—	—	—	17	7.2	
		double precision	—	—	—	—				24	10	
DEG DEGP		single precision	492	369	185	31	—	—	—	17	7.2	
		double precision	—	—	—	—				23	9.9	
SQR SQRP		single precision	4520	3398	1699	39	—	—	—	28	12	
		double precision	—	—	—	—				1812	780	
EXP EXPP	single precision	s = -10	5871	4414	2207	37	—	—	—	—	—	
		s = 1	5950	4474	2237					129	56	
	double precision	s = -10	—	—	—	—				—	—	
		s = 1	—	—	—	—				2386	1026	
LOG LOGP	single precision	s = 1	1191	896	448	37	—	—	—	113	49	
		s = 10	6839	5142	2571					—	—	
	double precision	s = 1	—	—	—	—				2146	924	
		s = 10	—	—	—	—				—	—	
RND RNDP			10	7.5	3.8	3.8	—	—	—	3.9	1.7	

Instruction	Processing (Device)	Processing time (μs)								
		QnA series CPU modules				System Q CPU modules				
		Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
SRND SRNDP		8.8	6.6	3.3	3.3	—	—	—	3.5	1.5
BSQR BSQRP	s = 0	16	12	6.0	6.0	—	—	—	6.2	2.7
	s = 9999	97	73	37	37				38	16
BDSQR BDSQRP	s = 0	17	13	6.5	6.5	—	—	—	6.2	2.7
	s = 99999999	88	66	33	33				38	16
BSIN BSINP		30	22	11	11	—	—	—	12	5.1
BCOS BCOSP		32	24	12	12	—	—	—	12	5.2
BTAN BTANP		30	22	11	11	—	—	—	12	5.2
BASIN BASINP		52	39	20	20	—	—	—	20	8.7
BACOS BACOSP		53	40	20	20	—	—	—	21	9.0
BATAN BATANP		56	42	21	21	—	—	—	22	9.6
LIMIT LIMITP		24	18	9.0	9.0	34	28	26	10	4.3
DLIMIT DLIMITP		28	21	11	11	41	34	30	11	4.7
BAND BANDP		24	18	9.0	9.0	33	28	25	9.8	4.2
DBAND DBANDP		28	21	11	11	40	34	30	11	4.9
ZONE ZONEP		24	18	9.0	9.0	31	25	24	9.1	3.9
DZONE DZONEP		28	21	11	11	37	29	28	11	4.6
RSET RSETP		19	14	7.0	7.0	—	18	16	6.8	2.9
QDRSET QDRSETP		322	242	121	121	—	—	—	205	88
QCDSET QCDSETP		218	164	82	82	—	—	—	147	63
DATERD DATERDP		36	27	14	14	30	25	23	13	5.5
DATEWR DATEWRP		42	31	16	16	69	57	54	15	6.4
DATE+ DATE+P	no digit increase	60	45	23	23	47	39	36	13	5.4
	digit increase	60	45	23	23	50	42	38	13	5.4
DATE- DATE-P	no digit increase	59	44	22	22	47	40	36	12	5.2
	digit increase	60	45	23	23	50	42	38	12	5.2
SECOND SECONDP		27	20	10	10	28	24	22	10	4.5
HOUR HOURP		31	23	12	12	38	32	29	12	5.2
MSG	1 character	7.2	5.4	2.7	2.7	—	—	—	3.0	1.3
	32 characters	7.4	5.6	2.8	2.8					
PKEY	initial time	51	38	19	19	—	—	—	20	8.6
	no acceptance	48	36	18	18				19	8.2
PSTOP PSTOPP		122	92	46	46	—	—	—	79	34
POFF POFFP		120	90	45	45	—	—	—	79	34
PSCAN PSCANP		122	92	46	46	—	—	—	75	32

Instruction	Processing (Device)		Processing time (μs)								
			QnA series CPU modules				System Q CPU modules				
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH
PLOW PLOWP			124	93	47	47	—	—	—	80	34
WDT WDTP			12	8.7	4.4	4.4	18	15	14	5.9	2.6
DUTY			1.6	1.2	0.6	0.6	41	36	32	9.3	4.0
ZRRDB ZRRDBP			19	14	6.9	6.9	—	24	22	7.9	3.4
ZRWRB ZRWRBP			21	16	7.8	7.8	—	27	24	9.4	4.0
ADRSET ADRSETP			13	9.3	4.7	4.7	23	19	18	4.9	2.1
KEY			43.4	32.4	16.2	16.2	—	—	—	17	7.3
ZPUSH ZPUSHP			27.6	20.6	10.3	10.3	38	33	30	11	4.7
ZPOP ZPOPP			12.7	9.5	4.8	4.8	37	31	29	5.1	2.2
EPROMWR EPROMWRP			62.6	46.7	23.4	23.4	—	—	—	—	—
ZCOM			4296.6	3206.4	1603.2	1603.2	105	82	80	691	289
READ			770.6	575.1	287.6	287.6	—	—	—	554	260
SREAD			858.9	641.0	320.5	320.5	—	—	—	588	278
WRITE			791.9	591.0	295.5	295.5	—	—	—	582	273
SWRITE			848.6	633.3	316.6	316.6	—	—	—	625	295
SEND			575.7	429.6	214.8	214.8	—	—	—	—	—
RECV			375.9	280.5	140.3	140.3	—	—	—	—	—
REQ			527.4	393.6	196.8	196.8	—	—	—	—	—
ZNFR			982.1	732.9	366.5	366.5	—	—	—	—	—
ZNTO			989.3	738.3	369.2	369.2	—	—	—	—	—
ZNRD		MELSECNET/10	598.6	446.7	223.4	223.4	—	—	—	—	—
		MELSECNET (II)	649.2	484.5	242.3	242.3	—	—	—	—	—
ZNWR		MELSECNET/10	614.3	458.4	229.2	229.2	—	—	—	—	—
		MELSECNET (II)	665.6	496.7	248.4	248.4	—	—	—	—	—
RFRP			590.9	441.0	220.5	220.5	—	—	—	—	—
RTOP			588.8	439.4	219.7	219.7	—	—	—	—	—
S.STMODE			—	—	—	22.6	—	—	—	—	—
S.CGMODE			—	—	—	19.4	—	—	—	—	—
S.TRUCK			—	—	—	43.7	—	—	—	—	—
S.SPREF		Buffer memory head adress = 0 (when A68AD is used)	—	—	—	407.1	—	—	—	—	—
		Buffer memory head adress = 0 (when A68DA is used)	—	—	—	331.4	—	—	—	—	—
UNIRD		n = 1	—	—	—	—	96	80	74	79	34
		n = 16	—	—	—	—	440	370	340		
TRACE		Start	—	—	—	—	—	—	—	176	76
		completion of the TRACE instruction	—	—	—	—	—	—	—	6.3	2.7
TRACER			—	—	—	—	—	—	—	19	8.2
FWRITE			—	—	—	—	—	—	—	84	36
FREAD			—	—	—	—	—	—	—	79	34
PLOADP			—	—	—	—	—	—	—	58	25
PUNLOADP			—	—	—	—	—	—	—	272	117
PSWAPP			—	—	—	—	—	—	—	308	133
RBMOV	Übertragungszeit	1 Wort	—	—	—	—	—	—	—	69	29
		1000 Worte	—	—	—	—	—	—	—	580	308

Instruction	Processing (Device)		Processing time (μs)									
			QnA series CPU modules				System Q CPU modules					
			Q2A Q2AS	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2	QnH	
COM <sup>1</sup>	With automatic refresh of CPU shared memory	Refresh range = 2 k words (0.5 k words for each CPU)	—	—	—	—	—	—	—	720	660	
		Refresh range = 4 k words (1 k words for each CPU)	—	—	—	—	—	—	—	860	730	
	Without automatic refresh of CPU shared memory		—	—	—	—	—	—	—	43	20	
FROM <sup>1</sup>	Read from shared memory of another CPU	n3 = 0	—	—	—	—	—	—	—	59	29	
		n3 = 1000	—	—	—	—	—	—	—	530	500	
	Read from buffer memory of a special function module <sup>2</sup>	n3 = 1	main base unit	—	—	—	—	—	—	—	51	24
			extension base unit	—	—	—	—	—	—	—	54	27
		n3 = 1000	main base unit	—	—	—	—	—	—	—	540	480
			extension base unit	—	—	—	—	—	—	—	1100	1050
S.TO	s2 = 1		—	—	—	—	—	—	—	74	33	
	s2 = 256		—	—	—	—	—	—	—	126	54	

<sup>1</sup> When the instruction is executed from several CPUs of a multi-CPU simultaneously, the processing time will be increased. The following formula is used to calculate the instruction processing time increase.  
 For a system which consists of a base unit only:  
 Instruction processing time increase [μs] = 0,54 x (number of points processed) x (number of CPU modules)

For a system which consists of a base unit only and extension base units:  
 Instruction processing time increase [μs] = 1,30 x (number of points processed) x (number of CPU modules)

<sup>2</sup> The instruction processing time for special function modules under control of the CPU which is executing the instruction is identical to the instruction processing time for special function modules under control of another CPU of the multi-CPU system.

Instruction	Processing (Device)		Processing time (μs)							
			QnA series CPU module				System Q CPU module			
			Q2A	Q3A	Q4A QnASH	Q4AR	Q00J	Q00	Q01	Q2
FROM (n1, n2, d, n3) FROMP (n1, n2, d, n3)	n3 = 1	253 <sup>1</sup>	217 <sup>1</sup>	160 <sup>1</sup>	160 <sup>1</sup>	—	—	—	—	—
		252 <sup>2</sup>	210 <sup>2</sup>	154 <sup>2</sup>	154 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	125 <sup>3</sup>	105 <sup>3</sup>	93 <sup>3</sup>	47 <sup>4</sup>	22 <sup>4</sup>
DFRO (n1, n2, d, n3) DFROP (n1, n2, d, n3)	n3 = 1000	4514 <sup>1</sup>	4286 <sup>1</sup>	4150 <sup>1</sup>	4150 <sup>1</sup>	—	—	—	—	—
		2855 <sup>2</sup>	2127 <sup>2</sup>	2038 <sup>2</sup>	2038 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	740 <sup>3</sup>	695 <sup>3</sup>	685 <sup>3</sup>	476 <sup>4</sup>	437 <sup>4</sup>
TO (n1, n2, d, n3) TOP (n1, n2, d, n3)	n3 = 1	260 <sup>1</sup>	221 <sup>1</sup>	165 <sup>1</sup>	165 <sup>1</sup>	—	—	—	—	—
		257 <sup>2</sup>	214 <sup>2</sup>	156 <sup>2</sup>	156 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	130 <sup>3</sup>	110 <sup>3</sup>	100 <sup>3</sup>	51 <sup>4</sup>	24 <sup>4</sup>
DTO (n1, n2, d, n3) DTOP (n1, n2, d, n3)	n3 = 500	4543 <sup>1</sup>	4271 <sup>1</sup>	4082 <sup>1</sup>	4082 <sup>1</sup>	—	—	—	—	—
		2883 <sup>2</sup>	2129 <sup>2</sup>	2064 <sup>2</sup>	2064 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	745 <sup>3</sup>	695 <sup>3</sup>	675 <sup>3</sup>	478 <sup>4</sup>	437 <sup>4</sup>
TO (n1, n2, d, n3) TOP (n1, n2, d, n3)	n3 = 1000	276 <sup>1</sup>	217 <sup>1</sup>	162 <sup>1</sup>	162 <sup>1</sup>	—	—	—	—	—
		254 <sup>2</sup>	211 <sup>2</sup>	154 <sup>2</sup>	154 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	120 <sup>3</sup>	105 <sup>3</sup>	92 <sup>3</sup>	48 <sup>4</sup>	20 <sup>4</sup>
DTO (n1, n2, d, n3) DTOP (n1, n2, d, n3)	n3 = 1	4500 <sup>1</sup>	4319 <sup>1</sup>	4188 <sup>1</sup>	4188 <sup>1</sup>	—	—	—	—	—
		2878 <sup>2</sup>	2155 <sup>2</sup>	2043 <sup>2</sup>	2043 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	735 <sup>3</sup>	680 <sup>3</sup>	645 <sup>3</sup>	479 <sup>4</sup>	412 <sup>4</sup>
DTO (n1, n2, d, n3) DTOP (n1, n2, d, n3)	n3 = 500	260 <sup>1</sup>	221 <sup>1</sup>	165 <sup>1</sup>	165 <sup>1</sup>	—	—	—	—	—
		257 <sup>2</sup>	216 <sup>2</sup>	157 <sup>2</sup>	157 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	130 <sup>3</sup>	110 <sup>3</sup>	99 <sup>3</sup>	50 <sup>4</sup>	23 <sup>4</sup>
DTO (n1, n2, d, n3) DTOP (n1, n2, d, n3)	n3 = 1000	4471 <sup>1</sup>	4315 <sup>1</sup>	4198 <sup>1</sup>	4198 <sup>1</sup>	—	—	—	—	—
		2819 <sup>2</sup>	2172 <sup>2</sup>	2062 <sup>2</sup>	2062 <sup>2</sup>	—	—	—	—	—
		—	—	—	—	740 <sup>3</sup>	680 <sup>3</sup>	640 <sup>3</sup>	457 <sup>4</sup>	416 <sup>4</sup>

<sup>1</sup> These are the processing times when a A38B/A1S38B main base is used in combination with an extension base.

<sup>2</sup> These processing times are for a A38HB/A1S38HB main base.

<sup>3</sup> The instruction processing time depends on the type of extension base used, the number of slots at the base unit and the number of modules actually installed.

<sup>4</sup> These processing times are for a Q312B main base unit and the QJ71C24 installed at slot 0.



**A.2.2 Processing times of the A series CPUs**

The processing time of an instruction depends on the applied processing mode for the input and output signals:

- Direct I/O control mode = ○
- Refresh I/O control mode = ○○

Instruction	Processing (Devices)		Processing time (μs)							
			A	AnN, AnS		A2A, A2AS	A3A			
			○	○○	○	○○	○○			
				no X, Y	with X, Y					
LD	X		2.3	1.0	2.3	0.2	0.15			
LDI										
AND										
ANI	Y, M, L, B, F, T, C		1.3	1.0	1.0	0.2	0.15			
OR										
ORI										
ANB										
ORB										
MPS										
MRD										
MPP										
OUT	Y	not changed	2.3	1.0	2.3	0.4	0.3			
		changed (OFF/ON)	2.3	1.0	2.3	0.4	0.3			
	M (excl. special M)	not changed	1.3	1.0	1.0	0.4	0.3			
		changed (OFF/ON)	1.3	1.0	1.0	0.4	0.3			
	special M			37	37	0.8	0.6			
	F	not executed		66	AnN: 61 AnS: 62	61	6.6	5.0		
		executed	displayed	700	AnN: 633 AnS: 270	AnN: 633 AnS: 267	99	74		
	display completed									
	T	Execution time of instruction		1.3	1.0	1.0	0.4	0.3		
		END processing	not executed		1.3	0	0	0.23	0.18	
			executed	after time out	15	11	11	4.5	3.3	
				added	K	30	24	24	7.7	5.7
					D	36	30	30	8.3	6.2
		C	Execution time of instruction		1.3	1.0	1.0	0.4	0.3	
			END processing	not executed		1.3	0	0	0.27	0.2
				executed	nicht gezählt	14	0	0	0.27	0.2
					after time out	14	0	0	0.27	0.2
					added	K	28	25	25	4.2
				D	33	30	30	4.8	3.6	
	SET	Y	not executed		2.3	1.0	2.3	0.4	0.3	
executed			not changed	2.3	1.0	2.3	0.4	0.3		
			changed	2.3	1.0	2.3	0.4	0.3		
M, L, S, B		not executed		3.7	1.0	1.0	0.4	0.3		
		executed	not changed	41	1.0	1.0	0.4	0.3		
			changed	41	1.0	1.0	0.4	0.3		
special M, B		not executed			2.0	3.0	0.8	0.6		
		executed			32	32	0.8	0.6		
F		not executed		3.7	AnN: 3.0 AnS: 2.7	AnN: 3.0 AnS: 3.2	2.0	1.5		
		executed	displayed	730	AnN: 638 AnS: 232	AnN: 638 AnS: 237	99	74		
display completed										

Instruction	Processing (Devices)		Processing time (μs)						
			A	AnN, AnS		A2A, A2AS	A3A		
				no X, Y	with X, Y				
RST	Y	not executed	2.3	1.0	2.3		0.4	0.3	
		executed	not changed	2.3	1.0	2.3		0.4	0.3
			changed (OFF/ON)	2.3	1.0	2.3		0.4	0.3
	M, L, S, B	not executed	3.7	1.0	1.0		0.4	0.3	
		executed	not changed	41	1.0	1.0		0.4	0.3
			changed (OFF/ON)	41	1.0	1.0		0.4	0.3
	Special M, B	not executed		3.0	3.0		0.8	0.6	
		executed		32	32		0.8	0.6	
	F	not executed	3.7	AnN: 3.0 AnS: 3.6	3.0		2.0	1.5	
		executed	displayed	680	AnN: 477 AnS: 296	AnN: 477 AnS: 283		150	112
			display completed						
	T, C	not executed	3.7	3.0	3.0		1.4	1.1	
		executed	57	43	43		5.6	4.2	
	D, W, A0, A1, V, Z	not executed	3.7	3.0	3.0		1.4	1.1	
		executed	34	28	28		8.4	6.3	
R	not executed	3.7	3.0	3.0		1.4	1.1		
	executed	41	35	35		4.6	3.5		
PLS PLF	Y	not executed	65	59	61		2.2	1.7	
		executed	ON	68	62	63		2.2	1.7
			OFF	64	60	62		2.2	1.7
	M, L, B, F	not executed	64	59	59		2.2	1.7	
		executed	ON	67	62	62		2.2	1.7
			OFF	63	61	61		2.2	1.7
SFT SFTP	Y	not executed	3.7	3.0	3.0		1.4	1.1	
		executed	49	38	39		4.4	3.3	
	M, L, B, F	not executed	3.7	3.0	3.0		1.4	1.1	
		executed	48	38	38		4.4	3.3	
MC	Y	not executed	85	43	44		1.2	0.9	
		executed	50	39	41		1.2	0.9	
	M, L, S, B, F	not executed	84	43	43		1.2	0.9	
		executed	49	39	39		1.2	0.9	
MCR		35	26	26		0.6	0.45		
FEND	M9084 = OFF	2400	2150	2150		435	327		
END	M9084 = ON	2400	2060	2060		285	214		
NOP		1.3	1.0	1.0		0.2	0.15		
LD=		95	70	70	87	3.8	2.9		
AND=		96	61	62	81	2.6	2.0		
OR=		94	67	66	85	2.8	2.1		
LD<>		98	69	69	86	4.1	3.1		
AND<>		92	60	60	79	2.6	2.0		
OR<>		96	66	66	84	2.8	2.1		
LD>		96	67	67	84	4.1	3.1		
AND>		92	60	60	79	2.6	2.0		
OR>		98	66	65	83	2.8	2.1		
LD<=		100	71	71	88	4.1	3.1		
AND<=		94	61	61	80	2.6	2.0		
OR<=		100	69	68	86	2.8	2.1		
LD<		96	69	69	86	4.1	3.1		
AND<		92	59	60	79	2.6	2.0		
OR<		96	66	65	84	2.8	2.1		
LD>=		100	71	71	88	4.1	3.1		

Instruction	Processing (Devices)	Processing time (μs)					
		A ○	AnN, AnS		A2A, A2AS ○○	A3A ○○	
			○○	○			
				no X, Y			with X, Y
AND>=		94	61	61	81	2.6	2.0
OR>=		100	69	68	86	2.8	2.1
LDD=		238	133	134	119	10	7.7
ANDD=		231	124	125	210	5.9	4.4
ORD=		236	133	133	218	6.3	4.7
LDD<>		235	131	132	217	10	7.7
ANDD<>		239	129	129	215	5.9	4.4
ORD<>		234	129	129	214	6.1	4.6
LDD>		238	133	133	219	9.7	7.3
ANDD>		240	131	131	217	5.8	4.4
ORD>		236	131	130	219	6.0	4.5
LDD<=		244	137	136	222	9.7	7.3
ANDD<=		238	127	128	213	5.8	4.4
ORD<=		246	137	136	221	6.0	4.5
LDD<		238	133	133	219	9.7	7.3
ANDD<		241	131	131	217	5.8	4.4
ORD<		236	131	130	215	6.0	4.5
LDD>=		243	137	137	222	9.7	7.3
ANDD>=		238	127	128	213	5.8	4.4
ORD>=		246	137	136	221	6.0	4.5
+ (s,d) +P (s,d)		72	44	45	59	2.8	2.1
+ (s1,s2,d) +P (s1,s2,d)		112	77	77	103	3.2	2.4
-(s,d) -P (s,d)		74	45	45	59	2.8	2.1
- (s1,s2,d) -P (s1,s2,d)		123	79	79	107	3.2	2.4
D+ (s,d) D+P (s,d)		110	69	69	90	4.0	3.0
D+ (s1,s2,d) D+P (s1,s2,d)		140	99	99	246	4.6	3.5
D-(s,d) D-P(s,d)		110	69	69	90	4.0	3.0
D- (s1,s2,d) D-P (s1,s2,d)		141	99	99	130	4.6	3.5
x (s1,s2,d) xP (s1,s2,d)		135	94	95	168	3.4	2.6
/ (s1,s2,d) /P (s1,s2,d)		144	102	103	99	11	8.6
Dx (s1,s2,d) DxP (s1,s2,d)		429	341	340	370	20	15
D/ (s1,s2,d) D/P (s1,s2,d)		289	393	394	412	36	27

Instruction	Processing (Devices)	Processing time (μs)					
		A	AnN, AnS		A2A, A2AS	A3A	
			no X, Y	with X, Y			
B+ (s,d) B+P (s,d)		210	123	123	183	6.4	4.8
B+ (s1,s2,d) B+P (s1,s2,d)		217	129	129	192	6.2	4.7
B- (s,d) B-P (s,d)		210	125	125	185	34	25
B- (s1,s2,d) B-P (s1,s2,d)		212	133	133	203	32	23
DB+ (s,d) DB+P (s,d)		320	175	176	280	14	11
DB+ (s1,s2,d) DB+P (s1,s2,d)		321	187	186	294	14	11
DB- (s,d) DB-P (s,d)		318	175	175	280	31	23
DB- (s1,s2,d) DB-P (s1,s2,d)		322	185	186	294	29	22
Bx (s1, s2, d) BxP (s1, s2, d)		410	299	300	358	14	11
B/ (s1, s2, d) B/P (s1, s2, d)		422	235	236	274	89	67
DBx (s1, s2, d) DBxP (s1, s2, d)		1158	941	939	1044	11	8.0
DB/ (s1, s2, d) DB/P (s1,s2,d)		998	896	894	954	62	47
INC INCP		46	29	29	38	2.0	1.5
DINC DINCP		66	42	42	132	2.4	1.8
DEC DECP		48	31	31	39	2.0	1.5
DDEC DDECP		66	42	42	54	2.4	1.8
BCD BCDP		110	82	83	90	3.0	2.3
DBCD DBCDP		329	219	220	284	13	9.5
		329	219	220	284	13	9.5
BIN BINP		104	79	78	86	3.0	2.3
DBIN DBINP		311	215	216	280	6.0	4.5
NEG NEGP		105	50	49	86	8.6	6.5

Instruction	Processing (Devices)	Processing time (μs)					
		A	AnN, AnS			A2A, A2AS	A3A
			○	○			
				no X, Y	with X, Y		
MOV MOV P		72	47	47	57	17	13
DMOV DMOV P		104	67	67	87	20	15
CML CML P		68	43	43	57	2.4	1.8
DCML DCML P		130	74	75	108	3.2	2.4
BMOV (s1, s2, n) BMOV P (s1, s2, n)		7498	399	400	7144	1444	1083
FMOV (s1, s2, n) FMOV P (s1, s2, n)		1118	229	228	1029	1427	1070
XCH XCH P		102	60	61	84	2.8	2.1
DXCH DXCH P		170	107	107	141	4.2	3.2
CJ	no index qualification	49	39	39		6.6	5.0
	index qualification		48	48		6.6	5.0
SCJ	no index qualification	54	71	71		6.6	5.0
	index qualification		81	81		6.6	5.0
JMP		50	39	39		6.6	5.0
EI		195	38	38		3.0	2.3
DI		46	66	66		3.2	2.4
IRET		249	120	120		3.4	2.6
WAND (s, d) WAND P (s, d)		90	60	59	72	2.8	2.1
WAND (s1, s2, d) WAND P (s1, s2, d)		179	96	96	152	7.6	5.7
DAND (s, d) DAND P (s, d)		276	140	139	240	13	9.5
DAND (s1, s2, d) DAND P (s1, s2, d)							
WOR (s, d) WOR P (s, d)		90	61	60	72	2.8	2.1
WOR (s1, s2, d) WOR P (s1, s2, d)		176	97	96	152	7.6	5.7
DOR (s, d) DOR P (s, d)		276	140	139	240	13	9.5
DOR (s1, s2, d) DOR P (s1, s2, d)							

Instruction	Processing (Devices)	Processing time (μs)					
		A	AnN, AnS		A2A, A2AS	A3A	
		○	○○	○	○○	○○	
		no X, Y	with X, Y				
WXOR (s, d) WXORP (s, d)		91	60	59	72	2.8	2.1
WXOR (s1, s2, d) WXORP (s1, s2, d)		178	97	96	152	7.6	5.7
DXOR (s, d) DXORP (s, d)		274	140	139	240	13	9.5
DXOR (s1, s2, d) DXORP (s1, s2, d)							
WXNR (s, d) WXNRP (s, d)		89	64	62	74	3.0	2.3
WXNR (s1, s2, d) WXNRP (s1, s2, d)		177	98	96	152	7.8	5.9
DXNR (s,d) DXNRP (s,d)		277	142	140	241	15	11
DXNR (s1,s2,d) DXNRP (s1,s2,d)							
ROR (n) RORP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	66	52	51	51	5.8	4.4
RCR (n) RCRP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	74	59	59	59	6.4	4.8
ROL (n) ROLP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	68	54	53	53	5.8	4.4
RCL (n) RCLP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	74	57	57	57	6.4	4.8
DROR (n) DRORP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	97	70	70	69	11	8.3
DRCR (n) DRCRP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	95	72	72	72	12	9.2
DROL (n) DROLP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	101	70	69	69	10	7.8
DRCL (n) DRCLP (n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	98	68	68	68	12	8.7

Instruction	Processing (Devices)	Processing time (μs)					
		A	AnN, AnS		A2A, A2AS	A3A	
			no X, Y	with X, Y			
SFR (s, n) SFRP (s, n)	n = 3/5 (n = 5 for all AnN CPUs, n = 3 for all CPUs except for the AnN CPU)	102	74	72	83	5.0	3.8
SFL (d, n) SFLP (d, n)	n = 5	106	74	73	84	4.8	3.6
BSFR (d, n) BSFRP (d, n)	n = 5	145	124	123	124	29	22
BSFL (d, n) BSFLP (d, n)	n = 5	158	134	133	134	28	21
DSFR (d, n) DSFRP (d, n)	n = 5	133	118	116	—	18.8	14.1
DSFL (d, n) DSFLP (d, n)	n = 5	134	118	17	—	22	17
BSET (d, n) BSETP (d, n)	n = 5	107	90	90	—	9.6	7.2
BRST (d, n) BRSTP (d, n)	n = 5	114	97	96	—	9.6	7.2
SER (s1, s2, d, n) SERP (s1, s2, d, n)	n = 5	230	200	200	—	33	25
SUM SUMP		164	115	114	131	15	11
DSUM DSUMP		267	200	199	231	34	25
DECO (s, d, n) DECOP (s, d, n)	n = 2	249	164	163	216	32	24
ENCO (s, d, n) ENCOP (s, d, n)	n = 2	478	164	163	195	41	31
SEG		170	91	91	155	6.4	4.8
DIS (s, d, n)	n = 4	180	154	153	—	25	19
DISP (s, d, n)	n = 4	180	154	153	120	25	19
UNI (s, d, n)	n = 4	159	131	131	—	31	24
UNIP (s, d, n)	n = 4	159	131	131	—	31	24
FOR		64	53	53		5.8	4.4
NEXT		2532	41	41		8.0	6.0
CALL (pn)	no index qualification	74	74	74		10	7.8
	index qualification		78	78		10	7.8
CALLP (pn)	no index qualification	74	70	70		10	7.8
	index qualification		78	78		10	7.8
RET		249	50	50		7.0	5.3

Instruction	Processing (Devices)	Processing time (μs)					
		A	AnN, AnS		A2A, A2AS	A3A	
		○	○	○	○	○	
		no X, Y	with X, Y				
CHG	M9084 = OFF	8546	2420	2420		—	—
	M9084 = ON	—	2340	2340		—	—
SUB SUBP	no index qualification	90	79	79		—	—
	index qualification	—	85	85		—	—
FIFW FIFWP		340	101	101	123	20	15
FIFR FIFRP		202	118	118	134	69	52
PR		—	226	226	226	74	56
PRC		—	141	141	141	37	28
LED		170	203	203	203	100	75
LEDC		210	265	265	265	142	106
LEDR		520	638	638	638	106	80
LEDA		170	202	202	202	—	—
LEDB		172	211	211	211	—	—
CHK (error check)	1 input contact	—	—	AnN: 771 AnS: 240		33	25
	50 input contacts	—	—	AnN: 3380 AnS: 3905		1257	943
	100 input contacts	—	—	AnN: 6687 AnS: 7820		2503	1877
	150 input contacts	—	—	AnA: 10137 AnS: 11470		3753	2815
CHK (generate flip-flop)		—	121	121	121	—	—
SLT	device memory	—	8448	8448	8448	2915	2186
	device memory + R	—	24598	24598	24598	9996	7497
SLTR		—	29	29	29	6.6	5.0
STRA		—	30	30	30	5.0	3.8
STRAR		—	28	28	28	5.0	3.8
STC		—	28	28	28	2.4	1.8
CLC		—	31	31	31	2.4	1.8
ASC		140	120	120	120	3.4	2.6
WDT WDTP		—	64	64	64	5.0	3.8
DUTY		—	68	68	68	14	11
LRDP (n1, s, d, n2)	n2 = 1	—	190	190	190	42	32
	n2 = 32	—	190	190	190	42	32
LWTP (n1, d, s, n2)	n2 = 1	—	200	200	200	49	37
	n2 = 32	—	446	446	446	89	66
RFRP (n1, n2, d, n3)	n3 = 1	—	172	172	172	32	24
	n3 = 32	—	172	172	172	63	47
RTOP (n1, n2, s, n3)	n3 = 1	—	176	176	176	68	51
	n3 = 32	—	176	176	176	34	26



Instruction	Processing (Devices)	Processing times (μs)						
		A	AnN, AnS		A2A, A2AS, A2J		A3A, A3U	
		○	○ + ○		○		○	
		—	no X, Y	with X, Y	no X, Y	with X, Y	no X, Y	with X, Y
FROM (n1, n2, d, n3)	n3 = 1	—	439	524	237	261	178	196
FROMP (n1, n2, d, n3)	n3 = 1000/112 <sup>1</sup>	—	6609	2358	5749	2789	4312	2092
DFRO (n1, n2, d, n3)	n3 = 1	—	449	529	244	266	183	199
DFROP (n1, n2, d, n3)	n3 = 500/56 <sup>2</sup>	—	6609	2109	5669	1669	4252	1252
TO (n1, n2, d, n3)	n3 = 1	—	449	539	243	266	182	200
TOP (n1, n2, d, n3)	n3 = 1000/112 <sup>1</sup>	—	6609	3918	5773	2117	4330	1588
DTO (n1, n2, d, n3)	n3 = 1	—	454	544	240	266	180	199
DTOP (n1, n2, d, n3)	n3 = 500/56 <sup>2</sup>	—	6609	1609	5747	1501	4310	1126

<sup>1</sup> CPUs without X and Y: n3 = 1000; other CPUs with X and Y: n3 = 112

<sup>2</sup> CPUs without X and Y: n3 = 500; other CPUs with X and Y: n3 = 56

### A.3 Comparison of the CPUs

The following table contains the characteristics, i.e. available devices, processing modes, special relays, etc. of the different CPUs (System Q, Q4AR, QnA, AnU, AnA, AnN, AnS).

#### A.3.1 Available devices

Device	System Q		Q series		A series			
	Q00J, Q00, Q01	Qn, QnH, QnPH	Q4AR	QnA	AnU	AnA	AnN	AnS
Number of inputs/outputs	Q00J: 256 Q00: 1024 Q01: 1024	4096 for Q02, Q02H, Q06H, Q12H and Q25H	4096	Q2A: 512 Q2A-S1: 1024 Q3A: 2048 Q4A: 4096	A2U: 512 A2U-S1: 1024 A3U: 2048 A4U: 4096	A2A: 512 A2A-S1:1024 A3A: 2048	A1N: 256 A2N: 512 A2N-S: 1024 A3N: 2048	A1S: 256 A1S-S1: 512 A2S: 512 A2S-S1: 1024
Internal relays	8192 <sup>1</sup>		8192	8192 <sup>1</sup>	Total 8192	Total 8192	Total 2048	256
Latch relays	2048 <sup>1</sup>	8192 <sup>1</sup>	8192	8192 <sup>1</sup>				1048 <sup>1</sup>
Step relays	Sequence-program	—		8192				—
	SFC	2048	8192	8192	—	—	—	
Annunciators	1024 <sup>1</sup>	2048 <sup>1</sup>	2048	2048 <sup>1</sup>	2048	2048	256	256
Edge triggered relays	1024 <sup>1</sup>	2048 <sup>1</sup>	2048	2048 <sup>1</sup>	—	—	—	—
Link relays	2048 <sup>1</sup>	8192 <sup>1</sup>	8192	8192 <sup>1</sup>	8192	4096	1024	1024
Special link relays	1024	2048 <sup>1</sup>	2048	2048 <sup>1</sup>	56	56	56	—
Timers	512 <sup>1</sup>	2048 <sup>1</sup>	2048	2048 <sup>1</sup>	Total 2048	Total 2048	Total 256	256 <sup>1</sup>
Retentive Timers	0 <sup>1</sup>	0 <sup>1</sup>	0 <sup>1</sup>	0 <sup>1</sup>				0 <sup>1</sup>
Counters	512 <sup>1</sup>	1024 <sup>1</sup>	1024	1024 <sup>1</sup>	1024	1024	256	256 <sup>1</sup>
Data registers	11136 <sup>1</sup>	12288 <sup>1</sup>	12288	12288 <sup>1</sup>	8192	6144	1024	1024
Link registers	2048 <sup>1</sup>	8192 <sup>1</sup>	8192	8192 <sup>1</sup>	8192	4096	1024	1024
Special link Registers	1024 <sup>1</sup>	2048 <sup>1</sup>	2048	2048 <sup>1</sup>	56	56	56	—
Function inputs	16 (FX0 to FXF)	16 (FX0 to FXF)	16 (FX0 to FXF)	16 (FX0 to FXF)	—	—	—	—
Function output	16 (FY0 to FYF)	16 (FY0 to FYF)	16 (FY0 to FYF)	16 (FY0 to FYF)	—	—	—	—
Special relays	1024	2048	2048	2048	256	256	256	256
Function registers	5 (FD0 to FD4)	16 (FD0 to FD15)	16 (FD0 to FD15)	16 (FD0 to FD15)	—	—	—	—
Special registers	1024	2048	2048	2048	256	256	256	256
Direct access link devices	Designated by J□□□		Designated by J□□□		—	—	—	—
Direct access special devices	Designated by U□□G□		Designated by U□□G□□		—	—	—	—

Device		System Q		Q series		A series			
		Q00J, Q00, Q01	Qn, QnH, QnPH	Q4AR	QnA	AnU	AnA	AnN	AnS
Index registers	Z	10 (Z0 to Z15)	16 (Z0 to Z15)	16 (Z0 to Z15)	16 (Z0 to Z15)	7 (Z, Z1 to Z6)	7 (Z, Z1 to Z6)	1 (Z)	1 (Z)
	V <sup>2</sup>	—		—		7 (V, V1 to V6)	7 (V, V1 to V6)	1 (V)	1 (V)
File registers		Q00JCPU: 0 Q00 and Q01CPU: 32767	32767 per block (R0 to R32767)  1042432 (ZR0 to ZR1042432)	32767 per block (R0 to R32767)  1042432 (ZR0 to ZR1042432)	32767 per block (R0 bis R32767)  1042432 (ZR0 to ZR1042432)	8192 per block (R0 to R8191)	8192 per block (R0 to R8191)	8192 per block (R0 to R8191)	0 <sup>1</sup>
Accumulators <sup>3</sup>		—		—		2	2	2	2
Nesting		15	15	15	15	8	8	8	8
Pointer		300	4096	4096	4096	256	256	256	256
Interrupt pointers		128	256	48	48	32	32	32	32
SFC blocks		—	320	320	320	—	—	—	—
SFC transition devices		—	512	512	512	—	—	—	—
Decimal constants		K-2147483648 to K2147483647		K-2147483648 to K2147483647		K-2147483648 to K2147483647			
Hexadecimal constants		H0 to HFFFFFFFF		H0 to HFFFFFFFF		H0 bis HFFFFFFFF			
Real number constants		—	E±1,17549-38 to E±3,40282+38	E±1,17549-38 to E±3,40282+38		—	—	—	—
Character strings		„QnA CPU“, „ABCD“ <sup>4</sup>	„QnA CPU“, „ABCD“	„QnA CPU“, „ABCD“		—	—	—	—

<sup>1</sup> The number of device points can be changed via parameters.

<sup>2</sup> The QnA CPU uses V for the edge relay.

<sup>3</sup> Instructions using accumulators with the AnN, AnA, and AnU CPUs have different formats than those with the QnA CPUs.

<sup>4</sup> Can only be used by the \$MOV instruction with the Q00JCPU, Q00CPU and Q01CPU.

**A.3.2 I/O control modes**

I/O control mode		Type of CPU				
		System Q	QnA/ QnAR	AnU	AnA	AnN
Refresh mode		●		●	●	● <sup>2</sup>
Direct input/output mode	Partial refresh instructions	●		●	●	●
	Dedicated instructions <sup>1</sup>	—		●	●	—
	Direct access inputs	●		—	—	—
	Direct access outputs	●		—	—	—
Direct mode		—		—	—	● <sup>2</sup>

<sup>1</sup> The DOUT, DSET, and SRST instructions are dedicated instructions for direct access outputs. There are no dedicated instructions for direct access inputs.

<sup>2</sup> With the AnN CPU refresh mode and direct mode are switched over via DIP switch.

**A.3.3 Data types**

Set Data		System Q CPU	QnA/QnAR CPU	AnU CPU	AnA CPU	AnN CPU
Bit Data	Bit device	●		●	●	●
	Word device	● (Bit designation required)		—	—	—
16-bit word data	Bit device	● (Digit designation required)		● (Digit designation required)	● (Digit designation required)	● (Digit designation required)
	Word device	●		●	●	●
32-bit word data	Bit device	● (Digit designation required)		● (Digit designation required)	● (Digit designation required)	● (Digit designation required)
	Word Operand	●		●	●	●
Real number data		● <sup>1</sup>		●	●	—
Character string data		● <sup>2</sup>		—	—	—

<sup>1</sup> Unusable for Q00JCPU, Q00CPU and Q01CPU

<sup>2</sup> Character string data can be used in the Q00JCPU, Q00CPU and Q01CPU in combination with the \$MOV instruction only.

**NOTE**

Refer to section 3.5 for detailed information on data types.

### A.3.4 Timer-Vergleich

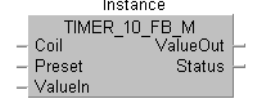
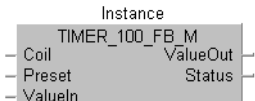
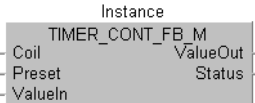
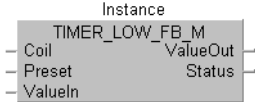
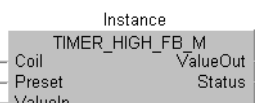
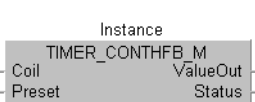
#### Timer functions

Name	Function		System Q QnA/QnAR- CPU	AnU-CPU*	AnA-CPU*	AnN/AnS-CPU*	
Low-speed timer	Measurement unit		100 ms (default) Setting range: 10 to 100 ms  This value is the factor the setting range (TV) is multiplied by.	Fixed at 100 ms			
	Programming (GX IEC Developer)	TIMER_M (regular/dedicated timers)	Setting value designation and timer start	●	●	●	●
		TIMER_VALUE_M (dedicated timers only)	Setting value designation	●	●	●	●
		TIMER_START_M (dedicated timers only)	Timer start	●	●	●	●
Programming (GX Developer)	Setting value designation and timer start		OUT Tn Setting value	OUT Tn Setting value			
High-speed timer	Measurement unit		10 ms (default) Setting range: 10 to 100 ms  This value is the factor the setting range (TV) is multiplied by.	Fixed at 100 ms			
	Programming (GX IEC Developer)	TIMER_M (regular/dedicated timers)	Setting value designation and timer start	●	●	●	●
		TIMER_VALUE_M (dedicated timers only)	Setting value designation	●	●	●	●
		TIMER_START_M (dedicated timers only)	Timer start	●	●	●	●
Programming (GX Developer)	Setting value designation and timer start		OUT Tn Setting value	OUT Tn Setting value			
Retentive low-speed timer	Measurement unit		100 ms (default) Setting range: 10 to 100 ms  This value is the factor the setting range (TV) is multiplied by.	Fixed at 100 ms			
	Programming (GX IEC Developer)	TIMER_H_M (regular/dedicated timers)	Setting value designation and timer start	●	●	●	●
		TIMER_VALUE_M (dedicated timers only)	Setting value designation	●	●	●	●
		TIMER_START_M (dedicated timers only)	Timer start	●	●	●	●
Programming (GX Developer)	Setting value designation and timer start		OUT Tn Setting value	OUT Tn Setting value			

Name	Function		System Q QnA/QnAR- CPU	AnU-CPU*	AnA-CPU*	AnN/AnS-CPU*	
Retentive High-speed timer	Measurement unit		10 ms (default) Setting range: 10 to 100 ms  This value is the factor the setting range (TV) is multiplied by.	—			
	Programming (GX IEC Developer)	TIMER_H_M (regular/dedicated timers)	Setting value designation and timer start	●	—	—	—
		TIMER_VALUE_M (dedicated timers only)	Setting value designation	●	—	—	—
		TIMER_START_M (dedicated timers only)	Timer start	●	—	—	—
Programming (GX Developer)	Setting value designation and timer start		OUTH STn Set value	—	—	—	
Setting range for setting value			1 to 32767	1 to 32767			
Processing of setting value 0			ON momentarily	No maximum (does not time out)			
Index qualification	Contact		Enabled (Z0 and Z1 useable only)	Capable		Not capable	
	Coil		Enabled (Z0 and Z1 useable only)	Not capable		Not capable	
	Setting value		Not capable	Not capable		Not capable	
	Istwert		Enabled (Z0 to Z15 are useable, Z0 to Z9 for Q00JCPU, Q00CPU and Q01CPU)	Capable		Capable	
Update processing for current value			At OUT Tn instruction execution	After END processing			
Contact ON/OFF processing							

\* The initial number for the different timers must be specified in the GX IEC Developer in the dialogbox "PLC Parameter - T/C Range"

Timer function blocks in the GX IEC Developer

Name	Function block	Type of CPU					
		System Q	QnA/ QnAR	AnU	AnA	AnN	AnS
10 ms timer	Instance 	—	●	●	●	●	●
100 ms timer	Instance 	—	●	●	●	●	●
retentive timer	Instance 	—	●	●	●	●	●
Low-speed timer	Instance 	—	●	—	—	—	—
High-speed timer	Instance 	—	●	—	—	—	—
retentive High-speed timer	Instance 	—	●	—	—	—	—

Timer function blocks (legend)

Term in function block	Meaning		Indication of regular timers	Indication of retentive timers
Coil	Coil	Execution condition for timer	TC	STC
Preset	Setting value	—	TValue	TValue
ValueIn	Initial value	Default: 0	—	—
ValueOut	Actual value	—	TN	STN
Status	Contact	Output contact is switched after time	TS	STS

Assign the function block to the instance label specified in the header and assign the input and output variables.

**NOTE Cautions on using timers**

During the execution of the OUT(H) T instruction, the present value of the timers is updated and the contact is switched ON or OFF. If the present value of the timer is larger than or equal to the set value when the timer coil is turned ON, the contact of that timer is turned ON.

In a program, in which the operation of a timer is started by another timer, the instruction for the timer which is started later must be processed first. For example, if the contact of T1 activates the coil of T2, the instruction for T2 must be placed in the program before the instruction for T1.

By doing so, it is prevented that all timer contact are turned ON at the same scan. This can happen if the instruction for a timer, which starts another timer is processed first and the setting value for high speed timers is smaller than the scan time or the setting value for slow speed timers is „1“.

**A.3.5 Comparison of counters**

Counter functions

Function			Type of CPU					
			System Q	QnA/ QnAR	AnU	AnA	AnN	AnS
Programming (GX IEC Developer)	Counter_M	Setting value designation and counter start	●		●	●	●	●
	Counter_Start_M	Setting value designation	●		●	●	●	●
	Counter_Value_M	Counter start	●		●	●	●	●
Programming (GX Developer)	OUT Cn Set value	Setting value designation and counter start	●		●	●	●	●
Index qualification	Contact		Enabled (Z0 and Z1 useable only)		Capable		Not capable	
	Coil		Enabled (Z0 and Z1 useable only)		Capable		Not capable	
	Setting value		Not capable		Not capable		Not capable	
	Current value		Enabled (Z0 to Z15 are useable, Z0 to Z9 for Q00JCPU, Q00CPU and Q01CPU)		Capable		Capable	
Update processing for current value			At OUT Tn instruction execution		After END processing			
Contact ON/OFF processing								



## Counter function blocks

Name	Function blocks	Type of CPU					
		System Q	QnA/ QnAR	AnU	AnA	AnN	AnS
Counter	<p>Instance COUNTER_FB_M - Coil      ValueOut - Preset      Status - ValueIn</p>	—	●	●	●	●	●

## Counter function blocks (legend)

Term in function block	Meaning		Indication of counter
Coil	Coil	Execution condition for counter	CC
Preset	Setting value		CValue
ValueIn	Initial value	Default: 0	—
ValueOut	Current value		CN
Status	Contact	Output contact is switched after the function block is processed.	CS

## A.3.6 Comparison of display instructions

Instruction	System Q CPU	QnA/QnAR-CPU	AnU-CPU	AnA-CPU	AnN-CPU	AnS-CPU
PR <sup>1</sup>	When SM701 is OFF: Output continued until 00 <sub>H</sub> encountered When SM701 is ON: 16 characters output		When M9049 is OFF: Output continued until 00 <sub>H</sub> encountered When M9049 is ON: 16 characters output			
PRC <sup>1</sup>	When SM701 is OFF: 32 character comment output When SM701 is ON: Upper 16 characters output					16-character comment output

<sup>1</sup> These instruction are not available for a Q00JCPU, Q00CPU or Q01CPU.

**A.3.7 Q series and System Q instructions equivalent to A series instructions**

Since System Q and QnA CPUs do not use accumulators (A0, A1), the format of the AnU, AnN, and AnN CPU instructions that use accumulators has changed.

Function	System Q CPU / QnA CPU		AnU CPU / AnA CPU / AnN CPU	
	Instruction format	Remark	Instruction format	Remark
16-bit rotation to right	ROR (d, n)	D: Rotation data	ROR (n)	Rotation data is set at A0
	RRCR (d, n)	D: Rotation data The carry flag uses SM700	RRCR (n)	Rotation data is set at A0 Carry flag uses M9012
16-bit rotation to left	ROL (d, n)	D: Rotation data	ROL (n)	Rotation data is set at A0
	RCL (d, n)	D: Rotation data The carry flag uses SM700	RCL (n)	Rotation data is set at A0 Carry flag uses M9012
32-bit rotation to right	DROR (d, n)	D: Rotation data	DROR (n)	Rotation data is set at A0 and A1
	DRCCR (d, n)	D: Rotation data The carry flag uses SM700	DRCCR (n)	Rotation data is set at A0 and A1 Carry flag uses M9012
32-bit rotation to left	DROL (d, n)	D: Rotation data	DROL (n)	Rotation data is set at A0 and A1
	DRCL (d, n)	D: Rotation data The carry flag uses SM700	DRCL (n)	Rotation data is set at A0 and A1 Carry flag uses M9012
16-bit data search	SER (s1, s2, d, n)	Search results are stored at the D and D+1 devices	SER (s1, s2, n)	Search results stored at A0 and A1
32-bit data search	DSER (s1, s2, d, n)	Search results are stored at the D and D+1 devices	DSER (s1, s2, n)	Search results stored at A0 and A1 Carry flag uses M9012
16-bit data bit Check	SUM (s, d)	Check results are stored at the D device	SUM (s)	Check results stored at A0
32-bit data bit Check	DSUM (s, d)	Check results are stored at the D device	DSUM (s)	Check results stored at A0
Partial refresh	RFS (s, n)	Added dedicated instruction	SEG (d, n)	Only when M9052 is ON
8 character ASCII conversion	\$MOV (s, d)		ASC (d)	
Carry flag set	SET (SM700)	No dedicated instruction	STC	
Carry flag reset	RST (SM700)	No dedicated instruction	CLC	
Jump to END instruction	GOEND	Added dedicated instruction	CJ (P255)	P255: END instruction designation
CHK instruction	CHKST CHK	Added CHKST instruction	CJ (Pn) CHK (P255)	

<sup>1</sup> Not for Q00JCPU, Q00CPU and Q01CPU

### A.3.8 Comparison between QnA/Q2AS CPU and MELSEC System Q CPU

The following new instructions are applicable for a System Q CPU only.

Function	Instruction
Reading of module information	UNIRD
Trace set	TRACE
Trace reset	TRACER
Writing of data to a designated file	S.FWRITE
Reading of data from a designated file	S.FREAD
Loading of a program from memory	PLOAD
Unloading (deletion) of a program from program memory	PUNLOAD
Unloading (deletion) of a program from program memory and loading of a program from memory	PSWAP
High-speed block transfer of file register	PBMOV
Writing in CPU shared memory	S.TO

The following table indicates QnA/Q2AS instructions which are not applicable for a System Q CPU.

Function	Instruction
Batch write operation to EEPROM file register	EROMWR
Setting sampling trace (can be substituted by TRACE)	STRA
Resetting sampling trace (can be substituted by TRACER)	STRAR
Setting status latch	SLT
Resetting status latch	SLTR
Setting of program trace	PTRA
Resetting of program Trace	PTRAR
Execution of program trace	PTRAEXE PTRAEEXEP
ASCII code LED display information	LED
LED display instructions for comments	LEDC

Please notice that the processing of the following instructions is different in a QnA/Q2AS CPU compared with a CPU of the System Q.

Function	Instruction
Output, setting and resetting of internal devices	OUT, SET, RST
Reading device comment data	COMRD
Print comment	PRC
Reset of error display and annunciator	LEDNR
Conversion of binary data	BIN
Conversion of binary data	DBIN
Reading clock data	DATERD
Reading clock data	DATEWR
Interrupt program mask	IMASK
Refresh instruction	COM
Network refresh instruction	ZCOM
Reading routing parameters	RTREAD
Writing routing parameters	RTWRITE
Setting of a closed loop control	PIDINT
Closed loop control	PIDCONT
Counter 1-phase input up or down	UDCNT1
Counter 2-phase input up or down	UDCNT2
Pulse density	SPD
Pulse output	PLSY
Pulse width modulation	PWM

**NOTE**

*When a program for a QnA CPU, which is used to access a special function module, is converted for a System Q CPU, please note the following:*

- *The System Q CPU (Q-Mode) is not compatible to A/AnS series special function and network modules. Use the FROM/TO instruction to read data from and to write data to these modules.*
- *Some A/AnS instructions can still be used if the QnA, Q2AS, A or AnS series special function modules are replaced by System Q special function modules. Refer to the appropriate manual of the special function module.*

## A.4 Overview of special relays

### A.4.1 Table of diagnostic special relays (MELSEC Q series and System Q)

Diagnostic special relays (SM) are internal relays the application of which is fixed in the PLC. Therefore, they cannot be used like other internal relays in a sequence program. However, some of them can be set ON or OFF in order to control the CPU.

**NOTE**

*The special relays SM1200 to SM1255 are used for QnA CPU. These relays are vacant with a System Q CPU.*

*The special relays from SM1500 onward are dedicated for Q4AR CPU.*

The table below describes the meanings of the headings in the following table:

Item	Meaning
Number	Indicates the number of the diagnostic special relay.
Name	Indicates the name of the diagnostic special relay.
Meaning	Contains the function of the diagnostic special relay in brief.
Description	Contains a detailed description of the diagnostic special relay.
Set by (if set)	<p>Indicates whether the diagnostic special relay was set by the system or the user.</p> <p>&lt;Set by&gt;</p> <p>S : Set by the system</p> <p>U : Set by the user (via sequence program or a programming terminal in test mode)</p> <p>S/U : Set by the system or user</p> <p>Is indicated only if the setting is done by the system.</p> <p>&lt;if set&gt;</p> <p>END processing : Set during END processing</p> <p>Initial : Set during initial processing (Power ON, STOP-&gt;RUN)</p> <p>Status change : Set after status change</p> <p>Error : Set after error</p> <p>Instruction execution : Set during instruction execution</p> <p>Request : Set for user request (through SM, etc.)</p>
A CPU M9[ ][ ][ ]	<p>Indicates special relay M9[ ][ ][ ] corresponding to the A CPU (Change and notation when contents changed).</p> <p>Items indicated as "New" were newly added to the Q-Series/System Q CPU.</p>
Valid for:	<p>Indicates the corresponding CPU:</p> <p>●: Can be applied to all types of CPU</p> <p>Q CPU: Can be applied to a System Q CPU</p> <p>QnA CPU: Can be applied to a CPU of the QnA series and Q2AS series</p> <p>CPU name: Can be applied only to the specific CPU (e.g. Q4AR CPU)</p> <p>Rem: Can be applied to a remote MELSECNET/H I/O module</p>

# Overview of special relays

## (1) Diagnostic information

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM0	Diagnostic errors	OFF: No error ON: Error	ON if diagnosis results show error occurrence (Includes external diagnosis). Stays ON subsequently even if normal operations restored.	S (Error)	New	● Rem
SM1	Self-diagnostic error	OFF: No self-diagnosis errors ON: Self-diagnosis	Comes ON when an error occurs as a result of self-diagnosis. Stays ON subsequently even if normal operations restored.	S (Error)	M9008	
SM5	Error common information	OFF: No error common information ON: Error common information	When SM0 is ON, ON if there is error common information.	S (Error)	New	
SM16	Error individual information	OFF: No error individual information ON: Error individual information	When SM0 is ON, ON if there is error individual information.	S (Error)	New	
SM50	Error reset	OFF → ON: Error reset	Conducts error reset operation. See Chapter 5 for further information.	U	New	
SM51	Battery low latch	OFF: Normal ON: Battery low	ON if battery voltage at CPU or memory card drops below rated value. Stays ON subsequently even after normal operation is restored. Synchronous with BAT. ALARM LED.	S (Error)	M9007	●
SM52	Battery low	OFF: Normal ON: Battery low	Same as SM51, but goes OFF subsequently when battery voltage returns to normal.	S (Error)	M9006	
SM53	AC DOWN detection	OFF: AC DOWN detected ON: AC DOWN not detected	Comes ON when a AC power supply module is used and a momentary power interruption not exceeding 20 ms has occurred; reset by turning the power OFF then ON again.	S (Error)	M9005	●
			Comes ON when a DC power supply module is used and a momentary power interruption not exceeding 10 ms has occurred; reset by turning the power OFF then ON again.			Q CPU
			Comes ON when a DC power supply module is used and a momentary power interruption not exceeding 1 ms has occurred; reset by turning the power OFF then ON again.			QnA CPU
SM54	MINI link errors	OFF: Normal ON: Error	Goes ON if MINI (S3) link error is detected at even one of the installed AJ71PT32 (S3) modules. Stays ON subsequently even after normal operation is restored.	S (Error)	M9004	QnA CPU
SM56	Operation errors	OFF: Normal ON: Operation error	ON when operation error is generated. Stays ON subsequently even if normal operation is restored.	S (Error)	M9011	●
SM60	Blown fuse detection	OFF: Normal ON: Module with blown fuse	Comes ON even if there is only one output module with a blown fuse and remains ON even after return to normal. Blown fuse state is checked even for remote I/O station output modules.	S (Error)	M9000	●
SM61	I/O module Verification error	OFF: Normal ON: Error	Comes ON if there is a discrepancy between the actual I/O modules and the registered information when the power is turned on. I/O module verification is also conducted for remote I/O station modules.	S (Error)	M9002	● Rem
SM62	Annunciator detection	OFF: Not detected ON: Detected	Goes ON if even one annunciator F goes ON.	S (Instruction execution)	M9009	●

## (1) Diagnostic information

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM80	CHK detection	OFF: Not detected ON: Detected	Goes ON if error is detected by CHK instruction. Stays ON subsequently even after normal operation is restored.	S (Instruction execution)	New	
SM90	Startup of watchdog timer for step transition (Enabled only when SFC program exists)	OFF: Not started (watchdog timer reset)  ON: Started (watchdog timer started)	Corresponds to SD90	U	M9108	QnA CPU,  Q CPU (except Q00J, Q00 and Q01CPU)
SM91			Corresponds to SD91		M9109	
SM92			Corresponds to SD92		M9110	
SM93			Corresponds to SD93		M9111	
SM94			Corresponds to SD94		M9112	
SM95			Corresponds to SD95		M9113	
SM96			Corresponds to SD96		M9114	
SM97			Corresponds to SD97		New	
SM98			Corresponds to SD98		New	
SM99			Corresponds to SD99		New	
SM100	Serial communication function in use	OFF: Serial communication is not in use  ON: Serial communication is used	Indicates whether the serial communication function in the serial communication setting parameter is selected or not.	S (power on or reset)	New	
SM101	Communication protocol status flag	OFF: Protocol for programming devices  ON: MC protocol	Indicates whether the device that is communicating via the RS232 interface is using the protocol for programming devices or the MC protocol.	S (RS232 communication)	New	
SM110	Protocol error	OFF: No error ON: Error	Turns ON when an abnormal protocol was used to make communication in the serial communication function. Remains ON if the protocol is restored to normal thereafter.	S (Error)	New	Q00J Q00 and Q01CPU
SM111	Communication status	OFF: No error ON: Error	Turns ON when the mode used to make communication was different from the setting in the serial communication function. Remains ON if the mode is restored to normal thereafter.	S (Error)	New	
SM112	Clear error information	ON: Clear diagnostic special relays and registers	When turned ON, the diagnostic special relays SM110 and SM111 are reset and the contents of the diagnostic special registers SD110 and SD111 is cleared.	U	New	
SM113	Overrun error	OFF: No error ON: Error	Turns ON when an overrun error (too much data) occurred during the serial communication.	S (Error)		
SM114	Parity error	OFF: No error ON: Error	Turns ON when a parity error occurred during the serial communication.	S (Error)		
SM115	Framing error	OFF: No error ON: Error	Turns ON when a framing error occurred during the serial communication.	S (Error)		

# Overview of special relays

## (2) System information

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM202	LED off command	OFF → ON : LED off	At change from OFF to ON, the LEDs corresponding to the individual bits at SD202 go off.	U	New	● (except Q00J, Q00 and Q01CPU)
SM203	STOP contact	STOP state	Goes ON at STOP state.	S (Status change)	M9042	●
SM204	PAUSE contact	PAUSE state	Goes ON at PAUSE state.	S (Status change)	M9041	
SM205	STEP-RUN contact	STEP-RUN state	Goes ON at STEP-RUN state.	S (Status change)	M9054	● (except Q00J, Q00 and Q01CPU)
SM206	PAUSE enable coil	OFF: PAUSE disabled ON: PAUSE enabled	PAUSE state is entered if this relay is ON when the remote PAUSE contact goes ON.	U	M9040	●
	Device test request acceptance status	OFF: Device test not yet executed ON: Device test executed	Comes ON when the device test mode is executed on the programming software.	S (Request)	New	Q00J Q00 and Q01 CPU
SM210	Clock data set request	OFF: Ignored ON: Set request	When this relay goes from OFF to ON, clock data being stored from SD210 through SD213 after execution of END instruction for changed scan is written to the clock device.	U	M9025	●
SM211	Clock data error	OFF: No error ON: Error	ON when error is generated in clock data (SD210 through SD213) value and OFF if no error is detected.	S (Request)	M9026	
SM212	Clock data display	OFF: Ignored ON: Display	Displays clock data as month, day, hour, minute and second at the LED display at front of CPU. (Enabled only for Q3A-CPU and Q4A-CPU)	U	M9027	Q3A, Q4A Q4AR CPU
SM213	Clock data read request	OFF: Ignored ON: Read request	When this relay is ON, clock data is read to SD210 through SD213 as BCD values.	U	M9028	● Rem
SM240	No. 1 CPU reset flag	OFF: No reset ON: CPU 1 has been reset	This flag comes ON when the CPU no. 1 has been reset or has been removed from the base. The other CPUs of the multi-CPU system are also put in reset status.	S (Status change)	New	Q02, Q02H, Q06H, Q12H, Q25H CPU with function ver. B or later
SM241	No. 2 CPU reset flag	OFF: No reset ON: CPU 2 has been reset	This flag comes ON when the CPU no. 2 has been reset or has been removed from the base. In the other CPUs of the multi-CPU system the error code 7000 ("MULTI CPU DOWN") will occur.	S (Status change)	New	
SM242	No. 3 CPU reset flag	OFF: No reset ON: CPU 3 has been reset	This flag comes ON when the CPU no. 3 has been reset or has been removed from the base. In the other CPUs of the multi-CPU system the error code 7000 ("MULTI CPU DOWN") will occur.	S (Status change)	New	
SM243	No. 4 CPU reset flag	OFF: No reset ON: CPU 4 has been reset	This flag comes ON when the CPU no. 4 has been reset or has been removed from the base. In the other CPUs of the multi-CPU system the error code 7000 ("MULTI CPU DOWN") will occur.	S (Status change)	New	



## (2) System information

Number	Name	Meaning	Description	Set by (if set)	A CPU M9 [ ] [ ] [ ]	Valid for:
SM244	No. 1 CPU error flag	OFF: No error ON: CPU no. 1 is stopped due to an error	The set flag indicates that an error has occurred which has stopped the CPU. The flag goes OFF when the CPU is normal or when an error occurs which will not stop the CPU.	S (Status change)	New	Q02, Q02H, Q06H, Q12H, Q25H CPU with function ver. B or later
SM245	No. 2 CPU error flag	OFF: No error ON: CPU no. 2 is stopped due to an error		S (Status change)	New	
SM246	No. 3 CPU error flag	OFF: No error ON: CPU no. 3 is stopped due to an error		S (Status change)	New	
SM247	No. 4 CPU error flag	OFF: No error ON: CPU no. 4 is stopped due to an error		S (Status change)	New	
SM250	Max. loaded I/O read	OFF: Ignored ON: Read	When this relay goes from OFF to ON, maximum loaded I/O number is read to SD250.	U	New	● (except Q00J, Q00 and Q01CPU)
SM251	I/O change flag	OFF: No replacement ON: Replacement	After the head I/O number of the I/O module being replaced is set in SD251 online, I/O module replacement is enabled when this relay is ON. (Only one module can be replaced at each setting.) To replace an I/O module in the RUN state, use the program or a peripheral device to turn this relay ON; to replace an I/O module in the STOP state, turn this relay ON in the test mode of a peripheral device. Do not switch between RUN and STOP states until I/O module replacement is completed.	S (END)	M9054	Q2A(S1) Q3A, Q4A Q4AR CPU
SM252	I/O change enabled	OFF: Replacement prohibited ON: Replacement enabled	Goes ON when I/O replacement is enabled.	S (END)	New	
SM254	All stations refresh command	OFF: Refresh the head station only ON: Refresh all stations	Effective for the batch refresh and the low-speed cycle. If this relay is ON, a refresh is made for all stations	S (END)	New	Q CPU (except Q00J, Q00 and Q01CPU)
SM255	MELSECNET/10 module 1 information	OFF: Operative network ON: Standby network	Goes ON for standby network. (If no designation has been made concerning active or standby, active is assumed.)	S (Initial)	New	● (except Q00J, Q00 and Q01CPU)
SM256		OFF: Reads ON: Does not read	For refresh from link to CPU (B, W, etc.) indicate whether to read from the link module.	U	New	
SM257		OFF: Writes ON: Does not write	For refresh from CPU to link (B, W etc.) designate whether to write to the link module.	U	New	
SM260	MELSECNET/10 module 2 information	OFF: Operative network ON: Standby network	Goes ON for standby network. (If no designation has been made concerning active or standby, active is assumed.)	S (Initial)	New	● (except Q00J, Q00 and Q01CPU)
SM261		OFF: Reads ON: Does not read	For refresh from link to CPU (B, W, etc.) indicate whether to read from the link module.	U	New	
SM262		OFF: Writes ON: Does not write	For refresh from CPU to link (B, W etc.) designate whether to write to the link module.	U	New	

# Overview of special relays

## (2) System information

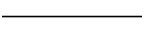
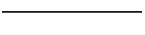
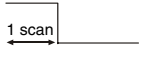
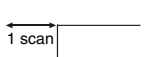
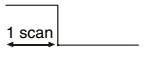
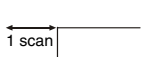
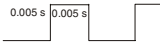



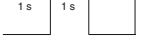
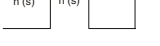
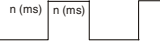
Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM265	MELSECNET/10 module 3 information	OFF: Operative network ON: Standby network	Goes ON for standby network. (If no designation has been made concerning active or standby, active is assumed.)	S (Initial)	New	● (except Q00J, Q00 and Q01CPU)
SM266		OFF: Reads ON: Does not read	For refresh from link to CPU (B, W, etc.) indicate whether to read from the link module.	U	New	
SM267		OFF: Writes ON: Does not write	For refresh from CPU to link (B, W etc.) designate whether to write to the link module.	U	New	
SM270	MELSECNET/10 module 4 information	OFF: Operative network ON: Standby network	Goes ON for standby network. (If no designation has been made concerning active or standby, active is assumed.)	S (Initial)	New	
SM271		OFF: Reads ON: Does not read	For refresh from link to CPU (B, W, etc.) indicate whether to read from the link module.	U	New	
SM272		OFF: Writes ON: Does not write	For refresh from CPU to link (B, W etc.) designate whether to write to the link module.	U	New	
SM280	CC-Link error	OFF: Normal ON: Error	Goes ON when a CC-Link error is detected in any of the installed QJ61QBT11. Goes OFF when normal operation is restored.	S (Error)	New	Q CPU (except Q00J, Q00 and Q01CPU)
			Goes ON when a CC-Link error is detected in any of the installed A(1s)J61QBT11. Stays ON even after normal operation is restored.	S (Error)	New	QnA CPU
SM315	Communication reserved time delay enable flag	OFF: Witout delay ON: With delay	The usage of this flag is enabled when the time reserved for communication has been set in SD315 When this flag is turned ON, the END processing is delayed by the time set in SD315 if no communication is performed. The scan time increases by the time set in SD315. When this flag is turned OFF, the END processing is performed without delay if there is no communication processing.	U	New	Q00J Q00 and Q01 CPU
SM320	Presence/absence of SFC program	OFF: SFC program absent ON: SFC program present	ON if SFC program is correctly registered, and OFF if not registered. Goes OFF if SFC dedicated instruction is not correct.	S (Initial)	M9100	
SM321	Start/stop SFC program	OFF: SFC program stop ON: SFC program start	Initial value is set at the same value as SM900. (Goes ON automatically if SFC program is present.) SFC program will not execute if this goes OFF prior to SFC program processing. Subsequently, starts SFC program when this goes from OFF to ON. Subsequently, stops SFC program when this goes from ON to OFF.	S/U (Initial)	M9101 format change	● (except Q00J, Q00 and Q01CPU)
SM322	SFC program start state	OFF: Initial start ON: Restart	Initial value is set at ON or OFF depending on parameters. When OFF, all execution states are cleared from time SFC program was stopped; starts from the initial step of block where the start request was made. When ON, starts from execution block and execution step active at time SFC program was stopped. (ON is enabled only when resumptive start has been designated at parameters.) SM902 is not automatically designated for latch.	S/U (Initial)	M9102 format change	

## (2) System information

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM323	Presence/absence of continuous transition for entire block	OFF: Continuous transition not effective ON: Continuous transition effective	When OFF, transition occurs at one scan/one step, for all blocks. When ON, transition occurs continuously for all blocks in one scan. In designation of individual blocks, priority is given to the continuous transition bit of the block. (Designation is checked when block starts.)	U	M9103	● (except Q00J, Q00 and Q01CPU)
SM324	Continuous transition prevention flag	OFF: When transition is executed ON: When no transition	When continuous transition is effective, goes ON when continuous transition is not being executed; goes OFF when continuous transition is being executed. Normally ON when continuous transition is not effective.	S (Instruction execution)	M9104	
SM325	Output mode at block stop	OFF: OFF ON: Preserves	When block stops, selects active step operation output. All coil outputs go OFF when OFF. Coil outputs are preserved when ON.	S (Status change)	M9196	
SM326	SFC device clear mode	OFF: Clear device ON: Preserves device	Selects the device status when the stopped CPU is run after the sequence program or SFC program has been modified when the SFC program exists.	U	New	
SM327	Output during end step execution	OFF: OFF ON: Preserves	Selects the output action of the step being held when a block is ended by executing the END step. When the relay is OFF, all coil outputs go OFF. When the relay is ON, all coil outputs are preserved.	S (Initial) U	New	
SM330	Operation mode for low-speed execution type programs	OFF: Asynchronous mode ON: Synchronous mode	Asynchronous mode: Mode where the operations for the low-speed execution type program are continued during excess time. Synchronous mode: Mode where the operations for the low-speed execution type program are started from the next scan even when there is excess time.	U (END)	New	

# Overview of special relays

## (3) System clocks/counters

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM400	Always ON	ON  OFF	This flag is normally ON	S (Every END processing)	M9036	●
SM401	Always ON	ON OFF 	This flag is normally OFF	S (Every END processing)	M9037	
SM402	ON for 1 scan only after RUN	ON  OFF	After RUN, ON for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	M9038	
SM403	After RUN, OFF for 1 scan only	ON  OFF	After RUN, OFF for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	M9039	
SM404	ON for 1 scan only after RUN	ON  OFF	After RUN, ON for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	New	● (except Q00J, Q00 and Q01CPU)
SM405	After RUN, OFF for 1 scan only	ON  OFF	After RUN, OFF for 1 scan only. This connection can be used for scan execution type programs only.	S (Every END processing)	New	
SM409	0.01 second clock		Repeatedly changes between ON and OFF at 5-ms interval. When power supply is turned OFF, or reset is performed, goes from OFF to start.	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)
SM410	0.1 second clock		Repeatedly changes between ON and OFF at each designated time interval. Operation continues even during STOP. When power supply is turned OFF, or reset is performed, goes from OFF to start.	S (Status change)	M9030	●
SM411	0.2 second clock				M9031	
SM412	1 second clock				M9032	
SM413	2 second clock				M9033	
SM414	2x n second clock				M9034 format change	
SM415	2 x n ms clock		Goes between ON and OFF in accordance with the number of milliseconds designated by SD415.	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)

## (3) System clocks/counters

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM420	User timing clock No. 0		<p>Relay repeats ON/OFF switching at fixed scan intervals.</p> <p>When power supply is turned ON, or reset is performed, goes from OFF to start.</p> <p>The ON/OFF intervals are set with the DUTY instruction.</p>	S (Every END processing)	M9020	●
SM421	User timing clock No.1				M9021	
SM422	User timing clock No. 2				M9022	
SM423	User timing clock No. 3				M9023	
SM424	User timing clock No. 4				M9024	
SM430	User timing clock No. 5					
SM431	User timing clock No. 6					
SM432	User timing clock No. 7					
SM433	User timing clock No. 8					
SM434	User timing clock No. 9					
			For use with SM420 through SM424 low speed programs.	S (Every END processing)	New	● (except Q00J, Q00 and Q01CPU)

## (4) Scan information

Number	Name	Meaning	Description	Set by (if set)	A-CPU M9[ ][ ][ ]	Valid for:
SM510	Low speed program execution flag	OFF: Completed or not executed ON: Execution under way	Goes ON when low-speed execution type program is executed.	S (Every END processing)	New	● (except Q00J, Q00 and Q01CPU)
SM551	Reads module service interval	OFF: Ignored ON: Read	When this goes from OFF to ON, the module service interval designated by SD550 is read to SD551 through 552.	U	New	

# Overview of special relays

## (5) Memory cards

Number	Name	Meaning	Description	Set by (if set)	A CPU M9 [ ] [ ] [ ]	Valid for:
SM600	Memory card A usable flags	OFF: Unusable ON: Use enabled	ON when memory card A is ready for use by user.	S (Initial)	New	● (except Q00J, Q00 and Q01CPU)
SM601	Memory card A protect flag	OFF: No protect ON: Protect	Goes ON when memory card A protect switch is ON.	S (Initial)	New	
SM602	Drive 1 flag	OFF: No drive 1 ON: Drive 1 present	Goes ON when drive 1 (card 1 RAM area) is present.	S (Initial)	New	
SM603	Drive 2 flag	OFF: No drive 2 ON: Drive 2 present	Goes ON when drive 2 (card 1 ROM area) is present.	S (Initial)	New	
SM604	Memory card A in-use flag	OFF: Not in use ON: In use	Goes ON when memory card A is in use.	S (Initial)	New	
SM605	Memory card A remove/insert prohibit flag	OFF: Remove/insert enabled ON: Remove/insert prohibited	Goes ON when memory card A cannot be inserted or removed.	U	New	
SM620	Memory card B usable flags	OFF: Unusable ON: Use enabled	Always ON	S (Initial)	New	Q CPU
			ON when memory card B is ready for use by user.	S (Initial)	New	Q2A(S1) Q3A Q4A Q4AR
SM621	Memory card B protect flag	OFF: No protect ON: Protect	Always ON	S (Initial)	New	Q CPU
			Goes ON when memory card B protect switch is ON.	S (Initial)	New	Q2A(S1) Q3A Q4A Q4AR
SM622	Drive 3 flag	OFF: No drive 3 ON: Drive 3 present	Always ON	S (Initial)	New	Q CPU
			Goes ON when drive 3 (card 2 RAM area) is present.		New	Q2A(S1) Q3A Q4A Q4AR
SM623	Drive 4 flag	OFF: No drive 4 ON: Drive 4 present	Always ON	S (Initial)	New	Q CPU
			Goes ON when drive 4 (card 2 ROM area) is present.	S (Initial)	New	
SM624	Memory card B in-use flag	OFF: Not in use ON: In Use	Goes ON when memory card B is in use.	S (Initial)	New	Q2A(S1) Q3A Q4A Q4AR
SM625	Memory card B remove/insert prohibit flag	OFF: Remove/insert enabled ON: Remove/insert prohibited	Goes ON when memory card B cannot be inserted or removed.	U	New	

## (5) Memory cards (continued)

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ] [ ] [ ]	Valid for:
SM640	File register use	OFF: File register not in use ON: File register in use	Goes ON when file register is in use.	S (Status change)	New	●
SM650	Comment use	OFF: Comment not used ON: Comment in use	Goes ON when comment file is in use.	S (Status change)	New	● (except Q00J, Q00 and Q01CPU)
SM660	Boot operation	OFF: Internal memory execution ON: Boot operation in progress	Goes ON while boot operation is in process Goes OFF if boot designation switch is OFF.	S (Status change)	New	●
SM672	Memory card A file register access range flag	OFF: Within access range ON: Outside access range	Goes ON when access is made to area outside the range of file register R of memory card A (set within END processing). Reset at user program.	S/U	New	● (except Q00J, Q00 and Q01CPU)
SM673	Memory card B file register access range flag	OFF: Within access range ON: Outside access range	Goes ON when access is made outside the range of file registers, R, of memory card B (set within END processing). Reset at user program	S/U	New	Q2A(S1) Q3A Q4A Q4AR

## (6) Instruction related diagnostic special relays

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ] [ ] [ ]	Valid for:
SM700	Carry flag	OFF: Carry OFF ON: Carry ON	Carry flag used in application instruction.	S (Instruction execution)	M9012	●
SM701	Number of output characters selection	OFF: 16 characters output ON: Outputs until NUL	When SM701 is OFF, 16 characters of ASCII code are output. When SM701 is ON, output conducted until NUL (00 <sub>H</sub> ) code is encountered.	U	M9049	● (except Q00J, Q00 and Q01CPU)
SM702	Search method	OFF: Search next ON: 2-part search	Designates method to be used by search instruction. Data must be arranged for 2-part search.	U	New	●
SM703	Sort order	OFF: Ascending order ON: Descending order	The sort instruction is used to designate whether data should be sorted in ascending order or in descending order.	U	New	
SM704	Block comparison	OFF: Non-match found ON: All match	Goes ON when all data conditions have been met for the BKCMP instruction.	S (Instruction execution)	New	
SM707	Selection of real number instruction processing type	OFF: Speed optimized ON: Accuracy optimized	When SM707 is OFF, real number instructions are processed at high speed When SM707 is ON, real number instructions are processed with high accuracy	U	New	Q4AR

# Overview of special relays

## (6) Instruction related diagnostic special relays

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ][ ][ ]	Valid for:
SM710	CHK instruction priority ranking flag	OFF: Conditions priority ON: Pattern priority	Remains as originally set when OFF. CHK priorities updated when ON.	S (Instruction execution)	New	● (except Q00J, Q00 and Q01CPU)
SM711	Divided transmission status	OFF: Other than during divided processing ON: During divided processing	In processing of AD57(S1), goes ON when screen is split for transfer, and goes OFF when split processing is completed.	S (Instruction execution)	M9065	OnA
SM712	Transmission processing selection	OFF: Batch transmission ON: Divided transmission	In processing of AD57(S1), goes ON when canvas screen is divided for transfer.	S (Instruction execution)	M9066	
SM714	Communication request registration area BUSY signal	OFF: Communication request to remote terminal module enabled ON: Communication request to remote terminal module disabled	Used to determine whether communications requests to remote terminal modules connected to the AJ71PT32-S3 or A2CCPU can be executed or not.	S (Instruction execution)	M9081	
SM715	EI flag	OFF: During DI ON: During EI	ON when EI instruction is being executed.	S (Instruction execution)	New	●
SM720	Comment read completion flag	OFF: Comment read not completed ON: Comment read completed	SM720 is set for one scan after the execution of the COMRD or PRC instruction	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)
SM721	File being accessed	OFF: File is not accessed ON: File is accessed	This flag is ON while a file is being accessed by the S.FWRITE, S.FREAD, COMRD, PRC, or LEDC instruction	S (Status change)	New	Q CPU
SM722	BIN/DBIN instruction error disabling flag	OFF: Error enabled ON: Error disabled	When this flag is set, an "OPERATION ERROR" is suppressed for both the BIN and the DBIN instruction	U	New	
SM730	BUSY signal for CC-Link communication request registration area	OFF: Request for communication with intelligent device station enabled ON: Request for communication with intelligent device station disabled	This flag is used for determination whether to enable or disable the communication request for the intelligent device station connected with A(1S)J61QBT11.	S (Instruction execution)	New	QnA CPU
SM736	PKEY instruction execution in progress flag	OFF: Instruction not executed ON: Instruction execution	ON when PKEY instruction is being executed. Goes OFF when CR is input, or when input character string reaches 32 characters.	S (Instruction execution)	New	● (except Q00J, Q00 and Q01CPU)
SM737	Keyboard input reception flag for PKEY instruction	OFF: Keyboard input reception enabled ON: Keyboard input reception disabled	Goes ON when keyboard input is being conducted. Goes when keyboard input has been stored at the CPU.	S (Instruction execution)	New	
SM738	MSG instruction reception flag	OFF: Instruction not executed ON: Instruction execution	Goes ON when MSG instruction is executed.	S (Instruction execution)	New	



## (6) Instruction related diagnostic special relays

Number	Name	Meaning	Description	Set by (if set)	A CPU M9[ ] [ ] [ ]	Valid for:
SM774	PID bumpless processing	OFF: Forces match ON: Does not force match	In manual mode, designates whether or not to force the SV value to match the PV value.	U	New	● (except Q00J, Q00 and Q01CPU)
SM775	Selection of link refresh processing during COM instruction execution	OFF: Performs link refresh ON: No link refresh performed	Select whether or not to perform link refresh processing in cases where only general data processing will be conducted during the execution of the COM instruction.	U	New	●
SM776	Enable local device at CALL	OFF: Local device disabled ON: Local device enabled	This flag specifies whether to enable or disable the local device in the program called at the CALL instruction.	U	New	● (except Q00J, Q00 and Q01CPU)
SM777	Enable local device in interrupt program	OFF: Local device disabled ON: Local device enabled	This flag specifies whether to enable or disable the local device at the execution of an interrupt program.	U	New	
SM780	CC-Link dedicated instruction executable	OFF: CC-Link dedicated instruction executable ON: CC-Link dedicated instruction not executable	This flag switches ON when the number of the CC-Link dedicated instructions that can be executed simultaneously reaches 32. When the number goes below 32, the flag is reset.	S (Status change)	New	QnA CPU

# Overview of special relays

## (7) Debugging

Number	Name	Meaning	Description	Set by (if set)	A-CPU M9[ ][ ][ ]	Valid for:
SM800	Trace preparation	OFF: Not prepared ON: Ready	Goes ON when the trace preparation is completed.	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)
	Sampling trace preparation		Goes ON when sampling trace is ready.	S (Status change)		QnA CPU
SM801	Trace start	OFF: Suspend ON: Start	Trace is started when this goes ON. Suspended when OFF (Related special M all OFF).	U	M9047	Q CPU (except Q00J, Q00 and Q01CPU)
	Sampling trace start		Sampling trace started when this goes ON. Suspended when OFF (Related special M all OFF).	U		QnA CPU
SM802	Trace execution in progress	OFF: Suspend ON: Start	Goes ON during execution of trace.	S (Status change)	M9046	Q CPU (except Q00J, Q00 and Q01CPU)
	Sampling trace execution in progress		Goes ON during execution of sampling trace.	S (Status change)		QnA CPU
SM803	Trace trigger	OFF → ON: Start	Sampling trace trigger goes ON when this goes from OFF to ON (Identical to TRACE instruction execution state).	U	M9044	Q CPU (except Q00J, Q00 and Q01CPU)
	Sampling trace trigger		Sampling trace trigger goes ON when this goes from OFF to ON (Identical to STRA instruction execution state)	U		QnA CPU
SM804	After Trace trigger	OFF: Not after trigger ON: After trigger	Goes ON after trace trigger is triggered.	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)
	After Sampling trace trigger		Goes ON after sampling trace is triggered.	S (Status change)		QnA CPU
SM805	Trace completed	OFF: Not completed ON: End	Goes ON at completion of trace.	S (Status change)	M9043	Q CPU (except Q00J, Q00 and Q01CPU)
	Sampling trace completed		Goes ON at completion of sampling trace.	S (Status change)		QnA CPU
SM806	Status latch preparation	OFF: Not prepared ON: Ready	Goes ON when status latch is ready.	S (Status change)	New	QnA CPU
SM807	Status latch command	OFF → ON: Latch	Runs status latch command.	U	New	
SM808	Status latch completion	OFF: Latch not completed ON: Latch completed	Comes ON when status latch is completed.	S (Status change)	M9055	
SM809	Status latch clear	OFF → ON: Clear	Enable next status latch.	U	New	
SM810	Program trace preparation	OFF: Not ready ON: Ready	Goes ON when program trace is ready.	S (Status change)	New	

## (7) Debugging

Number	Name	Meaning	Description	Set by (if set)	A-CPU M9[ ] [ ] [ ]	Valid for:
SM811	Start program trace	OFF: Suspend ON: Start	Program trace started when this goes ON. Suspended when OFF (Related special M all OFF).	U	New	QnA CPU
SM812	Program trace execution underway	OFF: Suspend ON: Start	ON when program trace execution is underway.	S (Status change)	New	
SM813	Program trace trigger	OFF → ON: Start	Program trace trigger goes ON when this goes from OFF to ON (Identical to PTR A instruction execution status).	U	New	
SM814	After program trace trigger	OFF: Not after trigger ON: After trigger	Goes ON after program trace trigger.	S (Status change)	New	
SM815	Program trace completion	OFF: Not completed ON: END	Goes ON at completion of program trace.	S (Status change)	New	
SM820	Step trace preparation	OFF: Not prepared ON: Ready	Goes ON after program trace registration, at ready.	S (Status change)	New	● (except Q00J, Q00 and Q01CPU)
SM821	Step trace starts	OFF: Suspend ON: Start	When this goes ON, step trace is started Suspended when OFF (Related special M all OFF)	U	M9182 format change	
SM822	Step trace execution underway	OFF: Suspend ON: Start	Goes ON when step trace execution is underway Goes OFF at completion or suspension	S (Status change)	M9181	
SM823	After step trace trigger	OFF: Not after trigger ON: Is after first trigger	Goes ON if even 1 block within the step trace being executed is triggered. Goes OFF when step trace is commenced.	S (Status change)	New	
SM824	Step trace After trigger	OFF: Is not after all triggers ON: Is after all triggers	Goes ON if all blocks within the step trace being executed are triggered. Goes OFF when step trace is commenced.	S (Status change)	New	
SM825	Step trace completed	OFF: Not completed ON: End	Goes ON at step trace completion. Goes OFF when step trace is commenced.	S (Status change)	M9180	
SM826	Trace error	OFF: Normal ON: Error	Goes ON if error occurs during execution of trace/sampling trace.	S (Status change)	New	Q CPU (except Q00J, Q00 and Q01CPU)
	Sampling trace error			S (Status change)		QnA CPU
SM827	Status latch error	OFF: Normal ON: Error	Goes ON if error occurs during execution of status latch	S (Status change)	New	QnA CPU
SM828	Program trace error	OFF: Normal ON: Error	Goes ON if error occurs during execution of program trace.	S (Status change)	New	

## (8) Latch Area

Number	Name	Meaning	Description	Set by (if set)	A-CPU M9[ ] [ ] [ ]	Valid for:
SM900	Power cut file	OFF: No power cut file ON: Power cut file present	Goes ON if power was interrupted while a file was being accessed .	S/U (Status change)	New	QnA CPU
SM910	RKEY registration flag	OFF: Keyboard input not registered ON: Keyboard input registered	Goes ON at registration of keyboard input. OFF if keyboard input is not registered.	S (Instruction execution)	New	● (except Q00J, Q00 and Q01CPU)

## Overview of special relays

### (9) A to System Q/QnA series conversion correspondences

For a conversion from the MELSEC A series to the MELSEC Q series or the MELSEC System Q the special relays M9000 through M9255 (A series) correspond to the diagnostic relays SM1000 through SM1255 (System Q/Q series).

These diagnostic special relays are all set by the system and cannot be changed by a user-program. Users intending to set or reset these relays should alter their programs so that only real System Q/QnA diagnostic special relays are applied. An exception are the special relays M9084 and M9200 through M9255. If a user can set or reset some of these special relays before conversion, the user can also set and reset the corresponding relays among SM1084 and SM1200 through SM1255 after the conversion.

Refer to the manuals of the CPUs and the networks MELSECNET and MELSECNET/B for detailed information on the special relays of the A series.

#### NOTE

*The processing time may be longer when converted special relays are used with a System Q CPU. Don't select "A-PLC: Use special relay/special register from SM/SD 1000" within the PC system setting in the GX Developer parameters when converted special relays are not used.*

*When a special relay for modification is provided, the device number should be changed to the provided System Q/QnA CPU special relay. When no special relay for modification is provided, the converted special relay can be used for the device number.*

Table of special relays and diagnostic relays

A CPU special relay	Special relay after conversion	Equivalent System Q/QnA diagnostic special relay	Name	Meaning	Valid for:
M9000	SM1000	—	Fuse blown	OFF: Normal ON: Fuse blown module with blown fuse present	System Q/ QnA CPU
M9002	SM1002	—	I/O module verification error	OFF: Normal ON: Error	
M9004	SM1004	—	MINI link error	OFF: Normal ON: Error	
M9005	SM1005	—	AC DOWN detection	OFF: AC DOWN not detected ON: AC DOWN detected	System Q/ QnA CPU
M9006	SM1006	—	Battery low	OFF: Normal ON: Battery low	
M9007	SM1007	—	Battery low (latched)	OFF: Normal ON: Battery low	
M9008	SM1008	SM1	Self-diagnostic error	OFF: No error ON: Error	
M9009	SM1009	SM62	Annunciator detection	OFF: No F number detected ON: F number detected	
M9011	SM1011	SM56	Operation error flag	OFF: No error ON: Error	
M9012	SM1012	SM700	Carry Flag	OFF: Carry OFF ON: Carry ON	
M9016	SM1016	The device does not work with a System Q/QnA CPU	Data memory clear flag	OFF: Ignored ON: Output cleared	
M9017	SM1017	The device does not work with a System Q/QnA CPU	Data memory clear flag	OFF: Ignored ON: Output cleared	

Table of special relays and diagnostic relays

A CPU special relay	Special relay after conversion	Equivalent System Q/QnA diagnostic special relay	Name	Meaning	Valid for:
M9020	SM1020	—	User timing clock No. 0		System Q/ QnA CPU
M9021	SM1021	—	User timing clock No. 1		
M9022	SM1022	—	User timing clock No. 2		
M9023	SM1023	—	User timing clock No. 3		
M9024	SM1024	—	User timing clock No. 4		
M9025	SM1025	—	Clock data set request	OFF: Ignored ON: Set request present used	System Q/ QnA CPU
M9026	SM1026	—	Clock data error	OFF: No error ON: Error	
M9027	SM1027	—	Clock data display	OFF: Ignored ON: Display	
M9028	SM1028	—	Clock data read request	OFF: Ignored ON: Read request	
M9029	SM1029	The device does not work with a System Q/QnA CPU	Batch processing of data communications request	OFF: Batch processing not conducted ON: Batch processing conducted	
M9030	SM1030	—	0.1 second clock		
M9031	SM1031	—	0.2 second clock		
M9032	SM1032	—	1 second clock		
M9033	SM1033	—	2 second clock		
M9034	SM1034	—	1 minute clock		
M9036	SM1036	—	Always ON	ON ————— OFF	
M9037	SM1037	—	Always OFF	ON OFF —————	
M9038	SM1038	—	ON for 1 scan only after RUN	ON OFF	

# Overview of special relays

Table of special relays and diagnostic special relays (continued)

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9039	SM1039	—	RUN flag (After RUN, OFF for 1 scan only)	ON  OFF	System Q/ QnA CPU
M9040	SM1040	SM206	PAUSE enable coil	OFF: PAUSE disabled ON: PAUSE enabled	
M9041	SM1041	SM204	PAUSE status contact	OFF: PAUSE not in effect ON: PAUSE in effect	
M9042	SM1042	SM203	STOP status contact	OFF: STOP not in effect ON: STOP in effect	
M9043	SM1043	SM805	Sampling trace completed	OFF: Sampling trace in progress ON: Sampling trace completed	
M9044	SM1044	SM803	Sampling trace	0 → 1 STRA Same as execution 1 → 0 STRAR Same as execution	
M9045	SM1045	The device does not work with a System Q/QnA CPU.	Watchdog timer (WDT) reset	OFF: Does not reset WDT ON: Resets WDT	
M9046	SM1046	SM802	Sampling trace	OFF: Trace not in progress ON: Trace in progress	
M9047	SM1047	SM801	Sampling trace preparations	OFF: Sampling Trace suspended ON: Sampling Trace started	
M9049	SM1049	SM701	Selection of number of characters output	OFF: Output until NUL ON: 16 characters output	
M9051	SM1051	The device does not work with a System Q/QnA CPU.	CHG instruction execution disable	OFF: Enabled ON: Disable	
M9052	SM1052	The device does not work with a System Q/QnA CPU.	SEG instruction switch	OFF: 7 segment display ON: I/O partial refresh	
M9054	SM1054	SM205	STEP RUN flag	OFF: STEP RUN not in effect ON: STEP RUN in effect	
M9055	SM1055	SM808	Status latch completion flag	OFF: Not completed ON: Completed	
M9056	SM1056	These devices do not work with a System Q/QnA CPU.	Main side P, I set request	OFF: Other than when P, I set being requested ON: P, I set being requested	System Q/ QnA CPU
M9057	SM1057		Sub side P, I set request		
M9058	SM1058		Main program P, I set completion	Momentarily ON at P, I set completion	
M9059	SM1059		Sub program P, I set completion	Momentarily ON at P, I set completion	
M9060	SM1060		Sub program 2 P, I set request	OFF: Other than when P, I set being requested ON: P, I set being requested	
M9061	SM1061		Sub program 3 P, I set request		

Table of special relays and diagnostic special relays (continued)

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9065	SM1065	SM711	Divided processing execution detection	OFF: Divided processing not underway ON: During divided processing	QnA CPU
M9066	SM1066	SM712	Divided processing request flag	OFF: Batch processing ON: Divided processing	
M9070	SM1070	The device does not work with a System Q/QnA CPU.	A8UPU/A8PUJ required search time	OFF: Read time not shortened ON: Read time shortened	System Q/QnA CPU
M9081	SM1081	SM714	Communication request registration area BUSY signal	OFF: Empty spaces in communication request registration area ON: No empty spaces in communication request registration area	QnA CPU
M9084	SM1084	The device does not work with a System Q/QnA CPU.	Error check	OFF: Error check executed ON: No error check	System Q/QnA CPU
M9091	SM1091	The device does not work with a System Q/QnA CPU.	Instruction error flag	OFF: No error ON: Error	
M9094	SM1094	SM251	I/O change flag	OFF: Replacement ON: No replacement	QnA CPU
M9100	SM1100	SM320	Presence/absence of SFC program	OFF: SFC programs not used ON: SFC programs used	System Q/QnA CPU
M9101	SM1101	SM321	Start/stop SFC program	OFF: SFC programs stop ON: SFC programs start	
M9102	SM1102	SM322	SFC program start state	OFF: Initial Start ON: Continue	
M9103	SM1103	SM323	Presence/absence of continuous transition	OFF: Continuous transition not effective ON: Continuous transition effective	
M9104	SM1104	SM324	Continuous transition suspension flag	OFF: When transition is completed ON: When no transition	
M9108	SM1108	SM90	Step transition watchdog timer start (equivalent of D9108)	OFF: Watchdog timer reset ON: Watchdog timer reset start	
M9109	SM1109	SM91	Step transition watchdog timer start (equivalent of D9109)		
M9110	SM1110	SM92	Step transition watchdog timer start (equivalent of D9110)		
M9111	SM1111	SM93	Step transition watchdog timer start (equivalent of D9111)		
M9112	SM1112	SM94	Step transition watchdog timer start (equivalent of D9112)		
M9113	SM1113	SM95	Step transition watchdog timer start (equivalent of D9113)		
M9114	SM1114	SM96	Step transition watchdog timer start (equivalent of D9114)		
M9180	SM1180	SM825	Active step sampling trace execution flag		
M9181	SM1181	SM822	Active step sampling trace execution flag	OFF: Trace not being executed ON: Trace execution under way	

# Overview of special relays

Table of special relays and diagnostic special relays (continued)

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9182	SM1182	SM821	Active step sampling trace permission	OFF: Trace disable/suspend ON: Trace enable	System Q/ QnA CPU
M9196	SM1196	SM325	Operation output at block stop	OFF: Coil output OFF ON: Coil output ON	
M9197 M9198	SM1197 SM1198	The device does not work with a System Q/QnA CPU	Switch between blown fuse and I/O verification error display	Display is changed depending on combination of M9197 ON/OFF state and M9198 ON/OFF state.	
M9199	SM1199	The device does not work with a System Q/QnA CPU	On-line recovery of sampling trace status latch data	OFF: Does not perform data recovery ON: Performs data recovery	
M9200	SM1200	—	LRDP instruction reception	OFF: Not accepted ON: Accepted	QnA CPU
M9201	SM1201	—	LRDP instruction completion	OFF: Not completed ON: End	
M9202	SM1202	—	LWTP instruction reception	OFF: Not accepted ON: Accepted	
M9203	SM1203	—	LWTP instruction completion	OFF: Not completed ON: End	
M9204	SM1204	—	LRDP instruction completion	OFF: Not completed ON: End	
M9205	SM1205	—	LWTP instruction completion	OFF: Not completed ON: End	
M9206	SM1206	—	Host station link parameter error	OFF: Normal ON: Abnormal	
M9207	SM1207	—	Link parameter check results	OFF: YES ON: NO	
M9208	SM1208	—	Sets master station B and W transmission range (for lower link master stations only).	OFF: Transmits to tier 2 and tier 3 ON: Transmits to tier 2 only	
M9209	SM1209	—	Link parameter check command (for lower link master stations only).	OFF: Executing the check function ON: Check non-execution	
M9210	SM1210	—	Link card error (for local station)	OFF: Normal ON: Abnormal	
M9211	SM1211	—	Link module error (for master station use)	OFF: Normal ON: Abnormal	
M9224	SM1224	—	Link state	OFF: Online ON: Offline, station-to-station test, or self-loopback test	
M9225	SM1225	—	Forward loop error	OFF: Normal ON: Abnormal	
M9226	SM1226	—	Reverse loop error	OFF: Normal ON: Abnormal	
M9227	SM1227	—	Loop test state	OFF: Not being executed ON: Forward or reverse loop test execution underway	



Table of special relays and diagnostic special relays (continued)

A CPU special relay	Special relay after conversion	Equivalent QnA diagnostic special relay	Name	Meaning	Valid for:
M9232	SM1232	—	Local station operation state	OFF: RUN or STEP RUN state ON: STOP or PAUSE state	QnA CPU
M9233	SM1233	—	Local station error detect state	OFF: No errors ON: Error detection	
M9235	SM1235	—	Local station, remote I/O station parameter error detect state	OFF: No errors ON: Error detection	
M9236	SM1236	—	Local station, remote I/O station parameter error detect state	OFF: No communications ON: Communications underway	
M9237	SM1237	—	Local station, remote I/O station error	OFF: Normal ON: Abnormal	
M9238	SM1238	—	Local station, remote I/O station forward or reverse loop error	OFF: Normal ON: Abnormal	
M9240	SM1240	—	Link state	OFF: Online ON: Offline, station-to-station test or self-loopback test	
M9241	SM1241	—	Forward loop line error	OFF: Normal ON: Abnormal	
M9242	SM1242	—	Reverse loop line error	OFF: Normal ON: Abnormal	
M9243	SM1243	—	Loopback implementation	OFF: Loopback not being conducted ON: Loopback implementation	
M9246	SM1246	—	Data not received	OFF: Reception ON: No reception	
M9247	SM1247	—	Data not received	OFF: Reception ON: No reception	
M9250	SM1250	—	Parameters not received	OFF: Reception ON: No reception	
M9251	SM1251	—	Link relay	OFF: Normal ON: Abort	
M9252	SM1252	—	Loop test state	OFF: Not being executed ON: Forward or reverse loop test execution underway	
M9253	SM1253	—	Master station operation state	OFF: RUN or STEP RUN state ON: STOP or PAUSE state	
M9254	SM1254	—	Local station other than host station operation state	OFF: RUN or STEP RUN state ON: STOP or PAUSE state	
M9255	SM1255	—	Local station other than host station error	OFF: Normal ON: Abnormal	

# Overview of special relays

---

## A.4.2 Table of special relays (M) (A series)

Special relays (M) are internal relays provided for a large number of application varieties like error indication, special functions, etc. The following table contains an overview of the entire MELSEC A series special relays including a description of their purposes.

In general there are two types of special relays:

- Special relays that are set automatically by the CPU and can only be reset by the user.
- Special relays that can be set or reset only under certain conditions depending on their functions.

### NOTE

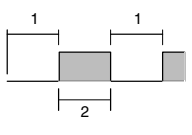
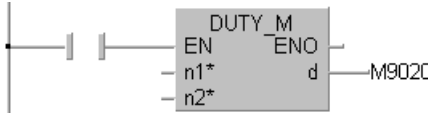
*The usage of special relays in a sequence program has to be checked accordingly.*

*Special relays that are tagged by ❶, ❷ or ❸ in the margin "Number" cannot be set or reset randomly. The according explanations are given following this table on page 37.*

*In how far a special relay can be used in combination with a certain CPU is listed in the table below:*

CPU		Meaning
		For all CPU types without restrictions
○	(AnA-CPU)	Not for the specified CPU(s)
●	A2C-CPU	For the specified CPU(s) only

# Overview of special relays

Number	Meaning	Status	Description	CPU
① M9000	Fuse blown	OFF: Normal ON: Fuse blown unit	Turned on when there is one or more output units of which fuse has been blown. Remains on if normal status is restored. Output modules of remote I/O stations are also checked for fuse condition.	○ Usable with all types of CPUs. (Only remote I/O station information is valid for A2C.)
① M9002	I/O unit verify error	OFF: Normal ON: Error	Turned on if the status of I/O is different from entered status when power is turned on. Remains on if normal status is restored. I/O module verification is done also to remote I/O station modules. (Reset is enabled only when special registers D9116 to D9123 are reset.)	○ Usable with all types of CPUs. (Only remote I/O station information is valid for A2C.)
M9004	MINI link master module error	OFF: Normal ON: Error	Turned on when the MINI(S3) link error is detected on even one of the AJ71PT32(S3) modules being loaded. Remains on if normal status is restored.	● Dedicated to AnA and AnU.
① M9005	AC DOWN detection	OFF: AC power good ON: AC power down	Turned on when an momentary power failure of 20 msec or less occurred. Reset when POWER switch is moved from OFF to ON position.	○ Usable with all types of CPUs.
M9006	Battery low	OFF: Normal ON: Battery low	Turned on when battery voltage reduces to less than specified. Turned off when battery voltage becomes normal.	○ Usable with all types of CPUs.
① M9007	Battery low latch	OFF: Normal ON: Battery low	Turned on when battery voltage reduces to less than specified. Remains on if battery voltage becomes normal.	○ Usable with all types of CPUs.
① M9008	Self-diagnostic error	OFF: No error ON: Error	Turned on when error is found as a result of self-diagnosis.	○ Usable with all types of CPUs.
M9009	Annunciator detection	OFF: No detection ON: Detected	Turned on when OUT F or SET F instruction is executed. Switched off when D9124 data is zeroed.	○ Usable with all types of CPUs.
M9010	Operation error flag	OFF: No error ON: Error	Turned on when operation error occurs during execution of application instruction. Turned off when error is eliminated.	○ Unusable with A3H, A3M, AnA(-F) and AnU.
① M9011	Operation error flag	OFF: No error ON: Error	Turned on when operation error occurs during execution of application instruction. Remains on if normal status is restored.	○ Usable with all types of CPUs.
M9012	Carry Flag	OFF: Carry off ON: Carry on	Carry flag used in application instruction.	○ Usable with all types of CPUs.
M9016	Data memory clear flag	OFF: No processing ON: Output clear	Clears the data memory including the latch range (other than special relays and special registers) in remote run mode from computer, etc. when M9016 is on.	○ Usable with all types of CPUs.
M9017	Data memory clear flag	OFF: No processing ON: Output clear	Clears the unlatched data memory (other than special relays and special registers) in remote run mode from computer, etc. when M9017 is on.	○ Usable with all types of CPUs.
M9020	User timing clock No. 0		Relay which repeats on/off at intervals of predetermined scan. When power is turned on or reset is performed, the clock starts with off. Set the intervals of on/off by DUTY instruction.	○ Usable with all types of CPUs.
M9021	User timing clock No. 1			
M9022	User timing clock No. 2			
M9023	User timing clock No. 3			
M9024	User timing clock No. 4			
		<sup>1</sup> Scan n2 <sup>2</sup> Scan n1		
② M9025	Clock data set request	OFF: No processing ON: Set requested	Writes clock data from D9025 - D9028 to the clock element after the END instruction is executed during the scan in which M9025 has changed from off to on.	● Unusable with An, A3H, A3M, A3V, A2C, A52G and A0J2H.
M9026	Clock data error	OFF: No error ON: Error	Switched on by clock date (D9025 to D9028) error.	● Unusable with An, A3H, A3M, A3V, A2C, A52G and A0J2H.

# Overview of special relays

Number	Meaning	Status	Description	CPU
M9027	Clock data display	OFF: No processing ON: Display	Clock data is read from D9025 to D9028 and month, day, hour, minute and minute are indicated on the CPU front LED display.	● Usable with A3N, A3N-F, A3A, A73 and A3N board.
② M9028	Clock data read request	OFF: No processing ON: Read request	Reads clock data to D9025 - D9028 in BCD when M9028 is on.	● Unusable with An, A3H, A3M, A3V, A2C and A0J2H.
M9030	0,1 second clock	OFF: 0,05 s ON: 0,05 s	0,1 second, 0,2 second, 1 second, 2 second and 1 minute clocks are generated. Not turned on and off per scan but turned on and off even during scan if corresponding time has elapsed. Starts with off when power is turned on or reset is performed.	Unusable with A3V
M9031	0,2 second clock	OFF: 0,1 s ON: 0,1 s		
M9032	1 second clock	OFF: 0,5 s ON: 0,5 s		
M9033	2 second clock	OFF: 1 s ON: 1 s		
M9034	1 minute clock	OFF: 30 s ON: 30 s		
M9036	Normally ON	Always ON		
M9037	Normally OFF	Always OFF	M9036 and M9037 are turned on and off without regard to position of key switch on CPU front. M9038 and M9039 are under the same condition as RUN status except when the key switch is at STOP position, and turned off and on. Switched off if the key switch is in STOP position. M9038 is on for one scan only and M9039 is off for one scan only if the key switch is not in STOP position.	
M9038	On only for 1 scan after run	ON: for one scan after run OFF: after one scan		
M9039	RUN flag (off only for 1 scan after run)	ON: after one scan OFF: for one run scan		
M9040	PAUSE enable coil	OFF: PAUSE disabled ON: PAUSE enabled	When RUN key switch is at PAUSE position or remote pause contact has turned on and if M9040 is on, PAUSE mode is set and M9041 is turned on.	Usable with all types of CPU
M9041	PAUSE status contact	OFF: Not during PAUSE ON: During PAUSE		
M9042	STOP status contact	OFF: During STOP ON: Not during STOP	Switched on when the RUN key switch is in STOP position.	Usable with all types of CPU
M9043	Sampling trace completion	OFF: During Sampling trace ON: Sampling trace completion	Turned on upon completion of sampling trace performed the number of times preset by parameter after STRA instruction is executed. Reset when STRAR instruction is executed.	○ Unusable with A1 and A1N
M9044	Sampling trace	OFF to ON: STRA Same as execution OFF to ON: STRAR Same as execution	Turning on/off M9044 can execute STRA/STRAR instruction. (M9044 is forcibly turned on/off by a peripheral device.) When switched from ON to OFF: STRA instruction When switched from ON to OFF: STRAR instruction The value stored in D9044 is used as the condition for the sampling trace. At scanning, at time ↔ time (10 msec unit).	● Unusable with A1 and A1N
M9046	Sampling trace	OFF: Except during trace ON: During trace	Switched on during sampling trace.	○ Unusable with A1 and A1N
M9047	Sampling trace preparation	OFF: Sampling trace stop ON: Sampling trace start	Switched on to start sampling trace. Switched off to stop sampling trace.	○ Unusable with A1 and A1N
M9049	Switching the number of output characters	OFF: Up to NUL code are output. ON: 16 characters are output.	When M9049 is off, all characters up to NUL (00 <sub>H</sub> ) code are output. When M9049 is on, ASCII codes of 16 characters are output.	○ Usable with An, A3V, A2C and A52G
② M9050	Operation result storage memory change contact (for CHG instruction)	OFF: not changed ON: Changed	Switched on to exchange the operation result storage memory data and the save area data (for details refer to section 7.6.8).	● Dedicated to A3
M9051	CHG instruction execution disable	OFF: Disable ON: Enable	Switched on to disable the CHG instruction. Switched on when program transfer is requested and automatically switched off when transfer is complete.	● Usable with A3, A3N(-F), A3H, A3M, A3V, A3A(-F), A3U, A4U, A73 and A3N board

## Overview of special relays

Number	Meaning	Status	Description	CPU			
② M9052	SEG instruction switching	OFF: 7SEG display ON: I/O partial refresh	Switched on to execute the SEG instruction as an I/O partial refresh instruction. Switched off to execute the SEG instruction as a 7SEG display instruction (for details refer to section 6.7.2 and 7.5.5).	○	Unusable with An, A3V and A3N board		
② M9053	EI/DI instruction switching	OFF: Sequence interrupt control ON: Link interrupt control	Switched on to execute the link refresh enable, disable (EI, DI) instructions. (for details refer to section 6.6.1 and 6.7.4)	●	Unusable with An, A3H, A3M, AnA, AnA-F, AnU and A3V		
M9054	STEP RUN flag	OFF: Other than step run ON: During step run	Switched on when the RUN key switch is in STEP RUN position.	○	Unusable with A1S, A2C, A0J2H and A52G		
M9055	Status Latch complete flag	OFF: Not complete ON: Complete	Turned on when status latch is completed. Turned off by reset instruction.	○	Unusable with A1 and A1N.		
M9056	Main program P, I set request	OFF: Other than P, I set request ON: P, I set request	Provides P, I set request after transfer of the other program (for example subprogram when main program is being run) is complete during run. Automatically switched off when P, I setting is complete.	●	Usable with A3, A3N, A3N-F, A3H, A3M, A3V, A3A, A3A-F, A73, A3U, A4U and A3N board		
M9057	Subprogram 1 P, I set request	OFF: Except during P, I set request ON: During P, I set request					
M9060	Subprogram 2 P, I set request					●	Dedicated to A4U
M9061	Subprogram 3 P, I set request						
M9060	Remote terminal error	OFF: Normal ON: Error	Turned on when one of remote terminal modules has become a faulty station. Communication error is detected when normal communication is not restored after the number of retries set at D9174. Turned off when communication with all remote terminal modules is restored to normal with automatic online return enabled. Remains on when automatic online return is disabled. Not turned on or off when communication is suspended at error detection.	●	Usable with A2C and A52G		
M9061	Communication error	OFF: Normal ON: Error	Turned on when communication with a remote terminal module or an I/O module is faulty. Communication error occurs due to the following reasons. Initial data error Cable breakage Power off for remote terminal modules or I/O modules Turned off when communication is restored to normal with automatic online return enabled. Remains on when communication is suspended at error detection with automatic online return disabled.	●	Usable with A2C and A52G		
M9065	Divided transfer status	OFF: Other than divided processing ON: Divided processing	Turned on when canvas screen transfer to AD57(S1)/AD58 is done by divided processing and turned off at completion of divided processing.	●	Usable with AnA(-F) and AnU		
② M9066	Transfer processing switching	OFF: Batch transfer ON: Divided transfer	Turned on when canvas screen transfer to AD57(S1)/AD58 is done by divided processing.	●	Usable with AnA(-F) and AnU		
M9067	I/O module error detection	OFF: Normal ON: Error	Turned on when one of I/O modules has become a faulty station. Communication error is detected when normal communication is not restored after the number of retries set at D9174. Turned off when communication with all I/O modules is restored to normal with automatic online return enabled. Remains on when automatic online return is disabled. Not turned on or off when communication is suspended at error detection.	●	Usable with A2C and A52G.		
M9068	Test mode	OFF: Automatic online return enabled Automatic online return disabled Communication suspended at online error ON: Line check	Turned on when line check with I/O modules and remote terminal modules is performed. Turned off when communication with I/O modules and remote terminal modules is performed.	●	Usable with A2C and A52G.		

# Overview of special relays

Number	Meaning	Status	Description	CPU
M9069	Output at line error	OFF: All outputs are turned off. ON: Outputs are retained.	Sets whether all outputs are turned off or retained at communication error. OFF All outputs are turned off at communication error. ON Outputs before communication error are retained.	● Usable with A2C and A52G.
M9081	Communication request to remote terminal modules	OFF: Communication request to remote terminal modules enabled ON: Communication request to remote terminal modules disabled	Indication of communication enable/disable to remote terminal modules connected to the AJ71PT32-S3, A2C or A52G.	● Usable with AnA, AnA-F, A2C and A52G.
M9082	Final station number disagreement	OFF: Final station number agreement ON: Final station number disagreement	Turned on when the final station number of the remote terminal modules and remote I/O modules connected to the A2C or A52G disagrees with the total number of stations set in the initial setting. Turned off when the final station number agrees with the total number of stations at STOP →RUN.	● Dedicated to A2C and A52G.
M9084	Error check	OFF: Checks enabled ON: Checks disabled	Specifies whether the following errors are to be checked or not after the END instruction is executed (to reduce END processing time): Fuse blown I/O unit verify error Battery error	○ Unusable with An, A2C and A3V.
M9086	BASIC program RUN flag	OFF: A3M-BASIC stop ON: A3M-BASIC run	Set when the A3M-BASIC is in RUN state and reset when it is in STOP state.	● Dedicated to A3M.
M9087	BASIC program PAUSE flag	OFF: A3M-BASIC RUN enable ON: A3M-BASIC disable	Specifies enable/disable of A3M-BASIC execution when the A3M-CPU is in PAUSE state. OFF: A3M-BASIC is executed. ON: A3M-BASIC is not executed.	● Dedicated to A3M.
M9089	Output at ERR terminal	OFF: no signal at ERR output ON: signal at ERR output	The internal relay is set, if the sequence program output an signal at the ERR terminals. The relay can only be reset, if M9089 and M9090 are reset simultaneously.	● Dedicated to A2C-CPU
M9090	Output at ERR terminal	OFF: no signal at ERR output ON: signal at ERR output	The internal relay is set, if an error occurs within MELSECNET/MINI or in the sequence program (when processing is stopped). The relay is reset, once the error in the network is cleared or the sequence program is restored.	● Dedicated to A2C-CPU
① M9091	Operation error detail flag	OFF: No error ON: error	Set when an operation error detail factor is stored at D9091 and remains set after normal state is restored.	● Usable with AnA(-F) and AnU.
	Microcomputer subroutine call error flag	OFF: No error ON: error	Set when an error occurred at execution of the microcomputer program package and remains set after normal state is restored.	○ Unusable with AnA(-F) and AnU.
②③ M9094	I/O change flag	OFF: Changed ON: Not changed	After the head address of the required I/O module is set to D9094, switching M9094 allows the I/O module to be changed in online mode. (One module is only allowed to be changed by one setting.) To be switched on in the program or peripheral device test mode to change the module during CPU RUN. To be switched on in peripheral device test mode to change the module during CPU STOP. RUN/STOP mode must not be changed until I/O module change is complete.	○ Unusable with An, A3H, A3V, A0J2H, A2C, A52G and A3N board.

**NOTE**

*After switching OFF the power supply, a latch clear or a RESET all special relays are reset.  
If the RUN key switch is switched to STOP the contents of the relays are retained.*

The special relays tagged ❶ even remain set, if the normal status is restored. They can be reset as follows:

- Insert a program line into the sequence program that resets the special relay via an RST instruction due to a specified execution condition.
- Force a RESET via a programming terminal.
- Reset the CPU by switching the key switch on the CPU to RESET.

The special relays tagged ❷ can only be set and reset by the sequence program.

The special relays tagged ❸ are set and reset in the test mode of a programming terminal.

# Overview of special relays

## A.4.3 Table of link relays (A series only)

Link relays are internal relays (in link operation) that are set or reset during data communications in a network depending on various conditions. Their status changes after the occurrence of an error in the program execution.

The processing of link relays depends on whether the CPU is installed in a master or a local station.

### Link relays in the master station

Number	Meaning	Status	Description
M9200	LRDP instruction received	OFF: Unreceived ON: Received	Depends on whether or not the LRDP (word device read) instruction has been received. Used in the program as an interlock for the LRDP instruction. Use the RST instruction to reset.
M9201	LRDP instruction complete	OFF: Incomplete ON: Complete	Depends on whether or not the LRDP (word device read) instruction execution is complete. Used as a condition contact for resetting M9200 and M9201 after the LRDP instruction is complete. Use the RST instruction to reset.
M9202	LWTP instruction received	OFF: Unreceived ON: Received	Depends on whether or not the LWTP (word device write) instruction has been received. Used in the program as an interlock for the LWTP instruction. Use the RST instruction to reset.
M9203	LWTP instruction complete	OFF: Incomplete ON: Complete	Depends on whether or not the LWTP (word device write) instruction execution is complete. Used as a condition contact to reset M9202 and M9203 after the LWTP instruction is complete. Use the RST instruction to reset.
M9206	Link parameter error in the host	OFF: Normal ON: Error	Depends on whether or not the link parameter setting of the host is valid.
M9207	Link parameter unmatched between master stations	OFF: Normal ON: Unmatched	Depends on whether or not the link parameter setting of the master station in tier two matches that of the master station in tier three in a three-tier system. (Valid only for the master stations in a three-tier system.)
M9208	Sets master station B and W transmission range (for lower link master stations only).	OFF: Transmits to tier 2 and tier 3 ON: Transmits to tier 2 only	The internal relay specifies, if the link data in B and W is transferred from the master station in the 1st level to the stations in the lower levels (sub stations). Data is only transferred if M9208 is not set.
M9209	Link parameter check command (for lower link master stations only).	OFF: Check ON: No check	The special relay is set, if the link devices (B and W) from the upper level should not be compared to the link devices (B and W) from the lower level. If M9209 is not set, the link devices from the upper and the lower levels are checked continuously.
M9210	Link card error	OFF: Normal ON: Error	Depends on presence or absence of the link card hardware error. Judged by the CPU.
M9224	Link status	OFF: Online ON: Offline, station-to-station test, or self-loopback test	Depends on whether the master station is online or offline or is in station-to-station test or self-loopback test mode.
M9225*	Forward loop error	OFF: Normal ON: Error	Depends on the error condition of the forward loop line.
M9226*	Reverse loop error	OFF: Normal ON: Error	Depends on the error condition of the reverse loop line.
M9227*	Loop test status	OFF: Unexecuted ON: Forward or reverse loop test being executed	Depends on whether or not the master station is executing a forward or a reverse loop test.
M9232	Local station operating status	OFF: RUN or STEP RUN mode ON: STOP or PAUSE mode	Depends on whether or not a local station is in STOP or PAUSE mode.



### Link relays in the master station

Number	Meaning	Status	Description
M9233	Local station error detect	OFF: No error ON: Error detected	Depends on whether or not a local station has detected an error in another station.
M9235	Local or remote I/O station parameter error detect	OFF: No error ON: Error detected	Depends on whether or not a local or a remote I/O station has detected any link parameter error in the master station.
M9236	Local or remote I/O station initial communicating status	OFF: Noncommunicating ON: Communicating	Depends on whether or not a local or a remote I/O station is communicating initial data (such as parameters) with the master station.
M9237	Local or remote I/O station error	OFF: Normal ON: Error	Depends on the error condition of a local or remote I/O station.
M9238*	Local or remote I/O station forward/reverse loop error	OFF: Normal ON: Error	Depends on the error condition of the forward and reverse loop lines of a local or a remote I/O station.

\* The tagged special relays cannot be applied within MELSECNET/B.

## Overview of special relays

### Link relays in the local station

Number	Meaning	Status	Description
M9204	LRDP instruction complete	OFF: Incomplete ON: Complete	On indicates that the LRDP instruction is complete at the local station.
M9205	LWTP instruction complete	OFF: Incomplete ON: Complete	On indicates that the LWTP instruction is complete at the local station.
M9211	Link card error (local station)	OFF: Normal ON: Error	Depends on presence or absence of the link card error. Judged by the CPU.
M9240*	Link status	OFF: Online ON: Offline, station-to-station test, or self-loopback test	Depends on whether the local station is online or offline, or is in station-to-station test or self-loopback test mode.
M9241*	Forward loop error	OFF: Normal ON: Error	Depends on the error condition of the forward loop line.
M9242*	Reverse loop error	OFF: Normal ON: Error	Depends on the error condition of the reverse loop line.
M9243	Loopback execution	OFF: NON-executed ON: Executed	Depends on whether or not loopback is occurring at the local station.
M9246	Data unreceived	OFF: Received ON: Unreceived	Depends on whether or not data has been received from the master station.
M9247	Data unreceived	OFF: Received ON: Unreceived	Depends on whether or not a tier three station has received data from its master station in a three-tier system.
M9250	Parameter unreceived	OFF: Received ON: Unreceived	Depends on whether or not link parameters have been received from the master station.
M9251	Link break	OFF: Normal ON: Break	Depends on the data link condition at the local station.
M9252	Loop test status	OFF: Unexecuted ON: Forward or reverse loop test is being executed.	Depends on whether or not the local station is executing a forward or a reverse loop test.
M9253	Master station operating status	OFF: RUN or STEP RUN mode ON: STOP or PAUSE mode	Depends on whether or not the master station is in STOP or PAUSE mode.
M9254	Operating status of other local stations	OFF: RUN or STEP RUN mode ON: STOP or PAUSE mode	Depends on whether or not a local station other than the host is in STOP or PAUSE mode.
M9255	Error status of other local stations	OFF: Normal ON: Error	Depends on whether or not a local station other than the host is in error.

\* The tagged special relays cannot be applied within MELSECNET/B.

## A.5 Table of Special Registers

### A.5.1 Table of special registers (MELSEC Q series and MELSEC System Q)

The special registers are internal registers with fixed applications in the programmable controller.

Therefore, they cannot be used like other special registers in a sequence program. However, data can be written to these registers in order to control the Q/QnA CPU. Data is usually stored in binary format except another format is required.

**NOTE**

*The special registers SD1200 to SD1255 are used for QnA CPU. These registers are vacant with a System Q CPU.*

*The special registers from SD1500 onward are dedicated for Q4AR CPU.*

The table below describes the meanings of the headings in the following table:

Item	Meaning
Number	Indicates the number of the special register.
Name	Indicates the name of the special register.
Meaning	Contains the function of the special register in brief.
Description	Contains a detailed description of the register.
Set by (if set)	Indicates whether the diagnostic special relay was set by the system or the user. <Set by> S : Set by the system U : Set by the user (via sequence program or a programming terminal in test mode) S/U : Set by the system or user  Is indicated only if the system set the status. <if set> END processing : Set during END processing Initial : Set during initial processing (Power ON, STOP->RUN) Status change : Set after status change Error : Set after error Instruction execution : Set during instruction execution Request : Set for user request (through SM, etc.)
Corresponding A CPU registers D9 [ ] [ ] [ ]	Indicates special register M9 [ ] [ ] [ ] corresponding to the A CPU (Change and notation when contents changed). Items indicated as "New" were newly added to the System Q/QnA CPU.
Valid for:	Indicates the corresponding CPU: ●: Can be applied to all types of CPU Q CPU: Can be applied to a CPU of the System Q QnA CPU: Can be applied to a CPU of the QnA series and Q2AS series CPU name: Can be applied only to the specific CPU (e.g. Q4AR CPU) Rem: Can be applied to a remote MELSECNET/H I/O module

For detailed information on the following topic refer to the manuals:

- Networks → Melsecnet/10 Network System Reference Manual for QnA
- SFC → QnA-CPU Programming Manual ( SFC )

# Table of Special Registers

## Table of special registers

### (1) Diagnostic information

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ][ ][ ]	Valid for:		
SD0	Diagnostic errors	Diagnosis error code	Error codes for errors found by diagnosis are stored as BIN data. Contents identical to latest fault history information.	S (Error)	D9008 format change			
SD1	Clock time for diagnosis error occurrence	Clock time for diagnosis error occurrence	Year (last two digits) and month that SD0 data was updated is stored as BCD 2-digit code. Example: <b>October 1995</b> <b>H9510</b> b15                      b8 b7                      b0 Year (0 to 99)                      Month (1 to 31)	S (Error)	New			
SD2			The day and hour that SD0 was updated is stored as BCD 2-digit code. Example: <b>10 p.m. on 25th</b> <b>H2510</b> b15                      b8 b7                      b0 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 50%;">Day (1 to 31)</td> <td style="width: 50%;">Hour (0 to 23)</td> </tr> </table>				Day (1 to 31)	Hour (0 to 23)
Day (1 to 31)			Hour (0 to 23)					
SD3	The minute and second that SD0 data was updated is stored as BCD 2-digit code. Example: <b>35 min 48s</b> <b>H3548</b> b15                      b8 b7                      b0 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 50%;">Minute (1 to 60)</td> <td style="width: 50%;">Second (1 to 60)</td> </tr> </table>	Minute (1 to 60)	Second (1 to 60)					
Minute (1 to 60)	Second (1 to 60)							
SD4	Error information categories	Error information category code	Category codes which help indicate what type of information is being stored in the common information areas (SD5 through SD15) and the individual information areas (SD16 through SD26) are stored here. b15                      b8 b7                      b0 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 50%;">Individual error info.</td> <td style="width: 50%;">Common error info.</td> </tr> </table> The common information category codes store the following codes: 0: No error 1: Unit/module No. 2: File name/Drive name 3: Time (value set) 4: Program error location  The individual information category codes store the following codes: 0: No error 1: (Open) 2: File name/Drive name 3: Time (value actually measured) 4: Program error location 5: Parameter number 6: Annunciator number 7: Check instruction malfunction number	Individual error info.	Common error info.	S (Error)	New	
Individual error info.	Common error info.							

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ] [ ]	Valid for:																																																																																																														
SD5	Error common information	Error common information	<p>Common information corresponding to the error codes (SD0) is stored here.</p> <p>The following four types of information are stored here:</p> <p>(1) Unit/module No.</p> <table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD5</td> <td>Station / module number</td> </tr> <tr> <td>SD6</td> <td>I/O number</td> </tr> <tr> <td>SD7</td> <td rowspan="8">Vacant</td> </tr> <tr> <td>SD8</td> </tr> <tr> <td>SD9</td> </tr> <tr> <td>SD10</td> </tr> <tr> <td>SD11</td> </tr> <tr> <td>SD12</td> </tr> <tr> <td>SD13</td> </tr> <tr> <td>SD14</td> </tr> <tr> <td>SD15</td> </tr> </tbody> </table> <p>(2) File name/Drive name</p> <p>Example: File name = ABCDEFGH.IJK</p> <table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD5</td> <td>Drive</td> </tr> <tr> <td>SD6</td> <td rowspan="2">File name</td> </tr> <tr> <td>SD7</td> <td>ASCII code: 8 characters</td> </tr> <tr> <td>SD8</td> <td rowspan="2">Extension</td> </tr> <tr> <td>SD9</td> <td>2EH(.)</td> </tr> <tr> <td>SD10</td> <td>ASCII code: 3 characters</td> </tr> <tr> <td>SD11</td> <td rowspan="5">Vacant</td> </tr> <tr> <td>SD12</td> </tr> <tr> <td>SD13</td> </tr> <tr> <td>SD14</td> </tr> <tr> <td>SD15</td> </tr> </tbody> </table> <table border="1"> <tr> <td>b15</td> <td>B</td> <td>A</td> </tr> <tr> <td></td> <td>D</td> <td>C</td> </tr> <tr> <td></td> <td>F</td> <td>E</td> </tr> <tr> <td></td> <td>H</td> <td>G</td> </tr> <tr> <td></td> <td>I</td> <td>.</td> </tr> <tr> <td></td> <td>K</td> <td>J</td> </tr> </table> <p>(3) Time (value set)</p> <table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD5</td> <td>Time: 1μs-steps (0 to 999 μs)</td> </tr> <tr> <td>SD6</td> <td>Time: 1ms-steps (0 to 999 ms)</td> </tr> <tr> <td>SD7</td> <td rowspan="8">Vacant</td> </tr> <tr> <td>SD8</td> </tr> <tr> <td>SD9</td> </tr> <tr> <td>SD10</td> </tr> <tr> <td>SD11</td> </tr> <tr> <td>SD12</td> </tr> <tr> <td>SD13</td> </tr> <tr> <td>SD14</td> </tr> <tr> <td>SD15</td> </tr> </tbody> </table> <p>(4) Program error location</p> <table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD5</td> <td rowspan="2">File name</td> </tr> <tr> <td>SD6</td> <td>(ASCII code: 8 characters)</td> </tr> <tr> <td>SD7</td> <td rowspan="2">Extension</td> </tr> <tr> <td>SD8</td> <td>2EH(.)</td> </tr> <tr> <td>SD9</td> <td>ASCII code: 3 characters</td> </tr> <tr> <td>SD10</td> <td>Pattern*</td> </tr> <tr> <td>SD11</td> <td>Block No.</td> </tr> <tr> <td>SD12</td> <td>Step / transition No.</td> </tr> <tr> <td>SD13</td> <td>Sequence step No. (L)</td> </tr> <tr> <td>SD14</td> <td>Sequence step No. (H)</td> </tr> <tr> <td>SD15</td> <td></td> </tr> </tbody> </table> <p>* Contents of pattern data</p> <table border="1"> <tr> <td>15</td><td>14</td><td>---</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>0</td><td>0</td><td>---</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td> </tr> </table> <p>← (Bit No.)</p> <p>not used</p> <p>SFC block designation present (1) / absent (0)</p> <p>SFC step designation present (1) / absent (0)</p> <p>SFC transition designation present (1) / absent (0)</p>	Number	Meaning	SD5	Station / module number	SD6	I/O number	SD7	Vacant	SD8	SD9	SD10	SD11	SD12	SD13	SD14	SD15	Number	Meaning	SD5	Drive	SD6	File name	SD7	ASCII code: 8 characters	SD8	Extension	SD9	2EH(.)	SD10	ASCII code: 3 characters	SD11	Vacant	SD12	SD13	SD14	SD15	b15	B	A		D	C		F	E		H	G		I	.		K	J	Number	Meaning	SD5	Time: 1μs-steps (0 to 999 μs)	SD6	Time: 1ms-steps (0 to 999 ms)	SD7	Vacant	SD8	SD9	SD10	SD11	SD12	SD13	SD14	SD15	Number	Meaning	SD5	File name	SD6	(ASCII code: 8 characters)	SD7	Extension	SD8	2EH(.)	SD9	ASCII code: 3 characters	SD10	Pattern*	SD11	Block No.	SD12	Step / transition No.	SD13	Sequence step No. (L)	SD14	Sequence step No. (H)	SD15		15	14	---	4	3	2	1	0	0	0	---	0	0	1	1	1	S (Error)	New	●
Number				Meaning																																																																																																																
SD5				Station / module number																																																																																																																
SD6				I/O number																																																																																																																
SD7				Vacant																																																																																																																
SD8																																																																																																																				
SD9																																																																																																																				
SD10																																																																																																																				
SD11																																																																																																																				
SD12																																																																																																																				
SD13																																																																																																																				
SD14																																																																																																																				
SD15																																																																																																																				
Number				Meaning																																																																																																																
SD5				Drive																																																																																																																
SD6	File name																																																																																																																			
SD7		ASCII code: 8 characters																																																																																																																		
SD8	Extension																																																																																																																			
SD9		2EH(.)																																																																																																																		
SD10	ASCII code: 3 characters																																																																																																																			
SD11	Vacant																																																																																																																			
SD12																																																																																																																				
SD13																																																																																																																				
SD14																																																																																																																				
SD15																																																																																																																				
b15	B	A																																																																																																																		
	D	C																																																																																																																		
	F	E																																																																																																																		
	H	G																																																																																																																		
	I	.																																																																																																																		
	K	J																																																																																																																		
Number	Meaning																																																																																																																			
SD5	Time: 1μs-steps (0 to 999 μs)																																																																																																																			
SD6	Time: 1ms-steps (0 to 999 ms)																																																																																																																			
SD7	Vacant																																																																																																																			
SD8																																																																																																																				
SD9																																																																																																																				
SD10																																																																																																																				
SD11																																																																																																																				
SD12																																																																																																																				
SD13																																																																																																																				
SD14																																																																																																																				
SD15																																																																																																																				
Number	Meaning																																																																																																																			
SD5	File name																																																																																																																			
SD6		(ASCII code: 8 characters)																																																																																																																		
SD7	Extension																																																																																																																			
SD8		2EH(.)																																																																																																																		
SD9	ASCII code: 3 characters																																																																																																																			
SD10	Pattern*																																																																																																																			
SD11	Block No.																																																																																																																			
SD12	Step / transition No.																																																																																																																			
SD13	Sequence step No. (L)																																																																																																																			
SD14	Sequence step No. (H)																																																																																																																			
SD15																																																																																																																				
15	14	---	4	3	2	1	0																																																																																																													
0	0	---	0	0	1	1	1																																																																																																													

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [][][]	Valid for
Meaning of the extensions:						
SD10 (SD9)		SD11 (SD10)		Extension name	File type	
Higher byte	Lower byte	Higher byte				
51H	50H	41H	QPA	Parameters		
51H	50H	47H	QPG	Sequence program		
51H	43H	44H	QCD	Device comment		
51H	44H	49H	QDI	Device initial value		
51H	44H	52H	QDR	File register		
51H	44H	53H	QDS	Simulation data		
51H	44H	4CH	QDL	Local device		
51H	54H	53H	QTS	Sampling trace data (QnA-CPU only)		
51H	54H	4CH	QTL	Status latch data (QnA-CPU only)		
51H	54H	50H	QTP	Program trace data (QnA-CPU only)		
51H	54H	52H	QTR	SFC trace file		
51H	46H	44H	QFD	Trouble history data		

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for																																						
SD16	Error individual information	Error individual information	<p>Individual information corresponding to the error codes (SD0) is stored here. The following six types of information are stored here:</p> <p>(1) File name/Drive name</p> <p>Example: <b>File name = ABCDEFGH.IJK</b></p> <table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD16</td> <td>Drive</td> </tr> <tr> <td>SD17</td> <td rowspan="4">File name ASCII code: 8 characters</td> </tr> <tr> <td>SD18</td> </tr> <tr> <td>SD19</td> </tr> <tr> <td>SD20</td> </tr> <tr> <td>SD21</td> <td>Extension 2E<sub>H</sub>(.)</td> </tr> <tr> <td>SD22</td> <td>ASCII code: 3 characters</td> </tr> <tr> <td>SD23</td> <td rowspan="4">Vacant</td> </tr> <tr> <td>SD24</td> </tr> <tr> <td>SD25</td> </tr> <tr> <td>SD26</td> </tr> </tbody> </table> <p>b15                      b0</p> <table border="1"> <tr> <td>B</td> <td>A</td> </tr> <tr> <td>D</td> <td>C</td> </tr> <tr> <td>F</td> <td>E</td> </tr> <tr> <td>H</td> <td>G</td> </tr> <tr> <td>I</td> <td>.</td> </tr> <tr> <td>K</td> <td>J</td> </tr> </table>	Number	Meaning	SD16	Drive	SD17	File name ASCII code: 8 characters	SD18	SD19	SD20	SD21	Extension 2E <sub>H</sub> (.)	SD22	ASCII code: 3 characters	SD23	Vacant	SD24	SD25	SD26	B	A	D	C	F	E	H	G	I	.	K	J	S (Error)	New	●								
Number				Meaning																																								
SD16				Drive																																								
SD17				File name ASCII code: 8 characters																																								
SD18																																												
SD19																																												
SD20																																												
SD21				Extension 2E <sub>H</sub> (.)																																								
SD22				ASCII code: 3 characters																																								
SD23				Vacant																																								
SD24																																												
SD25																																												
SD26																																												
B	A																																											
D	C																																											
F	E																																											
H	G																																											
I	.																																											
K	J																																											
SD17																																												
SD18																																												
SD19																																												
SD20																																												
SD21																																												
SD22																																												
SD23																																												
SD24																																												
SD25																																												
SD26	(2) Time (value actually measured)	<table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD16</td> <td>Time: 1μs-steps (0 to 999 μs)</td> </tr> <tr> <td>SD17</td> <td>Time: 1ms-steps (0 to 999 ms)</td> </tr> <tr> <td>SD18</td> <td rowspan="10">Vacant</td> </tr> <tr> <td>SD19</td> </tr> <tr> <td>SD20</td> </tr> <tr> <td>SD21</td> </tr> <tr> <td>SD22</td> </tr> <tr> <td>SD23</td> </tr> <tr> <td>SD24</td> </tr> <tr> <td>SD25</td> </tr> <tr> <td>SD26</td> </tr> </tbody> </table>	Number	Meaning	SD16	Time: 1μs-steps (0 to 999 μs)	SD17	Time: 1ms-steps (0 to 999 ms)	SD18	Vacant	SD19	SD20	SD21	SD22	SD23	SD24	SD25	SD26																										
Number	Meaning																																											
SD16	Time: 1μs-steps (0 to 999 μs)																																											
SD17	Time: 1ms-steps (0 to 999 ms)																																											
SD18	Vacant																																											
SD19																																												
SD20																																												
SD21																																												
SD22																																												
SD23																																												
SD24																																												
SD25																																												
SD26																																												
SD26		(3) Program error location	<table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD16</td> <td rowspan="4">File name (ASCII code: 8 characters)</td> </tr> <tr> <td>SD17</td> </tr> <tr> <td>SD18</td> </tr> <tr> <td>SD19</td> </tr> <tr> <td>SD20</td> <td>Extension 2E<sub>H</sub>(.)</td> </tr> <tr> <td>SD21</td> <td>(ASCII code: 3 characters)</td> </tr> <tr> <td>SD22</td> <td>Pattern*</td> </tr> <tr> <td>SD23</td> <td>Block No.</td> </tr> <tr> <td>SD24</td> <td>Step / transition No.</td> </tr> <tr> <td>SD25</td> <td>Sequence step No. (L)</td> </tr> <tr> <td>SD26</td> <td>Sequence step No. (H)</td> </tr> </tbody> </table> <p>* Contents of pattern data</p> <table border="1"> <tr> <td>15</td> <td>14</td> <td>...</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> <td>← (Bit No.)</td> </tr> <tr> <td>0</td> <td>0</td> <td>...</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td></td> </tr> </table> <p>not used                      SFC block designation present (1) / absent (0) SFC step designation present (1) / absent (0) SFC transition designation present (1) / absent (0)</p>	Number	Meaning	SD16	File name (ASCII code: 8 characters)	SD17	SD18	SD19	SD20	Extension 2E <sub>H</sub> (.)	SD21	(ASCII code: 3 characters)	SD22	Pattern*	SD23	Block No.	SD24	Step / transition No.	SD25	Sequence step No. (L)	SD26	Sequence step No. (H)	15	14	...	4	3	2	1	0	← (Bit No.)	0	0	...	0	0	0	0	0			
Number	Meaning																																											
SD16	File name (ASCII code: 8 characters)																																											
SD17																																												
SD18																																												
SD19																																												
SD20	Extension 2E <sub>H</sub> (.)																																											
SD21	(ASCII code: 3 characters)																																											
SD22	Pattern*																																											
SD23	Block No.																																											
SD24	Step / transition No.																																											
SD25	Sequence step No. (L)																																											
SD26	Sequence step No. (H)																																											
15	14	...	4	3	2	1	0	← (Bit No.)																																				
0	0	...	0	0	0	0	0																																					

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:																														
SD16	Error individual information	Error individual information	<p>(4) Parameter number</p> <p>(5) Annunciator number/ CHK instruction malfunction number</p> <table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD16</td> <td>Parameter No.</td> <td>SD16</td> <td>No.</td> </tr> <tr> <td>SD17</td> <td rowspan="10">Vacant</td> <td>SD17</td> <td rowspan="10">Vacant</td> </tr> <tr> <td>SD18</td> <td>SD18</td> </tr> <tr> <td>SD19</td> <td>SD19</td> </tr> <tr> <td>SD20</td> <td>SD20</td> </tr> <tr> <td>SD21</td> <td>SD21</td> </tr> <tr> <td>SD22</td> <td>SD22</td> </tr> <tr> <td>SD23</td> <td>SD23</td> </tr> <tr> <td>SD24</td> <td>SD24</td> </tr> <tr> <td>SD25</td> <td>SD25</td> </tr> <tr> <td>SD26</td> <td>SD26</td> </tr> </tbody> </table>	Number	Meaning	Number	Meaning	SD16	Parameter No.	SD16	No.	SD17	Vacant	SD17	Vacant	SD18	SD18	SD19	SD19	SD20	SD20	SD21	SD21	SD22	SD22	SD23	SD23	SD24	SD24	SD25	SD25	SD26	SD26	S (Error)	New	
Number				Meaning	Number	Meaning																														
SD16				Parameter No.	SD16	No.																														
SD17				Vacant	SD17	Vacant																														
SD18					SD18																															
SD19					SD19																															
SD20					SD20																															
SD21					SD21																															
SD22					SD22																															
SD23					SD23																															
SD24					SD24																															
SD25					SD25																															
SD26					SD26																															
SD17																																				
SD18																																				
SD19																																				
SD20																																				
SD21																																				
SD22																																				
SD23																																				
SD24																																				
SD25																																				
SD26	<p>(6) Intelligent function module parameter error (for System Q CPU only)</p> <table border="1"> <thead> <tr> <th>Number</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SD16</td> <td>Parameter No.</td> </tr> <tr> <td>SD17</td> <td>Error code for intelligent function module</td> </tr> <tr> <td>SD18</td> <td rowspan="9">Vacant</td> </tr> <tr> <td>SD19</td> </tr> <tr> <td>SD20</td> </tr> <tr> <td>SD21</td> </tr> <tr> <td>SD22</td> </tr> <tr> <td>SD23</td> </tr> <tr> <td>SD24</td> </tr> <tr> <td>SD25</td> </tr> <tr> <td>SD26</td> </tr> </tbody> </table>	Number	Meaning	SD16	Parameter No.	SD17	Error code for intelligent function module	SD18	Vacant	SD19	SD20	SD21	SD22	SD23	SD24	SD25	SD26																			
Number	Meaning																																			
SD16	Parameter No.																																			
SD17	Error code for intelligent function module																																			
SD18	Vacant																																			
SD19																																				
SD20																																				
SD21																																				
SD22																																				
SD23																																				
SD24																																				
SD25																																				
SD26																																				
SD50	Error reset	Error number that performs error reset	Stores error number that performs error reset	U	New																															
SD51	Battery low latch	Bit pattern indicating where battery voltage drop occurred	<p>All corresponding bits go ON when battery voltage drops. Subsequently, these remain ON even after battery voltage has been returned to normal.</p> <p>For a Q00J, Q00, and Q01CPU, only Bit 0 will be set.</p>	S (Error)	New																															
SD52	Battery low	Bit pattern indicating where battery voltage drop occurred	<p>Same configuration as SD51 above</p> <p>Subsequently, goes OFF when battery voltage is restored to normal.</p>	S (Error)	New																															
SD53	AC DOWN detection	Number of times for AC DOWN	1 is added to the stored value each time the input voltage becomes 80 % or less of the rating while the CPU module is operating, and the value is stored in BIN code.	S (Error)	D9005	Rem																														



## Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD54	MINI link errors	Error detection state	<p>(1) The relevant station bit goes ON when any of the installed MINI (-S3) X(n+0)/X(n+20), X(n+6)/(n+26), X(n+7)/(n+27) or X(n+8)/X(n+28) goes ON.</p> <p>(2) Goes ON when communications between the installed MINI (-S3) and the CPU are not possible.</p> <div style="text-align: center; margin-top: 10px;"> </div>	S (Error)	D9004 format change	QnA- CPU
SD60	Blown fuse number	Number of module with blown fuse	Value stored here is the lowest station number of the module with the blown fuse, divided by 16.	S (Error)	D9000	● Rem
SD61	I/O module verification error	I/O module verification error module number	The lowest number of the module where the I/O module verification number took place.	S (Error)	D9002	●
SD62	Annunciator number	Annunciator number	The first annunciator number to be detected is stored here.	S (Instruction execution)	D9009	●
SD63	Number of annunciators	Number of annunciators	Stores the number of annunciators searched.	S (Instruction execution)	D9124	●

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for																																																																																																																																																																																																																																										
SD64	Table of detected annunciator numbers	Annunciator detection number	<p>When F goes ON due to OUT F or SET F, the F numbers which go progressively ON from SD64 through SD79 are registered. F numbers turned OFF by RST F are deleted from SD64 to SD79, and are shifted to the data register following the data register where the deleted F numbers had been stored. Execution of the LEDR instruction shifts the contents of SD64 to SD79 up by one.</p> <p>(This can also be done by using the INDICATOR RESET switch on the front of the CPU of the Q3A/Q4ACPU.)</p> <p>After 16 annunciators have been detected, detection of the 17th will not be stored from SD64 through SD79.</p> <div style="text-align: center;"> <small>SET SET SET SET SET SET SET SET SET SET SET SET SET SET SET SET</small>  <small>F50 F25 F19 F25 F15 F70 F65 F38 F110F151F210 LEDR</small> </div> <p>SD62 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>99</td></tr></table> <small>Number detected</small></p> <p>SD63 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>8</td></tr></table> <small>Number of annunciators detected</small></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>SD64</td><td>0</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>50</td><td>99</td></tr> <tr><td>SD65</td><td>0</td><td>0</td><td>25</td><td>25</td><td>99</td><td>99</td><td>99</td><td>99</td><td>99</td><td>99</td><td>99</td><td>15</td></tr> <tr><td>SD66</td><td>0</td><td>0</td><td>0</td><td>99</td><td>0</td><td>15</td><td>15</td><td>15</td><td>15</td><td>15</td><td>15</td><td>15</td></tr> <tr><td>SD67</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>70</td><td>70</td><td>70</td><td>70</td><td>70</td><td>65</td></tr> <tr><td>SD68</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>65</td><td>65</td><td>65</td><td>65</td><td>65</td><td>38</td></tr> <tr><td>SD69</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>38</td><td>38</td><td>38</td><td>38</td><td>110</td></tr> <tr><td>SD70</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>110</td><td>110</td><td>110</td><td>151</td></tr> <tr><td>SD71</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>151</td><td>151</td><td>210</td></tr> <tr><td>SD72</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>210</td><td>0</td></tr> <tr><td>SD73</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>SD74</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>SD75</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>SD76</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>SD77</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>SD78</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>SD79</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	50	50	50	50	50	50	50	50	50	50	50	99	0	1	2	3	2	3	4	5	6	7	8	9	8	SD64	0	50	50	50	50	50	50	50	50	50	50	99	SD65	0	0	25	25	99	99	99	99	99	99	99	15	SD66	0	0	0	99	0	15	15	15	15	15	15	15	SD67	0	0	0	0	0	0	70	70	70	70	70	65	SD68	0	0	0	0	0	0	65	65	65	65	65	38	SD69	0	0	0	0	0	0	0	38	38	38	38	110	SD70	0	0	0	0	0	0	0	0	110	110	110	151	SD71	0	0	0	0	0	0	0	0	0	151	151	210	SD72	0	0	0	0	0	0	0	0	0	0	210	0	SD73	0	0	0	0	0	0	0	0	0	0	0	0	SD74	0	0	0	0	0	0	0	0	0	0	0	0	SD75	0	0	0	0	0	0	0	0	0	0	0	0	SD76	0	0	0	0	0	0	0	0	0	0	0	0	SD77	0	0	0	0	0	0	0	0	0	0	0	0	SD78	0	0	0	0	0	0	0	0	0	0	0	0	SD79	0	0	0	0	0	0	0	0	0	0	0	0	S (Instruction execution)	D9125	●
0				50	50	50	50	50	50	50	50	50	50	50	99																																																																																																																																																																																																																																	
0				1	2	3	2	3	4	5	6	7	8	9	8																																																																																																																																																																																																																																	
SD64				0	50	50	50	50	50	50	50	50	50	50	99																																																																																																																																																																																																																																	
SD65				0	0	25	25	99	99	99	99	99	99	99	15																																																																																																																																																																																																																																	
SD66				0	0	0	99	0	15	15	15	15	15	15	15																																																																																																																																																																																																																																	
SD67				0	0	0	0	0	0	70	70	70	70	70	65																																																																																																																																																																																																																																	
SD68				0	0	0	0	0	0	65	65	65	65	65	38																																																																																																																																																																																																																																	
SD69				0	0	0	0	0	0	0	38	38	38	38	110																																																																																																																																																																																																																																	
SD70				0	0	0	0	0	0	0	0	110	110	110	151																																																																																																																																																																																																																																	
SD71				0	0	0	0	0	0	0	0	0	151	151	210																																																																																																																																																																																																																																	
SD72				0	0	0	0	0	0	0	0	0	0	210	0																																																																																																																																																																																																																																	
SD73				0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																	
SD74				0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																	
SD75	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																				
SD76	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																				
SD77	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																				
SD78	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																				
SD79	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																				
SD65	D9126																																																																																																																																																																																																																																															
SD66	D9127																																																																																																																																																																																																																																															
SD67	D9128																																																																																																																																																																																																																																															
SD68	D9129																																																																																																																																																																																																																																															
SD69	D9130																																																																																																																																																																																																																																															
SD70	D9131																																																																																																																																																																																																																																															
SD71	D9132																																																																																																																																																																																																																																															
SD72	New																																																																																																																																																																																																																																															
SD74	New																																																																																																																																																																																																																																															
SD75	New																																																																																																																																																																																																																																															
SD76	New																																																																																																																																																																																																																																															
SD77	New																																																																																																																																																																																																																																															
SD78	New																																																																																																																																																																																																																																															
SD79	New																																																																																																																																																																																																																																															
SD80	CHK number	CHK number	Error codes detected by the CHK instruction are stored as BCD code.	S (Instruction execution)	New	● (except Q00J, Q00 and Q01CPU)																																																																																																																																																																																																																																										

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:					
SD90	Step transition watchdog timer setting value (Enabled only when SFC program exists)	F number for timer set value and time over error	Corresponds to SM90	<p>F numbers that are set ON at setting value of step transition watchdog timer and watchdog timer over errors.</p> <p>Timer is started by turning SM90 through SM99 ON during active step, and if the transition conditions for the relevant steps are not met within the timer limits, the designated annunciator (F) will go ON.</p>	U	D9108	● (except Q00J, Q00 and Q01CPU)				
SD91			Corresponds to SM91			D9109					
SD92			Corresponds to SM92			D9110					
SD93			Corresponds to SM93			D9111					
SD94			Corresponds to SM94			D9112					
SD95			Corresponds to SM95			D9113					
SD96			Corresponds to SM96			D9114					
SD97			Corresponds to SM97			New					
SD98			Corresponds to SM98			New					
SD99			Corresponds to SM99			New					
SD100			Transmission speed			Stores the transmission speed specified in the serial communication setting.		K96: 9600 bps, K192: 19.2 kbps, K384: 38.4 kbps, K576: 57.6 kbps, K1152: 115.2 kbps	S (power on or reset)	New	Q00JCPU Q00CPU Q01CPU
SD101			Communication settings			Stores the settings for serial communication		Bit 4 = OFF: Without sumcheck Bit 4 = ON: With sumcheck  Bit 5 = OFF: Online program correction disabled Bit 5 = ON: Online program correction enabled  The other bits have no function.		New	
SD102	Message waiting time	Stores the waiting time specified in the serial communication setting.	0: No waiting time 1 to F <sub>H</sub> : Waiting time (unit: 10 ms) Default: 0	New							
SD105	CH1 transmission speed setting (RS232)	Stores the present transmission speed.	K3: 300 bps, K6: 600 bps, K24: 2400 bps, K48: 4800 bps, K96: 9600 bps, K192: 19.2 kbps, K384: 38.4 kbps, K576: 57.6 kbps, K1152: 115.2 kbps	S	New	Q CPU (except Q00J, Q00 and Q01CPU)					
SD110	Data sending result	Stores the data sending result when the serial communication is used.	Stores the error code which occurred during transmission using the serial communication.	S (Error)	New	Q00JCPU Q00CPU Q01CPU					
SD111	Data receiving result	Stores the data receiving result when the serial communication is used.	Stores the error code which occurred when data was received using the serial communication.	S (Error)	New						
SD120	Error number for external power supply OFF	Module number which has external power supply error	Stores the smallest head number of the module whose external power supply is OFF.	S (Error)	New	Q CPU (except Q00J, Q00 and Q01CPU)					

# Table of Special Registers

## (2) System information

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:																																																																				
SD130	Modules with blown fuse	The bit pattern (16 Bit) indicates the modules with a blown fuse. 0 : No blown fuse 1 : Blown fuse detected	<ul style="list-style-type: none"> <li>The number of output modules whose fuses have blown are input as a bit pattern in units of 16 points. If the module numbers are set by parameter, the parameter-set numbers are stored.</li> <li>Blown fuses of remote station output modules will be detected also.</li> <li>A set bit is not automatically cleared when the module with the blown fuse is replaced. The flag is cleared by an error reset operation.</li> </ul> <div style="text-align: center;"> <table border="1" style="margin: 0 auto;"> <tr> <td></td> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>SD130</td> <td>0</td><td>0</td><td>0</td><td>1<sub>(YCD)</sub></td><td>0</td><td>0</td><td>0</td><td>1<sub>(YBD)</sub></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> <tr> <td>SD131</td> <td>1<sub>(YFD)</sub></td><td>0</td><td>0</td><td>0</td><td>0</td><td>1<sub>(Y1A)</sub></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table>   <table border="1" style="margin: 0 auto;"> <tr> <td>SD137</td> <td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1<sub>(YF3B)</sub></td><td>0</td><td>0</td><td>0</td> </tr> </table> <p style="text-align: center;">Blown fuse at the module with the head I/O number Y1F80.</p> </div>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	SD130	0	0	0	1 <sub>(YCD)</sub>	0	0	0	1 <sub>(YBD)</sub>	0	0	0	0	0	0	0	0	SD131	1 <sub>(YFD)</sub>	0	0	0	0	1 <sub>(Y1A)</sub>	0	0	0	0	0	0	0	0	0	0	SD137	0	0	0	0	1	0	0	0	0	0	0	0	1 <sub>(YF3B)</sub>	0	0	0	S (Error)	New	Q00JCPU Q00CPU Q01CPU
				b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																							
SD130				0	0	0	1 <sub>(YCD)</sub>	0	0	0	1 <sub>(YBD)</sub>	0	0	0	0	0	0	0	0																																																							
SD131				1 <sub>(YFD)</sub>	0	0	0	0	1 <sub>(Y1A)</sub>	0	0	0	0	0	0	0	0	0	0																																																							
SD137				0	0	0	0	1	0	0	0	0	0	0	0	1 <sub>(YF3B)</sub>	0	0	0																																																							
SD131																																																																										
SD132																																																																										
SD133																																																																										
SD134																																																																										
SD135																																																																										
SD136																																																																										
SD137																																																																										
SD150	I/O module verification error	The bit pattern (16 Bit) indicates the modules with verification errors. 0 : No I/O verification error 1 : I/O verification error present	<ul style="list-style-type: none"> <li>When the power is turned on, the module numbers of the I/O modules whose information differs from the registered I/O module information are set in this register (in units of 16 points).</li> <li>I/O module information is also detected.</li> </ul> <div style="text-align: center;"> <table border="1" style="margin: 0 auto;"> <tr> <td></td> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>SD150</td> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1<sub>(XYB0)</sub></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1<sub>(XYD)</sub></td> </tr> <tr> <td>SD151</td> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1<sub>(XY19C)</sub></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table>   <table border="1" style="margin: 0 auto;"> <tr> <td>SD157</td> <td>0</td><td>0</td><td>0</td><td>0</td><td>1<sub>(XYFB0)</sub></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table> <p style="text-align: center;">Verification error for the module with the head I/O number X/YFB0.</p> </div>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	SD150	0	0	0	0	0	0	0	1 <sub>(XYB0)</sub>	0	0	0	0	0	0	0	1 <sub>(XYD)</sub>	SD151	0	0	0	0	0	0	1 <sub>(XY19C)</sub>	0	0	0	0	0	0	0	0	0	SD157	0	0	0	0	1 <sub>(XYFB0)</sub>	0	0	0	0	0	0	0	0	0	0	0	S (Error)	New	
				b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																							
SD150				0	0	0	0	0	0	0	1 <sub>(XYB0)</sub>	0	0	0	0	0	0	0	1 <sub>(XYD)</sub>																																																							
SD151				0	0	0	0	0	0	1 <sub>(XY19C)</sub>	0	0	0	0	0	0	0	0	0																																																							
SD157				0	0	0	0	1 <sub>(XYFB0)</sub>	0	0	0	0	0	0	0	0	0	0	0																																																							
SD151																																																																										
SD152																																																																										
SD153																																																																										
SD154																																																																										
SD155																																																																										
SD156																																																																										
SD157																																																																										

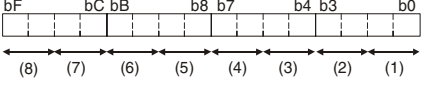
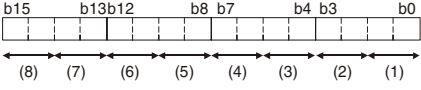
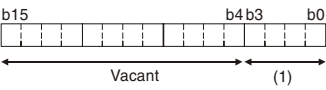
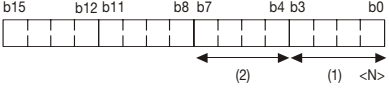
# Table of Special Registers

## (2) System information

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:				
SD200	State of switch	State of CPU switch	<p>The status of the remote I/O module is stored in the following format:</p> <div style="text-align: center;"> <p style="font-size: small;">(1) Remote I/O module switch status Always 1: STOP</p> </div>	S (Continuous)	New	Remote				
			<p>The CPU switch state is stored in the following format:</p> <div style="text-align: center;"> <table border="1" style="margin: 10px auto; font-size: x-small;"> <tr> <td>(1) CPU switch status</td> <td>(0): RUN (1): STOP</td> </tr> <tr> <td>(2) Memory card switch</td> <td>Always OFF</td> </tr> </table> </div>	(1) CPU switch status	(0): RUN (1): STOP	(2) Memory card switch	Always OFF		New	Q00JCPU Q00CPU Q01CPU
			(1) CPU switch status	(0): RUN (1): STOP						
			(2) Memory card switch	Always OFF						
<p>The CPU switch state is stored in the following format:</p> <div style="text-align: center;"> <table border="1" style="margin: 10px auto; font-size: x-small;"> <tr> <td>(1) CPU switch status</td> <td>(0): RUN (1): STOP (2): L.CLR</td> </tr> <tr> <td>(2) Memory card switch</td> <td>Always OFF</td> </tr> <tr> <td>(3) DIP-Switch</td> <td>b8 to bC correspond to SW1 through SW5 of system setting switch 1. 0: OFF, 1: ON bD,bE and bF are vacant</td> </tr> </table> </div>	(1) CPU switch status	(0): RUN (1): STOP (2): L.CLR	(2) Memory card switch	Always OFF	(3) DIP-Switch	b8 to bC correspond to SW1 through SW5 of system setting switch 1. 0: OFF, 1: ON bD,bE and bF are vacant	S (Every END processing)	New	Q CPU (except Q00J, Q00 and Q01CPU)	
(1) CPU switch status	(0): RUN (1): STOP (2): L.CLR									
(2) Memory card switch	Always OFF									
(3) DIP-Switch	b8 to bC correspond to SW1 through SW5 of system setting switch 1. 0: OFF, 1: ON bD,bE and bF are vacant									
<p>The CPU switch state is stored in the following format:</p> <div style="text-align: center;"> <table border="1" style="margin: 10px auto; font-size: x-small;"> <tr> <td>(1): CPU Status</td> <td>(0): RUN (1): STOP (2): L.CLR</td> </tr> <tr> <td>(2): Memory card switch</td> <td>B4 corresponds to card A, B5 corresponds to card B OFF for 0; ON for 1</td> </tr> <tr> <td>(3): DIP switch</td> <td>B8 to B15 correspond to SW1 to SW8 OFF for 0; ON for 1</td> </tr> </table> </div>	(1): CPU Status	(0): RUN (1): STOP (2): L.CLR	(2): Memory card switch	B4 corresponds to card A, B5 corresponds to card B OFF for 0; ON for 1	(3): DIP switch	B8 to B15 correspond to SW1 to SW8 OFF for 0; ON for 1	S (Every END processing)	New	QnA CPU	
(1): CPU Status	(0): RUN (1): STOP (2): L.CLR									
(2): Memory card switch	B4 corresponds to card A, B5 corresponds to card B OFF for 0; ON for 1									
(3): DIP switch	B8 to B15 correspond to SW1 to SW8 OFF for 0; ON for 1									

# Table of Special Registers

## (2) System information

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD201	LED status	State of CPU-LED	<p>The following bit patterns are used to store the statuses of the LEDs of the CPU:</p>  <p>(1) : RUN (5) : BOOT            (2) : ERROR (6) : Vacant            (3) : USER (7) : Vacant            (4) : BAT.ALARM (8) : MODE            Bitpatterns for MODE            0 : OFF            1 : Green            2 : Orange</p> <p>The areas 3 to 8 are not available for a Q00JCPU, Q00CPU or Q01CPU.</p>	S (Status change)	New	System Q CPU
			<p>Information concerning which of the following states the LEDs on the CPU are stored in the following bit patterns:            0 is off, 1 is on, and 2 is flicker</p>  <p>(1) : RUN (5) : BOOT            (2) : ERROR (6) : Card A (memory card)            (3) : USER (7) : Card B (memory card)            (4) : BAT.ALARM (8) : Vacant</p>	S (Status change)	New	QnA CPU
SD202	LED off	Bit pattern of LED that is turned off	<p>Stored bit patterns of LEDs turned off            (Only USER and BOOT enabled)            Turned off at 1, not turned off at 0</p>	U	New	QnA CPU
SD203	Operating state of CPU	Operating state of CPU	<p>The operating status of the remote I/O module is stored in the following format:</p>  <p>(1) Remote I/O module operating status      Always 2: STOP</p>	S (Continuous)	New	Remote
			<p>The CPU operating state is stored as indicated in the following figure:</p>  <div style="border: 1px solid black; padding: 5px;"> <p>(1) : Operating state of CPU0 : RUN              1 : STEP-RUN              2 : STOP              3 : PAUSE</p> <p>(2) : STOP/PAUSE cause              0 : Key switch              1 : Remote contact              2 : Peripheral, computer link, or operation from some other remote source              3 : Internal program instruction              4 : Error</p> <p>Remark: Only the error that occurred first is stored.</p> </div>	S (Every END processing)	D9015 (format change)	●

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ] [ ]	Valid for:															
SD206	Device test execution type	Indicates the kind of device test	When a device test is being executed by a programming device, the contents of this register reflects the state of the test: 0 = Test not yet executed 1 = Test of input devices (X) 2 = Test of output devices (Y) 3 = Test of input and output devices (X/Y)	S (Request)	New	Remote															
SD207	LED display priority ranking	Priorities 1 to 4	When error is generated, the LED display (flicker) is made according to the error number setting priorities. The setting areas for priorities are as follows:  <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">SD207</td> <td style="padding: 2px;">Priority 4</td> <td style="padding: 2px;">Priority 3</td> <td style="padding: 2px;">Priority 2</td> <td style="padding: 2px;">Priority 1</td> </tr> <tr> <td style="padding: 2px;">SD208</td> <td style="padding: 2px;">Priority 8</td> <td style="padding: 2px;">Priority 7</td> <td style="padding: 2px;">Priority 6</td> <td style="padding: 2px;">Priority 5</td> </tr> <tr> <td style="padding: 2px;">SD209</td> <td colspan="2" style="padding: 2px;"></td> <td style="padding: 2px;">Priority 10</td> <td style="padding: 2px;">Priority 9</td> </tr> </table> <p style="text-align: center; margin-left: 100px;">                         (4321<sub>H</sub>)                          (8765<sub>H</sub>)                          (00A9<sub>H</sub>)                     </p> No display is made if "0" is set. However, even if "0" has been set, information concerning CPU operation stop (including parameter settings) errors will be indicated by the LEDs without conditions.	SD207	Priority 4	Priority 3	Priority 2	Priority 1	SD208	Priority 8	Priority 7	Priority 6	Priority 5	SD209			Priority 10	Priority 9	U	D9038	● (except Q00J, Q00 and Q01CPU)
SD207		Priority 4		Priority 3	Priority 2	Priority 1															
SD208		Priority 8		Priority 7	Priority 6	Priority 5															
SD209			Priority 10	Priority 9																	
SD208	Priorities 5 to 8	D9039 (format change)																			
SD209	Priorities 9 to 10	New																			
SD210	Clock data	Clock data (year, month)	The year (last two digits) and month are stored as BCD code at SD210 as shown below:  <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 0 5px;">b15</td><td style="padding: 0 5px;">b12b11</td><td style="padding: 0 5px;">b8 b7</td><td style="padding: 0 5px;">b4 b3</td><td style="padding: 0 5px;">b0</td> </tr> <tr> <td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td> </tr> <tr> <td colspan="2" style="text-align: center;">Year</td><td colspan="3" style="text-align: center;">Month</td> </tr> </table> Example: July 1993 = <b>H9307</b>	b15	b12b11	b8 b7	b4 b3	b0						Year		Month			S/U (Request)	D9025	● Rem
b15	b12b11	b8 b7	b4 b3	b0																	
Year		Month																			
SD211	Clock data	Clock data (day, hour)	The day and hour are stored as BCD code at SD211 as shown below:  <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 0 5px;">b15</td><td style="padding: 0 5px;">b12b11</td><td style="padding: 0 5px;">b8 b7</td><td style="padding: 0 5px;">b4 b3</td><td style="padding: 0 5px;">b0</td> </tr> <tr> <td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td> </tr> <tr> <td colspan="2" style="text-align: center;">Day</td><td colspan="3" style="text-align: center;">Hour</td> </tr> </table> Example: 31st, 10 a. m. = <b>H3110</b>	b15	b12b11	b8 b7	b4 b3	b0						Day		Hour			D9026		
b15	b12b11	b8 b7	b4 b3	b0																	
Day		Hour																			
SD212	Clock data	Clock data (minute, second)	The minutes and seconds (after the hour) are stored as BCD code at SD212 as shown below:  <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 0 5px;">b15</td><td style="padding: 0 5px;">b12b11</td><td style="padding: 0 5px;">b8 b7</td><td style="padding: 0 5px;">b4 b3</td><td style="padding: 0 5px;">b0</td> </tr> <tr> <td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td> </tr> <tr> <td colspan="2" style="text-align: center;">Minute</td><td colspan="3" style="text-align: center;">Second</td> </tr> </table> Example: 35 min, 48 sec. = <b>H3548</b>	b15	b12b11	b8 b7	b4 b3	b0						Minute		Second			D9027		
b15	b12b11	b8 b7	b4 b3	b0																	
Minute		Second																			

# Table of Special Registers

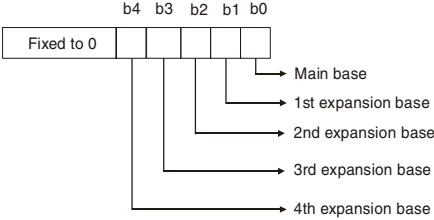
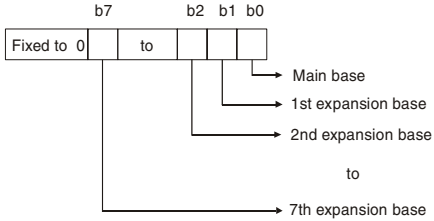
Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:																								
SD213	Clock data	Clock data (day of the week)	<p>The day of the week is stored as BCD code at SD213 as shown below:</p> <p>Higher digits of year (0 to 99)</p> <table border="1"> <tr><th>Day of week</th></tr> <tr><td>0</td><td>Sunday</td></tr> <tr><td>1</td><td>Monday</td></tr> <tr><td>2</td><td>Tuesday</td></tr> <tr><td>3</td><td>Wednesday</td></tr> <tr><td>4</td><td>Thursday</td></tr> <tr><td>5</td><td>Friday</td></tr> <tr><td>6</td><td>Saturday</td></tr> </table>	Day of week	0	Sunday	1	Monday	2	Tuesday	3	Wednesday	4	Thursday	5	Friday	6	Saturday	S/U (Request)	D9028	Q CPU Rem									
			Day of week																											
0	Sunday																													
1	Monday																													
2	Tuesday																													
3	Wednesday																													
4	Thursday																													
5	Friday																													
6	Saturday																													
<p>The day of the week is stored as BCD code at SD213 as shown below:</p> <p>Always "0"</p> <table border="1"> <tr><th>Day of week</th></tr> <tr><td>0</td><td>Sunday</td></tr> <tr><td>1</td><td>Monday</td></tr> <tr><td>2</td><td>Tuesday</td></tr> <tr><td>3</td><td>Wednesday</td></tr> <tr><td>4</td><td>Thursday</td></tr> <tr><td>5</td><td>Friday</td></tr> <tr><td>6</td><td>Saturday</td></tr> </table>	Day of week	0	Sunday	1	Monday	2	Tuesday	3	Wednesday	4	Thursday	5	Friday	6	Saturday	S/U (Request)	QnA CPU													
Day of week																														
0	Sunday																													
1	Monday																													
2	Tuesday																													
3	Wednesday																													
4	Thursday																													
5	Friday																													
6	Saturday																													
SD220	LED display data	Display indicator data	<p>LED display ASCII data (16 characters) stored here.</p> <p>b15 to b8    b7 to b0</p> <table border="1"> <tr> <td>SD220</td> <td>15th character from the right</td> <td>16th character from the right</td> </tr> <tr> <td>SD221</td> <td>13th character from the right</td> <td>14th character from the right</td> </tr> <tr> <td>SD222</td> <td>11th character from the right</td> <td>12th character from the right</td> </tr> <tr> <td>SD223</td> <td>9th character from the right</td> <td>10th character from the right</td> </tr> <tr> <td>SD224</td> <td>7th character from the right</td> <td>8th character from the right</td> </tr> <tr> <td>SD225</td> <td>5th character from the right</td> <td>6th character from the right</td> </tr> <tr> <td>SD226</td> <td>3rd character from the right</td> <td>4th character from the right</td> </tr> <tr> <td>SD227</td> <td>1st character from the right</td> <td>2nd character from the right</td> </tr> </table>	SD220	15th character from the right	16th character from the right	SD221	13th character from the right	14th character from the right	SD222	11th character from the right	12th character from the right	SD223	9th character from the right	10th character from the right	SD224	7th character from the right	8th character from the right	SD225	5th character from the right	6th character from the right	SD226	3rd character from the right	4th character from the right	SD227	1st character from the right	2nd character from the right	S (Status change)	New	●
SD220			15th character from the right	16th character from the right																										
SD221			13th character from the right	14th character from the right																										
SD222			11th character from the right	12th character from the right																										
SD223			9th character from the right	10th character from the right																										
SD224			7th character from the right	8th character from the right																										
SD225			5th character from the right	6th character from the right																										
SD226			3rd character from the right	4th character from the right																										
SD227	1st character from the right	2nd character from the right																												
SD221																														
SD222																														
SD223																														
SD224																														
SD225																														
SD226																														
SD227																														
SD240	Base mode	0: Automatic mode 1: Detail mode	Stores the base mode	S (Initial)	New	Q CPU Rem																								
SD241	Number of extension bases	0: Basic only 1 to 7: Number of extension bases	Stores the number of extension bases being installed	S (Initial)	New																									



# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD242	A/Q base differentiation	0: QA [ ] [ ] B is installed (A mode) 1: Q [ ] [ ] B is installed (Q mode)		S (Initial)	New	Q00JCPU Q00CPU Q01CPU
						System Q CPU (except Q00JCPU, Q00CPU, Q01CPU)
SD243	Number of base slots	Number of base slots The areas for the 5th to 7th expansion base are fixed to "0" for a Q00JCPU, Q00CPU or Q01CPU		S (Initial)	New	System Q CPU
SD244			The number of slots being installed is stored in the respective areas for the basic base and the extension bases (ext.).			
SD250	Loaded maximum I/O	Loaded maximum I/O No.	When SM250 goes from OFF to ON, the upper 2 digits of the final I/O number plus 1 of the modules loaded are stored as BIN values.	S (Request END)	New	●
SD251	Head I/O No. for replacement	Head I/O number for module replacement	Stores upper two digits of the first I/O number of an I/O module that is removed/replaced in the online status.	U	D9094	Q2A (S1) Q3A Q4A Q4AR
SD253	RS422 baud rate	RS422 baud rate	Stores the baud rate of RS422: 0: 9600 bps, 1: 19,2 bps, 2: 38,4 bps	S (When changed)	New	QnA CPU

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD254	MELSECNET/10 information	Number of modules installed	Indicates the number of modules installed on NET/10	S (Initial)	New	●          (except Q00JCPU Q00CPU Q01CPU)
SD255		I/O No.	NET/10 I/O number of first module installed			
SD256		Network No.	NET/10 network number of first module installed			
SD257		Group Number	NET/10 group number of first module installed			
SD258		Station No.	NET/10 station number of first module installed			
SD259		Standby information	In the case of standby stations, the module number of the standby station is stored. (1 to 4)			
SD260 – SD264		Information from 2nd module	Configuration is identical to that for the first module.			
SD265 – SD269		Information from 3rd module	Configuration is identical to that for the first module.			
SD270 – SD274		Information from 4th module	Configuration is identical to that for the first module.			
SD280	CC-Link error	Error detection status	<p>(1) When Xn0 of the installed CC-Link goes ON, the bit corresponding to the station switches ON.</p> <p>(2) When either Xn1 or XnF of the installed CC-Link switch OFF, the bit corresponding to the station switches ON.</p> <p>(3) Switches ON when the CPU cannot communicate with the installed CC-Link.</p>	S (error)	New	Q CPU
			<p>(1) When Xn0 of the installed CC-Link goes ON, the bit corresponding to the station switches ON.</p> <p>(2) When either Xn1 or XnF of the installed CC-Link switch OFF, the bit corresponding to the station switches ON.</p>	S (error)	New	QnA

## Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:	
SD290	Device allocation (Same as parameter contents)	Number of points allocated for X	● Stores the number of points currently set for X	S (Initial)	New	● Rem	
SD291		Number of points allocated for Y	● Stores the number of points currently set for Y				
SD292		Number of points allocated for M	● Stores the number of points currently set for M				
SD293		Number of points allocated for L	● Stores the number of points currently set for L			●	
SD294		Number of points allocated for B	● Stores the number of points currently set for B				● Rem
SD295		Number of points allocated for F	● Stores the number of points currently set for F				●
SD296		Number of points allocated for SB	● Stores the number of points currently set for SB				● Rem
SD297		Number of points allocated for V	● Stores the number of points currently set for V				●
SD298		Number of points allocated for S	● Stores the number of points currently set for S				
SD299		Number of points allocated for T	● Stores the number of points currently set for T				
SD300		Number of points allocated for ST	● Stores the number of points currently set for ST			●	
SD301		Number of points allocated for C	● Stores the number of points currently set for C				
SD302		Number of points allocated for D	● Stores the number of points currently set for D				
SD303	Device allocation (Same as parameter contents)	Number of points allocated for W	● Stores the number of points currently set for W	● Rem			
SD304	Number of points allocated for SW	● Stores the number of points currently set for SW					
SD315	Time reserved for communication processing	Time reserved for communication processing	Reserves the designated time for communication processing with the GX developer or other units. The greater the value is designated, the shorter the response time for communication with other devices (GX Developer, serial communication units) becomes. Setting range: 1 to 100 ms. If the specified value is out of range, it is assumed to no setting. The scan time becomes longer by the specified time.	END processing	New	System Q CPU	

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ] [ ]	Valid for:	
SD340	Ethernet information	Number of modules installed	● Indicates the number of modules installed on Ethernet.	S (Initial)	New	System Q CPU Rem	
SD341		Information from 1st module	I/O number				● Ethernet I/O number of the first module installed.
SD342			Network number				● Ethernet network number of the first module installed.
SD343			Group number				● Ethernet group number of the first module installed.
SD344			Station number				● Ethernet station number of the first module installed.
SD345 and SD346			Vacant			● Vacant (With a System Q CPU, the Ethernet IP address of the first module is stored in buffer memory.)	
SD347		Vacant	● Vacant (With System Q CPU, the Ethernet error code of the first module is read with the ERRORRD instruction.)				
SD348 to SD354		Information from 2nd module	● Configuration is identical to that for the first module.				
SD355 to SD361		Information from 3rd module	● Configuration is identical to that for the first module.				
SD362 to SD368		Information from 4th module	● Configuration is identical to that for the first module.				
SD340	Ethernet information	Number of modules installed	● Indicates the number of modules installed on Ethernet.	S (Initial)	New	QnA CPU	
SD341		Information from 1st module	I/O number				● Ethernet I/O number of the first module installed.
SD342			Network number				● Ethernet network number of the first module installed.
SD343			Group number				● Ethernet group number of the first module installed.
SD344			Station number				● Ethernet station number of the first module installed.
SD345 and SD346			IP address				● Ethernet IP address of the first module installed.
SD347		Error code	● Ethernet error code of the first module installed.				
SD348 to SD354		Information from 2nd module	● Configuration is identical to that for the first module.				
SD355 to SD361		Information from 3rd module	● Configuration is identical to that for the first module.				
SD362 to SD368		Information from 4th module	● Configuration is identical to that for the first module.				

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ][ ][ ]	Valid for:
SD380	Ethernet instruction reception status	Instruction reception status of the 1st module	<p>b15 b8 b7 b6 b5 b4 b3 b2 b1 b0</p> <p>0 ... 0 Not used</p> <p>Status of channel 1 Status of channel 2 Status of channel 3 Status of channel 4 Status of channel 5 Status of channel 6 Status of channel 7 Status of channel 8</p> <p>ON: Received (Channel is used) OFF: Not received (Channel is not used)</p>	S (Initial)	New	QnA CPU
SD381		Instruction reception status of the 2nd module	● Configuration is identical to that for the first module.			
SD382		Instruction reception status of the 3rd module	● Configuration is identical to that for the first module.			
SD383		Instruction reception status of the 4th module	● Configuration is identical to that for the first module.			
SD392	Software version	Internal system software version	<ul style="list-style-type: none"> <li>The software version is stored in the high byte of SD392 in ASCII code. For example, Version "A" is stored as 41<sub>H</sub>.</li> <li>The data in the low byte is indefinite.</li> <li>Note: The internal system software version may differ from the version indicated by the version symbol printed on the case.</li> </ul>	S (Initial)	D9060	
SD395	Multiple CPU number	1: CPU No. 1 2: CPU No. 2 3: CPU No. 3 4: CPU No. 4	Stores the number of the CPU when she is operated in a multi-CPU system.	S (Initial)	New	Q02CPU Q02HCPU Q06HCPU Q12HCPU Q25HCPU with function Ver. B or later

# Table of Special Registers

## (3) System clocks/counters

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD412	1 second counter	Number of counts in 1-second units	Following programmable controller CPU RUN, 1 is added each second. Count repeats from 0 to 32767 to -32768 to 0	S (Status change)	D9022	●
SD414	n = 1 second steps	2n second clock units	Stores value n of 2n second clock (Default is 30). Setting can be made between 1 and 32767.	U	New	
SD415	n = 1 ms steps	2n ms clock units	Stores value n of 2n ms clock (Default is 30). Setting can be made between 1 and 32767.	U	New	System Q CPU (except Q00JCPU Q00CPU Q01CPU)
SD420	Scan counter	Number of counts in each scan	Incremented by 1 for each scan execution after the PC CPU is set to RUN. Count repeats from 0 to 32767 to -32768 to 0.	S (Every END processing)	New	●
SD430	Low speed scan counter	Number of counts in each scan	Incremented by 1 for each scan execution after the PC CPU is set to RUN. Count repeats from 0 to 32767 to -32768 to 0. Used only for low speed execution type programs.	S (Every END processing)	New	● (except Q00JCPU Q00CPU Q01CPU)

# Table of Special Registers

## (4) Scan information

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD500	Execution program No.	Execution type of program being executed	Program number of program currently being executed is stored as BIN value.	S (Status change)	New	●  (except Q00JCPU Q00CPU Q01CPU)
SD510	Low speed program No.	File name of low speed execution in progress	Program number of low speed program currently being executed is stored as BIN value. Enabled only when SM510 is ON.	S (Every END processing)	New	
SD520	Current scan time	Current scan time (in 1 ms units)	Stores current scan time (in 1 ms units) Range from 0 to 65535	S (Every END processing)	D9017 (format change)	●
SD521		Current scan time (in 1 μs units)	Stores current scan time (in 1 μs units) Range from 00000 to 900 (Example) A current scan of 23.6 ms would be stored as follows: D520 = 23 D521 = 600		New	
SD522	Initial scan time	Initial scan time (in 1 ms units)	Stores scan time for first scan (in 1 ms units). Range from 0 to 65535	S (First END processing)	New	●  (except Q00JCPU Q00CPU Q01CPU)
SD523		Initial scan time (in 100 μs units)	Stores scan time for first scan (in 1 μs units). Range of 000 to 900			
SD524	Minimum scan time	Minimum scan time (in 1 ms units)	Stores minimum value of scan time (in 1 ms units). Range from 0 to 65535	S (Every END processing)	D9018 (format change)	●
SD525		Minimum scan time (in 100 μs units)	Stores minimum value of scan time (in 100 μs units). Range of 000 to 900		New	
SD526	Maximum scan time	Maximum scan time (in 1 ms units)	Stores maximum value of scan time, excepting the first scan. (in 1 ms units). Range from 0 to 65535	S (Every END processing)	D9019 (format change)	●
SD527		Maximum scan time (in 100 μs units)	Stores maximum value of scan time, excepting the first scan. (in 100 μs units). Range of 000 to 900		New	
SD528	For low speed execution type programs current scan time	Current scan time (in 1 ms units)	Stores current scan time for low speed execution type program (in 1 ms units).	S (Every END processing)	New	●
SD529		Current scan time (in 100 μs units)	Stores current scan time for low speed execution type program (in 100 μs units). Range of 000 to 900			
SD532	Minimum scan time for low speed execution type programs	Minimum scan time (in 1 ms units)	Stores minimum value of scan time for low speed execution type program (in 1 ms units). Range from 0 to 65535	S (Every END processing)	New	●  (except Q00JCPU Q00CPU Q01CPU)
SD533		Minimum scan time (in 100 μs units)	Stores minimum value of scan time for low speed execution type program (in 100 μs units). Range of 000 to 900			
SD534	Maximum scan time for low speed execution type programs	Maximum scan time (in 1 ms units)	Stores the maximum scan time for all except low speed execution type program s first scan (in 1 ms units). Range from 0 to 65535	S (Every END processing)	New	●
SD535		Maximum scan time (in 100 μs units)	Stores the maximum scan time for all except low speed execution type program s first scan (in 100 μs units). Range of 000 to 900			

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD540	END processing time	END processing time (in 1 ms units)	Stores time from completion of scan program to start of next scan (in 1 ms units). Range from 0 to 65535	S (Every END processing)	New	●
SD541		END processing time (in 100 μs units)	Stores time from completion of scan program to start of next scan (in 100 μs units). Range of 000 to 900			
SD542	Constant scan wait time	Constant scan wait time (in 1 ms units)	Stores wait time when constant scan time has been set (in 1 ms units). Range from 0 to 65535	S (First END processing)	New	
SD543		Constant scan wait time (in 100 μs units)	Stores wait time when constant scan time has been set (in 100 μs units). Range of 000 to 900			
SD544	Cumulative execution time for low speed execution type programs	Cumulative execution time for low speed execution type programs (in 1 ms units)	Stores cumulative execution time for low speed execution type programs (in 1 ms units). Range from 0 to 65535 Cleared to 0 following 1 low speed scan	S (Every END processing)	New	● (except Q00JCPU Q00CPU Q01CPU)
SD545		Cumulative execution time for low speed execution type programs (in 100 μs units)	Stores cumulative execution time for low speed execution type programs (in 100 μs units). Range of 000 to 900 Cleared to 0 following 1 low speed scan			
SD546	Execution time for low speed execution type programs	Execution time for low speed execution type programs (in 1 ms units)	Stores low speed program execution time during 1 scan (in 1 ms units). Range from 0 to 65535 Stores each scan	S (Every END processing)	New	
SD547		Execution time for low speed execution type programs (in 100 μs units)	Stores low speed program execution time during 1 scan (in 100 μs units). Range of 000 to 900 Stores each scan			
SD548	Scan program execution time	Scan program execution time (in 1 ms units)	Stores execution time for scan execution type program during 1 scan (in 1 ms units). Range from 0 to 65535 Stores each scan	S (Every END processing)	New	●
SD549		Scan program execution time (in 100 μs units)	Stores execution time for scan execution type program during 1 scan (in 100 μs units). Range of 000 to 900 Stores each scan			
SD550	Service interval measurement module	Unit/module No.	Sets I/O number for module that measures service interval.	U	New	
SD551	Service interval time	Module service interval (in 1 ms units)	When SM 551 is ON, stores service interval for module designated by SD 550 (in 1 ms units). Range from 0 to 65535	S (Request)	New	● (except Q00JCPU Q00CPU Q01CPU)
SD552		Module service interval (in 100 μs units)	When SM551 is ON, stores service interval for module designated by SD550 (in 1 μs units). Range from 000 to 999			



# Table of Special Registers

## (5) Memory cards

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:		
SD600	Memory card A models	Memory card A models	Indicates memory card A model installed. 	S (Initial and card removal)	New	System Q CPU (except Q00JCPU, Q00CPU, Q01CPU)		
			Indicates memory card A model installed. 	S (Initial and card removal)	New	QnA CPU		
SD602	Drive 1 (RAM) capacity	Drive 1 capacity	Drive 1 capacity is stored in 1 k byte units	S (Initial and card removal)	New	●		
SD603	Drive 2 (ROM) capacity	Drive 2 capacity	Drive 2 capacity is stored in 1 k byte units	S (Initial and card removal)	New	(except Q00JCPU, Q00CPU, Q01CPU)		
SD604	Memory card A use conditions	Memory card A use conditions	The use conditions for memory card A are stored as bit patterns (in use when ON). The significance of these bit patterns is indicated below: <table border="1" style="width: 100%; margin-top: 10px;"> <tr> <td style="width: 50%; vertical-align: top;">                             b0 : BOOT operation (QBT)                              b1 : Parameters (QPT)                              b2 : Device comments (QCD)                              b3 : Device initial value (QDI)                              b4 : File Register (QDR)                              b5 : Trace (QTS)                              b6 :                              b7 :                         </td> <td style="width: 50%; vertical-align: top;">                             b8 :                              b9 : CPU fault history (QFD)                              bA : SFC trace (QTS)                              bB : Local device (QDL)                              bC :                              bD :                              bE :                              bF :                         </td> </tr> </table>	b0 : BOOT operation (QBT) b1 : Parameters (QPT) b2 : Device comments (QCD) b3 : Device initial value (QDI) b4 : File Register (QDR) b5 : Trace (QTS) b6 : b7 :	b8 : b9 : CPU fault history (QFD) bA : SFC trace (QTS) bB : Local device (QDL) bC : bD : bE : bF :	S (Status change)	New	System Q CPU (except Q00JCPU, Q00CPU, Q01CPU)
			b0 : BOOT operation (QBT) b1 : Parameters (QPT) b2 : Device comments (QCD) b3 : Device initial value (QDI) b4 : File Register (QDR) b5 : Trace (QTS) b6 : b7 :	b8 : b9 : CPU fault history (QFD) bA : SFC trace (QTS) bB : Local device (QDL) bC : bD : bE : bF :				
The use conditions for memory card A are stored as bit patterns (in use when ON). The significance of these bit patterns is indicated below: <table border="1" style="width: 100%; margin-top: 10px;"> <tr> <td style="width: 50%; vertical-align: top;">                             b0 : BOOT operation (QBT)                              b1 : Parameters (QPT)                              b2 : Device comments (QCD)                              b3 : Device initial value (QDI)                              b4 : File Register (QDR)                              b5 : Sampling trace (QTS)                              b6 : Status latch (QTL)                              b7 : Program trace (QTP)                         </td> <td style="width: 50%; vertical-align: top;">                             b8 : Simulation data (QDS)                              b9 : CPU fault history (QFD)                              bA : SFC trace (QTS)                              bB : Local device (QDL)                              bC :                              bD :                              bE :                              bF :                         </td> </tr> </table>	b0 : BOOT operation (QBT) b1 : Parameters (QPT) b2 : Device comments (QCD) b3 : Device initial value (QDI) b4 : File Register (QDR) b5 : Sampling trace (QTS) b6 : Status latch (QTL) b7 : Program trace (QTP)	b8 : Simulation data (QDS) b9 : CPU fault history (QFD) bA : SFC trace (QTS) bB : Local device (QDL) bC : bD : bE : bF :	S (Status change)	New	QnA CPU			
b0 : BOOT operation (QBT) b1 : Parameters (QPT) b2 : Device comments (QCD) b3 : Device initial value (QDI) b4 : File Register (QDR) b5 : Sampling trace (QTS) b6 : Status latch (QTL) b7 : Program trace (QTP)	b8 : Simulation data (QDS) b9 : CPU fault history (QFD) bA : SFC trace (QTS) bB : Local device (QDL) bC : bD : bE : bF :							

# Table of Special Registers

## (5) Memory cards

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:																
SD620	Memory card B models	Memory card B models	<p>Indicates memory card B models installed</p> <p>The value for drive 4 is fixed to "3" because it has built-in FLASH ROM.</p>	S (Initial)	New	System Q CPU																
			<p>Indicates memory card B models installed</p>	S (Initial)	New	QnA CPU																
SD622	Drive 3 (RAM) capacity	Drive 3 capacity	<p>Drive 3 capacity is stored in 1k byte units With a Q CPU, this value is fixed to "61" because of the built-in 61k RAM.</p>	S (Initial)	New	System Q CPU																
			<p>Drive 3 capacity is stored in 1k byte units</p>	S (Initial)	New	Q2(S1) Q3A Q4A Q4AR CPU																
SD623	Drive 4 (ROM) capacity	Drive 4 capacity	Drive 4 capacity is stored in 1k byte units	S (Initial)	New	Q2(S1) Q3A Q4A Q4AR System Q CPU																
SD624	Drive 3 use conditions	Drive 3 use conditions	<p>The use condition of drive 3 is indicated by bit 4: b4 = OFF: Drive 3 is not used b4 = ON: Drive 3 is used to store file registers</p>	S (Status change)	New	Q00JCPU Q00CPU Q01CPU																
	Drive 3 and 4 use conditions	Drive 3 and 4 use conditions	<p>The use conditions for memory card B are stored as bit patterns (In use when ON) The significance of these bit patterns is indicated below:</p> <table border="1"> <tr> <td>b0 : BOOT operation (QBT)</td> <td>b8 : CPU fault history (QFD)</td> </tr> <tr> <td>b1 : Parameters (QPA)</td> <td>b9 : SFC trace (QTS)</td> </tr> <tr> <td>b2 : Device comments (QCD)</td> <td>bA : Local device (QDL)</td> </tr> <tr> <td>b3 : Device initial value (QDI)</td> <td>bC :</td> </tr> <tr> <td>b4 : File R (QDR)</td> <td>bD :</td> </tr> <tr> <td>b5 : Trace (QTS)</td> <td>bE :</td> </tr> <tr> <td>b6 :</td> <td>bF :</td> </tr> <tr> <td>b7 :</td> <td></td> </tr> </table>	b0 : BOOT operation (QBT)	b8 : CPU fault history (QFD)	b1 : Parameters (QPA)	b9 : SFC trace (QTS)	b2 : Device comments (QCD)	bA : Local device (QDL)	b3 : Device initial value (QDI)	bC :	b4 : File R (QDR)	bD :	b5 : Trace (QTS)	bE :	b6 :	bF :	b7 :		S (Status change)	New	System Q CPU (except Q00JCPU Q00CPU Q01CPU)
	b0 : BOOT operation (QBT)	b8 : CPU fault history (QFD)																				
b1 : Parameters (QPA)	b9 : SFC trace (QTS)																					
b2 : Device comments (QCD)	bA : Local device (QDL)																					
b3 : Device initial value (QDI)	bC :																					
b4 : File R (QDR)	bD :																					
b5 : Trace (QTS)	bE :																					
b6 :	bF :																					
b7 :																						
Memory card B use conditions	Memory card B use conditions	<p>The use conditions for memory card B are stored as bit patterns (In use when ON) The significance of these bit patterns is indicated below:</p> <table border="1"> <tr> <td>b0 : BOOT operation (QBT)</td> <td>b8 : Simulation data (QDS)</td> </tr> <tr> <td>b1 : Parameters (QPT)</td> <td>b9 : CPU fault history (QFD)</td> </tr> <tr> <td>b2 : Device comments (QCD)</td> <td>bA : SFC trace (QTS)</td> </tr> <tr> <td>b3 : Device initial value (QDI)</td> <td>bB : Local device (QDL)</td> </tr> <tr> <td>b4 : File Register (QDR)</td> <td>bC :</td> </tr> <tr> <td>b5 : Sampling trace (QTS)</td> <td>bD :</td> </tr> <tr> <td>b6 : Status latch (QTL)</td> <td>bE :</td> </tr> <tr> <td>b7 : Program trace (QTP)</td> <td>bF :</td> </tr> </table>	b0 : BOOT operation (QBT)	b8 : Simulation data (QDS)	b1 : Parameters (QPT)	b9 : CPU fault history (QFD)	b2 : Device comments (QCD)	bA : SFC trace (QTS)	b3 : Device initial value (QDI)	bB : Local device (QDL)	b4 : File Register (QDR)	bC :	b5 : Sampling trace (QTS)	bD :	b6 : Status latch (QTL)	bE :	b7 : Program trace (QTP)	bF :	S (Status change)	New	Q2(S1) Q3A Q4A Q4AR CPU	
b0 : BOOT operation (QBT)	b8 : Simulation data (QDS)																					
b1 : Parameters (QPT)	b9 : CPU fault history (QFD)																					
b2 : Device comments (QCD)	bA : SFC trace (QTS)																					
b3 : Device initial value (QDI)	bB : Local device (QDL)																					
b4 : File Register (QDR)	bC :																					
b5 : Sampling trace (QTS)	bD :																					
b6 : Status latch (QTL)	bE :																					
b7 : Program trace (QTP)	bF :																					

# Table of Special Registers

Table of special registers (continued)

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:																					
SD640	File register drive	Drive number	Stores drive number being used by file register	S (Status change)	New																						
SD641	File register file name	File register file name	Stores file register file name (with extension) selected at parameters or by use of QDRSET instruction as ASCII code.  <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b8 b7</td> <td style="width: 33%; text-align: right;">b0</td> </tr> <tr> <td style="border: none;">SD641</td> <td style="border: 1px solid black; text-align: center;">2nd character</td> <td style="border: 1px solid black; text-align: center;">1st character</td> </tr> <tr> <td style="border: none;">SD642</td> <td style="border: 1px solid black; text-align: center;">4th character</td> <td style="border: 1px solid black; text-align: center;">3rd character</td> </tr> <tr> <td style="border: none;">SD643</td> <td style="border: 1px solid black; text-align: center;">6th character</td> <td style="border: 1px solid black; text-align: center;">5th character</td> </tr> <tr> <td style="border: none;">SD644</td> <td style="border: 1px solid black; text-align: center;">8th character</td> <td style="border: 1px solid black; text-align: center;">7th character</td> </tr> <tr> <td style="border: none;">SD645</td> <td style="border: 1px solid black; text-align: center;">1st char. of extension</td> <td style="border: 1px solid black; text-align: center;">2EH (.)</td> </tr> <tr> <td style="border: none;">SD646</td> <td style="border: 1px solid black; text-align: center;">3rd char. of extension</td> <td style="border: 1px solid black; text-align: center;">2nd char. of extension</td> </tr> </table>		b8 b7	b0	SD641	2nd character	1st character	SD642	4th character	3rd character	SD643	6th character	5th character	SD644	8th character	7th character	SD645	1st char. of extension	2EH (.)	SD646	3rd char. of extension	2nd char. of extension	S (Status change)	New	●
				b8 b7	b0																						
SD641				2nd character	1st character																						
SD642				4th character	3rd character																						
SD643				6th character	5th character																						
SD644				8th character	7th character																						
SD645				1st char. of extension	2EH (.)																						
SD646	3rd char. of extension	2nd char. of extension																									
SD642																											
SD643																											
SD644																											
SD645																											
SD646																											
SD647	File register capacity	File register capacity	Stores the data capacity of the currently selected file register in 1 K word units.	S (Status change)	New																						
SD648	File register block number	File register block number	Stores the currently selected file register block number.	S (Status change)	D9035																						
SD650	Comment drive	Comment drive	Stores the comment drive number selected at the parameters or by the QCDESET instruction.	S (Status change)	New																						
SD651	Comment file name	Comment file name	Stores the comment file name (with extension) selected at the parameters or by the QCDESET instruction in ASCII code.  <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b8 b7</td> <td style="width: 33%; text-align: right;">b0</td> </tr> <tr> <td style="border: none;">SD651</td> <td style="border: 1px solid black; text-align: center;">2nd character</td> <td style="border: 1px solid black; text-align: center;">1st character</td> </tr> <tr> <td style="border: none;">SD652</td> <td style="border: 1px solid black; text-align: center;">4th character</td> <td style="border: 1px solid black; text-align: center;">3rd character</td> </tr> <tr> <td style="border: none;">SD653</td> <td style="border: 1px solid black; text-align: center;">6th character</td> <td style="border: 1px solid black; text-align: center;">5th character</td> </tr> <tr> <td style="border: none;">SD654</td> <td style="border: 1px solid black; text-align: center;">8th character</td> <td style="border: 1px solid black; text-align: center;">7th character</td> </tr> <tr> <td style="border: none;">SD655</td> <td style="border: 1px solid black; text-align: center;">1st char. of extension</td> <td style="border: 1px solid black; text-align: center;">2EH (.)</td> </tr> <tr> <td style="border: none;">SD656</td> <td style="border: 1px solid black; text-align: center;">3rd char. of extension</td> <td style="border: 1px solid black; text-align: center;">2nd char. of extension</td> </tr> </table>		b8 b7	b0	SD651	2nd character	1st character	SD652	4th character	3rd character	SD653	6th character	5th character	SD654	8th character	7th character	SD655	1st char. of extension	2EH (.)	SD656	3rd char. of extension	2nd char. of extension	S (Status change)	New	●
				b8 b7	b0																						
SD651				2nd character	1st character																						
SD652				4th character	3rd character																						
SD653				6th character	5th character																						
SD654				8th character	7th character																						
SD655				1st char. of extension	2EH (.)																						
SD656	3rd char. of extension	2nd char. of extension																									
SD652																											
SD653																											
SD654																											
SD655																											
SD656																											
SD660	Boot operation designation file	Boot designation file drive number	Stores the drive number where the boot designation file (*.QBT) is being stored.	S (Initial)	New	● (except Q00JCPU Q00CPU Q01CPU)																					
SD661		File name of boot designation file	Stores the file name of the boot designation file (*.QBT).  <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;"></td> <td style="width: 33%; text-align: center;">b8 b7</td> <td style="width: 33%; text-align: right;">b0</td> </tr> <tr> <td style="border: none;">SD661</td> <td style="border: 1px solid black; text-align: center;">2nd character</td> <td style="border: 1px solid black; text-align: center;">1st character</td> </tr> <tr> <td style="border: none;">SD662</td> <td style="border: 1px solid black; text-align: center;">4th character</td> <td style="border: 1px solid black; text-align: center;">3rd character</td> </tr> <tr> <td style="border: none;">SD663</td> <td style="border: 1px solid black; text-align: center;">6th character</td> <td style="border: 1px solid black; text-align: center;">5th character</td> </tr> <tr> <td style="border: none;">SD664</td> <td style="border: 1px solid black; text-align: center;">8th character</td> <td style="border: 1px solid black; text-align: center;">7th character</td> </tr> <tr> <td style="border: none;">SD665</td> <td style="border: 1px solid black; text-align: center;">1st char. of extension</td> <td style="border: 1px solid black; text-align: center;">2EH (.)</td> </tr> <tr> <td style="border: none;">SD666</td> <td style="border: 1px solid black; text-align: center;">3rd char. of extension</td> <td style="border: 1px solid black; text-align: center;">2nd char. of extension</td> </tr> </table>		b8 b7		b0	SD661	2nd character	1st character	SD662	4th character	3rd character	SD663	6th character	5th character	SD664	8th character	7th character	SD665	1st char. of extension	2EH (.)	SD666	3rd char. of extension	2nd char. of extension	S (Initial)	New
				b8 b7	b0																						
SD661				2nd character	1st character																						
SD662				4th character	3rd character																						
SD663				6th character	5th character																						
SD664				8th character	7th character																						
SD665	1st char. of extension	2EH (.)																									
SD666	3rd char. of extension	2nd char. of extension																									
SD662																											
SD663																											
SD664																											
SD665																											
SD666																											

# Table of Special Registers

## (6) Instruction related registers

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:												
SD705	Mask pattern	Mask pattern	During block operations, turning SM705 ON makes it possible to use the mask pattern being stored at SD705 (or at SD705 and SD706 if double words are being used) to operate on all data in the block with the masked values.	U	New	● (except Q00JCPU, Q00CPU, Q01CPU)												
SD706																		
SD714	Number of vacant communication request registration areas	0 to 32	Stores the number of vacant blocks in the communications request area for remote terminal modules connected to the AJ71PT32-S3.	S (During execution)	M9081	QnA CPU												
SD715	IMASK instruction mask pattern	Mask pattern	Patterns masked by use of the IMASK instruction are stored in the following manner:  <table style="border-collapse: collapse; margin-left: 20px;"> <tr> <td style="text-align: right; padding-right: 5px;">b15</td> <td style="text-align: center;">.....</td> <td style="text-align: left; padding-left: 5px;">b0</td> </tr> <tr> <td style="border-right: 1px solid black; padding-right: 2px;">SD715</td> <td style="border-right: 1px solid black; padding-right: 2px;"> 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td> <td></td> </tr> <tr> <td style="border-right: 1px solid black; padding-right: 2px;">SD716</td> <td style="border-right: 1px solid black; padding-right: 2px;"> 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td> <td></td> </tr> <tr> <td style="border-right: 1px solid black; padding-right: 2px;">SD717</td> <td style="border-right: 1px solid black; padding-right: 2px;"> 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32</td> <td></td> </tr> </table>	b15	.....	b0	SD715	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		SD716	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16		SD717	47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32		S (During execution)	New	●
b15				.....	b0													
SD715				15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0														
SD716	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16																	
SD717	47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32																	
SD716																		
SD717																		
SD718	Accumulator	Accumulator	For use as replacement for accumulators used in A-series programs.	S/U	New													
SD719																		
SD720	Program No. destination for PLOAD instruction	Program number destination for PLOAD instruction	Stores the program number of the program to be loaded by the PLOAD instruction when designated. The destination range is from 1 to 124.	U	New	System Q CPU												
SD730	No. of vacant registration area for CC-Link communication request	0 to 32	Stores the number of vacant registration areas for the request for communication with the intelligent device station connected to A(1S)J61QBT61.	S (During execution)	New	QnA CPU												
SD736	PKEY input	PKEY input	SD that temporarily stores keyboard data input by means of the PKEY instruction.	S (During execution)	New	● (except Q00JCPU, Q00CPU, Q01CPU)												





# Table of Special Registers

## (8) Latch area

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:																																																			
SD900	Drive where power was interrupted	Access file drive number during power loss	Stores drive number if file was being accessed during power loss.	S (Status change)	New	QnA CPU																																																			
SD901	File name active during power loss	Access file name during power loss	Stores file name (with extension) in ASCII code if file was being accessed during power loss.	S (Status change)	New																																																				
SD902			<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">b15</td> <td style="text-align: center;">b8 b7</td> <td style="text-align: center;">b0</td> </tr> <tr> <td>SD901</td> <td style="border: 1px solid black; text-align: center;">2nd character</td> <td style="border: 1px solid black; text-align: center;">1st character</td> </tr> <tr> <td>SD902</td> <td style="border: 1px solid black; text-align: center;">4th character</td> <td style="border: 1px solid black; text-align: center;">3rd character</td> </tr> <tr> <td>SD903</td> <td style="border: 1px solid black; text-align: center;">6th character</td> <td style="border: 1px solid black; text-align: center;">5th character</td> </tr> <tr> <td>SD904</td> <td style="border: 1px solid black; text-align: center;">8th character</td> <td style="border: 1px solid black; text-align: center;">7th character</td> </tr> <tr> <td>SD905</td> <td style="border: 1px solid black; text-align: center;">1st char. of extension</td> <td style="border: 1px solid black; text-align: center;">2EH (.)</td> </tr> <tr> <td>SD906</td> <td style="border: 1px solid black; text-align: center;">3rd char. of extension</td> <td style="border: 1px solid black; text-align: center;">2nd char. of extension</td> </tr> </table>				b15	b8 b7	b0	SD901	2nd character	1st character	SD902	4th character	3rd character	SD903	6th character	5th character	SD904	8th character	7th character	SD905	1st char. of extension	2EH (.)	SD906	3rd char. of extension	2nd char. of extension																														
b15							b8 b7	b0																																																	
SD901							2nd character	1st character																																																	
SD902							4th character	3rd character																																																	
SD903							6th character	5th character																																																	
SD904							8th character	7th character																																																	
SD905	1st char. of extension	2EH (.)																																																							
SD906	3rd char. of extension	2nd char. of extension																																																							
SD903																																																									
SD904																																																									
SD905																																																									
SD906																																																									
SD910	RKEY input	RKEY input	Stored in sequence that PU key code was entered.	S (During execution)	New																																																				
SD911			<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">b15</td> <td style="text-align: center;">b8 b7</td> <td style="text-align: center;">b0</td> </tr> <tr> <td>SD910</td> <td style="border: 1px solid black; text-align: center;">2nd character</td> <td style="border: 1px solid black; text-align: center;">1st character</td> </tr> <tr> <td>SD911</td> <td style="border: 1px solid black; text-align: center;">4th character</td> <td style="border: 1px solid black; text-align: center;">3rd character</td> </tr> <tr> <td>SD912</td> <td style="border: 1px solid black; text-align: center;">6th character</td> <td style="border: 1px solid black; text-align: center;">5th character</td> </tr> <tr> <td>SD913</td> <td style="border: 1px solid black; text-align: center;">8th character</td> <td style="border: 1px solid black; text-align: center;">7th character</td> </tr> <tr> <td>SD914</td> <td style="border: 1px solid black; text-align: center;">10th character</td> <td style="border: 1px solid black; text-align: center;">9th character</td> </tr> <tr> <td>SD915</td> <td style="border: 1px solid black; text-align: center;">12th character</td> <td style="border: 1px solid black; text-align: center;">11th character</td> </tr> <tr> <td>SD916</td> <td style="border: 1px solid black; text-align: center;">14th character</td> <td style="border: 1px solid black; text-align: center;">13th character</td> </tr> <tr> <td>SD917</td> <td style="border: 1px solid black; text-align: center;">16th character</td> <td style="border: 1px solid black; text-align: center;">15th character</td> </tr> <tr> <td>SD918</td> <td style="border: 1px solid black; text-align: center;">18th character</td> <td style="border: 1px solid black; text-align: center;">17th character</td> </tr> <tr> <td>SD919</td> <td style="border: 1px solid black; text-align: center;">20th character</td> <td style="border: 1px solid black; text-align: center;">19th character</td> </tr> <tr> <td>SD920</td> <td style="border: 1px solid black; text-align: center;">22nd character</td> <td style="border: 1px solid black; text-align: center;">21th character</td> </tr> <tr> <td>SD921</td> <td style="border: 1px solid black; text-align: center;">24th character</td> <td style="border: 1px solid black; text-align: center;">23th character</td> </tr> <tr> <td>SD922</td> <td style="border: 1px solid black; text-align: center;">26th character</td> <td style="border: 1px solid black; text-align: center;">25th character</td> </tr> <tr> <td>SD923</td> <td style="border: 1px solid black; text-align: center;">28th character</td> <td style="border: 1px solid black; text-align: center;">27th character</td> </tr> <tr> <td>SD924</td> <td style="border: 1px solid black; text-align: center;">30th character</td> <td style="border: 1px solid black; text-align: center;">29th character</td> </tr> <tr> <td>SD925</td> <td style="border: 1px solid black; text-align: center;">32th character</td> <td style="border: 1px solid black; text-align: center;">31st character</td> </tr> </table>				b15	b8 b7	b0	SD910	2nd character	1st character	SD911	4th character	3rd character	SD912	6th character	5th character	SD913	8th character	7th character	SD914	10th character	9th character	SD915	12th character	11th character	SD916	14th character	13th character	SD917	16th character	15th character	SD918	18th character	17th character	SD919	20th character	19th character	SD920	22nd character	21th character	SD921	24th character	23th character	SD922	26th character	25th character	SD923	28th character	27th character	SD924	30th character	29th character	SD925	32th character	31st character
b15							b8 b7	b0																																																	
SD910							2nd character	1st character																																																	
SD911							4th character	3rd character																																																	
SD912							6th character	5th character																																																	
SD913							8th character	7th character																																																	
SD914							10th character	9th character																																																	
SD915							12th character	11th character																																																	
SD916							14th character	13th character																																																	
SD917							16th character	15th character																																																	
SD918						18th character	17th character																																																		
SD919						20th character	19th character																																																		
SD920						22nd character	21th character																																																		
SD921						24th character	23th character																																																		
SD922						26th character	25th character																																																		
SD923	28th character	27th character																																																							
SD924	30th character	29th character																																																							
SD925	32th character	31st character																																																							
SD912																																																									
SD913																																																									
SD914																																																									
SD915																																																									
SD916																																																									
SD917																																																									
SD918																																																									
SD919																																																									
SD920																																																									
SD921																																																									
SD922																																																									
SD923																																																									
SD924																																																									
SD925																																																									

# Table of Special Registers

## (9) Blown fuse detection module

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD1300	Fuse blown module	Bit pattern in units of 16 points, indicating the modules whose fuses have blown 0 : No blown fuse 1 : Blown fuse present	<p>The numbers of output modules whose fuses have blown are input as a bit pattern (in units of 16 points). (If the module numbers are set by parameter, the parameter-set numbers are stored.) Also detects blown fuse condition at remote station output modules.</p>	S (Error)	D9100	● (except Q00JCPU Q00CPU Q01CPU)
SD1301					D9101	
SD1302					D9102	
SD1303					D9103	
SD1304					D9104	
SD1305					D9105	
SD1306					D9106	
SD1307					D9107	
SD1308					New	
SD1309 — SD1330					New	
SD1331					New	
SD1350 to SD1381	External power supply disconnected	Bit pattern in units of 16 points, indicating the modules whose external power supply has been disconnected 0 : External power supply disconnected 1 : External power supply is not disconnected	<p>● The numbers of output modules whose external power supply is disconnected are stored as bit pattern (in units of 16 points). (If the module numbers are set by parameter, the parameter-set numbers are stored.)</p>	S (Error)	New	System Q CPU (except Q00JCPU Q00CPU Q01CPU)



# Table of Special Registers

## (10) I/O module verification

Number	Name	Meaning	Description	Set by (if set)	ACPU register D9 [ ] [ ] [ ]	Valid for:
SD1400	I/O module verification error	Bit pattern, in units of 16 points, indicating the modules with verification errors. 0: No I/O verification errors 1: I/O verification error present	<p>When the power is turned on, the module numbers of the I/O modules whose information differs from the registered I/O module information are set in this register (in units of 16 points). (If the I/O numbers are set by parameter, the parameter-set numbers are stored.) Also detects I/O module information.</p> <pre>           b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 SD1400  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0 SD1401  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0       ↓       :       : SD1431  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0           ↑           Indication of I/O module           with verification error                     </pre>	S (Error)	D9116	● (except Q00JCPU Q00CPU Q01CPU)
SD1401					D9117	
SD1402					D9118	
SD1403					D9119	
SD1404					D9120	
SD1405					D9121	
SD1406					D9122	
SD1407					D9123	
SD1408					New	
SD1409 to SD1430					New	
SD1431					New	

## Table of Special Registers

### (11) MELSEC A to MELSEC QnA series/MELSEC System Q conversion correspondences

For a conversion from the A series to the Q series or the System Q the special registers D9000 through D9255 (A series) correspond to the diagnostic special registers SD1000 to SD1255 of the Q series and the System Q.

These diagnostic special registers are all set by the system and cannot be changed by a user-program. Users intending to set or reset these registers should alter their programs so that only real System Q/QnA diagnostic special registers are applied.

An exception are the special registers D9200 through D9255. The data in these registers can be changed by the user. Therefore, the user can change the data in the diagnostic special registers SD1200 to SD1255 after the conversion.

Refer to the manuals of the CPUs and the networks MELSECNET and MELSECNET/B for detailed information on the special relays of the A series.

#### NOTE

For the device numbers for which a equivalent System Q/QnA diagnostic special register for modification is specified, modify it to the special register for a System Q/QnA CPU. If no equivalent System Q/QnA diagnostic special register is specified, the special register after conversion can be used.

Table of special relays and diagnostic special relays

A CPU special register	Special register after conversion	Equivalent Q/QnA diagnostic special register	Name	Meaning	Valid for:
D9000	SD1000		Fuse blown	Number of module with blown fuse	System Q QnA CPU
D9001	SD1001		Fuse blown	Number of module with blown fuse	
D9002	SD1002		I/O module verification error	I/O module verification error module number	
D9004	SD1004		MINI link errors	Stores setting status made at parameters (modules 1 to 8)	QnA CPU
D9005	SD1005		AC DOWN counter	Number of times for AC DOWN	System Q QnA CPU
D9008	SD1008	SD0	Self-diagnostic error	Self-diagnostic error number	
D9009	SD1009	SD62	Annunciator detection	F number at which external failure has occurred	
D9010	SD1010	No function for a System Q/QnA CPU	Error step	Step number at which operation error has occurred.	
D9011	SD1011		Error step	Step number at which operation error has occurred.	
D9014	SD1014		I/O control mode	I/O control mode number	
D9015	SD1015	SD203	Operating state of CPU	Operating state of CPU	
D9016	1016	No function for a System Q/QnA CPU	Program number	Stores sequence program under execution as BIN value	
D9017	SD1017	SD520	Scan time	Minimum scan time (10 ms units)	
D9018	SD1018	SD524	Scan time	Scan time (10 ms units)	
D9019	SD1019	SD526	Scan time	Maximum scan time (10 ms units)	
D9020	SD1020	No function for a System Q/QnA CPU	Constant scan	Constant scan time (User sets in 10 ms units)	

## Table of Special Registers

Table of special relays and diagnostic special relays

A CPU special register	Special register after conversion	Equivalent Q/QnA diagnostic special register	Name	Meaning	Valid for:
D9021	SD1021		Scan time	Scan time (in 1 ms units)	System Q QnA CPU
D9022	SD1022	SD412	Time	Time	
D9025	SD1025	SD210	Clock data	Clock data (year, month)	
D9026	SD1026	SD211	Clock data	Clock data (day, hour)	
D9027	SD1027	SD212	Clock data	Clock data (minute, second)	
D9028	SD1028	SD213	Clock data	Clock data (day of week)	
D9035	SD1035	SD648	Extension file register	Use block No.	
D9036	SD1036	No function for a System Q/QnA CPU	Extension file register for designation of device number	Device number when individual devices from extension file register are directly accessed	
D9037	SD1037				
D9038	SD1038	SD207	LED display priority ranking	Priorities 1 to 4	
D9039	SD1039	SD208		Priorities 5 to 7	
D9044	SD1044	No function for a System Q/QnA CPU	For sampling trace	Step or time during sampling trace	
D9049	SD1049		Work area for SFC	Block number of extension file register	
D9050	SD1050		SFC program error number	Error code generated by SFC program	
D9051	SD1051		Error block	Block number where error occurred	
D9052	SD1052		Error step	Step number where error occurred	
D9053	SD1053		Error transition	Transition condition number where error occurred	
D9054	SD1054		Error sequence step	Sequence step number where error occurred	
D9055	SD1055	SD812	Status latch	Status latch step	
D9060	SD1060	SD392	Software version	Software version of internal software	
D9072	SD1072	No function for a System Q/QnA CPU	PC communications check	Computer link data check	System Q QnA CPU
D9081	SD1081	SD714	Number of empty blocks in communications request registration area	Number of empty blocks in communications request registration area	
D9085	SD1085	No function for a System Q/QnA CPU	Register for setting time check value	Default value 10s	
D9090	SD1090		AnN: Address area for sub routine of microcomputer program AnA: No function QnA: Total of special function modules	AnN: Address area for sub routine of microcomputer program AnA: No function QnA: Total of special function modules	
D9091	SD1091		Detailed error code	Self-diagnosis detailed error code	
D9094	SD9094		SD251	Head I/O number for replacement	

# Table of Special Registers

Table of special relays and diagnostic special relays

A CPU special register	Special register after conversion	Equivalent Q/QnA diagnostic special register	Name	Meaning	Valid for:
D9100	SD1100	—	Fuse blown module	Bit pattern in units of 16 points, indicating the modules whose fuses have blown	
D9101	SD1101				
D9102	SD1102				
D9103	SD1103				
D9104	SD1104				
D9105	SD1105				
D9106	SD1106				
D9107	SD1107				
D9108	SD1108	—	Step transfer monitoring timer setting	Sets value for the step transfer monitoring timer in the lower byte. The setting range is from 1 to 255 seconds. Sets the number of F which turn on when the monitoring time is over in the higher byte. The monitoring timer starts when any relay from SM1108 through SM1114 is set. If the transfer condition following a step which corresponds to the timer is not established within the set time, the set annunciator (F) is turned on.	
D9109	SD1109				
D9110	SD1110				
D9111	SD1111				
D9112	SD1112				
D9113	SD1113				
D9114	SD1114				
D9116	SD1116	—	I/O module verification error	Bit pattern in units of 16 points, indicating the modules with verification errors	System Q QnA CPU
D9117	SD1117				
D9118	SD1118				
D9119	SD1119				
D9120	SD1120				
D9121	SD1121				
D9122	SD1122				
D9123	SD1123				
D9124	SD9124	SD63	Annunciator detection quantity	Annunciator detection quantity	
D9125	SD9125	SD64	Annunciator detection number	Annunciator detection number	
D9126	SD9126	SD65			
D9127	SD9127	SD66			
D9128	SD9128	SD67			
D9129	SD9129	SD68			
D9130	SD9130	SD69			
D9131	SD9131	SD70			
D9132	SD9132	SD71			

## Table of Special Registers

Table of special relays and diagnostic special relays

A CPU special register	Special register after conversion	Equivalent Q/QnA diagnostic special register	Name	Meaning	Valid for:
D9200	SD1200	—	LRDP processing results	0 : Normal End 2 : LRDP instruction setting fault 3 : Error at relevant station 4 : Relevant station LRDP execution disabled	QnA CPU
D9201	SD1201	—	LWTP processing results	0 : Normal End 2 : LRDP instruction setting fault 3 : Error at relevant station 4 : Relevant station LWTP execution disabled	
D9202	SD1202	—	Local station link type	Stores conditions for up to numbers 1 to 16	
D9203	SD1203	—		Stores conditions for up to numbers 17 to 32	
D9241	SD1241	—		Stores conditions for up to numbers 33 to 48	
D9242	SD1242	—		Stores conditions for up to numbers 49 to 64	
D9204	SD1204	—	Link state	0: Forward loop, during data link 1: Reverse loop, during data link 2: Loopback implemented in forward/reverse directions 3: Loopback implemented only forward direction 4: Loopback implemented only inreverse direction 5: Data link disabled	
D9205	SD1205	—	Station implementing loopback	Station that implemented forward loopback	
D9206	SD1206	—	Station implementing loopback	Station that implemented forward loopback	
D9207	SD1207	—	Link scan time	Maximum value	
D9208	SD1208	—		Minimum value	
D9209	SD1209	—		Present value	
D9210	SD1210	—	Number of retries	Stored as cumulative value	
D9211	SD1211	—	Number of times loop selected	Stored as cumulative value	
D9212	SD1212	—	Local station operation state	Stores conditions for up to Stations 1 to 16	
D9213	SD1213	—		Stores conditions for up to Stations 17 to 32	
D9214	SD1214	—		Stores conditions for up to Stations 33 to 48	
D9215	SD1215	—		Stores conditions for up to Numbers 49 to 64	

# Table of Special Registers

Table of special relays and diagnostic special relays

A CPU special register	Special register after conversion	Equivalent Q/QnA diagnostic special register	Name	Meaning	Valid for:
D9216	SD1216	—	Local station error detect state	Stores conditions for up to Numbers 1 to 16	QnA CPU
D9217	SD1217	—		Stores conditions for up to Numbers 17 to 32	
D9218	SD1218	—		Stores conditions for up to Numbers 33 to 48	
D9219	SD1219	—		Stores conditions for up to Numbers 49 to 64	
D9220	SD1220	—	Local station parameters non-conforming; remote I/O station I/O allocation error	Stores conditions for up to Numbers 1 to 16	
D9221	SD1221	—		Stores conditions for up to Numbers 17 to 32	
D9222	SD1222	—		Stores conditions for up to Numbers 33 to 48	
D9223	SD1223	—		Stores conditions for up to Numbers 49 to 64	
D9224	SD1224	—	Local station and remote I/O station initial communications underway	Stores conditions for up to Numbers 1 to 16	
D9225	SD1225	—		Stores conditions for up to Numbers 17 to 32	
D9226	SD1226	—		Stores conditions for up to Numbers 33 to 48	
D9227	SD1227	—		Stores conditions for up to Numbers 49 to 64	
D9228	SD1228	—	Local station and remote I/O station error	Stores conditions for up to Numbers 1 to 16	
D9229	SD1229	—		Stores conditions for up to Numbers 17 to 32	
D9230	SD1230	—		Stores conditions for up to Numbers 33 to 48	
D9231	SD1231	—		Stores conditions for up to Numbers 49 to 64	

## Table of Special Registers

Table of special relays and diagnostic special relays

A CPU special register	Special register after conversion	Equivalent Q/QnA diagnostic special register	Name	Meaning	Valid for:
D9232	SD1232	—	Local station and remote I/O station loop error	Stores conditions for up to Numbers 1 to 8	QnA CPU
D9233	SD1233	—		Stores conditions for up to Numbers 9 to 16	
D9234	SD1234	—		Stores conditions for up to Numbers 17 to 24	
D9235	SD1235	—		Stores conditions for up to Numbers 25 to 32	
D9236	SD1236	—		Stores conditions for up to Numbers 33 to 40	
D9237	SD1237	—		Stores conditions for up to Numbers 41 to 48	
D9238	SD1238	—		Stores conditions for up to Numbers 49 to 56	
D9239	SD1239	—		Stores conditions for up to Numbers 57 to 64	
D9240	SD1240	—	Number of times communications errors detected	Stores cumulative total of receive errors	
D9243	SD1243	—	Station number information for host station	Stores station number (0 to 64)	
D9244	SD1244	—	Number of link device stations	Stores number of slave stations	
D9245	SD1245	—	Number of times communications errors detected	Stores cumulative total of receive errors	
D9248	SD1248	—	Local station operation state	Stores conditions for up to Numbers 1 to 16	
D9249	SD1249	—		Stores conditions for up to Numbers 17 to 32	
D9250	SD1250	—		Stores conditions for up to Numbers 33 to 48	
D9251	SD1251	—		Stores conditions for up to Numbers 49 to 64	
D9252	SD1252	—	Local station error conditions	Stores conditions for up to Numbers 1 to 16	
D9253	SD1253	—		Stores conditions for up to Numbers 17 to 32	
D9254	SD1254	—		Stores conditions for up to Numbers 33 to 48	
D9255	SD1255	—		Stores conditions for up to Numbers 49 to 64	

# Table of Special Registers

## A.5.2 Table of special registers (D) (A series only)

Special registers (D) are data registers provided for specific applications inside the CPU. Therefore, do not write data to the special registers in the program (except for the registers tagged by ❶).

In general there are two types of special registers:

- Special registers that are written to automatically by the CPU and can only be read (and reset) by the user.
- Special registers that can be written to only under certain conditions.

The usage of special registers in a sequence program has to be checked accordingly.

The following table contains an overview of the entire MELSEC A series special registers including a description of their purposes.

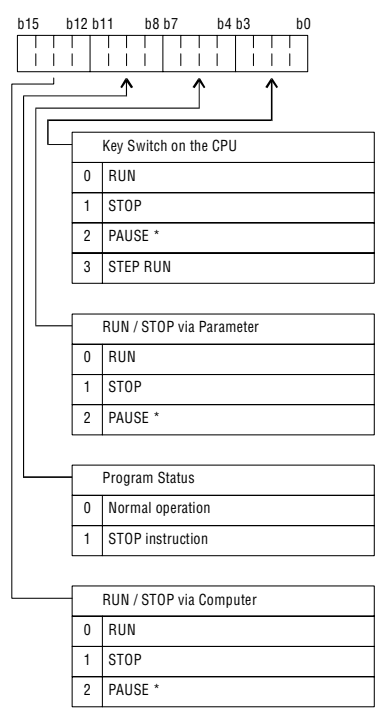

Table of special registers (A series)

Number	Meaning	Description	Details	CPU
D9000	Fuse blown	Fuse blown module number	When fuse blown modules are detected, the lowest number of detected units is stored in hexadecimal (example: When fuses of Y50 to 6F output modules have blown, '50' is stored in hexadecimal). In order to monitor the number by peripheral devices, perform monitor operation given in hexadecimal. (Cleared when all contents of D9100 to D9107 are reset to 0.) Fuse blow check is executed also to the output modules of remote I/O stations.	
D9002	I/O module verify error	I/O module verify error unit number	If I/O modules, of which data are different from data entered, are detected when the power is turned on, the first I/O number of the lowest number unit among the detected units is stored in hexadecimal. (Storing method is the same as that of D9000.) To monitor the number by peripheral devices, perform monitor operation given in hexadecimal. (Cleared when all contents of D9116 to D9123 are reset to 0.) I/O module verify check is executed also to the modules of remote I/O terminals.	
❶ D9004	MINI link master module error	Error detection status	Error status of the MINI(S3) link detected on loaded (AJ71PT32(S3)) is stored. Please refer to the MELSECNET/MINI-S3 manual for more informations.	Only for AnA-, AnAS-, AnU-CPU
❶ D9005	AC DOWN counter	AC DOWN count	1 is added each time input voltage becomes 80 % or less of rating while the CPU unit is performing operation, and the value is stored in BIN code.	
❶ D9008	Self-diagnostic error	Self-diagnostic error number	When error is found as a result of self-diagnosis, error number is stored in BIN code.	Only for AnS-CPU, A2C-CPU
D9009	Annunciator detection	F number at which external failure has occurred	When one of F0 to 255 is turned ON by OUT F or SET F, the F number, which has been detected earliest among the F numbers which have turned on, is stored in BIN code. D9009 can be cleared by RST F or LEDR instruction. If another F number has been detected, the clearing of D9009 causes the next number to be stored in D9009.	Not for A3N-CPU, A3M-CPU, A3A-CPU, A3H-CPU
			When one of F0 to F255 is turned on by OUT F or SET F, the F number, which has been detected earliest among the F numbers which have turned on, is stored in BIN code. D9009 can be cleared by executing RST F or LEDR instruction or moving INDICATOR RESET switch on CPU front to ON position. If another F number has been detected, the clearing of D9009 causes the next number to be stored in D9009.	Only for A3N-CPU, A3M-CPU, A3A-CPU, A3H-CPU
D9010	Error step	Step number at which operation error has occurred	When operation error has occurred during execution of application instruction, the step number, at which the error has occurred, is stored in BIN code. Thereafter, each time operation error occurs, the contents of D9010 are renewed.	Not for A3H-CPU, A3M-CPU
❶ D9011	Error step	Step number at which operation error has occurred	When operation error has occurred during execution of application instruction, the step number, at which the error has occurred, is stored in BIN code. Since storage into D9011 is made when M9011 changes from off to on, the contents of D9010 cannot be renewed unless M9011 is cleared by user program.	




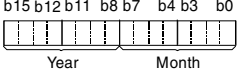

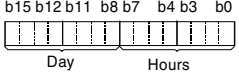

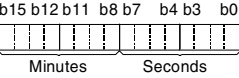

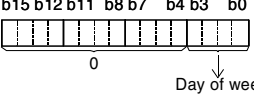
# Table of Special Registers

Table of special registers (A series)

Number	Meaning	Description	Details	CPU
D9014	I/O control mode	I/O control mode number	The I/O control mode set is returned in any of the following numbers: 0: Both input and output in direct mode 1: Input in refresh mode, output in direct mode 3: Both input and output in refresh mode	Not for A3H-CPU, A3M-CPU, An-CPU
D9015	CPU operating states	Operating states of CPU	The operating states of CPU as shown below are stored in D9015:   <p>* When the CPU is in RUN mode and M9040 is off, the CPU remains in RUN mode if changed to PAUSE mode.</p>	
D9016	ROM/RAM setting	0: ROM 1: RAM 2: EEPROM	Indicates the setting of memory select chip. One value of 0 to 2 is stored in BIN code.	Only for A1-CPU, A1N-CPU
	Program number	0: Main program (ROM) 1: Main program (RAM) 2: Subprogram (RAM)	Indicates which sequence program is run presently. One value of 0 to 2 is stored in BIN code. "2" is not stored when A1S, A0J2H, A2C, A2, A2N, A2N-F, A2A, A2A-F and A2U is used.	Not for A1-CPU, A1N-CPU
D9017	Scan time	Minimum scan time (per 10 ms)	If scan time is smaller than the contents of D9017, the value is newly stored at each END. Namely, the minimum value of scan time is stored into D9017 in BIN code.	
D9018	Scan time	Scan time (per 10 ms)	Scan time is stored in BIN code at each END and always rewritten.	
D9019	Scan time	Maximum scan time (per 10 ms)	If scan time is larger than the contents of D9019, the value is newly stored at each END. Namely, the maximum value of scan time is stored into D9019 in BIN code.	
 D9020	Constant scan	Constant scan time (Set by user in 10 ms increments)	Sets the interval between consecutive program starts in multiples of 10 ms. 0: No setting 1 to 200: Set. Program is executed at intervals of (set value) x 10 ms	Not for A-CPU
D9021	Scan time	Scan time (1 msec unit)	Scan time is stored and updated in BIN code after every END.	Only for AnA-, AnAS-, AnU-CPU

# Table of Special Registers

Table of special registers (A series)

Number	Meaning	Description	Details	CPU
D9022	1 second counter	Counts 1 every second.	When the PC CPU starts running, it starts counting 1 every second. It starts counting up from 0 to 32767, then down to -32768 and then again up to 0. Counting repeats this routine.	Only for AnA-, AnAS-, AnU-CPU's
 D9025	Clock data	(Year, month)	Stores the year (2 lower digits) and month in BCD: Example: 1992, July = H9207 	Only for AnA-CPU's, AnU-CPU's, AnN-CPU's, A1S-CPU
 D9026	Clock data	(Day, hour)	Stores the day and hour in BCD: Example: 31st, 10 o'clock = H3110 	Only for AnA- AnAS-, AnU-CPU's, AnN-CPU's, AnS-CPU's
 D9027	Clock data	(Minute, second)	Stores the minute and second in BCD: Example: 35 minutes, 48 seconds = H3548 	Only for AnA-, AnAS-, AnU-CPU's, AnN-CPU's, AnS-CPU's
 D9028	Clock data	(Day of week)	Stores the day of the week in BCD: (0=Sunday, 1=Monday, 2=Tuesday etc.): 	Only for AnA-, AnAS-, AnU-CPU's, AnN-CPU's, AnS-CPU's

# Table of Special Registers

Table of special registers (A series)

Number	Meaning	Description	Details	CPU																																
D9021	Remote terminal parameter setting	1 to 61	<p>Sets the head station number (1 to 61) of remote terminal modules connected to A2C and A52G. Setting is not necessarily in the order of station numbers.</p> <p>Data configuration:</p> <table border="1" style="margin-left: 40px;"> <tr> <td>D9021</td> <td>Remote terminal module No. 1 area</td> </tr> <tr> <td>D9022</td> <td>Remote terminal module No. 2 area</td> </tr> <tr> <td colspan="2" style="text-align: center;">:</td> </tr> <tr> <td colspan="2" style="text-align: center;">:</td> </tr> <tr> <td colspan="2" style="text-align: center;">:</td> </tr> <tr> <td colspan="2" style="text-align: center;">:</td> </tr> <tr> <td>D9032</td> <td>Remote terminal module No. 12 area</td> </tr> <tr> <td>D9033</td> <td>Remote terminal module No. 13 area</td> </tr> <tr> <td>D9034</td> <td>Remote terminal module No. 14 area</td> </tr> </table>	D9021	Remote terminal module No. 1 area	D9022	Remote terminal module No. 2 area	:		:		:		:		D9032	Remote terminal module No. 12 area	D9033	Remote terminal module No. 13 area	D9034	Remote terminal module No. 14 area	Only for A2C-CPU														
D9021				Remote terminal module No. 1 area																																
D9022				Remote terminal module No. 2 area																																
:																																				
:																																				
:																																				
:																																				
D9032				Remote terminal module No. 12 area																																
D9033				Remote terminal module No. 13 area																																
D9034				Remote terminal module No. 14 area																																
D9022																																				
D9023																																				
D9024																																				
D9025																																				
D9026																																				
D9027																																				
D9028																																				
D9029																																				
D9030																																				
D9031																																				
D9032																																				
D9033																																				
D9034																																				
D9035	Attribute of remote terminal module	0: MINI standard protocol 1: No protocol	<p>Sets attribute of each remote terminal module connected to A2C and A52G with 0 or 1 at each bit.</p> <p>0: Conforms to the MINI standard protocol or remote terminal unit. 1: No-protocol mode of AJ35PTF-R2</p> <p>Data configuration:</p> <table border="1" style="margin-left: 40px;"> <tr> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table> <p style="margin-left: 100px;">↑ Indication of module with blown fuse</p> <p>Bit b0 specifies the attribute for special function module 1, bit b1 for module 2, b2 for module 3 etc.</p>	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																					
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0																					
D9036	Total number of stations	1 to 64	Sets the total number of stations (1 to 64) of I/O modules and remote terminal modules which are connected to an A2C or A52G.	Only for A2C-CPU																																
D9036	For designating extension file register device numbers	The device number used for getting direct access to each device for extension file register	Designate the device number for the extension file register for direct read and write in 2 words at D9036 and D9037 in BIN data. Use consecutive numbers beginning with R0 of block No. 1 to designate device numbers.	Only for AnA-, AnAS-, AnU-CPU																																
D9037																																				
D9038	LED indication priority	Priority 1 to 4	<p>Sets priority of ERROR LEDs which illuminate (or flicker) to indicate errors with error code numbers.</p> <p>Configuration of the priority setting areas is as shown below.</p> <table border="1" style="margin-left: 40px;"> <tr> <td>Priority 4</td> <td>Priority 3</td> <td>Priority 2</td> <td>Priority 1</td> </tr> <tr> <td></td> <td></td> <td>Priority 0</td> <td></td> </tr> </table> <p>For details, refer to the applicable CPUs User s Manual and the ACPU (Fundamentals) Programming manual.</p>	Priority 4	Priority 3	Priority 2	Priority 1			Priority 0		Only for AnA-, AnAS-, AnU-CPU, A2C-CPU, AnS-CPU																								
Priority 4		Priority 3		Priority 2	Priority 1																															
		Priority 0																																		
D9039	Priority 5 to 7																																			
D9055	Status latch step number	Status latch step	Stores the step number when status latch was executed.	Only for AnA-, AnAS-, AnU-CPU																																

# Table of Special Registers

Table of special registers (A series)

Number	Meaning	Description	Details	CPU																																																																																					
D9056 to D9059	Indication of an erroneous station	0: Normal 1: Error	<p>If an error occurs during the communication with a remote module, the corresponding bit of this module is set "1" in the register. This bit is set "1", if the communication failed after the number of retries specified in D9174. The bit even remains set, if the error has been corrected and the station has been re-joined.</p> <p>Data configuration:</p> <table border="1"> <tr> <td></td> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>D9056</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9057</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9058</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9059</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </table> <p>Station numbers</p> <p>The corresponding bit is even set "1", if the fuse of a remote I/O module is blown.</p>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9056	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9057	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9058	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9059	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	Only for A2C-CPU
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																									
D9056	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																									
D9057	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																									
D9058	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																									
D9059	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																									
D9061	Communication error code	0: Normal 1: Initial data error 2: Line error	<p>Stores error code when M9061 is turned on (communication with I/O modules or remote terminal modules fails).</p> <p>1: Total number of stations of I/O modules or remote terminal modules or number of retries is not normal. Initial program contains an error. 2: Cable breakage or power supply of I/O modules or remote terminal modules is turned off.</p>	Only for A2C-CPU																																																																																					
D9072	PC communication check	Data check by AJ71C24(S3/S6/S8)	In the loopback test mode of individual AJ71C24(S3/S6/S8), the AJ71C24(S3/S6/S8) executes data write/read and communication check.	Only for AnA-, AnAS-, AnU-CPU																																																																																					
D9073	Clock data	Year and month	The functions correspond to that of the special register D9035.	Only for A2C-CPU C24(-PRF)																																																																																					
D9074	Clock data	Day and hour	The functions correspond to that of the special register D9036.																																																																																						
D9075	Clock data	Minute and second	The functions correspond to that of the special register D9037.																																																																																						
D9076	Clock data	Day of week	The functions correspond to that of the special register D9038.																																																																																						
D9081	Number of communication request executed to remote terminal modules	0 to 32	Stores the number of communication requests executed to remote terminal modules connected to AJ71PT32(S3), A2C and A52G. Subtracts 1 at completion of communication with a remote terminal module.	Only for AnA-, AnAS-, AnU-CPU, A2C-CPU																																																																																					
D9082	Final connected station number	Final connected station number	Stores the final station number of I/O modules and remote terminal modules connected to A2C and A52G.	Only for A2C-CPU																																																																																					
D9090	Microcomputersubroutine input data area head device number	Depends on the microcomputer program package to be used.	For details, refer to the manual of each microcomputer program package.	Not for AnA-, AnAS-, AnU-CPU																																																																																					
D9091	Instruction error	Instruction error detail number	Stores the detail code of cause of an instruction error.	Only for AnA- AnAS, AnU-CPU																																																																																					
D9091	Microcomputersubroutine call error code	Depends on the microcomputer program package to be used.	For details, refer to the manual of each microcomputer program package.	Not for AnA- AnAS, AnU-CPU																																																																																					
②③ D9094	Changed I/O module head address	Changed I/O module head address	<p>Stores upper 2 digits of the head I/O address of I/O modules to be loaded or unloaded during online mode in BIN code.</p> <p>Example: Input module X2F0 = H2F</p>	Only for AnN-CPU, AnA- AnAS-, AnU-CPU																																																																																					
① D9100	Fuse blow module	Fuse blow module bit pattern	<p>Stores the output module number of the fuses have blown in the bit pattern.</p> <table border="1"> <tr> <td></td> <td>b15</td><td>b14</td><td>b13</td><td>b12</td><td>b11</td><td>b10</td><td>b9</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td> </tr> <tr> <td>D9100</td> <td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table> <p>↑ Indication of module with blown fuse</p>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9100	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	Only for A1S-CPU																																																			
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																									
D9100	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0																																																																									

# Table of Special Registers

Table of special registers (A series)

Number	Meaning	Description	Details	CPU
① D9100	Fuse blown module	Bit pattern in units of 16 points of fuse blow modules	<p>Output module numbers (in units of 16 points), of which fuses have blown, are entered in bit pattern. (Preset output unit numbers when parameter setting has been performed.)</p> <pre>           b15 b14 b13 b12 b11 b10 b9  b8  b7  b6  b5  b4  b3  b2  b1  b0 D9100  [ 0  0  0  1  0  0  0 ] 0  0  0  0  0  0  0  0  0 D9101  [ 1  0  0  0  0  1  0 ] 0  0  0  0  0  0  0  0  0  D9107  [ 0  0  0  1  0  0 ] 0  0  0  0  0  1  0  0  0                     </pre> <p>Fuse blow check is executed also to the output module of remote I/O station. (If normal status is restored, clear is not performed. Therefore, it is required to perform clear by user program.)</p>	Not for A1S-CPU, with AnS-CPU only D9100 to D9103
① D9101				
① D9102				
① D9103				
① D9104				
① D9105				
① D9106				
① D9107				
① D9116	I/O modules with verification error	Bit pattern of module numbers	<p>If the current status of an I/O module differs from the prescribed status after POWER ON, within the bit pattern of D9116 the bit assigned to the according I/O module is set (assignment via parameters). After restoring the normal status the bit has to be reset to 0 by the sequence program.</p>	Only for A1S-CPU
D9116	I/O module verify error	Bit pattern in units of 16 points of verify error units	<p>When I/O modules, of which data are different from those entered at power-on, have been detected, the I/O unit numbers (in units of 16 points) are entered in bit pattern. (Preset I/O unit numbers when parameter setting has been performed.)</p> <pre>           b15 b14 b13 b12 b11 b10 b9  b8  b7  b6  b5  b4  b3  b2  b1  b0 D9116  [ 0  0  0  1  0  0  0 ] 0  0  0  0  0  0  0  0  0 D9117  [ 1  0  0  0  0  1  0 ] 0  0  0  0  0  0  0  0  0       ↓       :       : D9123  [ 0  1  0  0  0  0  0 ] 0  0  0  0  0  0  1  0  0  0                     ↑                     Indication of I/O module                     with verification error                     </pre> <p>I/O module verify check is executed also to remote I/O station modules. (If normal status is restored, clear is not performed. Therefore, it is required to perform clear by user program.)</p>	Not for A1S-CPU, with AnS-CPU only D9116 to D9119)
D9117				
D9118				
D9119				
D9120				
D9121				
D9122				
D9123				
D9124	Annunciator detection quantity	Annunciator detection quantity	<p>When one of F0 to 255 is turned on by OUT F or SET F 1 is added to the contents of D9124. When RST F or LEDR instruction is executed, 1 is subtracted from the contents of D9124. (If the INDICATOR RESET switch is provided to the CPU, pressing the switch can execute the same processing.) Quantity, which has been turned on by OUT F or SET F is stored into D9124 in BIN code. The value of D9124 is maximum 8.</p>	



# Table of Special Registers

Table of special registers (A series)

Number	Meaning	Description	Details	CPU																																																																	
D9141 — D9172	Number of times of retry execution	Number of retries	<p>Stores the number of retries executed to I/O modules or remote terminal modules which caused communication error. (Retry processing is executed the number of times set at D9174.)</p> <p>Data becomes 0 when communication is restored to normal.</p> <p>Station number setting of I/O modules and remote terminal modules is as shown below.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">D9141</td> <td style="text-align: center;">b15</td> <td style="text-align: center;">-</td> <td style="text-align: center;">b8</td> <td style="text-align: center;">b7</td> <td style="text-align: center;">-</td> <td style="text-align: center;">b0</td> </tr> <tr> <td></td> <td colspan="2" style="text-align: center;">Station 2</td> <td colspan="2"></td> <td colspan="2" style="text-align: center;">Station 1</td> </tr> <tr> <td style="text-align: center;">D9142</td> <td colspan="2" style="text-align: center;">Station 4</td> <td colspan="2"></td> <td colspan="2" style="text-align: center;">Station 3</td> </tr> <tr> <td style="text-align: center;">D9143</td> <td colspan="2" style="text-align: center;">Station 6</td> <td colspan="2"></td> <td colspan="2" style="text-align: center;">Station 5</td> </tr> <tr> <td style="text-align: center;">↓</td> <td style="text-align: center;">:</td> <td colspan="2"></td> <td style="text-align: center;">:</td> <td colspan="2" style="text-align: center;">:</td> </tr> <tr> <td style="text-align: center;">D9171</td> <td colspan="2" style="text-align: center;">Station 62</td> <td colspan="2"></td> <td colspan="2" style="text-align: center;">Station 61</td> </tr> <tr> <td style="text-align: center;">D9172</td> <td colspan="2" style="text-align: center;">Station 64</td> <td colspan="2"></td> <td colspan="2" style="text-align: center;">Station 63</td> </tr> </table> <p>Retry counter uses 8 bits for one station.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">b(n+7)</td> <td style="text-align: center;">b(n+6)</td> <td style="text-align: center;">b(n+5)</td> <td style="text-align: center;">b(n+4)</td> <td style="text-align: center;">b(n+3)</td> <td style="text-align: center;">b(n+2)</td> <td style="text-align: center;">b(n+1)</td> <td style="text-align: center;">b(n+0)</td> </tr> <tr> <td style="text-align: center;">0/1</td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> </table> <p style="text-align: center;">Number of retries</p> <p>0: Normal 1: Station error</p> <p>"n" is determined by station number of I/O module or remote terminal module.</p> <p>Odd number stations: b0 to b7 (n = 0) Even number stations: b8 to b15 (n = 8)</p>	D9141	b15	-	b8	b7	-	b0		Station 2				Station 1		D9142	Station 4				Station 3		D9143	Station 6				Station 5		↓	:			:	:		D9171	Station 62				Station 61		D9172	Station 64				Station 63		b(n+7)	b(n+6)	b(n+5)	b(n+4)	b(n+3)	b(n+2)	b(n+1)	b(n+0)	0/1								Only for A2C-CPU
D9141	b15	-	b8	b7	-	b0																																																															
	Station 2				Station 1																																																																
D9142	Station 4				Station 3																																																																
D9143	Station 6				Station 5																																																																
↓	:			:	:																																																																
D9171	Station 62				Station 61																																																																
D9172	Station 64				Station 63																																																																
b(n+7)	b(n+6)	b(n+5)	b(n+4)	b(n+3)	b(n+2)	b(n+1)	b(n+0)																																																														
0/1																																																																					
D9173	Mode setting	0: Automatic online return enabled 1: Automatic online return disabled 2: Transmission stop at online error 3: Line check	<p>Mode setting</p> <p>Mode 0 (Automatic online return enabled): When an I/O module or a remote terminal module caused a communication error, the station is placed offline. The communication with normal stations is continued. When the faulty station returns to normal, it is placed online.</p> <p>Mode1 (Automatic online return disabled): When an I/O module or a remote terminal module caused a communication error, the station is placed offline. The communication with normal stations is continued. Though a faulty station returned to normal, communication is not restored unless the station module is restarted.</p> <p>Mode 2 (Transmission stop at online error): When an I/O module or a remote terminal module caused a communication error, communications with all stations is stopped. Though a faulty station returned to normal, communication is not restored unless the station module is restarted.</p> <p>Mode 3 (Line check): Checks hardware and connecting cables of I/O modules and remote terminal modules.</p>	Only for A2C-CPU																																																																	
D9174	Setting of the number of retries	Number of retries	<p>Sets the number of retries executed to I/O modules and remote terminal modules and remote terminal modules which caused communication error.</p> <p>Sets for 5 times at power on.</p> <p>Set range: 0 to 32</p> <p>If communication with an I/O module or a remote terminal module is not restored to normal after set number of retries, such module is regarded as a faulty station.</p>	Only for A2C-CPU																																																																	
D9175	Line error retry counter	Number of retries	<p>Stores the number of retries executed at line error (time out).</p> <p>Data becomes 0 when line is restored to normal and communication with I/O modules and remote terminal modules is resumed.</p>	Only for A2C-CPU																																																																	

# Table of Special Registers

Table of special registers (A series)

Number	Meaning	Description	Details	CPU																																																																																					
D9180 — D9193	Remote terminal module error number	Error code (0: normal)	<p>Stores error code of a faulty remote terminal module when M9060 is turned on.</p> <p>The error code storage areas for each remote terminal module are as shown below.</p> <table style="margin-left: 40px;"> <tr><td>D9180</td><td>Remote special module No.1</td></tr> <tr><td>D9181</td><td>Remote special module No.2</td></tr> <tr><td>D9182</td><td>Remote special module No.3</td></tr> <tr><td>↓</td><td>:</td></tr> <tr><td>D9192</td><td>Remote special module No.13</td></tr> <tr><td>D9193</td><td>Remote special module No.14</td></tr> </table> <p>Error code is cleared in the following cases.            When the RUN key switch is moved from STOP to RUN.            (D9180 to D9183 are all cleared.)            When Yn4 of each remote terminal is set from OFF to ON.</p>	D9180	Remote special module No.1	D9181	Remote special module No.2	D9182	Remote special module No.3	↓	:	D9192	Remote special module No.13	D9193	Remote special module No.14	Only for A2C-CPU																																																																									
D9180	Remote special module No.1																																																																																								
D9181	Remote special module No.2																																																																																								
D9182	Remote special module No.3																																																																																								
↓	:																																																																																								
D9192	Remote special module No.13																																																																																								
D9193	Remote special module No.14																																																																																								
D9196 — D9199	Faulty station detection	Bit pattern of the faulty station	<p>Bit which corresponds to faulty I/O module or remote terminal module is set (1).            (Bit which corresponds to a faulty station is set when normal communication cannot be restored after executing the number of retries set at D9174.)            If automatic online return is enabled, bit which corresponds to a faulty station is reset (0) when the station is restored to normal.</p> <p>Data configuration</p> <table style="margin-left: 40px;"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr><td>D9196</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr> <tr><td>D9197</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td></tr> <tr><td>D9198</td><td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td></tr> <tr><td>D9199</td><td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td></tr> </tbody> </table>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9196	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9197	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9198	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9199	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	Only for A2C-CPU
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																									
D9196	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																									
D9197	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																									
D9198	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																									
D9199	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																									

**NOTE**

After switching OFF the power supply, a latch clear or a RESET all special registers are reset.  
 If the RUN key switch is switched to STOP the contents of the registers are retained.

The contents of the special registers tagged ❶ are even retained, if the normal status is restored. They can be reset as follows:

- Insert a program line into the sequence program that resets the special register via an RST instruction due to a specified execution condition.
- Force a RESET via a programming terminal.
- Reset the CPU by switching the key switch on the CPU to RESET.

The special registers tagged ❷ can only be set and reset by the sequence program.

The special registers tagged ❸ are set and reset in the test mode of a programming terminal.

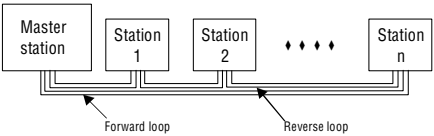
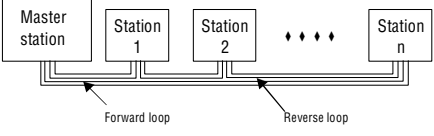


## A.5.3 Table of link registers (A series only)

Link registers are set or reset during data communications in a network depending on various conditions. They store the status of communications and errors within the network as a numeric value. By monitoring a link register any station number with a fault diagnosis can be read.

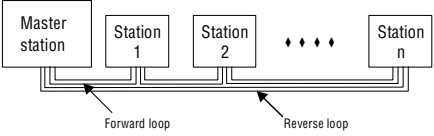
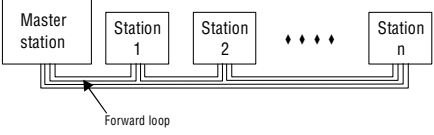
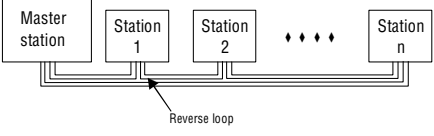
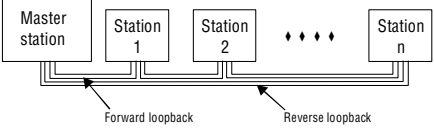
The processing of link registers depends on whether the CPU is installed in a master or a local station.

### Link registers in the master station

Number	Meaning	Description	Details
D9200	LRDP processing result	0: Normal 2: LRDP instruction setting fault 3: Corresponding station error 4: LRDP cannot be executed in the corresponding station	Stores the execution result of the LRDP (word device read) instruction.  LRDP instruction setting fault: Faulty setting of the LRDP instruction constant, source, and/or destination. Corresponding station error: One of the stations is not communicating. LRDP cannot be executed in the corresponding station: The specified station is a remote I/O station.
D9201	LWTP processing result	0: Normal 2: LWTP instruction setting fault 3: Corresponding station error 4: LWTP cannot be executed in the corresponding station	Stores the execution result of the LWTP (word device write) instruction.  LWTP instruction setting fault: Faulty setting of the LWTP instruction constant, source, and/or destination. Corresponding station error: One of the stations is not communicating. LWTP cannot be executed in the corresponding station: The specified station is a remote I/O station.
D9202	Link type of a local station (see also D9241, D9242)	Status of stations 1 to 16	The data registers store the compatibility of the slave stations to the MELSECNET or MELSECNET/II. If a slave station is compatible to the MELSECNET/II the corresponding bit of the special register stores a "1". If it is compatible to the MELSECNET a "0" is stored.
D9203		Status of stations 17 to 32	
D9204 (Continue)	Link status	0: Data link in forward loop 1: Data link in reverse loop 2: Loopback in forward/reverse direction 3: Loopback in forward direction 4: Loopback in reverse direction 5: Data link impossible	Stores the present path status of the data link.  Data link in forward loop   Data link in reverse loop 

# Table of Special Registers

## Link registers in the master station

Number	Meaning	Description	Details
D9204	Link status		<p>Loopback in forward/reverse loops</p>  <p>Loopback in forward loop only</p>  <p>Loopback in reverse loop only</p> 
D9205*	Loop executing station	Station executing forward loopback	Stores the local or remote I/O station number at which loopback is being executed.
D9206*	Loop executing station	Station executing reverse loopback	 <p>In the above example, 1 is stored into D9205 and 3 into D9206. If data link returns to normal status (data link in forward loop), values in D9205 and D9206 remain 1 and 3. Reset using sequence program or the RESET key.</p>
D9207*	Link scan time	Maximum value	Stores the data link processing time with all local and remote I/O stations.  Input (X), output (Y), link relay (B), and link register (W) assigned in link parameters communication with the corresponding stations every link scan.  Link scan is a period of time during which data link is executed with all connected slave stations, independently of the sequence program scan time.
D9208*	Link scan time	Minimum value	
D9209*	Link scan time	Present value	
D9210*	Retry count	Total number stored	Stores the number of retry times due to transmission error. Count stops at a maximum of "FFFF <sub>H</sub> ". RESET to return the count to 0.
D9211*	Loop switching count	Total number stored	Stores the number of times the loop line has been switched to reverse loop or loopback.

# Table of Special Registers

## Link registers in the master station

Number	Meaning	Description	Details																																																																																					
D9212	Local station operating status	Stores the status of stations 1 to 16	<p>Stores the local station numbers which are in STOP or PAUSE mode.</p> <table border="1"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9212</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9213</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9214</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9215</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </tbody> </table> <p>Station numbers</p> <p>When a local station is switched to STOP or PAUSE mode, the bit corresponding to the station number in the register becomes "1".</p> <p>Example: When station 7 switches to STOP mode, bit 6 in D9212 becomes "1", and when D9212 is monitored, its value is "64 (49h)".</p>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9212	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9213	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9214	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9215	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																						
D9212		16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																						
D9213		32		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																						
D9214	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																								
D9215	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																								
D9213	Stores the status of stations 17 to 32																																																																																							
D9214*	Stores the status of stations 33 to 48																																																																																							
D9215*	Stores the status of stations 49 to 64																																																																																							
D9216	Local station error detection	Stores the status of stations 1 to 16	<p>Stores the local station numbers which are in error.</p> <table border="1"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9216</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9217</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9218</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9219</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </tbody> </table> <p>Station numbers</p> <p>If a local station detects an error, the bit corresponding to the station number becomes "1".</p> <p>Example: When station 6 and 12 detect an error, bits 5 and 11 in D9216 become "1", and when D9216 is monitored, its value is "2080 (820h)".</p>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9216	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9217	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9218	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9219	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																						
D9216		16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																						
D9217		32		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																						
D9218	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																								
D9219	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																								
D9217	Stores the status of stations 17 to 32																																																																																							
D9218*	Stores the status of stations 33 to 48																																																																																							
D9219*	Stores the status of stations 49 to 64																																																																																							
D9220	Local station parameter mismatched or remote station I/O assignment error	Stores the status of stations 1 to 16	<p>Stores the local station numbers which contain mismatched parameters or of remote station numbers for which incorrect I/O assignment has been made.</p> <table border="1"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9220</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9221</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9222</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9223</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </tbody> </table> <p>Station numbers</p> <p>If a local station acting as the master station of tier 3 detects a parameter error or a remote station contains an invalid I/O assignment, the bit corresponding to the station number becomes "1".</p> <p>Example: When local station 5 and remote I/O station 14 detect an error, bits 4 and 13 in D9220 become "1", and when D9220 is monitored, its value is "8208 (2010h)".</p>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9220	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9221	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9222	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9223	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																						
D9220		16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																						
D9221		32		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																						
D9222	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																								
D9223	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																								
D9221	Stores the status of stations 17 to 32																																																																																							
D9222*	Stores the status of stations 33 to 48																																																																																							
D9223*	Stores the status of stations 49 to 64																																																																																							
D9224	Initial communication between local or remote I/O stations	Stores the status of stations 1 to 16	<p>Stores the local or remote station numbers while they are communicating the initial data with their relevant master station.</p> <table border="1"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9224</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9225</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9226</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9227</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </tbody> </table> <p>Station numbers</p> <p>The bit corresponding to the station number which is currently communicating the initial settings becomes "1".</p> <p>Example: When stations 23 and 45 are communicating, bit 6 of D9225 and bit 12 of D9226 become "1", and when D9225 is monitored, its value is "64 (40h)", and when D9226 is monitored, its value is "4096 (1000h)".</p>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9224	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9225	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9226	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9227	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																						
D9224		16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																						
D9225		32		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																						
D9226	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																								
D9227	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																								
D9225	Stores the status of stations 17 to 32																																																																																							
D9226*	Stores the status of stations 33 to 48																																																																																							
D9227*	Stores the status of stations 49 to 64																																																																																							

# Table of Special Registers

## Link registers in the master station

Number	Meaning	Description	Details																																																																																																																																																																																																																																																																																																	
D9228	Local or remote I/O station error	Stores the status of stations 1 to 16	Stores the local or remote station numbers which are in error.  <table border="1"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9228</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9229</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9230</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9231</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </tbody> </table>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9228	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9229	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9230	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9231	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																																																																																																																																																												
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																																																																																																																																																																																																																																		
D9228		16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																																																																																																																																																																																																																																		
D9229		32		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																																																																																																																																																																																																																																		
D9230	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																																																																																																																																																																																																																																				
D9231	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																																																																																																																																																																																																																																				
D9229	Stores the status of stations 17 to 32																																																																																																																																																																																																																																																																																																			
D9230*	Stores the status of stations 33 to 48																																																																																																																																																																																																																																																																																																			
D9231*	Stores the status of stations 49 to 64																																																																																																																																																																																																																																																																																																			
D9232	Local or remote I/O station loop error	Stores the status of stations 1 to 8	Stores the local or remote station numbers which a forward or reverse loop error has occurred.  <table border="1"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9232</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>D9233</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>D9234</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>D9235</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>D9236</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>D9237</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>D9238</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>D9239</td> <td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td><td>R</td><td>V</td> </tr> <tr> <td></td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </tbody> </table> <p>The bit corresponding to the station number with the error becomes "1".</p> <p>Example: When local station 3 and remote I/O station 14 have an error, bits 2 and 13 of D9228 become "1", and when D9228 is monitored, its value is "8196 (2004<sub>H</sub>)".</p>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9232	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		8	7	6	5	4	3	2	1									D9233	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		16	15	14	13	12	11	10	9									D9234	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		24	23	22	21	20	19	18	17									D9235	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		32	31	30	29	28	27	26	25									D9236	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		40	39	38	37	36	35	34	33									D9237	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		48	47	46	45	44	43	42	41									D9238	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		56	55	54	53	52	51	50	49									D9239	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V		64	63	62	61	60	59	58	57								
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																																																																																																																																																																																																																																		
D9232		R		V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																		
		8		7	6	5	4	3	2	1																																																																																																																																																																																																																																																																																										
D9233		R		V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																		
		16		15	14	13	12	11	10	9																																																																																																																																																																																																																																																																																										
D9234		R		V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																		
		24		23	22	21	20	19	18	17																																																																																																																																																																																																																																																																																										
D9235		R		V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																		
	32	31	30	29	28	27	26	25																																																																																																																																																																																																																																																																																												
D9236	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																				
	40	39	38	37	36	35	34	33																																																																																																																																																																																																																																																																																												
D9237	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																				
	48	47	46	45	44	43	42	41																																																																																																																																																																																																																																																																																												
D9238	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																				
	56	55	54	53	52	51	50	49																																																																																																																																																																																																																																																																																												
D9239	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	V																																																																																																																																																																																																																																																																																				
	64	63	62	61	60	59	58	57																																																																																																																																																																																																																																																																																												
D9233	Stores the status of stations 9 to 16																																																																																																																																																																																																																																																																																																			
D9234	Stores the status of stations 17 to 24																																																																																																																																																																																																																																																																																																			
D9235	Stores the status of stations 25 to 32																																																																																																																																																																																																																																																																																																			
D9236	Stores the status of stations 33 to 40																																																																																																																																																																																																																																																																																																			
D9237	Stores the status of stations 41 to 48																																																																																																																																																																																																																																																																																																			
D9238	Stores the status of stations 49 to 56																																																																																																																																																																																																																																																																																																			
D9239	Stores the status of stations 57 to 64																																																																																																																																																																																																																																																																																																			
D9240	Number of receive error detection times	Total number stored	Stores the number of times the following transmission errors have been detected: CRC, OVER, AB,IF Count is made to a maximum of FFFH. RESET to return the count to 0.																																																																																																																																																																																																																																																																																																	
D9241*	Link type of local station (see also D9202, D9203)	Stores the status of stations 33 to 48	The data registers store the compatibility of the slave stations to the MELSECNET or MELSECNET/II. If a slave station is compatible to the MELSECNET/II the corresponding bit of the special register stores a "1". If it is compatible to the MELSECNET a "0" is stored.																																																																																																																																																																																																																																																																																																	
D9242*		Stores the status of stations 49 to 64																																																																																																																																																																																																																																																																																																		
D9243	Own station number check	Stores a station number (0 to 64)	Allows a local station to confirm its own station number.																																																																																																																																																																																																																																																																																																	
D9244	Total number of slave stations	Stores the number of slave stations	Indicates the number of slave stations in one loop.																																																																																																																																																																																																																																																																																																	
D9245	Number of receive error detection times	Total number stored	Stores the number of times the following transmission errors have been detected: CRC, OVER, AB,IF Count is made to a maximum of FFFF <sub>H</sub> . RESET to return the count to 0.																																																																																																																																																																																																																																																																																																	

# Table of Special Registers

## Link registers in the master station

Number	Meaning	Description	Details																																																																																					
D9248	Local station operating status	Stores the status of stations 1 to 16	Stores the local station number which is in STOP or PAUSE mode.  <table border="1" style="font-size: small; border-collapse: collapse; margin: 10px auto;"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9196</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9197</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9198</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9199</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </tbody> </table>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9196	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9197	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9198	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9199	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																						
D9196		16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																						
D9197		32		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																						
D9198	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																								
D9199	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																								
D9249	Stores the status of stations 17 to 32																																																																																							
D9250*	Stores the status of stations 33 to 48																																																																																							
D9251*	Stores the status of stations 49 to 64	The bit corresponding to the station number which is in STOP or PAUSE mode, becomes "1".  Example: When local stations 7 and 15 are in STOP mode, bits 6 and 14 of D9248 become "1", and when D9248 is monitored, its value is "16448 (4040 <sub>h</sub> )".																																																																																						
D9252	Local station error	Stores the status of stations 1 to 16	Stores the local station number other than the host, which is in error.  <table border="1" style="font-size: small; border-collapse: collapse; margin: 10px auto;"> <thead> <tr> <th></th> <th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th> </tr> </thead> <tbody> <tr> <td>D9196</td> <td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td> </tr> <tr> <td>D9197</td> <td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td> </tr> <tr> <td>D9198</td> <td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td> </tr> <tr> <td>D9199</td> <td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td> </tr> </tbody> </table>		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	D9196	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	D9197	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	D9198	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	D9199	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
		b15		b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0																																																																						
D9196		16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																																																						
D9197		32		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17																																																																						
D9198	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33																																																																								
D9199	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49																																																																								
D9253	Stores the status of stations 17 to 32																																																																																							
D9254*	Stores the status of stations 33 to 48																																																																																							
D9255*	Stores the status of stations 49 to 64	The bit corresponding to the station number which is in error, becomes "1".  Example: When local station 12 is in error, bit 11 of D9252 becomes "1", and when D9252 is monitored, its value is "2048 (800 <sub>h</sub> )".																																																																																						

\* The tagged special registers cannot be applied within MELSECNET/B.



# Index

A	B
A series compatible data link instructions . . . . .	8-72
A to System Q/QnA series conversion	
special registers . . . . .	A-104
special relays . . . . .	A-58
ACOS, ACOSP . . . . .	7-356
Adding clock data . . . . .	7-434
Address configuration of the data storage area	12-3
Addressing of	
arrays . . . . .	3-20
registers . . . . .	3-19
AnAS/AnUS CPUs . . . . .	1-2
AnA/AnU CPUs . . . . .	1-2
ANB . . . . .	5-11
AND . . . . .	5-4
ANDF . . . . .	5-8
ANDP . . . . .	5-8
ANI . . . . .	5-4
AnN CPUs . . . . .	1-2
AnS CPUs . . . . .	1-2
ANY (data type) . . . . .	3-9
Application instructions, Part 1 . . . . .	6-1
Application Instructions, Part 2 . . . . .	7-1
Applying user-created microcomputer programs	12-2
Arcus cosine calculation	
from BCD data . . . . .	7-396
of floating point values . . . . .	7-357
Arcus sine calculation	
from BCD data . . . . .	7-393
from floating point values . . . . .	7-354
Arcus tangent calculation	
from BCD data . . . . .	7-399
of floating point values . . . . .	7-360
Arithmetic operation instructions . . . . .	6-25
Arithmetic Operation instructions (overview) . .	2-15
Array address/ initial address conversion . . . .	3-21
Arrays (addressing) . . . . .	3-20
ASC (A series) . . . . .	7-316
ASC, ASCP (Q series) . . . . .	7-313
ASIN, ASINP . . . . .	7-353
ATAN, ATANP . . . . .	7-359
BACOS, BACOSP . . . . .	7-395
BAND, BANDP, DBAND, DBANDP . . . . .	7-406
BASIN, BASINP . . . . .	7-392
BATAN, BATANP . . . . .	7-398
Batch	
reset of bits . . . . .	7-74
save and recovery of index register	
contents . . . . .	7-489
write of data to an EEPROM file register	7-492
BBLKRD, BBLKRD . . . . .	11-27
BBLKWR, BBLKWRP . . . . .	11-30
BCD data arithmetic operation instructions . .	6-27
BCD 4-digit	
addition and subtraction operations . . . . .	6-44
multiplication and division operations . . . .	6-54
BCD 8-digit	
addition and subtraction operations . . . . .	6-49
multiplication and division operations . . . .	6-57
BCDDA, BCDDAP, DBCDDA, DBCDDAP . .	7-258
BCD, BCDP, DBCD, DBCDP . . . . .	6-83
BCOS, BCOSP . . . . .	7-386
BIN block	
addition and subtraction operations . . . . .	6-69
data comparisons . . . . .	6-21
data exchange . . . . .	6-142
data transfer . . . . .	6-133
BIN data	
arithmetic operation instructions . . . . .	6-27
exchange . . . . .	6-139
inversion . . . . .	6-128
transfer . . . . .	6-119
BIN 16-bit	
addition and subtraction operations . . . . .	6-29
increment and decrement operations . . . . .	6-76
multiplication and division . . . . .	6-37
BIN 16-bit and 32-bit	
dead band control . . . . .	7-407
zone control . . . . .	7-411
BIN 16-bit data	
comparisons . . . . .	6-6
BIN 32-bit	
addition and subtraction operations . . . . .	6-34
increment and decrement operations . . . . .	6-79
multiplication and division . . . . .	6-41
BIN 32-bit data	
comparison . . . . .	6-9
BINDA, BINDAP, DBINDA, DBINDAP . . . . .	7-248
BINHA, BINHAP, DBINHA, DBINHAP . . . . .	7-253
BIN, BINP, DBIN, DBINP . . . . .	6-86
Bit blocks . . . . .	3-12
of double word data . . . . .	3-14
of word data . . . . .	3-13

- Bit devices
    - of double word data. . . . . 3-14
    - of word data . . . . . 3-12
    - usage in instructions . . . . . 3-11
  - Bit processing instructions . . . . . 7-65
  - Bit processing instructions (overview). . . . . 2-37
  - BKAND, BKANDP. . . . . 7-11
  - BKBCD, BKBCDP. . . . . 6-111
  - BKBIN, BKBINP . . . . . 6-114
  - BKCMP, BKCMP\_P . . . . . 6-20
  - BKOR, BKORP. . . . . 7-20
  - BKRST, BKRSTP . . . . . 7-73
  - BKXNR, BKXNRP. . . . . 7-39
  - BKXOR, BKXORP . . . . . 7-29
  - BK+, BK+P, BK-, BK-P . . . . . 6-68
  - Block transfer (for System Q). . . . . 9-42
  - Blown fuse detection (special registers) . . . . . A-102
  - BMOV, BMOVP . . . . . 6-132
  - BREAK, BREAKP. . . . . 7-129
  - BSET, BSETP, BRST, BRSTP. . . . . 7-66
  - BSFR, BSFRP, BSFL, BSFLP . . . . . 7-59
  - BSIN, BSINP . . . . . 7-383
  - BSQR, BSQRP, BDSQR, BDSQRP . . . . . 7-379
  - BTAN, BTANP . . . . . 7-389
  - Buffer memory access instructions. . . . . 7-185
  - Buffer memory access instructions (overview). 2-44
  - BUFRCV . . . . . 11-34
  - BUFRCVS (ETHERNET interface module) . . 11-39
  - BUFRCVS (Serial communication modules) . . 11-3
  - BUFSND . . . . . 11-42
  - Building an input matrix. . . . . 6-204
  - BXCH, BXCHP . . . . . 6-141
  - Bx, BxP, B/, B/P . . . . . 6-53
  - Byte exchange . . . . . 6-145
  - B+, B+P, B-, B-P. . . . . 6-43
- 
- C**
- Calculating
    - totals of 16-bit BIN data blocks. . . . . 7-122
    - totals of 32-bit BIN data blocks. . . . . 7-124
  - Calculation of program steps
    - examples . . . . . 3-39
  - Calling
    - a subroutine program . . . . . 7-133
    - a subroutine program in a program file . . 7-142
  - CALL, CALLP. . . . . 7-132
  - Categories of instructions . . . . . 2-1
  - CGMODE. . . . . 10-4
  - Changing the clock data format . . . . . 7-444
  - Character string
    - linking operations . . . . . 6-73
    - searching . . . . . 7-333
    - storing and moving . . . . . 7-327
    - STRING instruction. . . . . 3-22
  - Character string data
    - comparison . . . . . 6-16
    - transfer . . . . . 6-125
  - Character string processing instructions. . . . 7-245
  - Character string processing instructions
    - (overview) . . . . . 2-47
  - Check data bits . . . . . 7-85
  - CHG . . . . . 7-147
  - CHG instruction
    - and counting of counters. . . . . 7-152
    - and processing of OUT instructions . . . . 7-153
    - and timing of timers . . . . . 7-153
    - in conjunction with a PLS instruction . . . 7-150
    - in conjunction with a pulsed instruction . 7-151
  - CHK (A series only) . . . . . 7-226
  - CHKCIR, CHKEND . . . . . 7-234
  - CHKST, CHK (Q series and System Q only) 7-218
  - CJ, SCJ, JMP . . . . . 6-148
  - Clock data
    - adding of . . . . . 7-434
    - format changing . . . . . 7-444
    - reading of . . . . . 7-426
    - subtraction of . . . . . 7-439
    - writing of . . . . . 7-430
  - Clock instructions. . . . . 7-424
  - Clock instructions (overview). . . . . 2-56
  - CLOSE . . . . . 11-56
  - CML, CMLP, DCML, DCMLP . . . . . 6-127
  - COM . . . . . 6-172
  - Comparison
    - Counters . . . . . A-38
    - Display instructions . . . . . A-39
    - of CPU modules. . . . . A-32
    - QnA CPU and System Q CPU . . . . . A-41
    - Timer. . . . . A-35
  - Comparison operation instructions (overview) 2-10
  - Comparison operation instructions. . . . . 6-2
  - COMRD, COMRDP . . . . . 7-277
  - Configuration of data memory area . . . . . 12-4
  - Connection instructions . . . . . 5-11
  - Connection instructions (overview) . . . . . 2-7
  - Connection (ETHERNET)
    - close . . . . . 11-58
    - open of . . . . . 11-50
  - Constants (use in an instruction) . . . . . 3-1
  - Conversion
    - from BCD block data into BIN block data 6-115
    - from BCD data into BIN data. . . . . 6-87
    - from BIN block data into BCD block data 6-112
    - from BIN data into BCD data. . . . . 6-84
    - from BIN data into floating point data . . . 6-91
    - from BIN data into Gray code data. . . . . 6-101
    - from BIN 16-bit data into BIN 32-bit data . 6-97
    - from BIN 32-bit data into BIN 16-bit data . 6-99
    - from degrees into radian as floating
      - point value. . . . . 7-363
    - from floating point data into BIN data . . . 6-94
    - from Gray code data into BIN data. . . . . 6-104



- Conversion
    - from radian in floating point format
      - into degrees . . . . . 7-366
  - Conversion of
    - BIN 16-bit data into ASCII code . . . . . 7-314
    - BIN 16-/32-bit binary data into
      - character strings . . . . . 7-285
    - character string data into ASCII code . . . 7-317
    - character string data into decimal
      - floating point data . . . . . 7-308
    - character strings into BIN 16-/32-bit
      - binary data . . . . . 7-293
    - decimal ASCII data into BIN 16-/32-bit
      - binary data . . . . . 7-264
    - decimal ASCII data into 4-/8-digit BCD
      - data . . . . . 7-273
    - floating point data into character
      - string data . . . . . 7-299
    - floating point data into the decimal format 7-340
    - floating point number into the BCD format 7-337
    - hexadecimal ASCII data into BIN 16-/32-bit
      - binary data . . . . . 7-269
    - hexadecimal ASCII values into
      - binary values . . . . . 7-319
    - 16-/32-bit binary data into decimal
      - values in ASCII code . . . . . 7-249
    - 16-/32-bit binary data into hexadecimal
      - values in ASCII code . . . . . 7-254
    - 4-/ 8-digit BCD data into ASCII code . . . 7-259
  - Cosine calculation
    - from BCD data . . . . . 7-387
    - from floating point values . . . . . 7-348
  - COS, COSP . . . . . 7-347
  - Counter function blocks . . . . . A-39
  - Counter functions . . . . . A-38
  - CPU shared memory
    - reading from . . . . . 9-50
    - writing to . . . . . 9-47
  - CPU table in instruction discription . . . . . 4-2
  - CPU types . . . . . 1-2
- 
- D**
- DABCD, DABCDP, DDABCD, DDABCDP . . 7-272
  - DABIN, DABINP, DDABIN, DDABINP . . . . 7-263
  - Data control instructions . . . . . 7-401
  - Data control instructions (overview) . . . . . 2-54
  - Data conversion instructions . . . . . 6-81
  - Data conversion instructions (overview) . . . . . 2-22
  - Data Link Instructions . . . . . 8-1
  - Data processing instructions . . . . . 7-76
  - Data processing instructions (overview) . . . . . 2-38
  - Data read/write ranges . . . . . 8-3
  - Data refresh instructions . . . . . 8-6
  - Data refresh instructions (overview) . . . . . 2-28
  - Data rotation
    - to the left (16-bit) . . . . . 7-47
    - to the left (32-bit) . . . . . 7-53
    - to the right (16-bit) . . . . . 7-44
    - to the right (32-bit) . . . . . 7-50
  - Data rotation instructions . . . . . 7-42
  - Data rotation instructions (overview) . . . . . 2-35
  - Data shift instructions . . . . . 7-55
  - Data table instructions (overview) . . . . . 2-43
  - Data table operation instructions . . . . . 7-167
  - Data transfer for Q4ARCPU . . . . . 10-6
  - Data transfer instructions . . . . . 6-117
  - Data transfer instructions (overview) . . . . . 2-25
  - Data types . . . . . 3-9
  - DATERD, DATERDP . . . . . 7-425
  - DATEWR, DATEWRP . . . . . 7-429
  - DATE+, DATE+P . . . . . 7-433
  - DATE-, DATE-P . . . . . 7-438
  - DBL, DBLP . . . . . 6-96
  - DBx, DBxP, DB/, DB/P . . . . . 6-56
  - DB+, DB+P, DB-, DB-P . . . . . 6-48
  - Debugging
    - Instructions . . . . . 7-217
    - Instructions for System Q . . . . . 9-7
    - Special registers . . . . . A-100
    - Special relays . . . . . A-56
  - Debugging instructions (overview) . . . . . 2-46
  - Decoding from 8 to 256 bits . . . . . 7-88
  - DECO, DECOP . . . . . 7-87
  - Dedicated data link instructions
    - for the QnA series . . . . . 8-11
    - General . . . . . 8-4
  - Dedicated data link instructions (overview) . . 2-60
  - Dedicated instructions . . . . . 3-6
  - DEG, DEGP . . . . . 7-365
  - Delete a program from memory . . . . . 9-37
  - Deleting and inserting data blocks in a
    - data table . . . . . 7-181
  - Destination of data (d) . . . . . 3-2
  - Detecting the length of character strings . . . 7-282
  - Devices
    - MELSEC A . . . . . 4-3
    - MELSEC Q . . . . . 4-4
  - Diagnostic information
    - special registers . . . . . A-74
  - DINC, DINCP, DDEC, DDECP . . . . . 6-78
  - DINT (data type) . . . . . 3-19
  - Direct read of one byte from a file register . . 7-474
  - Direct write of one byte to a file register . . . 7-478
  - Display instructions . . . . . 7-194
  - Display instructions (overview) . . . . . 2-45
  - Disuniting and uniting data in byte units . . . 7-107
  - Disuniting or uniting of data in random
    - bit groupings . . . . . 7-102
  - Disuniting 16-bit data . . . . . 7-96
  - DIS, DISP . . . . . 7-95

DI, EI, IMASK . . . . . 6-156  
 DROL, DROL P, DRCL, DRCLP . . . . . 7-52  
 DROR, DROR P, DRCP, DRCP P . . . . . 7-49  
 DSFR, DSFR P, DSFL, DSFLP . . . . . 7-62  
 DUTY . . . . . 7-470  
 DWORD . . . . . 3-19  
 DWSUM, DWSUM P . . . . . 7-123  
 Dx, DxP, D/, D/P . . . . . 6-40  
 D+, D+P, D-, D-P . . . . . 6-32  
 D=, D, D>, D= . . . . . 6-8

**E**

ECALL, ECALLP . . . . . 7-141  
 EFCALL, EFCALLP . . . . . 7-144  
 EI, DI . . . . . 6-175  
 EMOD, EMOD P . . . . . 7-336  
 EMOV, EMOV P . . . . . 6-121  
 EN input . . . . . 3-35  
 Encoding from 256 to 8 bits . . . . . 7-90  
 ENCO, ENCO P . . . . . 7-89  
 End of subroutine program . . . . . 7-136  
 ENEG, ENEG P . . . . . 6-109  
 ENO output . . . . . 3-35  
 EREXP, EREXP P . . . . . 7-339  
 EROMWR, EROMWR P . . . . . 7-491  
 ERRCLR . . . . . 11-61  
 Error code (ETHERNET module)  
   clearing of . . . . . 11-63  
   reading of . . . . . 11-69  
 Error codes  
   A series (except AnA and AnAS) . . . . . 13-39  
   AnA and AnAS CPUs . . . . . 13-43  
   QnA-series and System Q  
     (except Q00J, Q00, Q01 CPU) . . . . . 13-12  
   Q00JCPU, Q00CPU, Q01CPU . . . . . 13-2  
 ERRRD . . . . . 11-67  
 ESTR, ESTR P . . . . . 7-298  
 ETHERNET  
   Deletion of "ERR"-LED . . . . . 11-63  
   reading of an error code . . . . . 11-69  
   Re-initialisation . . . . . 11-74  
 EVAL, EVALP . . . . . 7-307  
 Exclusive NOR operations . . . . . 7-40  
 Exclusive OR operations . . . . . 7-30  
 Execution conditions . . . . . 2-5  
   for a link refresh . . . . . 6-176  
   for comparison operation instructions . . . . . 6-3  
 Execution conditions of the instructions . . . . . 3-34  
 EXP, EXP P . . . . . 7-371  
 Extraction of character string data . . . . . 7-323  
 Ex, ExP, E/, E/P . . . . . 6-65  
 E+, E+P, E-, E-P . . . . . 6-60  
 E=, E, E>, E= . . . . . 6-11

**F**

Failure check for bidirectional operations  
   A series only . . . . . 7-227  
   QnA series and System Q . . . . . 7-219  
 Failure diagnosis and debugging . . . . . 7-217  
 FCALL, FCALLP . . . . . 7-137  
 FDEL, FDEL P, FINS, FINS P . . . . . 7-180  
 FIFR, FIFR P . . . . . 7-172  
 FIFW, FIFW P . . . . . 7-168  
 File register switching instructions . . . . . 7-414  
 File register switching instructions (overview) . . . . . 2-55  
 Finding an instruction . . . . . 1-3  
 Floating point data  
   addition and subtraction operations . . . . . 6-61  
   comparisons . . . . . 6-12  
   multiplication and division operations . . . . . 6-66  
   transfer . . . . . 6-122  
 Floating point values as exponent  
   of the base e . . . . . 7-372  
 FLT, FLTP, DFLT, DFLTP . . . . . 6-90  
 FMOV, FMOV P . . . . . 6-135  
 FOR, NEXT . . . . . 7-126  
 FPOP, FPOP P . . . . . 7-176  
 FREAD . . . . . 9-20  
 FROM, DFRO . . . . . 7-186  
 FROM/FROM P . . . . . 9-50  
 Functions . . . . . 4-6  
 FWRITE . . . . . 9-9

**G**

GBIN, GBIN P, DGBIN, DGBIN P . . . . . 6-103  
 Generating check circuits for the  
   CHK instruction . . . . . 7-235  
 GETE, GETE P . . . . . 11-6  
 GOEND . . . . . 6-153  
 GRY, GRYP, DGRY, DGRYP . . . . . 6-100

**H**

HABIN, HABIN P, DHABIN, DHABIN P . . . . . 7-268  
 Header of the MELSEC-AWL . . . . . 3-7  
 HEX, HEX P . . . . . 7-318  
 Hierarchy of data types ANY . . . . . 3-9

<b>I</b>	
Identical BIN block data transfer . . . . .	6-136
IEC Commands for comparison . . . . .	6-3
INC, INCP, DEC, DECP . . . . .	6-75
Index qualification	
AnA, AnAS, and AnU CPUs . . . . .	3-28
General . . . . .	3-24
System Q and QnA CPU . . . . .	3-26
Index qualification of entire program parts . . .	7-160
Indexed device numbers	
storing in an index qualification table . . .	7-165
Indirect address	
storing . . . . .	7-481
Initial registers . . . . .	3-19
Input instructions . . . . .	5-4
Input instructions (overview) . . . . .	2-6
Instruction name . . . . .	2-4
Instruction related registers . . . . .	A-98
Instruction (structure) . . . . .	3-1
Instructions for a multi-CPU system . . . . .	9-46
Instructions for Q4ARCPU . . . . .	10-1
Instructions for special function modules . . .	11-1
INSTR, INSTRP . . . . .	7-332
Interrupt control instructions . . . . .	6-157
Interrupt control instructions (overview) . . . .	2-27
INT, INTp, DINT, DINTP . . . . .	6-93
INV . . . . .	5-17
IRET . . . . .	6-163
IXDEV, IXSET . . . . .	7-164
IX, IXEND . . . . .	7-159
I/O module verification (special registers) . .	A-103
I/O partial refresh	
A series . . . . .	6-169
Q series and System Q . . . . .	6-167
<b>J</b>	
Jump instructions . . . . .	6-149
Jump to the end of a program . . . . .	6-154
<b>K</b>	
KEY . . . . .	7-482
Key input of data at peripheral devices . . . .	7-454
<b>L</b>	
Ladder block parallel connection . . . . .	5-12
Ladder block series connection . . . . .	5-12
Latch Area	
special registers . . . . .	A-101
special relays . . . . .	A-57
LD . . . . .	5-4
LDF . . . . .	5-8
LDI . . . . .	5-4
LDP . . . . .	5-8
LED . . . . .	7-205
LEDA, LEDB . . . . .	7-211
LEDC . . . . .	7-208
LEDR . . . . .	7-213
LEN, LENP . . . . .	7-281
Limitation of output values . . . . .	7-403
LIMIT, LIMITP, DLIMIT, DLIMITP . . . . .	7-402
Link refresh . . . . .	6-173
Link Refresh Instructions . . . . .	6-165
Logarithm (ln) calculation from floating point values	7-375
Logical	
AND . . . . .	7-6
exclusive NOR . . . . .	7-33
OR . . . . .	7-15
product with 16-bit data blocks . . . . .	7-12
sum with 16-bit data blocks . . . . .	7-21
Logical operation instruction (overview) . . . .	2-31
Logical operation instructions . . . . .	7-2
LOG, LOGP . . . . .	7-374
LRDP . . . . .	8-81
LWTP . . . . .	8-85
<b>M</b>	
Master control instructions (overview) . . . . .	2-9
MAX, MAXP, DMAX, DMAXP . . . . .	7-111
MEF . . . . .	5-19
MELSECNET	
Pairing setting of stations . . . . .	11-80
Memory cards	
special registers . . . . .	A-95
special relays . . . . .	A-52
Memory map . . . . .	12-3
MEP . . . . .	5-19
Microcomputer Mode (AnN(S)) . . . . .	12-1
Microcomputer program	
Application . . . . .	12-2
calling . . . . .	7-157
MIDR, MIDRP, MIDW, MIDWP . . . . .	7-326
Miscellaneous instructions (overview) . . . . .	2-9
Module information, reading of . . . . .	9-2
MOV, MOVP, DMOV, DMOVp . . . . .	6-118
MPP . . . . .	5-14
MPS . . . . .	5-14
MRD . . . . .	5-14
MSG . . . . .	7-450
MTR . . . . .	6-203
M_REAL_TO_REAL . . . . .	3-18
M_REAL_TO_REAL_E . . . . .	3-18

**N**

NDIS, NDISP, NUNI, NUNIP . . . . .	7-101
NEG, NEGP, DNEG, DNEGP . . . . .	6-106
Network data refresh . . . . .	8-8
Network refresh instructions (overview) . . . . .	2-60
Notation of instructions . . . . .	3-3
Number of program steps	
For an AnA, AnAS, and AnU CPU . . . . .	3-38
Number (n) . . . . .	3-2
Numerical key input . . . . .	7-483

**O**

OPEN . . . . .	11-47
Operation errors	
in the explanation of the instruction . . . . .	4-6
when an instruction is executed . . . . .	3-31
Operation result	
conversion into pulse . . . . .	5-20
inversion . . . . .	5-18
processing . . . . .	5-15
Operation start . . . . .	5-5
OR . . . . .	5-4
ORB . . . . .	5-11
ORF . . . . .	5-8
ORI . . . . .	5-4
ORP . . . . .	5-8
Other convenient Instructions, Part 1 . . . . .	6-178
Other convenient instructions, Part 2 . . . . .	7-465
Other instructions (overview) . . . . .	2-58
Output instructions (overview) . . . . .	2-8
Output of messages to peripheral devices . . . . .	7-451
Output to a LED display	
LED . . . . .	7-206
LEDA, LEDB . . . . .	7-212
LEDC . . . . .	7-209
Output to a peripheral device	
PR . . . . .	7-197
PRC . . . . .	7-202
Overview	
Arithmetic operation instructions . . . . .	2-15
Bit processing instructions . . . . .	2-37
Buffer memory access instructions . . . . .	2-44
Character string processing instructions . . . . .	2-47
Clock instructions . . . . .	2-56
Comparison operation instructions . . . . .	2-10
Connection instructions . . . . .	2-7
Data control instructions . . . . .	2-54
Data conversion instructions . . . . .	2-22
Data processing instructions . . . . .	2-38
Data refresh instructions . . . . .	2-28
Data table instructions . . . . .	2-43
Data transfer instructions . . . . .	2-25
Data types . . . . .	A-34
Debugging instructions . . . . .	2-46
Dedicated data link instructions . . . . .	2-60

Devices for control data (READ) . . . . .	8-14
Devices for control data (RECV) . . . . .	8-46
Devices for control data (REQ) . . . . .	8-52
Devices for control data (SEND) . . . . .	8-38
Devices for control data (SREAD) . . . . .	8-19
Devices for control data (SWRITE) . . . . .	8-32
Devices for control data (WRITE) . . . . .	8-25
Devices for control data (ZNFR) . . . . .	8-61
Devices for control data (ZNT0) . . . . .	8-67
Display instructions . . . . .	2-45
Error code numbers . . . . .	7-232
File register switching instructions . . . . .	2-55
Further manuals . . . . .	1-1
Input instructions . . . . .	2-6
Instruction Tables . . . . .	2-1
Instructions for System Q CPUs . . . . .	9-1
Interrupt control instructions . . . . .	2-27
I/O control modes . . . . .	A-34
Logical operation instructions . . . . .	2-31
Master control instructions . . . . .	2-9
Miscellaneous instructions . . . . .	2-9
Network refresh instructions . . . . .	2-60
Other instructions . . . . .	2-58
Output instructions . . . . .	2-8
Peripheral device instructions . . . . .	2-57
Processing times (definition) . . . . .	A-1
Program branch instructions . . . . .	2-27
Program instructions . . . . .	2-57
Program termination instructions . . . . .	2-9
Rotation instructions . . . . .	2-35
Routing information . . . . .	2-61
Shift instructions . . . . .	2-8, 2-36
Special function instructions . . . . .	2-51
Special relays . . . . .	A-43
Structured program instructions . . . . .	2-41
System Q/Q series instructions	
equivalent to A series instructions . . . . .	A-40
Timer function blocks . . . . .	A-37

**P**

PAIRSET . . . . .	11-79
Parallel connection . . . . .	5-5
Peripheral device instructions (overview) . . . . .	2-57
Peripheral device instructions . . . . .	7-449
PKEY . . . . .	7-453
PLC parameters . . . . .	1-3
PLOADP . . . . .	9-33
PLOW, PLOWP . . . . .	7-463
PLSY . . . . .	6-199
POFF, POFFP . . . . .	7-459
Positioning instruction for rotary tables . . . . .	6-192
PPC-CPU-CPU686(MS)-128 . . . . .	1-2
PPC-CPU686(MS)-64 . . . . .	1-2
PR . . . . .	7-196
PRC . . . . .	7-201

Presetting the number of execution scans of a device . . . . .	7-471
Processing	
of bit data . . . . .	3-11
of double word data (32 bits) . . . . .	3-14
of word data (16 bits) . . . . .	3-12
Program	
Delete and load . . . . .	9-39
Delete from program memory . . . . .	9-37
loading from a memory card . . . . .	9-34
Program Branch Instructions . . . . .	6-147
Program branch instructions (overview) . . . . .	2-27
Program control instructions . . . . .	7-456
Program Execution Control Instructions . . . . .	6-155
Program instructions	
overview . . . . .	2-57
Program jumps . . . . .	6-149
Program organisation unit (POU) . . . . .	3-19
Program termination instructions (overview) . . . . .	2-9
Programm instructions	
for System Q . . . . .	9-33
Programmable (teaching) Timer . . . . .	6-186
Programming	
of dedicated instructions . . . . .	3-6
of variables . . . . .	3-7
PRR, PRRP . . . . .	11-18
PSCAN, PSCANP . . . . .	7-461
PSTOP, PSTOPP . . . . .	7-457
PSWAPP . . . . .	9-38
PTRA, PTRAR, PTRAEXE, PTRAEXEP . . . . .	7-243
Pulse	
parallel connection . . . . .	5-9
series connection . . . . .	5-9
Pulse density measurement . . . . .	6-198
Pulse output with adjustable number of pulses . . . . .	6-200
Pulse width modulation . . . . .	6-202
PUNLOADP . . . . .	9-36
PUTE, PUTEF . . . . .	11-11
PWM . . . . .	6-201

**Q**

QCDSET, QCDSETP . . . . .	7-421
QDRSET, QDRSETP . . . . .	7-418
QnA CPUs . . . . .	1-2
Q00CPU . . . . .	1-2
Q00JCPU . . . . .	1-2
Q01CPU . . . . .	1-2
Q02CPU . . . . .	1-2
Q02HCPU . . . . .	1-2
Q06HCPU . . . . .	1-2
Q12HCPU . . . . .	1-2
Q4ARCPU	
Data transfer . . . . .	10-6
Q4ARCPU, Mode setting . . . . .	10-2

**R**

RAD, RADP . . . . .	7-362
RAMP . . . . .	6-195
Ramp signal . . . . .	6-196
Randomizing values and series update . . . . .	7-378
RBMOV, RBMOVP . . . . .	9-41
READ . . . . .	8-12
Reading	
routing information . . . . .	8-103
Reading and writing routing information . . . . .	8-101
Reading clock data . . . . .	7-426
Reading data	
entered first from a data table . . . . .	7-173
entered last from a data table . . . . .	7-177
from a local station . . . . .	8-82
from a remote station . . . . .	8-91
from a special function module . . . . .	7-187
from other stations . . . . .	8-74
from special function modules in remote I/O stations . . . . .	8-62
Reading device comment data . . . . .	7-278
Reading of	
user registered frames . . . . .	11-8
Reading of data	
from a communication modul . . . . .	11-4
from a PROFIBUS interface module . . . . .	11-28
from another PLC (CC-Link) . . . . .	11-118
from another station on CC-Link . . . . .	11-111
from automatic updating buffer (A series) . . . . .	11-168
from automatic updating buffer (Q series) . . . . .	11-172
from fixed buffer . . . . .	11-36
from intelligent device station (CC-Link) . . . . .	11-138
from intell. device station (CC-Link) . . . . .	11-144
in an interrupt program . . . . .	11-40
Reading word device data from another station (READ) . . . . .	8-15
Reading word device data from another station (SREAD) . . . . .	8-20
REAL (data type) . . . . .	3-17
REAL_TO_M_REAL . . . . .	3-18
REAL_TO_M_REAL_E . . . . .	3-18
Receiving sent data from other stations . . . . .	8-47
RECV . . . . .	8-45
REQ . . . . .	8-50
Request data	
during RUN/STOP operation at a remote station . . . . .	8-55
from other stations . . . . .	8-56
Request/response data during write/read operation of clock data . . . . .	8-53
Resetting	
annunciators and error displays . . . . .	7-214
the watchdog timer . . . . .	7-467
Resetting outputs	
in subroutine programs . . . . .	7-138
in subroutine programs in program files . . . . .	7-145

RET	7-135
Return from an interrupt program	6-164
RFRP	8-89
RFS, RFSP	6-166
RIFR (A series)	11-168
RIFR (QnA series, System Q)	11-172
RIGHT, RIGHTP, LEFT, LEFTP	7-322
RIRCV (A series)	11-136
RIRCV (QnA series, System Q)	11-142
RIRD (A series)	11-108
RIRD (QnA series, System Q)	11-114
RISEND (A series)	11-148
RISEND (QnA series, System Q)	11-154
RITO (A series)	11-160
RITO (QnA series, System Q)	11-164
RIWT (A series)	11-122
RIWT (QnA series, System Q)	11-128
RLPA (A-Serie)	11-83
RLPASET, RLPASET_P	11-89
RND, RNDP, SRND, SRNDP	7-377
ROL, ROLP, RCL, RCLP	7-46
ROR, RORP, RCR, RCRP	7-43
Rotation instructions (overview)	2-35
ROTC	6-191
Routing information (overview)	2-61
RRPA (A series)	11-101
RTOP	8-95
RTREAD	8-102
RTWRITE	8-104

## S

Scan information	
special registers	A-93
special relays	A-51
Search data	7-79
Search for character strings	7-333
Searching	
maximum values in 16-/32-bit data	7-112
minimum values in 16-/32-bit data	7-115
SECOND, SECONDP, HOUR, HOURP	7-443
SEG	6-168
SEG, SEGP	7-91
SEND	8-37
Sending data to other stations	8-40
Sequence instructions	5-1
SER, SERP, DSER, DSERP	7-78
Setting	
comment files	7-422
file register blocks	7-416
file register files	7-419
Setting a program	
into the low-speed execution mode	7-464
into the scan execution mode	7-462
into the stand-by mode	7-458
into the stand-by mode including reset of the outputs	7-460

Setting and resetting	
sampling trace	7-242
single bits	7-67
status latch	7-240
the carry flag	7-469
Setting, resetting, and executing	
program trace	7-244
SFR, SFRP, SFL, SFLP	7-56
Shift instructions (overview)	2-8, 2-36
Shifting	
a 16-bit data word by n bits	7-57
n bit devices by 1 bit	7-60
n word devices by 1 address	7-63
Sign reversal	
for BIN data (complement of 2)	6-107
for floating point data	6-110
Sine calculation	
from BCD data	7-384
from floating point values	7-345
SIN, SINP	7-344
SLT, SLTR	7-239
Software	1-2
Sorting 16-/32-bit data	7-118
SORT, SORTP, DSORT, DSORTP	7-117
Source of data (s)	3-1
SPD	6-197
Special function instructions (overview)	2-51
Special function timer	6-188
Special functions	7-342
SPREF	10-11
SQR, SQRP	7-368
Square root calculation	
from 4-digit or 8-digit BCD data	7-380
of floating point values	7-369
SREAD	8-17
STC, CLC	7-468
STMODE	10-2
STMR, STMRH	6-187
Storage area	
Microcomputer program	12-1
Storage capacities and memory areas	12-1
Storing and moving parts of character strings	7-327
STRA, STRAR	7-241
Structured program instructions	7-125
Structured program instructions (overview)	2-41
STR, STRP, DSTR, DSTRP	7-284
Subtracting clock data	7-439
SUB, SUBP	7-156
SUM, SUMP, DSUM, DSUMP	7-84
SWAP, SWAPP	6-144
Switching between MAIN and SUB program	7-148
SWRITE	8-30
System clocks/counters	
special registers	A-92
special relays	A-50
System information	
special registers	A-82
special relays	A-46

System Q	
Multi processor type CPUs	1-2
Single processor type CPUs	1-2
System Q CPUs	1-2
S.TO, SP.TO	9-46

**T**

Table of	
Available devices	A-32
Diagnostic special relays (SM)	
(Q series and System Q)	A-43
Error codes	
A series (except AnA and AnAS)	13-39
AnA and AnAS CPUs	13-43
QnA series and System Q	13-12
System Q (Q00J, Q00, Q01 CPU)	13-2
Link registers (A series only)	A-120
Link relays (A series only)	A-70
Processing times	A-2
Special registers (D) (A series only)	A-110
Special registers (Q series and System Q)	A-73
Special relays and diagnostic	
relays (conversion)	A-58
Special relays (M) (A series only)	A-64
Tangent calculation	
from BCD data	7-390
from floating point values	7-351
TAN, TANP	7-350
Terminating a FOR/NEXT loop	7-130
Test of single bits in 16- / 32-bit data words	7-70
TEST, TESTP, DTEST, DTESTP	7-69
Timer functions	A-35
TO, DTO	7-190
TRACE, TRACER	9-7
Transfer instructions for System Q	9-41
TRUCK	10-6
TTMR	6-185

**U**

UDCNT1	6-179
UDCNT2	6-182
UINI	11-72
UNIRD, UNIRDP	9-2
Uniting 16-bit data	7-99
UNI, UNIP	7-98
Unloading and loading of a program	9-39
Upper and lower byte exchanges	6-145
User frames	
deletion	11-13
reading of	11-8
registration	11-13
transmission	11-20

**V**

VAL, VALP, DVAL, DVALP	7-292
Variables	4-5
Variables (use in an instruction)	3-1
Verification	
of device data	3-33
of the device range	3-31

**W**

WAND, WANDP, DAND, DANDP	7-4
WDT, WDTP	7-466
Word devices	
specification of bits	3-11
usage in an instruction	3-13
WORD, WORDP	6-98
WOR, WORP, DOR, DORP	7-14
WRITE	8-23
Writing	
routing information	8-105
Writing clock data	7-430
Writing data	
to a data table	7-169
to a local station	8-86
to a remote station	8-97
to other stations	8-78
to special function modules in remote	
I/O stations	8-68
Writing od data	
to intelligent device station (CC-Link)	11-156
Writing of data	
to a PROFIBUS interface module	11-31
to another Station (CC-Link)	11-125
to automatic updating buffer (A series)	11-160
to automatic updating buffer (Q series)	11-164
to fixed buffer (ETHERNET)	11-44
to intell. device station (CC-Link)	11-150
to station on CC-Link	11-132
Writing to and reading from files (System Q)	9-9
Writing to the buffer memory of a	
special function module	7-191
Writing word device data to another station	
(SWRITE)	8-34
Writing word device data to another station	
(WRITE)	8-27
WSUM, WSUMP	7-121
WTOB, WTOBP, BTOW, BTOWP	7-106
WXNR, WXNRP, DXNR, DXNRP	7-32
WXOR, WXORP, DXOR, DXORP	7-23

**X**

XCH, XCHP, DXCH, DXCHP	6-138
x, xP, /, /P	6-36

**Z**

ZCOM .....	8-7
ZNFR .....	8-60
ZNRD .....	8-73
ZNTO .....	8-66
ZNWR .....	8-77
ZONE, ZONEP, DZONE, DZONEP .....	7-410
ZPUSH, ZPUSHP, ZPOP, ZPOPP .....	7-488
ZRRDB, ZRRDBP .....	7-473
ZRWRB, ZRWRBP .....	7-477

**Symbols**

\$ =, \$ , \$ >, \$ = .....	6-15
\$MOV, \$MOVP .....	6-124
\$+, \$+P .....	6-72
+, +P, -, -P .....	6-28
=, , >, = .....	6-5

**Numbers**

1-phase count-up/-down counter .....	6-180
2-phase count-up/-down counter .....	6-183
7-segment data. ....	7-93
7-Segment decoding. ....	7-92





