

Modicon Ladder Logic Block Library User Guide Volume 3

840USE10100

Version 5.0

043505766 79



Telemecanique

Document Set

At a Glance

This manual consists of four volumes.

Volume 1

- General Information and Instruction Descriptions (A - D)

Volume 2

- Instruction Descriptions (E)

Volume 3

- Instruction Descriptions (F - N)

Volume 4

- General Information and Instruction Descriptions (O - X) and Appendix

Table of Contents



	Safety Information	xxxi
	About the Book	xxxiii
Part I	General Information	1
	Introduction	1
Chapter 1	Ladder Logic Overview	3
	At a Glance	3
	Segments and Networks in Ladder Logic	4
	How a PLC Solves Ladder Logic	7
	Ladder Logic Elements and Instructions	8
Chapter 2	Memory Allocation in a PLC	15
	At a Glance	15
	User Memory	16
	State RAM Values	18
	State RAM Structure	20
	The Configuration Table	22
	The I/O Map Table	27
Chapter 3	Ladder Logic Opcodes	29
	At a Glance	29
	Translating Ladder Logic Elements in the System Memory Database	30
	Translating DX Instructions in the System Memory Database	33
	Opcode Defaults for Loadables	37
Chapter 4	Instructions	39
	Parameter Assignment of Instructions	39
Chapter 5	Instruction Groups	41
	At a Glance	41
	Instruction Groups	42
	ASCII Functions	43
	Counters and Timers Instructions	44
	Fast I/O Instructions	45

	Loadable DX	46
	Math Instructions	47
	Matrix Instructions	49
	Miscellaneous	50
	Move Instructions	51
	Skips/Specials	52
	Special Instructions	53
	Coils, Contacts and Interconnects	54
Chapter 6	Equation Networks	55
	At a Glance	55
	Equation Network Structure	56
	Mathematical Equations in Equation Networks	59
	Mathematical Operations in Equation Networks	64
	Mathematical Functions in Equation Networks	69
	Data Conversions in an Equation Network	72
	Roundoff Differences in PLCs without a Math Coprocessor	74
	Benchmark Performance	75
Chapter 7	Closed Loop Control / Analog Values	77
	At a Glance	77
	Closed Loop Control / Analog Values	78
	PCFL Subfunctions	79
	A PID Example	83
	PID2 Level Control Example	87
Chapter 8	Formatting Messages for ASCII READ/WRITE Operations	91
	At a Glance	91
	Formatting Messages for ASCII READ/WRITE Operations	92
	Format Specifiers	93
	Special Set-up Considerations for Control/Monitor Signals Format	96
Chapter 9	Coils, Contacts and Interconnects	99
	At a Glance	99
	Coils	100
	Contacts	102
	Interconnects (Shorts)	104
Chapter 10	Interrupt Handling	105
	Interrupt Handling	105
Chapter 11	Subroutine Handling	107
	Subroutine Handling	107
Chapter 12	Installation of DX Loadables	109
	Installation of DX Loadables	109

Part II	Instruction Descriptions (A to D)	111
	At a Glance	111
Chapter 13	1X3X - Input Simulation	113
	At A Glance	113
	Short Description: 1X3X - Input Simulation	114
	Representation: 1X3X - Input Simulation	115
Chapter 14	AD16: Ad 16 Bit	117
	At a Glance	117
	Short Description	118
	Representation: AD16 - 16-bit Addition	119
Chapter 15	ADD: Addition	121
	At a Glance	121
	Short Description	122
	Representation: ADD - Single Precision Add	123
Chapter 16	AND: Logical And	125
	At a Glance	125
	Short Description	126
	Representation: AND - Logical And	127
	Parameter Description	129
Chapter 17	BCD: Binary to Binary Code	131
	At a Glance	131
	Short Description	132
	Representation: BCD - Binary Coded Decimal Conversion	133
Chapter 18	BLKM: Block Move	135
	At a Glance	135
	Short Description	136
	Representation: BLKM - Block Move	137
Chapter 19	BLKT: Block to Table	139
	At a Glance	139
	Short Description	140
	Representation: BLKT - Block-to-Table Move	141
	Parameter Description	142
Chapter 20	BMDI: Block Move with Interrupts Disabled	143
	At a Glance	143
	Short Description: BMDI - Block Move Interrupts Disabled	144
	Representation: BMDI - Block Move Interrupts Disabled	145
Chapter 21	BROT: Bit Rotate	147
	At a Glance	147

	Short Description	148
	Representation: BROT - Bit Rotate	149
	Parameter Description	150
Chapter 22	CALL: Activate Immediate or Deferred DX Function	151
	AT A GLANCE	151
	Short Description: CALL - Activate Immediate or Deferred DX Function.	152
	Representation: CALL - Activate Immediate DX Function.	153
	Representation: CALL - Activate Deferred DX Function	156
Chapter 23	CANT - Interpret Coils, Contacts, Timers, Counters, and the SUB Block.	159
	At A Glance	159
	Short Description: CANT - Interpret Coils, Contacts, Timers, Counters, and the SUB Block.	160
	Representation: CANT - Interpret Coils, Contacts, Timers, Counters, and the SUB Block.	161
	Parameter Description: CANT - Interpret Coils, Contacts, Timers, Counters, and the SUB Block.	162
Chapter 24	CHS: Configure Hot Standby	165
	At a Glance	165
	Short Description	166
	Representation: CHS - Configure Hot Standby	167
	Detailed Description.	169
Chapter 25	CKSM: Check Sum.	173
	At a Glance	173
	Short Description	174
	Representation: CKSM - Checksum	175
	Parameter Description	177
Chapter 26	CMPR: Compare Register	179
	At a Glance	179
	Short Description	180
	Representation: CMPR - Logical Compare	181
	Parameter Description	182
Chapter 27	Coils	183
	At A Glance	183
	Short Description: Coils	184
	General Usage Guidelines: Coils.	185
Chapter 28	COMM - ASCII Communications Function	187
	At A Glance	187
	Short Description: COMM - ASCII Communications Block	188
	Representation: COMM - ASCII Communications Function	189

Chapter 29	COMP: Complement a Matrix	191
	At a Glance	191
	Short Description	192
	Representation: COMP - Logical Compliment	193
	Parameter Description	195
Chapter 30	Contacts	197
	At A Glance	197
	Short Description: Contacts	198
	Representation: Contacts	199
Chapter 31	CONV - Convert Data	201
	At A Glance	201
	Short Description: CONV - Convert Data	202
	Representation: CONV - Convert Data	203
Chapter 32	CTIF - Counter, Timer, and Interrupt Function.	205
	At A Glance	205
	Short Description: CTIF - Counter, Timer, and Interrupt Function	206
	Representation: CTIF - Counter, Timer, Interrupt Function.	207
	Parameter Description: CTIF - Register Usage Table (Top Node)	208
Chapter 33	DCTR: Down Counter.	215
	At a Glance	215
	Short Description	216
	Representation: DCTR - Down Counter	217
Chapter 34	DIOH: Distributed I/O Health	219
	At a Glance	219
	Short Description	220
	Representation: DIOH - Distributed I/O Health	221
	Parameter Description	223
Chapter 35	DISA - Disabled Discrete Monitor.	225
	At A Glance	225
	Short Description: DISA - Disabled Discrete Monitor	226
	Representation: DISA - Disabled Discrete Monitor	227
Chapter 36	DIV: Divide.	229
	At a Glance	229
	Short Description	230
	Representation: DIV - Single Precision Division	231
	Example	233
Chapter 37	DLOG: Data Logging for PCMCIA Read/Write Support.	235
	At a Glance	235
	Short Description	236

	Representation: DLOG	237
	Parameter Description	238
	Run Time Error Handling	240
Chapter 38	DMTH - Double Precision Math	241
	At a Glance	241
	Short Description: DMTH - Double Precision Math - Addition, Subtraction, Multiplication, and Division.	242
	Representation: DMTH - Double Precision Math - Addition, Subtraction, Multiplication, and Division.	243
Chapter 39	DRUM: DRUM Sequencer	251
	At a Glance	251
	Short Description	252
	Representation: DRUM	253
	Parameter Description	254
Chapter 40	DV16: Divide 16 Bit	257
	At a Glance	257
	Short Description	258
	Representation: DV16 - 16-bit Division	259
	Example	260
Part III	Instruction Descriptions (E)	261
	At a Glance	261
Chapter 41	EARS - Event/Alarm Recording System	263
	At A Glance	263
	Short Description: EARS - Event/Alarm Recording System	264
	Representation: EARS - Event/Alarm Recording System	265
	Parameter Description: EARS - Event/Alarm Recording System	267
Chapter 42	EMTH: Extended Math	271
	At a Glance	271
	Short Description	272
	Representation: EMTH - Extended Math Functions	273
	Parameter Description	274
	Floating Point EMTH Functions	276
Chapter 43	EMTH-ADDDP: Double Precision Addition	277
	At a Glance	277
	Short Description	278
	Representation: EMTH - ADDDP - Double Precision Math - Addition	279
	Parameter Description	281
Chapter 44	EMTH-ADDFP: Floating Point Addition	283
	At a Glance	283

	Short Description	284
	Representation: EMTH - ADDFP - Floating Point Math - Addition	285
	Parameter Description	286
Chapter 45	EMTH-ADDIF: Integer + Floating Point Addition	287
	At a Glance	287
	Short Description	288
	Representation: EMTH - ADDIF - Integer + Floating Point Addition	289
	Parameter Description	290
Chapter 46	EMTH-ANLOG: Base 10 Antilogarithm	291
	At a Glance	291
	Short Description	292
	Representation: EMTH - ANLOG - integer Base 10 Antilogarithm	293
	Parameter Description	295
Chapter 47	EMTH-ARCOS: Floating Point Arc Cosine of an Angle (in Radians)	297
	At a Glance	297
	Short Description	298
	Representation: EMTH - ARCOS - Floating Point Math - Arc Cosine of an Angle (in Radians)	299
	Parameter Description	301
Chapter 48	EMTH-ARSIN: Floating Point Arcsine of an Angle (in Radians)	303
	At a Glance	303
	Short Description	304
	Representation: EMTH - ARSIN - Arcsine of an Angle (in Radians)	305
	Parameter Description	306
Chapter 49	EMTH-ARTAN: Floating Point Arc Tangent of an Angle (in Radians)	307
	At a Glance	307
	Short Description	308
	Representation: Floating Point Math - Arc Tangent of an Angle (in Radians)	309
	Parameter Description	311
Chapter 50	EMTH-CHSIN: Changing the Sign of a Floating Point Number	313
	At a Glance	313
	Short Description	314
	Representation: EMTH - CHSIN - Change the Sign of a Floating Point Number	315
	Parameter Description	317

Chapter 51	EMTH-CMPFP: Floating Point Comparison	319
	At a Glance	319
	Short Description	320
	Representation: EMTH - CMFPF - Floating Point Math Comparison	321
	Parameter Description	323
Chapter 52	EMTH-CMPIF: Integer-Floating Point Comparison	325
	At a Glance	325
	Short Description	326
	Representation: EMTH - CMPIF - Floating Point Math - Integer/Floating Point Comparison	327
	Parameter Description	329
Chapter 53	EMTH-CNVD R: Floating Point Conversion of Degrees to Radians	331
	At a Glance	331
	Short Description	332
	Representation: EMTH - CNVDR - Conversion of Degrees to Radians	333
	Parameter Description	335
Chapter 54	EMTH-CNVFI: Floating Point to Integer Conversion	337
	At a Glance	337
	Short Description	338
	Representation: EMTH - CNVFI - Floating Point to Integer Conversion	339
	Parameter Description	341
	Runtime Error Handling	342
Chapter 55	EMTH-CNVIF: Integer-to-Floating Point Conversion	343
	At a Glance	343
	Short Description	344
	Representation: EMTH - CNVIF - Integer to Floating Point Conversion	345
	Parameter Description	347
	Runtime Error Handling	348
Chapter 56	EMTH-CNVRD: Floating Point Conversion of Radians to Degrees	349
	At a Glance	349
	Short Description	350
	Representation: EMTH - CNVRD - Conversion of Radians to Degrees	351
	Parameter Description	353
Chapter 57	EMTH-COS: Floating Point Cosine of an Angle (in Radians)	355
	At a Glance	355
	Short Description	356
	Representation: EMTH - COS - Cosine of an Angle (in Radians)	357
	Parameter Description	358

Chapter 58	EMTH-DIVDP: Double Precision Division	359
	At a Glance	359
	Short Description	360
	Representation: EMTH - DIVDP - Double Precision Math - Division	361
	Parameter Description	363
	Runtime Error Handling	364
Chapter 59	EMTH-DIVFI: Floating Point Divided by Integer	365
	At a Glance	365
	Short Description	366
	Representation: EMTH - DIVFI - Floating Point Divided by Integer	367
	Parameter Description	368
Chapter 60	EMTH-DIVFP: Floating Point Division	369
	At a Glance	369
	Short Description	370
	Representation: EMTH - DIVFP - Floating Point Division	371
	Parameter Description	372
Chapter 61	EMTH-DIVIF: Integer Divided by Floating Point	373
	At a Glance	373
	Short Description	374
	Representation: EMTH - DIVIF - Integer Divided by Floating Point	375
	Parameter Description	376
Chapter 62	EMTH-ERLOG: Floating Point Error Report Log	377
	At a Glance	377
	Short Description	378
	Representation: EMTH - ERLOG - Floating Point Math - Error Report Log	379
	Parameter Description	381
Chapter 63	EMTH-EXP: Floating Point Exponential Function	383
	At a Glance	383
	Short Description	384
	Representation: EMTH - EXP - Floating Point Math - Exponential Function	385
	Parameter Description	387
Chapter 64	EMTH-LNFP: Floating Point Natural Logarithm	389
	At a Glance	389
	Short Description	390
	Representation: EMTH - LNFP - Natural Logarithm	391
	Parameter Description	393
Chapter 65	EMTH-LOG: Base 10 Logarithm	395
	At a Glance	395
	Short Description	396
	Representation: EMTH - LOG - Integer Math - Base 10 Logarithm	397

	Parameter Description	399
Chapter 66	EMTH-LOGFP: Floating Point Common Logarithm	401
	At a Glance	401
	Short Description	402
	Representation: EMTH - LOGFP - Common Logarithm	403
	Parameter Description	405
Chapter 67	EMTH-MULDP: Double Precision Multiplication	407
	At a Glance	407
	Short Description	408
	Representation: EMTH - MULDP - Double Precision Math - Multiplication	409
	Parameter Description	411
Chapter 68	EMTH-MULFP: Floating Point Multiplication	413
	At a Glance	413
	Short Description	414
	Representation: EMTH - MULFP - Floating Point - Multiplication	415
	Parameter Description	416
Chapter 69	EMTH-MULIF: Integer x Floating Point Multiplication	417
	At a Glance	417
	Short Description	418
	Representation: EMTH - MULIF - Integer Multiplied by Floating Point	419
	Parameter Description	421
Chapter 70	EMTH-PI: Load the Floating Point Value of "Pi"	423
	At a Glance	423
	Short Description	424
	Representation: EMTH - PI - Floating Point Math - Load the Floating Point Value of PI	425
	Parameter Description	426
Chapter 71	EMTH-POW: Raising a Floating Point Number to an Integer Power	427
	At a Glance	427
	Short Description	428
	Representation: EMTH - POW - Raising a Floating Point Number to an Integer Power.	429
	Parameter Description	430
Chapter 72	EMTH-SINE: Floating Point Sine of an Angle (in Radians)	431
	At a Glance	431
	Short Description	432
	Representation: EMTH - SINE - Floating Point Math - Sine of an Angle (in Radians)	433
	Parameter Description	435

Chapter 73	EMTH-SQRFP: Floating Point Square Root	437
	At a Glance	437
	Short Description	438
	Representation: EMTH - SQRFP - Square Root	439
	Parameter Description	440
Chapter 74	EMTH-SQRT: Floating Point Square Root	441
	At a Glance	441
	Short Description	442
	Representation: EMTH - SQRT - Square Root	443
	Parameter Description	445
Chapter 75	EMTH-SQRTP: Process Square Root	447
	At a Glance	447
	Short Description	448
	Representation: EMTH - SQRTP - Double Precision Math - Process Square Root	449
	Parameter Description	451
	Example	452
Chapter 76	EMTH-SUBDP: Double Precision Subtraction	453
	At a Glance	453
	Short Description	454
	Representation: EMTH - SUBDP - Double Precision Math - Subtraction	455
	Parameter Description	457
Chapter 77	EMTH-SUBFI: Floating Point - Integer Subtraction	459
	At a Glance	459
	Short Description	460
	Representation: EMTH - SUBFI - Floating Point minus Integer	461
	Parameter Description	462
Chapter 78	EMTH-SUBFP: Floating Point Subtraction	463
	At a Glance	463
	Short Description	464
	Representation: EMTH - SUBFP - Floating Point - Subtraction	465
	Parameter Description	466
Chapter 79	EMTH-SUBIF: Integer - Floating Point Subtraction	467
	At a Glance	467
	Short Description	468
	Representation: EMTH - SUBIF - Integer minus Floating Point	469
	Parameter Description	470
Chapter 80	EMTH-TAN: Floating Point Tangent of an Angle (in Radians)	471
	At a Glance	471

	Short Description	472
	Representation: EMTH - TAN - Tangent of an Angle (in Radians)	473
	Parameter Description	474
Chapter 81	ESI: Support of the ESI Module	475
	At a Glance	475
	Short Description	476
	Representation.	477
	Parameter Description	478
	READ ASCII Message (Subfunction 1)	481
	WRITE ASCII Message (Subfunction 2)	485
	GET DATA (Subfunction 3)	486
	PUT DATA (Subfunction 4)	488
	ABORT (Middle Input ON)	492
	Run Time Errors.	493
Chapter 82	EUCA: Engineering Unit Conversion and Alarms	495
	At a Glance	495
	Short Description	496
	Representation: EUCA - Engineering Unit and Alarm	497
	Parameter Description	498
	Examples	500
Part IV	Instruction Descriptions (F to N)	507
	At a Glance	507
Chapter 83	FIN: First In	509
	At a Glance	509
	Short Description	510
	Representation: FIN - First in.	511
	Parameter Description	512
Chapter 84	FOUT: First Out	513
	At a Glance	513
	Short Description	514
	Representation: FOUT - First Out	515
	Parameter Description	517
Chapter 85	FTOI: Floating Point to Integer	519
	At a Glance	519
	Short Description	520
	Representation: FTOI - Floating Point to Integer Conversion	521
Chapter 86	GD92 - Gas Flow Function Block	523
	At A Glance	523
	Short Description: GD92 - Gas Flow Function Block	524
	Representation: GD92 - Gas Flow Function Block	525

	Parameter Description - Inputs: GD92 - Gas Flow Function Block	527
	Parameter Description - Outputs: GD92 - Gas Flow Function Block	533
	Parameter Description - Optional Outputs: GD92 - Gas Flow Function Block	534
Chapter 87	GFNX AGA#3 '85 and NX19 '68 Gas Flow Function Block	535
	At A Glance	535
	Short Description: GFNX - Gas Flow Function Block	536
	Representation: GFNX - Gas Flow Function Block	537
	Parameter Description - Inputs: GFNX - Gas Flow Function Block	539
	Parameter Description - Outputs: GFNX - Gas Flow Function Block	546
	Parameter Description - Optional Outputs: GFNX - Gas Flow Function Block	547
Chapter 88	GG92 AGA #3 1992 Gross Method Gas Flow Function Block	549
	At A Glance	549
	Short Description: GG92 - Gas Flow Function Block	550
	Representation: GG92 - Gas Flow Function Block	551
	Parameter Description - Inputs: GG92 - Gas Flow Function Block	553
	Parameter Description - Outputs: GG92 - Gas Flow Function Block	558
	Parameter Description - Optional Outputs: GG92 - Gas Flow Function Block	559
Chapter 89	GM92 AGA #3 and #8 1992 Detail Method Gas Flow Function Block	561
	At A Glance	561
	Short Description: GM92 - Gas Flow Function Block	562
	Representation: GM92 - Gas Flow Function Block	563
	Parameter Description - Inputs: GM92 - Gas Flow Function Block	565
	Parameter Description - Outputs: GM92 - Gas Flow Function Block	571
	Parameter Description - Optional Outputs: GM92 - Gas Flow Function Block	572
Chapter 90	G392 AGA #3 1992 Gas Flow Function Block	573
	At A Glance	573
	Short Description: G392 - Gas Flow Function Block	574
	Representation: G392 - Gas Flow Function Block	575
	Parameter Description - Inputs: G392 - Gas Flow Function Block	577
	Parameter Description - Outputs: G392 - Gas Flow Function Block	582
	Parameter Description - Optional Outputs: G392 - Gas Flow Function Block	583
Chapter 91	HLTH: History and Status Matrices.	585
	At a Glance	585
	Short Description.	586
	Representation: HLTH - System Health	587
	Parameter Description.	588
	Parameter Description Top Node (History Matrix)	589
	Parameter Description Middle Node (Status Matrix)	594
	Parameter Description Bottom Node (Length)	599

Chapter 92	HSBY - Hot Standby	601
	At A Glance	601
	Short Description: HSBY - Hot Standby	602
	Representation: HSBY - Hot Standby	603
	Parameter Description Top Node: HSBY - Hot Standby	605
	Parameter Description Middle Node: HSBY - Hot Standby	606
Chapter 93	IBKR: Indirect Block Read	607
	At a Glance	607
	Short Description	608
	Representation: IBKR - Indirect Block Read	609
Chapter 94	IBKW: Indirect Block Write	611
	At a Glance	611
	Short Description	612
	Representation: IBKW - Indirect Block Write	613
Chapter 95	ICMP: Input Compare	615
	At a Glance	615
	Short Description	616
	Representation: ICMP - Input Compare	617
	Parameter Description	618
	Cascaded DRUM/ICMP Blocks	621
Chapter 96	ID: Interrupt Disable	623
	At a Glance	623
	Short Description: ID - Interrupt Disable	624
	Representation: ID - Interrupt Disable	625
	Parameter Description: ID - Interrupt Disable	626
Chapter 97	IE: Interrupt Enable	627
	At a Glance	627
	Short Description: IE - Interrupt Enable	628
	Representation: IE - Interrupt Enable	629
	Parameter Description: IE - Interrupt Enable	630
Chapter 98	IMIO: Immediate I/O	631
	At a Glance	631
	Short Description: IMIO - Immediate I/O	632
	Representation: IMIO - Immediate I/O	633
	Parameter Description: IMIO - Immediate I/O	634
	Run Time Error Handling: IMIO - Immediate I/O	636
Chapter 99	IMOD: Interrupt Module Instruction	637
	At a Glance	637
	Short Description: IMOD - Interrupt Module	638
	Representation: IMOD - Interrupt Module	639

	Parameter Description: IMOD - Interrupt Module	641
Chapter 100	ITMR: Interrupt Timer	647
	At a Glance	647
	Short Description: ITMR - Interval Timer Interrupt	648
	Representation: ITMR - Interval Timer Interrupt	649
	Parameter Description: ITMR - Interval Timer Interrupt	651
Chapter 101	ITOF: Integer to Floating Point	653
	At a Glance	653
	Short Description.	654
	Representation: ITOF - integer to Floating Point Conversion	655
Chapter 102	JSR: Jump to Subroutine.	657
	At a Glance	657
	Short Description.	658
	Representation: JSR - Jump to Subroutine.	659
Chapter 103	LAB: Label for a Subroutine	661
	At a Glance	661
	Short Description.	662
	Representation: LAB - Label.	663
	Parameter Description.	664
Chapter 104	LOAD: Load Flash	665
	At a Glance	665
	Short Description.	666
	Representation: LOAD - Load.	667
	Parameter Description.	668
Chapter 105	MAP 3: MAP Transaction	669
	At a Glance	669
	Short Description.	670
	Representation: MAP 3 - Map Transaction	671
	Parameter Description.	672
Chapter 106	MATH - Integer Operations	677
	At A Glance	677
	Short Description: MATH - Integer Operations - Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10).	678
	Representation: MATH - Integer Operations - Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10).	679
Chapter 107	MBIT: Modify Bit	685
	At a Glance	685
	Short Description.	686
	Representation: MBIT - Logical Bit Modify	687

	Parameter Description	688
Chapter 108	MBUS: MBUS Transaction	689
	At a Glance	689
	Short Description	690
	Representation: MBUS - Modbus II Transfer	691
	Parameter Description	692
	The MBUS Get Statistics Function	694
Chapter 109	MRTM: Multi-Register Transfer Module	699
	At a Glance	699
	Short Description	700
	Representation: MRTM - Multi-Register Transfer Module	701
	Parameter Description	702
Chapter 110	MSPX (Seriplex)	705
	At A Glance	705
	Short Description: MSPX (Seriplex)	706
	Representation: MSPX (Seriplex)	707
Chapter 111	MSTR: Master	709
	At a Glance	709
	Short Description	711
	Representation: MSTR - Master Instruction	712
	Parameter Description	713
	Write MSTR Operation	717
	READ MSTR Operation	719
	Get Local Statistics MSTR Operation	721
	Clear Local Statistics MSTR Operation	723
	Write Global Data MSTR Operation	725
	Read Global Data MSTR Operation	726
	Get Remote Statistics MSTR Operation	727
	Clear Remote Statistics MSTR Operation	729
	Peer Cop Health MSTR Operation	731
	Reset Option Module MSTR Operation	734
	Read CTE (Config Extension Table) MSTR Operation	736
	Write CTE (Config Extension Table) MSTR Operation	738
	Modbus Plus Network Statistics	740
	TCP/IP Ethernet Statistics	745
	Run Time Errors	746
	Modbus Plus and SY/MAX Ethernet Error Codes	747
	SY/MAX-specific Error Codes	749
	TCP/IP Ethernet Error Codes	751
	CTE Error Codes for SY/MAX and TCP/IP Ethernet	754
Chapter 112	MU16: Multiply 16 Bit	755
	At a Glance	755

	Short Description	756
	Representation: MU16 - 16-Bit Multiplication	757
Chapter 113	MUL: Multiply	759
	At a Glance	759
	Short Description	760
	Representation: MUL - Single Precision Multiplication	761
	Example	762
Chapter 114	NBIT: Bit Control	763
	At a Glance	763
	Short Description	764
	Representation: NBIT - Normal Bit	765
Chapter 115	NCBT: Normally Closed Bit	767
	At a Glance	767
	Short Description	768
	Representation: NCBT - Bit Normally Closed	769
Chapter 116	NOBT: Normally Open Bit	771
	At a Glance	771
	Short Description	772
	Representation: NOBT - Bit Normally Open	773
Chapter 117	NOL: Network Option Module for Lonworks	775
	At a Glance	775
	Short Description	776
	Representation: NOL - Network Option Module for Lonworks	777
	Detailed Description	778
Part V	Instruction Descriptions (O to Q)	781
	At a Glance	781
Chapter 118	OR: Logical OR	783
	At a Glance	783
	Short Description	784
	Representation: OR - Logical Or	785
	Parameter Description	787
Chapter 119	PCFL: Process Control Function Library	789
	At a Glance	789
	Short Description	790
	Representation: PCFL - Process Control Function Library	791
	Parameter Description	792
Chapter 120	PCFL-AIN: Analog Input	797
	At a Glance	797

	Short Description	798
	Representation: PCFL - AIN - Convert Inputs to Scaled Engineering Units . . .	799
	Parameter Description	800
Chapter 121	PCFL-ALARM: Central Alarm Handler	803
	At a Glance	803
	Short Description	804
	Representation: PCFL - ALRM - Central Alarm Handler for a P(v) Input.	805
	Parameter Description	806
Chapter 122	PCFL-AOUT: Analog Output	809
	At a Glance	809
	Short Description	810
	Representation: PCFL - AOUT - Convert Outputs to Values in the 0 through 4095 Range	811
	Parameter Description	812
Chapter 123	PCFL-AVER: Average Weighted Inputs Calculate	813
	At a Glance	813
	Short Description	814
	Representation: PCFL - AVER - Average Weighted Inputs.	815
	Parameter Description	816
Chapter 124	PCFL-CALC: Calculated preset formula	819
	At a Glance	819
	Short Description	820
	Representation: PCFL - CALC - Calculate Present Formula.	821
	Parameter Description	822
Chapter 125	PCFL-DELAY: Time Delay Queue	825
	At a Glance	825
	Short Description	826
	Representation: PCFL - DELY - Time Delay Queue	827
	Parameter Description	828
Chapter 126	PCFL-EQN: Formatted Equation Calculator	829
	At a Glance	829
	Short Description	830
	Representation: PCFL - EQN - Formatted Equation Calculator	831
	Parameter Description	832
Chapter 127	PCFL-INTEG: Integrate Input at Specified Interval	835
	At a Glance	835
	Short Description	836
	Representation: PCFL - INTG - Integrate Input at Specified Interval.	837
	Parameter Description	838

Chapter 128	PCFL-KPID: Comprehensive ISA Non Interacting PID	839
	At a Glance	839
	Short Description	840
	Representation: PCFL - KPID - Comprehensive ISA Non-Interacting	
	Proportional-Integral-Derivative	841
	Parameter Description	842
Chapter 129	PCFL-LIMIT: Limiter for the Pv	845
	At a Glance	845
	Short Description	846
	Representation: PCFL - LIMIT - Limiter for the P(v)	847
	Parameter Description	848
Chapter 130	PCFL-LIMV: Velocity Limiter for Changes in the Pv	849
	At a Glance	849
	Short Description	850
	Representation: PCFL - LIMV - Velocity Limiter for Changes in the P(v)	851
	Parameter Description	852
Chapter 131	PCFL-LKUP: Look-up Table	853
	At a Glance	853
	Short Description	854
	Representation: PCFL - LKUP - Look-up Table	855
	Parameter Description	856
Chapter 132	PCFL-LLAG: First-order Lead/Lag Filter	859
	At a Glance	859
	Short Description	860
	Representation: PCFL - LLAG - First-Order Lead/Lag Filter	861
	Parameter Description	862
Chapter 133	PCFL-MODE: Put Input in Auto or Manual Mode	863
	At a Glance	863
	Short Description	864
	Representation: PCFL - MODE - Put Input in Auto or Manual Mode	865
	Parameter Description	866
Chapter 134	PCFL-ONOFF: ON/OFF Values for Deadband	867
	At a Glance	867
	Short Description	868
	Representation: PCFL - ONOFF - Specifies ON/OFF Values for Deadband	869
	Parameter Description	870
Chapter 135	PCFL-PI: ISA Non Interacting PI	873
	At a Glance	873
	Short Description	874
	Representation: PCFL - PI	875

	Parameter Description	876
Chapter 136	PCFL-PID: PID Algorithms	879
	At a Glance	879
	Short Description	880
	Representation: PCFL - PID - Algorithms	881
	Parameter Description	882
Chapter 137	PCFL-RAMP: Ramp to Set Point at a Constant Rate	885
	At a Glance	885
	Short Description	886
	Representation: PCFL - RAMP - Ramp to Set Point at Constant Rate	887
	Parameter Description	888
Chapter 138	PCFL-RATE: Derivative Rate Calculation over a Specified Timeme	891
	At a Glance	891
	Short Description	892
	Representation: PCFL - RATE - Derivative Rate Calculation Over a Specified Time	893
	Parameter Description	894
Chapter 139	PCFL-RATIO: Four Station Ratio Controller	895
	At a Glance	895
	Short Description	896
	Representation: PCFL - RATIO - Four-Station Ratio Controller	897
	Parameter Description	898
Chapter 140	PCFL-RMPLN: Logarithmic Ramp to Set Point.	901
	At a Glance	901
	Short Description	902
	Representation: PCFL - RMPLN - Logarithmic Ramp to Set Point	903
	Parameter Description	904
Chapter 141	PCFL-SEL: Input Selection	905
	At a Glance	905
	Short Description	906
	Representation: PCFL - SEL - High/Low/Average Input Selection	907
	Parameter Description	908
Chapter 142	PCFL-TOTAL: Totalizer for Metering Flow	911
	At a Glance	911
	Short Description	912
	Representation: PCFL - TOTAL - Totalizer for Metering Flow	913
	Parameter Description	914

Chapter 143	PEER: PEER Transaction	917
	At a Glance	917
	Short Description	918
	Representation: PEER - Modbus II Identical Transfer	919
	Parameter Description	920
Chapter 144	PID2: Proportional Integral Derivative	921
	At a Glance	921
	Short Description	922
	Representation: PID2 - Proportional/Integral/Derivative	923
	Detailed Description	924
	Parameter Description	927
	Run Time Errors	932
Part VI	Instruction Descriptions (R to Z)	935
	At a Glance	935
Chapter 145	R --> T: Register to Table	937
	At a Glance	937
	Short Description	938
	Representation: R → T - Register to Table Move	939
	Parameter Description	940
Chapter 146	RBIT: Reset Bit	941
	At a Glance	941
	Short Description	942
	Representation: RBIT - Reset Bit	943
Chapter 147	READ: Read	945
	At a Glance	945
	Short Description	946
	Representation: READ - Read ASCII Port	947
	Parameter Description	948
Chapter 148	RET: Return from a Subroutine	951
	At a Glance	951
	Short Description	952
	Representation: RET - Return to Scheduled Logic	953
Chapter 149	RTTI - Register to Input Table	955
	At A Glance	955
	Short Description: RTTI - Register to Input Table	956
	Representation: RTTI - Register to Input Table	957
Chapter 150	RTTO - Register to Output Table	959
	At A Glance	959
	Short Description: RTTO - Register to Output Table	960

	Representation: RTTO - Register to Output Table	961
Chapter 151	RTU - Remote Terminal Unit	963
	At A Glance	963
	Short Description: RTU - Remote Terminal Unit	964
	Representation: RTU - Remote Terminal Unit	965
Chapter 152	SAVE: Save Flash	969
	At a Glance	969
	Short Description	970
	Representation: SAVE - Save	971
	Parameter Description	972
Chapter 153	SBIT: Set Bit	973
	At a Glance	973
	Short Description	974
	Representation: SBIT - Set Bit	975
Chapter 154	SCIF: Sequential Control Interfaces.	977
	At a Glance	977
	Short Description	978
	Representation: SCIF - Sequential Control Interface	979
	Parameter Description	981
Chapter 155	SENS: Sense	983
	At a Glance	983
	Short Description	984
	Representation: SENS - Logical Bit-Sense	985
	Parameter Description	986
Chapter 156	Shorts	987
	At A Glance	987
	Short Description: Shorts	988
	Representation: Shorts	989
Chapter 157	SKP - Skipping Networks	991
	At A Glance	991
	Short Description: SKP - Skipping Networks	992
	Representation: SKP - Skipping Networks	993
Chapter 158	SRCH: Search.	995
	At a Glance	995
	Short Description	996
	Representation: SRCH - Search	997
	Parameter Description	999
Chapter 159	STAT: Status	1001
	At a Glance	1001

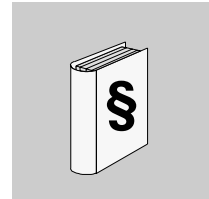
	Short Description	1002
	Representation: STAT - Status	1003
	Parameter Description	1004
	Description of the Status Table	1005
	Controller Status Words 1 - 11 for Quantum and Momentum	1009
	I/O Module Health Status Words 12 - 20 for Momentum	1014
	I/O Module Health Status Words 12 - 171 for Quantum	1016
	Communication Status Words 172 - 277 for Quantum	1018
	Controller Status Words 1 - 11 for TSX Compact and Atrium	1023
	I/O Module Health Status Words 12 - 15 for TSX Compact	1026
	Global Health and Communications Retry Status Words 182 ... 184 for TSX Compact	1027
Chapter 160	SU16: Subtract 16 Bit	1029
	At a Glance	1029
	Short Description	1030
	Representation: SU16 - 16-bit Subtraction	1031
Chapter 161	SUB: Subtraction	1033
	At a Glance	1033
	Short Description	1034
	Representation: SUB - Subtraction	1035
Chapter 162	SWAP - VME Bit Swap	1037
	At A Glance	1037
	Short Description: SWAP - VME Bit Swap	1038
	Representation: SWAP - VME Bit Swap	1039
Chapter 163	TTR - Table to Register	1041
	At A Glance	1041
	Short Description: TTR - Table to Register	1042
	Representation: TTR - Table to Register	1043
Chapter 164	T --> R Table to Register	1045
	At a Glance	1045
	Short Description	1046
	Representation: T → R - Table to Register Move	1047
	Parameter Description	1049
Chapter 165	T --> T: Table to Table	1051
	At a Glance	1051
	Short Description	1052
	Representation: T → T - Table to Table Move	1053
	Parameter Description	1055
Chapter 166	T.01 Timer: One Hundredth Second Timer	1057
	At a Glance	1057

	Short Description	1058
	Representation: T.01 - One Hundredth of a Second Timer	1059
Chapter 167	T0.1 Timer: One Tenth Second Timer	1061
	At a Glance	1061
	Short Description	1062
	Representation: T0.1 - One Tenth of a Second Timer	1063
Chapter 168	T1.0 Timer: One Second Timer	1065
	At a Glance	1065
	Short Description	1066
	Representation: T1.0 - One Second Timer	1067
Chapter 169	T1MS Timer: One Millisecond Timer	1069
	At a Glance	1069
	Short Description	1070
	Representation: T1MS - One Millisecond Timer	1071
	Example	1072
Chapter 170	TBLK: Table to Block.	1073
	At a Glance	1073
	Short Description	1074
	Representation: TBLK - Table-to-Block Move	1075
	Parameter Description	1077
Chapter 171	TEST: Test of 2 Values	1079
	At a Glance	1079
	Short Description	1080
	Representation: TEST - Test of 2 Values	1081
Chapter 172	UCTR: Up Counter	1083
	At a Glance	1083
	Short Description	1084
	Representation: UCTR - Up Counter	1085
Chapter 173	VMER - VME Read	1087
	At A Glance	1087
	Short Description: VMER - VME Read	1088
	Representation: VMER - VME Read	1089
	Parameter Description: VMER - VME Read	1090
Chapter 174	VMEW - VME Write.	1091
	At A Glance	1091
	Short Description: VMEW - VME Write	1092
	Representation: VMEW - VME Write	1093
	Parameter Description: VMEW - VME Write	1095

Chapter 175	WRIT: Write	1097
	At a Glance	1097
	Short Description	1098
	Representation: WRIT - Write ASCII Port	1099
	Parameter Description	1100
Chapter 176	XMIT - Transmit	1103
	At A Glance	1103
	General Description: XMIT - Transmit	1104
	XMIT Modbus Functions	1105
Chapter 177	XMIT Communication Block	1111
	At A Glance	1111
	Short Description: XMIT Communication Block	1112
	Representation: XMIT Communication Block	1113
	Parameter Description: Middle Node - Communication Control Table	1115
	Parameter Description: XMIT Communication Block	1119
	Parameter Description: XMIT Communications Block	1121
Chapter 178	XMIT Port Status Block	1123
	At A Glance	1123
	Short Description: XMIT Port Status Block	1124
	Representation: XMIT Port Status Block	1125
	Parameter Description: Middle Node - XMIT Conversion Block	1127
Chapter 179	XMIT Conversion Block	1131
	At A Glance	1131
	Short Description: XMIT Conversion Block	1132
	Representation: XMIT Conversion Block	1133
	Parameter Description: XMIT Conversion Block	1135
Chapter 180	XMRD: Extended Memory Read	1139
	At a Glance	1139
	Short Description	1140
	Representation: XMRD - Extended Memory Read	1141
	Parameter Description	1142
Chapter 181	XMWT: Extended Memory Write	1145
	At a Glance	1145
	Short Description	1146
	Representation: XMWT - Extended Memory Write	1147
	Parameter Description	1148
Chapter 182	XOR: Exclusive OR	1151
	At a Glance	1151
	Short Description	1152
	Representation: XOR - Boolean Exclusive Or	1153

Parameter Description	1155
Appendices	1157
Optimizing RIO Performance with the Segment Scheduler	1157
Appendix A Appendix A	1159
Optimizing RIO Performance with the Segment Scheduler	1159
Scan Time	1160
How to Measure Scan Time	1164
Maximizing Throughput	1165
Order of Solve	1167
Using Segment Scheduler to Improve Critical I/O Throughput	1168
Using Segment Scheduler to Improve System Performance	1169
Using Segment Scheduler to Improve Communication Port Servicing	1170
Sweep Functions	1171
Glossary	xxxv
Index	lvii

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.



DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death, serious injury, or equipment damage.



WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.



CAUTION

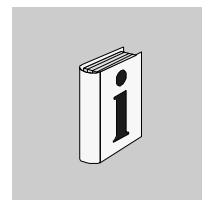
CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

PLEASE NOTE

Electrical equipment should be serviced only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material. This document is not intended as an instruction manual for untrained persons.

© 2004 Schneider Electric. All Rights Reserved.

About the Book



At a Glance

Document Scope This documentation will help you configure LL 984 instructions to any controller using ProWorx NxT, ProWorx 32 or Modbus Plus. Examples in this book are used with ProWorx 32. For LL 984 using Concept software, see Concept Block Library LL984 (840USE49600).

Validity Note The data and illustrations found in this book are not binding. We reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be construed as a commitment by Schneider Electric.

Related Documents

Title of Documentation	Reference Number
Concept Block Library LL 984	840USE49600

Product Related Warnings

Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When controllers are used for applications with technical safety requirements, please follow the relevant instructions.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this product related warning can result in injury or equipment damage.

User Comments

We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

Instruction Descriptions (F to N)



At a Glance

Introduction

In this part instruction descriptions are arranged alphabetically from F to N.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
83	FIN: First In	509
84	FOUT: First Out	513
85	FTOI: Floating Point to Integer	519
86	GD92 - Gas Flow Function Block	523
87	GFNX AGA#3 '85 and NX19 '68 Gas Flow Function Block	535
88	GG92 AGA #3 1992 Gross Method Gas Flow Function Block	549
89	GM92 AGA #3 and #8 1992 Detail Method Gas Flow Function Block	561
90	G392 AGA #3 1992 Gas Flow Function Block	573
91	HLTH: History and Status Matrices	585
92	HSBY - Hot Standby	601
93	IBKR: Indirect Block Read	607
94	IBKW: Indirect Block Write	611
95	ICMP: Input Compare	615
96	ID: Interrupt Disable	623
97	IE: Interrupt Enable	627
98	IMIO: Immediate I/O	631
99	IMOD: Interrupt Module Instruction	637
100	ITMR: Interrupt Timer	647
101	ITOF: Integer to Floating Point	653
102	JSR: Jump to Subroutine	657
103	LAB: Label for a Subroutine	661

Chapter	Chapter Name	Page
104	LOAD: Load Flash	665
105	MAP 3: MAP Transaction	669
106	MATH - Integer Operations	677
107	MBIT: Modify Bit	685
108	MBUS: MBUS Transaction	689
109	MRTM: Multi-Register Transfer Module	699
110	MSPX (Seriplex)	705
111	MSTR: Master	709
112	MU16: Multiply 16 Bit	755
113	MUL: Multiply	759
114	NBIT: Bit Control	763
115	NCBT: Normally Closed Bit	767
116	NOBT: Normally Open Bit	771
117	NOL: Network Option Module for Lonworks	775

FIN: First In

83

At a Glance

Introduction

This chapter describes the instruction FIN.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	510
Representation: FIN - First in	511
Parameter Description	512

Short Description

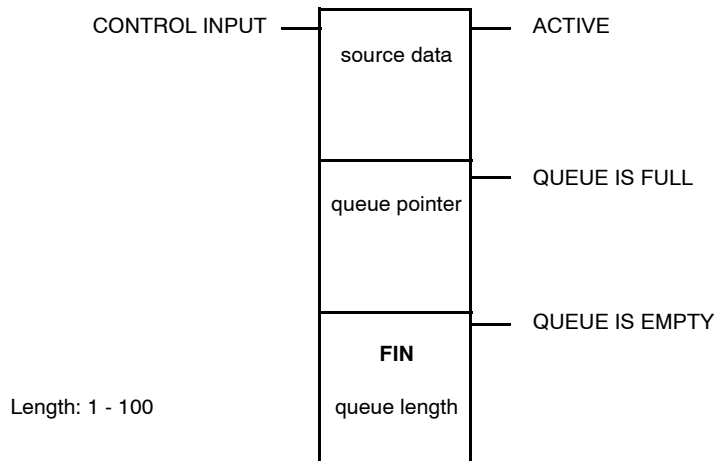
**Function
Description**

The FIN instruction is used to produce a first-in queue. A FOUT instruction needs to be used to clear the register at the bottom of the queue. An FIN instruction has one control input and can produce three possible outputs.

Representation: FIN - First in

Symbol

Representation of the instruction



Parameter Description

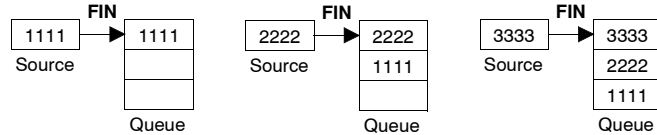
Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = copies source bit pattern into queue
source data (top node)	0x, 1x, 3x, 4x	ANY_BIT	Source data, will be copied to the top of the destination queue in the current logic scan
queue pointer (middle node)	4x	WORD	First of a queue of 4x registers, contains queue pointer; the next contiguous register is the first register in the queue
queue length (bottom node)		INT, UINT	Number of 4x registers in the destination queue. Range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = queue full, no more source data can be copied to the queue
Bottom output	0x	None	ON = queue empty (value in queue pointer register = 0)

Parameter Description

Mode of Functioning

The FIN instruction is used to produce a first-in queue. It copies the source data from the top node to the first register in a queue of holding registers. The source data is always copied to the register at the top of the queue. When a queue has been filled, no further source data can be copied to it.



Source Data (Top Node)

When using register types 0x or 1x:

- First 0x reference in a string of 16 contiguous coils or discrete outputs
- First 1x reference in a string of 16 discrete inputs

Queue Pointer (Middle Node)

The 4x register entered in the middle node is a queue pointer. The first register in the queue is the next contiguous 4x register following the pointer. For example, if the middle node displays a pointer reference of 400100, then the first register in the queue is 400101.

The value posted in the queue pointer equals the number of registers in the queue that are currently filled with source data. The value of the pointer cannot exceed the integer maximum queue length value specified in the bottom node.

If the value in the queue pointer equals the integer specified in the bottom node, the middle output passes power and no further source data can be written to the queue until an FOUT instruction clears the register at the bottom of the queue.

FOUT: First Out

84

At a Glance

Introduction

This chapter describes the instruction FOUT.

What's in this Chapter?


This chapter contains the following topics:

Topic	Page
Short Description	514
Representation: FOUT - First Out	515
Parameter Description	517

Short Description

**Function
Description**

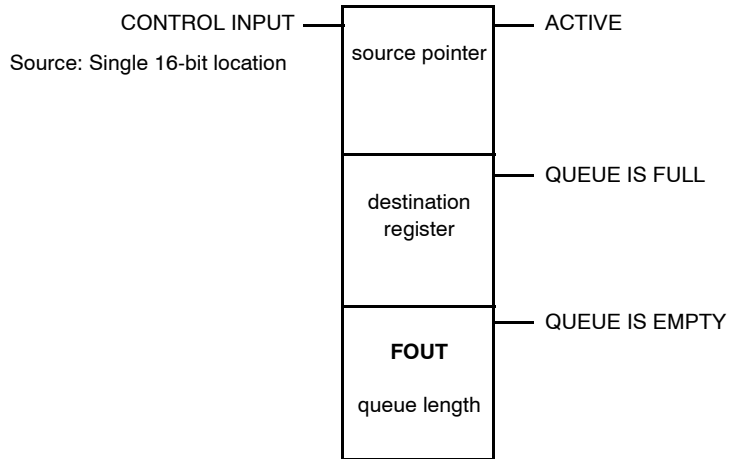
The FOUT instruction works together with the FIN instruction to produce a first in-first out (FIFO) queue. It moves the bit pattern of the holding register at the bottom of a full queue to a destination register or to word that stores 16 discrete outputs. An FOUT instruction has one control input and can produce three possible outputs.

	<p>DANGER</p>
	<p>Overriding any disabled coils</p> <p>FOUT will override any disabled coils within a destination register without enabling them. This can cause injury if a coil has been disabled for repair or maintenance because the coil's state can change as a result of the FOUT operation.</p> <p>Failure to follow this precaution will result in death, serious injury, or equipment damage.</p>

Representation: FOUT - First Out

Symbol

Representation of the instruction



**Parameter
Description**

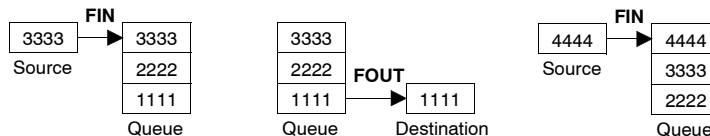
Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = clears source bit pattern from the queue
source pointer (top node)	4x	WORD	<p>First of a queue of 4x registers, contains source pointer; the next contiguous register is the first register in the queue</p> <p>In the FOUT instruction, the source data comes from the 4xxx register at the bottom of a full queue. The next contiguous 4xxx register following the source pointer register in the top node is the first register in the queue. For example, if the top node displays pointer register 40100, then the first register in the queue is 40101.</p> <p>The value posted in the source pointer equals the number of registers in the queue that are currently filled. The value of the pointer cannot exceed the integer maximum queue length value specified in the bottom node. If the value in the source pointer equals the integer specified in the bottom node, the middle output passes power and no further FIN data can be written to the queue until the FOUT instruction clears the register at the bottom of the queue to the destination register.</p>
destination register (middle node)	0x, 4x	ANY_BIT	<p>Destination register</p> <p>The destination specified in the middle node can be a 0xxx reference or 4xxx register.</p> <p>When the queue has data and the top control input to the FOUT passes power, the source data is cleared from the bottom register in the queue and is written to the destination register.</p>
queue length (bottom node)		INT, UINT	Number of 4x registers in the queue. Range: 1 ... 100
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON = queue full, no more source data can be copied to the queue
Bottom output	0x	None	ON = queue empty (value in queue pointer re

Parameter Description

Mode of Functioning

The FOUT instruction works together with the FIN instruction to produce a first in-first out (FIFO) queue. It moves the bit pattern of the holding register at the bottom of a full queue to a destination register or to word that stores 16 discrete outputs.



Note: The FOUT instruction should be placed before the FIN instruction in the ladder logic FIFO to ensure removal of the oldest data from a full queue before the newest data is entered. If the FIN block were to appear first, any attempts to enter the new data into a full queue would be ignored.

Source Pointer (Top Node)

In the FOUT instruction, the source data comes from the 4x register at the bottom of a full queue. The next contiguous 4x register following the source pointer register in the top node is the first register in the queue. For example, if the top node displays pointer register 400100, then the first register in the queue is 400101.

The value posted in the source pointer equals the number of registers in the queue that are currently filled. The value of the pointer cannot exceed the integer maximum queue length value specified in the bottom node. If the value in the source pointer equals the integer specified in the bottom node, the middle output passes power and no further FIN data can be written to the queue until the FOUT instruction clears the register at the bottom of the queue to the destination register.

Destination Register (Middle Node)

The destination specified in the middle node can be a 0x reference or 4x register. When the queue has data and the top input to the FOUT passes power, the source data is cleared from the bottom register in the queue and is written to the destination register.

FTOI: Floating Point to Integer

85

At a Glance

Introduction

This chapter describes the instruction FTOI.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	520
Representation: FTOI - Floating Point to Integer Conversion	521

Short Description

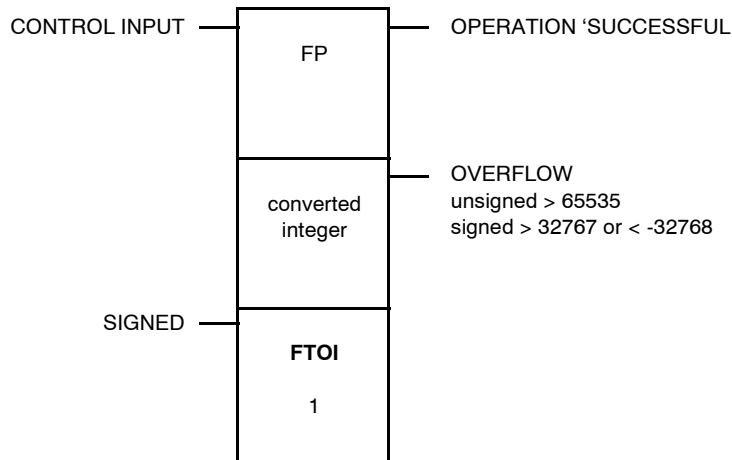
**Function
Description**

The FTOI instruction performs the conversion of a floating value to a signed or unsigned integer (stored in two contiguous registers in the top node), then stores the converted integer value in a 4x register in the middle node.

Representation: FTOI - Floating Point to Integer Conversion

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables conversion
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
FP (top node)	4x	REAL	First of two contiguous holding registers where the floating point value is stored
converted integer (middle node)	4x	INT, UINT	Converted integer value is posted here
1 (bottom node)		INT, UINT	A constant value of 1 (can not be changed)
Top output	0x	None	ON = integer conversion completed successfully
Bottom output	0x	None	ON = converted integer value is out of range: unsigned integer > 65 535 -32 768 > signed integer > 32 767

GD92 - Gas Flow Function Block

86

At A Glance

Introduction

This chapter describes the instruction GD92 AGA #3 and AGA #8 1992 Detail Method.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: GD92 - Gas Flow Function Block	524
Representation: GD92 - Gas Flow Function Block	525
Parameter Description - Inputs: GD92 - Gas Flow Function Block	527
Parameter Description - Outputs: GD92 - Gas Flow Function Block	533
Parameter Description - Optional Outputs: GD92 - Gas Flow Function Block	534

Short Description: GD92 - Gas Flow Function Block

Function Description

The Gas Flow Loadable Function Block allows you to run AGA 3 (1992) and AGA 8 (1992) equations. The computed flow rates agree within 1 ppm of the published AGA standards.

The GD92 instruction uses the detail method of characterization requiring detailed knowledge of the gas composition.

The GD92 Gas Flow Loadable function block is available only on certain Compact and Micro controllers.

Note: GD92 does not support API 21.1 audit trail. GD92 only supports a single meter run.

Important: LSUP loadable must be installed before GD92.

More Information

For detailed information about the Gas Flow Function Block loadables, especially the:

- System warning/error codes (4x+0) for each instruction
- Program warning/error codes (4x+1) for each instruction
- API 21.1 Audit Trail
- GET_LOGS.EXE utility
- SET_SIZE.EXE utility

please see *Modicon Starling Associates Gas Flow Loadable Function Block User Guide, 890 USE 137 00, Version 2.0*. The *User Guide* is available in PDF format and can be accessed through Schneider Electric's Web site.

The following instructions assume that you are using a Windows-based PC and a mouse with left and right buttons. Browser response should be similar whether you are using *Internet Explorer* or *Navigator*.

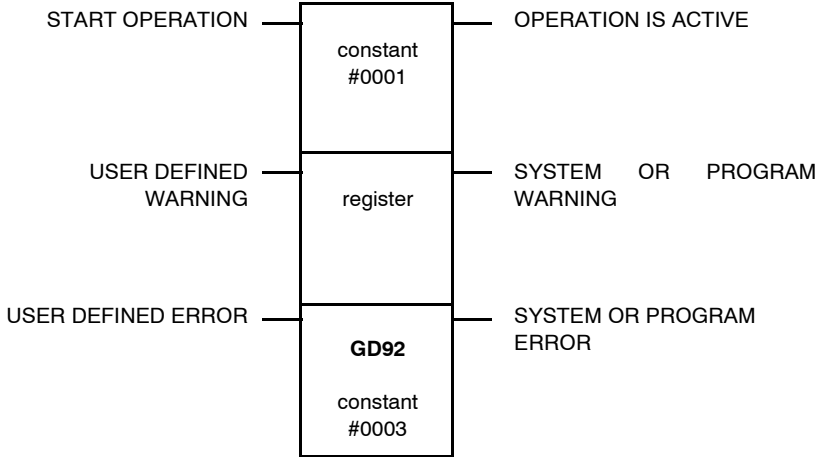
To access the Web site and view the *User Guide* follow these steps:

1. Enter this URL into your browser:
<http://www.schneider-automation.com>
2. When the home page appears, type the following term in the **Search** field:
"gas loadable"
3. Execute the search.
4. The *User Guide*'s link should display in the list that appears in the search results.
5. Double click on the title/link.
A second window appears.
6. In the **Attachment:** field of the second, smaller window right-click on **890USE13700.pdf (8036.5K)**
7. From the shortcut menu select **Save Target As...**
8. Save the **.pdf** file to your desktop or a folder.
9. The *User Guide* should open with *Adobe Acrobat Reader*.

Representation: GD92 - Gas Flow Function Block

Symbol

Representation of the instruction



**Parameter
Description**

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	<p>ON = solving</p> <p>This input starts the calculation of the gas flow. The calculations are based on your parameters entered into the input registers.</p> <p>Important: Never detach the top input while the block is running. You will generate an error 188 and the data in this block could be corrupted.</p> <p>Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 527.</i>)</p>
Middle input	0x, 1x	None	<p>Allows you to set a warning.</p> <p>Allows you to capture any user-defined warnings or errors as needed in your applications.</p> <p>Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 527.</i>)</p>
Bottom input	0x, 1x	None	<p>Allows you to set an error and STOP the flow function.</p> <p>Allows you to capture any user-defined warnings or errors as needed in your applications.</p> <p>Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 527.</i>)</p>
constant #0001 (top node)	4x	INT, UINT	The top node must contain a constant, #0001.
register (middle node)	4x	INT, UINT	<p>The 4x register entered in the middle node is the first in a group of contiguous holding registers that comprise the configuration parameters and values associated with the Gas Flow Block.</p> <p>Important: Do not attempt to change the middle node 4x register while the Gas Flow Block is running. You will lose your data and generate an error 302. If you need to change the 4x register, first STOP the PLC.</p>
#0003 (bottom node)		INT, UINT	The bottom node specifies the calculation type and must contain a constant, #0003.
Top output	0x	None	ON = Operation successful
Middle output	0x	None	ON = System or program warning
Bottom output	0x	None	ON = System or program error

Parameter Description - Inputs: GD92 - Gas Flow Function Block

Configuration Table

You MUST fill in all pertinent values in the configuration table using the reference data editor either in ProWORX or Concept, or the DX Zoom screens in Modsoft, or Meter Manager. The following inputs table lists all the configuration parameters that MUST be filled in.

The outputs (Outputs Results Table) and the optional outputs (Optional Outputs Results Table) show the calculation results of the block. Some of those parameters are required.

Important: Only valid entries are allowed. Entries outside the valid ranges are not accepted. Illegal entries result in errors or warnings.

Important: Concept 2.1 or higher may be used to load the Gas blocks. However, Concept and ProWORX do NOT provide help or DX zoom screens for configuration. When using Concept or ProWORX panel software we recommend you use Meter Manager for your configuration needs.

Inputs

The following is a detailed description of configuration variables for the GD92 gas flow function block.

Inputs	Description
4xxxx+3: 1 through 2	Location of Taps 1 - Upstream 2 - Downstream
4xxxx+3: 3 through 4	Meter Tube Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 5 through 6	Orifice Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 7 through 8	Reserved for Future Use (Do not use)
4xxxx+3: 9 through 10	Optional Outputs 1 - Yes 2 - No Note: When using only the standard outputs, the loadable uses 157 4xxxx registers. When using the optional outputs, the loadable uses 181 4xxxx registers.
4xxxx+3: 11 through 16	Reserved for Future Use (Do not use)
4xxxx+4: 1	Absolute/Gauge Pressure 0 - Static Pressure Measured in Absolute Units 1 - Static Pressure Measured in Gauge Units

Inputs	Description
4xxxx+4: 2	Low Flow Cut Off 0 - Do Not Use Flow Cut Off 1 - Use Flow Cut Off
4xxxx+4: 3 through 6	Load Command 0 - Ready to Accept Command 1 - CMD: Send Configuration to Internal Table from 4xxxx 2 - CMD: Read Configuration from Internal Table to 4xxxx 3 - CMD: Reset API 21.1 configuration change log
4xxxx+4: 7 through 8	Input Type 1 - 3xxxx Pointers entered in 4x+6 ... 4x+10 2 - Input Values entered in 4x+6 ... 4x+10
4xxxx+4: 9 through 10	Mole % Error Limits 1 - Enable 2 - Disable
4xxxx+4: 11 through 12	Dual Range Differential Pressure Option 1 - Yes 2 - No
4xxxx+4: 13 through 14	Compressible/Incompressible 1 - Compressible 2 - Incompressible
4xxxx+4: 15 through 16	Averaging Methods 0 - Flow Dependent Time Weighted Linear 1 - Flow Dependent Time Weighted Formulaic 2 - Flow Weighted Linear 3 - Flow Weighted Formulaic Note: For most applications you will use 0.
4xxxx+5: 1 through 2	Measurement Units 1 - US 2 - Metric (SI)
4xxxx+5: 3 through 16	Reserved for Future Use (Do not use)
4xxxx+6	Temperature 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+7	Pressure (absolute) 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+8	Differential Pressure 1 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+9	Differential Pressure 2 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+10	Analog Input Raw Value Minimum Temperature Data type: Unsigned integer value
4xxxx+11	Analog Input Raw Value Maximum Temperature Data type: Unsigned integer value

Inputs	Description
4xxxx+12	Analog Input Raw Value Minimum Pressure Data type: Signed integer value
4xxxx+13	Analog Input Raw Value Maximum Pressure Data type: Signed integer value
4xxxx+14	Analog Input Raw Value Minimum Differential Pressure 1 Data type: Signed integer value
4xxxx+15	Analog Input Raw Value Maximum Differential Pressure 1 Data type: Signed integer value
4xxxx+16	Analog Input Raw Value Minimum Differential Pressure 2 Data type: Signed integer value
4xxxx+17	Analog Input Raw Value Maximum Differential Pressure 2 Data type: Signed integer value
4xxxx+18 through 19	Engineering Unit Temperature Minimum -200 through 760°F (-128.89 through 404.4°C) Data type: Floating point number
4xxxx+20 through 21	Engineering Unit Temperature Maximum -200 through 760°F (-128.89 through 404.4°C) Data type: Floating point number
4xxxx+22 through 23	Engineering Unit Pressure Minimum 0 through 40,000psia (0 through 275,790.28kPa) Data type: Floating point number
4xxxx+24 through 25	Engineering Unit Pressure Maximum 0 through 40,000psia (0 through 275,790.28kPa) Data type: Floating point number
4xxxx+26 through 27	Engineering Unit Differential Pressure 1 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+28 through 29	Engineering Unit Differential Pressure 1 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+30 through 31	Engineering Unit Differential Pressure 2 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+32 through 33	Engineering Unit Differential Pressure 2 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+34 through 35	Orifice Plate Diameter, d_r (0 < d_r < 100in) (0 < d_r < 2540mm) Data type: Floating point number

Inputs	Description
4xxxx+36 through 37	Orifice Plate Diameter Measurement Temperature, T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+38 through 39	Meter Tube Internal Diameter D_r (0 < D_r < 100in) (0 < D_r < 2540mm) Data type: Floating point number
4xxxx+40 through 41	Measured Meter Tube Internal Diameter Temperature T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+42 through 43	Base Temperature, T_b (32.0 ≤ T_b < 77.0°F) (0 ≤ T_b < 25°C) Data type: Floating point number
4xxxx+44 through 45	Base Pressure, P_b (13.0 ≤ P_b < 16.0PSIA) (89.63 ≤ P_b < 110.32kPa) Data type: Floating point number
4xxxx+46 through 47	Reference Temperature for Relative Density, T_{gr} (32.0 ≤ T_{gr} < 77.0°F) (0 ≤ T_{gr} < 25°C) Data type: Floating point number
4xxxx+48 through 49	Reference Pressure for Relative Density, P_{gr} (13.0 ≤ P_{gr} < 16.0PSIA) (89.63 ≤ P_{gr} < 110.32kPa) Data type: Floating point number
4xxxx+50 through 57	Reserved for Future Use (Do not use)
4xxxx+58 through 59	User Input Correction Factor, F_u (0 < F_u < 2.0) Data type: Floating point number
4xxxx+60 through 61	Absolute Viscosity of Flowing Fluid, μ_c (0.005 ≤ μ_c ≤ 0.5 centipoise) Data type: Floating point number
4xxxx+62 through 63	Isentropic Exponent, k (1.0 ≤ k < 2.0) Data type: Floating point number
4xxxx+64	Beginning of Day Hour (0 ... 23) Data type: Unsigned integer value
4xxxx+65 through 78	Reserved for Future Use (Do not use)
4xxxx+79 through 80	Atmospheric Pressure P_{at} (3 ≤ P_{at} < 30psi) (20.684 ≤ P_{at} < 206.843kPa) Data type: Floating point number

Inputs	Description
4xxxx+81 through 82	Low Flow Cut Off Level ($\geq 0\text{ft}^3/\text{Hr}$) ($\geq 0\text{m}^3/\text{Hr}$) Used if enabled in 4x+4: 2. Data type: Floating point number
4xxxx+83 through 84	Mole % of Methane, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+85 through 86	Mole % of Nitrogen, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+87 through 88	Mole % of Carbon Dioxide, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+89 through 90	Mole % of Ethane, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
xxx+91 through 92	Mole % of Propane, x_i $*(0.0 \leq x_i \leq 12)$ Data type: Floating point number
4xxxx+93 through 94	Mole % of Water, x_i $*(0.0 \leq x_i \leq 10)$ Data type: Floating point number
4xxxx+95 through 96	Mole % of Hydrogen Sulfide, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+97 through 98	Mole % of Hydrogen, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+99 through 100	Mole % of Carbon Monoxide, x_i $*(0.0 \leq x_i \leq 3)$ Data type: Floating point number
4xxxx+101 through 102	Mole % of Oxygen, x_i $*(0.0 \leq x_i \leq 21)$ Data type: Floating point number
4xxxx+103 through 104	Mole % of I-Butane, x_i $*(0.0 \leq x_i \leq 6)$ for combined butanes Data type: Floating point number

Inputs	Description
4xxxx+105 through 106	Mole % of n-Butane, x_i *(0.0 <= x_i <= 6) for combined butanes Data type: Floating point number
4xxxx+107 through 108	Mole % of I-Pentane, x_i *(0.0 <= x_i <= 4) for combined pentanes Data type: Floating point number
4xxxx+109 through 110	Mole % of n-Pentane, x_i *(0.0 <= x_i <= 4) for combined pentanes Data type: Floating point number
4xxxx+111 through 112	Mole % of Hexane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+113 through 114	Mole % of Heptane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+115 through 116	Mole % of Octane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+117 through 118	Mole % of Nonane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+119 through 120	Mole % of Decane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+121 through 122	Mole % of Helium, x_i *(0.0 <= x_i <= 30) Data type: Floating point number
4xxxx+123 through 124	Mole % of Argon, x_i *(0.0 <= x_i <= 100) Data type: Floating point number

*Valid range

Parameter Description - Outputs: GD92 - Gas Flow Function Block

Outputs Results Table

The outputs show the calculation results of the block.

Outputs	Description
4xxxx+0	System Warning/Error Code (Displayed in Hex mode)
4xxxx+1	Program Warning/Error Code
4xxxx+2	Version Number (Displayed in Hex mode)
4xxxx+125 through 126	Temperature at Flowing Conditions (T_f) (F or C)
4xxxx+127 through 128	Pressure (P_f) (psia or kPa)
4xxxx+129 through 130	Differential Pressure (h_w) (in H ₂ O or kPa)
4xxxx+131 through 132	Integral Value (IV)
4xxxx+133 through 134	Integral Multiplier Value (IMV)
4xxxx+135 through 136	Volume Flow Rate at Base Conditions (T_b , P_b), Q_b (ft ₃ /hr or m ₃ /hr)
4xxxx+137 through 138	Mass Flow Rate (Q_m) (lbm/hr or Kg/hr)
4xxxx+139 through 140	Accumulated Volume Current Day
4xxxx+141 through 142	Accumulated Volume Last Hour
4xxxx+143 through 144	Accumulated Volume Last Day
4xxxx+145 through 146	Average Temperature Last Day
4xxxx+147 through 148	Average Pressure Last Day
4xxxx+149 through 150	Average Differential Pressure Last Day
4xxxx+151 through 152	Average IV Last Day
4xxxx+153 through 154	Average Volume Flow Rate at Base Conditions (T_b , P_b) for the Last Day
4xxxx+155: 13	4xxxx Table Differs from Actual Configuration
4xxxx+155: 14	Flow Rate Solve Complete Heartbeat
4xxxx+155: 15	Block is Functioning Heartbeat
4xxxx+155: 16	End of Day Flag

Parameter Description - Optional Outputs: GD92 - Gas Flow Function Block

Optional Outputs Configuration Table

The optional outputs show the calculation results of the block. These are only active if 4x+3: 9 ... 10 is 1.

Optional Outputs	Description
4xxxx+156 through 157	Compressibility at Flowing Conditions (T_f , P_f), Z_f
4xxxx+158 through 159	Compressibility at Base Conditions (T_b , P_b), Z_b
4xxxx+160 through 161	Compressibility at Standard Conditions (T_s , P_s), Z_s
4xxxx+162 through 163	Density at Fluid Flowing Conditions ($P_{t,p}$)
4xxxx+164 through 165	Density of Fluid at Base Conditions (ρ)
4xxxx+166 through 167	Supercompressibility (F_{pv})
4xxxx+168 through 169	Gas Relative Density (G_r)
4xxxx+170 through 171	Orifice Plate Coefficient of Discharge (C_d)
4xxxx+172 through 173	Expansion Factor (Y)
4xxxx+174 through 175	Velocity of Approach Factor (E_v)
4xxxx+176 through 177	Volume Flow Rate at Flowing Conditions (T_f , P_f), Q_f
4xxxx+178 through 179	Reserved for Future Use (Do not use)
4xxxx+180	Orifice Plate Coefficient of Discharge Bounds Flag within Iteration Scheme (C_{d-f})

GFNX AGA#3 '85 and NX19 '68 Gas Flow Function Block

87

At A Glance

Introduction

This chapter describes the instruction GFNX AGA#3 '85 AND NX19 '68.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: GFNX - Gas Flow Function Block	536
Representation: GFNX - Gas Flow Function Block	537
Parameter Description - Inputs: GFNX - Gas Flow Function Block	539
Parameter Description - Outputs: GFNX - Gas Flow Function Block	546
Parameter Description - Optional Outputs: GFNX - Gas Flow Function Block	547

Short Description: GFNX - Gas Flow Function Block

Function Description

The GFNX AGA #3 '85 and NX19 API 21.1 Gas Flow Loadable function block is available only on certain Compact and Micro controllers.

The Gas Flow Loadable Function Block allows you to run AGA 3 (1992) and AGA 8 (1992) equations. The computed flow rates agree within 1 ppm of the published AGA standards.

The GFNX instruction uses the detail method of characterization requiring detailed knowledge of the gas composition.

Important: LSUP loadable must be installed before GFNX.

More Information

For detailed information about the Gas Flow Function Block loadables, especially the:

- System warning/error codes (4x+0) for each instruction
- Program warning/error codes (4x+1) for each instruction
- API 21.1 Audit Trail
- GET_LOGS.EXE utility
- SET_SIZE.EXE utility

please see *Modicon Starling Associates Gas Flow Loadable Function Block User Guide, 890 USE 137 00, Version 2.0*. The *User Guide* is available in PDF format and can be accessed through Schneider Electric's Web site.

The following instructions assume that you are using a Windows-based PC and a mouse with left and right buttons. Browser response should be similar whether you are using *Internet Explorer* or *Navigator*.

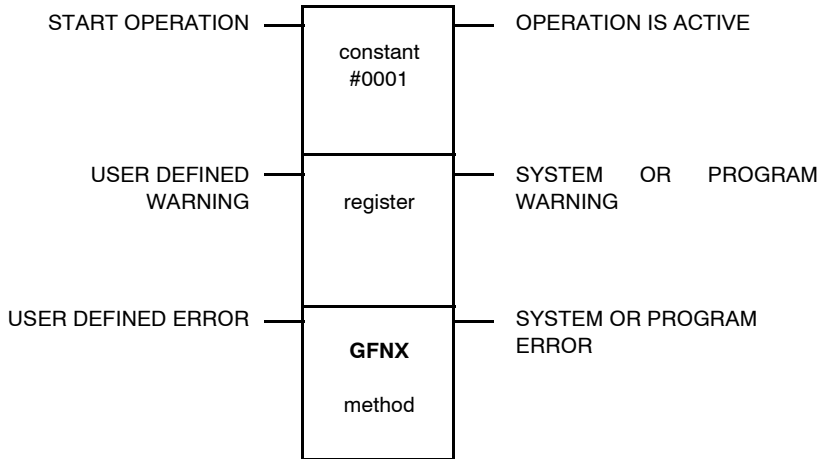
To access the Web site and view the *User Guide* follow these steps:

1. Enter this URL into your browser:
<http://www.schneider-automation.com>
 2. When the home page appears, type the following term in the **Search** field:
"gas loadable"
 3. Execute the search.
 4. The *User Guide*'s link should display in the list that appears in the search results.
 5. Double click on the title/link.
A second window appears.
 6. In the **Attachment:** field of the second, smaller window right-click on **890USE13700.pdf (8036.5K)**
 7. From the shortcut menu select **Save Target As...**
 8. Save the **.pdf** file to your desktop or a folder.
 9. The *User Guide* should open with *Adobe Acrobat Reader*.
-

Representation: GFNX - Gas Flow Function Block

Symbol

Representation of the instruction



**Parameter
Description**

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = solving This input starts the calculation of the gas flow. The calculations are based on your parameters entered into the input registers. Important: Never detach the top input while the block is running. You will generate an error 188 and the data in this block could be corrupted. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 539.</i>)
Middle input	0x, 1x	None	Allows you to set a warning. Allows you to set a warning and log peripheral activities in the audit trail even log without stopping the block. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 539.</i>)
Bottom input	0x, 1x	None	Allows you to set an error and STOP the flow function. Allows you to set an error, log peripheral errors in the audit trail event log, and STOP the flow function. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 539.</i>)
constant #0001 (top node)	4x	INT, UINT	The top node must contain a constant, #0001.
register (middle node)	4x	INT, UINT	The 4x register entered in the middle node is the first in a group of contiguous holding registers that comprise the configuration parameters and values associated with the Gas Flow Block. Important: Do not attempt to change the middle node 4x register while the Gas Flow Block is running. You will lose your data and generate an error 302. If you need to change the 4x register, first STOP the PLC.
method (bottom node)		INT, UINT	The bottom node specifies the calculation type and must contain a constant. Important: Use only valid entries. Other entries deny access to the block's DX zoom screens.
Top output	0x	None	ON = Operation successful
Middle output	0x	None	ON = System or program warning
Bottom output	0x	None	ON = System or program error

Parameter Description - Inputs: GFNX - Gas Flow Function Block

Configuration Table

You MUST fill in all pertinent values in the configuration table using the reference data editor either in ProWORX or Concept, or the DX Zoom screens in Modsoft, or Meter Manager. The following inputs table lists all the configuration parameters that MUST be filled in.

The outputs (Outputs Results Table) and the optional outputs (Optional Outputs Results Table) show the calculation results of the block. Some of those parameters are required.

Important: Only valid entries are allowed. Entries outside the valid ranges are not accepted. Illegal entries result in errors or warnings.

Important: Concept 2.1 or higher may be used to load the Gas blocks. However, Concept and ProWORX do NOT provide help or DX zoom screens for configuration. When using Concept or ProWORX panel software we recommend you use Meter Manager for your configuration needs.

Inputs

The following is a detailed description of configuration variables for the GFNX gas flow function block.

Inputs	Description
4xxxx+3: 1 through 2	Location of Taps 1 - Upstream 2 - Downstream
4xxxx+3: 3 through 4	Meter Tube Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 5 through 6	Orifice Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 7 through 8	Reserved for Future Use (Do not use)
4xxxx+3: 9 through 10	Optional Outputs 1 - Yes 2 - No Note: When using only the standard outputs, the loadable uses 157 4xxxx registers. When using the optional outputs, the loadable uses 181 4xxxx registers.
4xxxx+3: 11 through 16	Reserved for Future Use (Do not use)
4xxxx+4: 1	Absolute/Gauge Pressure 0 - Static Pressure Measured in Absolute Units 1 - Static Pressure Measured in Gauge Units

Inputs	Description
4xxxx+4: 2	Low Flow Cut Off 0 - Do Not Use Flow Cut Off 1 - Use Flow Cut Off
4xxxx+4: 3 through 6	Load Command 0 - Ready to Accept Command 1 - CMD: Send Configuration to Internal Table from 4xxxx 2 - CMD: Read Configuration from Internal Table to 4xxxx 3 - CMD: Reset API 21.1 configuration change log
4xxxx+4: 7 through 8	Input Type 1 - 3xxxx Pointers entered in 4x+6 ... 4x+10 2 - Input Values entered in 4x+6 ... 4x+10
4xxxx+4: 9 through 10	Mole % Error Limits 1 - Enable 2 - Disable
4xxxx+4: 11 through 12	Dual Range Differential Pressure Option 1 - Yes 2 - No
4xxxx+4: 13 through 14	Compressible/Incompressible 1 - Compressible 2 - Incompressible
4xxxx+4: 15 through 16	Averaging Methods 0 - Flow Dependent Time Weighted Linear 1 - Flow Dependent Time Weighted Formulaic 2 - Flow Weighted Linear 3 - Flow Weighted Formulaic Note: For most applications you will use 0.
4xxxx+5: 1 through 2	Measurement Units 1 - US 2 - Metric (SI)
4xxxx+5: 3 through 14	Reserved for Future Use (Do not use)
4xxxx+5: 15 through 16	Reserved for API 21.1
4xxxx+6	Temperature 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+7	Pressure (absolute) 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+8	Differential Pressure 1 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+9	Differential Pressure 2 3xxxx Pointer or Input Value Data type: Unsigned integer value

Inputs	Description
4xxxx+10	Analog Input Raw Value Minimum Temperature Data type: Unsigned integer value
4xxxx+11	Analog Input Raw Value Maximum Temperature Data type: Unsigned integer value
4xxxx+12	Analog Input Raw Value Minimum Pressure Data type: Signed integer value
4xxxx+13	Analog Input Raw Value Maximum Pressure Data type: Signed integer value
4xxxx+14	Analog Input Raw Value Minimum Differential Pressure 1 Data type: Signed integer value
4xxxx+15	Analog Input Raw Value Maximum Differential Pressure 1 Data type: Signed integer value
4xxxx+16	Analog Input Raw Value Minimum Differential Pressure 2 Data type: Signed integer value
4xxxx+17	Analog Input Raw Value Maximum Differential Pressure 2 Data type: Signed integer value
4xxxx+18 through 19	Engineering Unit Temperature Minimum -40 through 240°F (-40 through 115.5556°C) Data type: Floating point number
4xxxx+20 through 21	Engineering Unit Temperature Maximum -40 through 240°F (-40 through 115.5556°C) Data type: Floating point number
4xxxx+22 through 23	Engineering Unit Pressure Minimum 0 through 5,000psia (0 through 34,473.785kPa) Data type: Floating point number
4xxxx+24 through 25	Engineering Unit Pressure Maximum 0 through 5,000psia (0 through 34,473.785kPa) Data type: Floating point number
4xxxx+26 through 27	Engineering Unit Differential Pressure 1 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+28 through 29	Engineering Unit Differential Pressure 1 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+30 through 31	Engineering Unit Differential Pressure 2 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number

Inputs	Description
4xxxx+32 through 33	Engineering Unit Differential Pressure 2 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+34 through 35	Orifice Plate Diameter, d_r (0 < d_r < 100in) (0 < d_r < 2540mm) Data type: Floating point number
4xxxx+36 through 37	Orifice Plate Diameter Measurement Temperature, T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+38 through 39	Meter Tube Internal Diameter D_r (0 < D_r < 100in) (0 < D_r < 2540mm) Data type: Floating point number
4xxxx+40 through 41	Measured Meter Tube Internal Diameter Temperature T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+42 through 43	Base Temperature, T_b (32.0 ≤ T_b < 77.0°F) (0 ≤ T_b < 25°C) Data type: Floating point number
4xxxx+44 through 45	Base Pressure, P_b (13.0 ≤ P_b < 16.0PSIA) (89.63 ≤ P_b < 110.32kPa) Data type: Floating point number
4xxxx+46 through 57	Reserved for Future Use (Do not use)
4xxxx+58 through 59	User Input Correction Factor, F_u (0 < F_u < 2.0) Data type: Floating point number
4xxxx+60 through 63	Reserved for Future Use (Do not use)
4xxxx+64	Beginning of Day Hour (0 ... 23) Data type: Unsigned integer value
4xxxx+65 through 78	Reserved for API 21.1
4xxxx+79 through 80	Atmospheric Pressure P_{at} (3 ≤ P_{at} < 30psi) (20.684 ≤ P_{at} < 206.843kPa) Data type: Floating point number
4xxxx+81 through 82	Low Flow Cut Off Level (≥ 0ft ³ /Hr) (≥ 0m ³ /Hr) Used if enabled in 4x+4: 2. Data type: Floating point number

Inputs Detail Method 11

The following inputs apply to Detail Method 11.

Inputs	Description
Applies when using Detail Method 11	
4xxxx+83 through 84	Mole % of Methane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+85 through 86	Mole % of Nitrogen, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+87 through 88	Mole % of Carbon Dioxide, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+89 through 90	Mole % of Ethane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
xxx+91 through 92	Mole % of Propane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+93 through 94	Mole % of Water, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+95 through 96	Mole % of Hydrogen Sulfide, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+97 through 98	Mole % of Hydrogen, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+99 through 100	Mole % of Carbon Monoxide, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+101 through 102	Mole % of Oxygen, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+103 through 104	Mole % of I-Butane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number

Inputs	Description
Applies when using Detail Method 11	
4xxxx+105 through 106	Mole % of n-Butane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+107 through 108	Mole % of I-Pentane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+109 through 110	Mole % of n-Pentane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+111 through 112	Mole % of Hexane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+113 through 114	Mole % of Heptane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+115 through 116	Mole % of Octane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+117 through 118	Mole % of Nonane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+119 through 120	Mole % of Decane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number
4xxxx+121 through 122	Mole % of Helium, x_i *(0.0 <= x_i <= 30) Data type: Floating point number
4xxxx+123 through 124	Reserved for Future Use (Do not use)

*Valid range

**Inputs Gross
Methods 10, 12,
and 13**

The following inputs apply to Gross Methods 10, 12, and 13.

Inputs	Description
Applies when using Gross Methods 10, 12, and 13	
4xxxx+83 through 84	Mole % of Methane, x_i *(0.0 <= x_i <= 100) Data type: Floating point number (Required for method 13 ONLY)
4xxxx+85 through 86	Mole % of Nitrogen, x_i *(0.0 <= x_i <= 100) Data type: Floating point number (Required for methods 10, 12, and 13)
4xxxx+87 through 88	Mole % of Carbon Dioxide, x_i *(0.0 <= x_i <= 100) Data type: Floating point number (Required for methods 10, 12, and 13)
4xxxx+93 through 94	Specific Gravity, G_r (0.07 <= G_r < 1.52) Data type: Floating point number (Required for methods 10, 12, and 13)
4xxxx+95 through 96	Heating Value, HV (0.07 HV < 1800) Data type: Floating point number (Required for method 12 ONLY)

*Valid range

Parameter Description - Outputs: GFNX - Gas Flow Function Block

Outputs Results Table

The outputs show the calculation results of the block.

Outputs	Description
4xxx+0	System Warning/Error Code (Displayed in Hex mode)
4xxx+1	Program Warning/Error Code
4xxx+2	Version Number (Displayed in Hex mode)
4xxx+125 through 126	Temperature at Flowing Conditions (T_f) (F or C)
4xxx+127 through 128	Pressure (P_f) (psia or kPa)
4xxx+129 through 130	Differential Pressure (h_w) (in H ₂ O or kPa)
4xxx+131 through 132	Integral Value (IV)
4xxx+133 through 134	Integral Multiplier Value (IMV)
4xxx+135 through 136	Volume Flow Rate at Base Conditions (T_b , P_b), Q_b ft ³ /hr or m ³ /hr
4xxx+137 through 138	Reserved for Future Use (Do not use)
4xxx+139 through 140	Accumulated Volume Current Day
4xxx+141 through 142	Accumulated Volume Last Hour
4xxx+143 through 144	Accumulated Volume Last Day
4xxx+145 through 152	Reserved for API 21.1
4xxx+153	User-definable warning/error value (Use for API 21.1)
4xxx+155: 13	4xxx Table Differs from Actual Configuration
4xxx+155: 14	Flow Rate Solve Complete Heartbeat
4xxx+155: 15	Block is Functioning Heartbeat
4xxx+155: 16	End of Day Flag Note: This status bit does not appear in the DX Zoom screen but may be used in program logic.

Parameter Description - Optional Outputs: GFNX - Gas Flow Function Block

Optional Outputs Configuration Table

The optional outputs show the calculation results of the block. These are only active if 4x+3: 9 ... 10 is 1.

Optional Outputs	Description
4xxxx+156 through 165	Reserved for Future Use (Do not use)
4xxxx+166 through 167	Supercompressibility, F_{pv}
4xxxx+168 through 169	Gas Relative Density, G_r
4xxxx+170 through 171	Reserved for Future Use (Do not use)
4xxxx+172 through 173	Expansion Factor, Y
4xxxx+174 through 180	Reserved for Future Use (Do not use)

GG92 AGA #3 1992 Gross Method Gas Flow Function Block



88

At A Glance

Introduction

This chapter describes the instruction GG92 AGA #3 and AGA #8 1992 Gross Method Gas Flow Function Block.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: GG92 - Gas Flow Function Block	550
Representation: GG92 - Gas Flow Function Block	551
Parameter Description - Inputs: GG92 - Gas Flow Function Block	553
Parameter Description - Outputs: GG92 - Gas Flow Function Block	558
Parameter Description - Optional Outputs: GG92 - Gas Flow Function Block	559

Short Description: GG92 - Gas Flow Function Block

Function Description

The GG92 Gas Flow Loadable function block is available only on certain Compact and Micro controllers.

The Gas Flow Loadable Function Block allows you to run AGA 3 (1992) and AGA 8 (1992) equations. The computed flow rates agree within 1 ppm of the published AGA standards. The GG92 allows the API 21.1 audit trail. The GG92 permits 8 passes.

The GG92 instruction uses the detail method of characterization requiring detailed knowledge of the gas composition.

Important: LSUP loadable must be installed before GG92.

More Information

For detailed information about the Gas Flow Function Block loadables, especially the:

- System warning/error codes (4x+0) for each instruction
- Program warning/error codes (4x+1) for each instruction
- API 21.1 Audit Trail
- GET_LOGS.EXE utility
- SET_SIZE.EXE utility

please see *Modicon Starling Associates Gas Flow Loadable Function Block User Guide, 890 USE 137 00, Version 2.0*. The *User Guide* is available in PDF format and can be accessed through Schneider Electric's Web site.

The following instructions assume that you are using a Windows-based PC and a mouse with left and right buttons. Browser response should be similar whether you are using *Internet Explorer* or *Navigator*.

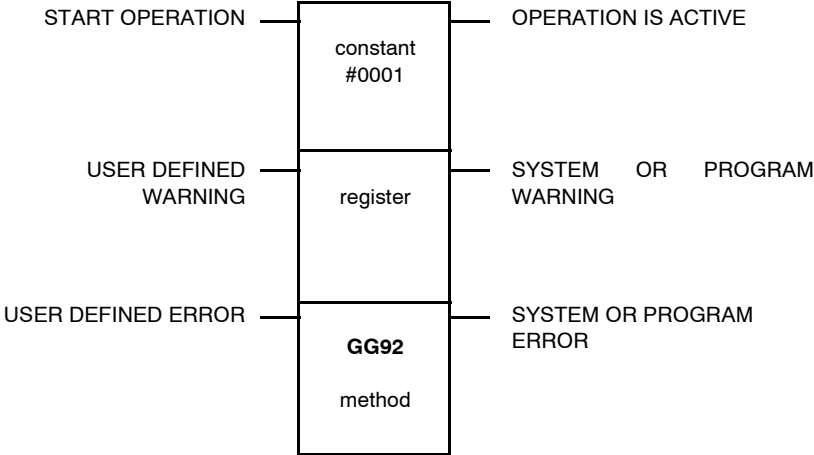
To access the Web site and view the *User Guide* follow these steps:

1. Enter this URL into your browser:
<http://www.schneider-automation.com>
 2. When the home page appears, type the following term in the **Search** field:
"gas loadable"
 3. Execute the search.
 4. The *User Guide*'s link should display in the list that appears in the search results.
 5. Double click on the title/link.
A second window appears.
 6. In the **Attachment:** field of the second, smaller window right-click on **890USE13700.pdf (8036.5K)**
 7. From the shortcut menu select **Save Target As...**
 8. Save the **.pdf** file to your desktop or a folder.
 9. The *User Guide* should open with *Adobe Acrobat Reader*.
-

Representation: GG92 - Gas Flow Function Block

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = solving This input starts the calculation of the gas flow. The calculations are based on your parameters entered into the input registers. Important: Never detach the top input while the block is running. You will generate an error 188 and the data in this block could be corrupted. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 553.</i>)
Middle input	0x, 1x	None	Allows you to set a warning. Allows you to set a warning and log peripheral activities in the audit trail event log without stopping the block. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 553.</i>)
Bottom input	0x, 1x	None	Allows you to set an error and STOP the flow function. Allows you to set an error, log peripheral errors in the audit trail event log, and STOP the flow function. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 553.</i>)
constant #0001 (top node)	4x	INT, UINT	The top node must contain a constant, #0001.
register (middle node)	4x	INT, UINT	The 4x register entered in the middle node is the first in a group of contiguous holding registers that comprise the configuration parameters and values associated with the Gas Flow Block. Important: Do not attempt to change the middle node 4x register while the Gas Flow Block is running. You will lose your data. If you need to change the 4x register, first STOP the PLC.
method (bottom node)		INT, UINT	The bottom node specifies the calculation type and must contain a constant, #0003. The integer value entered in the bottom node specifies the <i>characterization method</i> : <ul style="list-style-type: none"> ● 1 - Gross Method 1 (HV-Gr-CO₂) ● 2 - Gross Method 2 (Gr-CO₂-N₂)
Top output	0x	None	ON = Operation successful
Middle output	0x	None	ON = System or program warning
Bottom output	0x	None	ON = System or program error

Parameter Description - Inputs: GG92 - Gas Flow Function Block

Configuration Table

You MUST fill in all pertinent values in the configuration table using the reference data editor either in ProWORX or Concept, or the DX Zoom screens in Modsoft, or Meter Manager. The following inputs table lists all the configuration parameters that MUST be filled in.

The outputs (Outputs Results Table) and the optional outputs (Optional Outputs Results Table) show the calculation results of the block. Some of those parameters are required.

Important: Only valid entries are allowed. Entries outside the valid ranges are not accepted. Illegal entries result in errors or warnings.

Important: Concept 2.1 or higher may be used to load the Gas blocks. However, Concept and ProWORX do NOT provide help or DX zoom screens for configuration. When using Concept or ProWORX panel software we recommend you use Meter Manager for your configuration needs.

Inputs

The following is a detailed description of configuration variables for the GG92 gas flow function block.

Inputs	Description
4xxxx+3: 1 through 2	Location of Taps 1 - Upstream 2 - Downstream
4xxxx+3: 3 through 4	Meter Tube Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 5 through 6	Orifice Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 7 through 8	Reserved for Future Use (Do not use)
4xxxx+3: 9 through 10	Optional Outputs 1 - Yes 2 - No Note: When using only the standard outputs, the loadable uses 157 4xxxx registers. When using the optional outputs, the loadable uses 181 4xxxx registers.
4xxxx+3: 11 through 16	Reserved for Future Use (Do not use)
4xxxx+4: 1	Absolute/Gauge Pressure 0 - Static Pressure Measured in Absolute Units 1 - Static Pressure Measured in Gauge Units

Inputs	Description
4xxxx+4: 2	Low Flow Cut Off 0 - Do Not Use Flow Cut Off 1 - Use Flow Cut Off
4xxxx+4: 3 through 6	Load Command 0 - Ready to Accept Command 1 - CMD: Send Configuration to Internal Table from 4xxxx 2 - CMD: Read Configuration from Internal Table to 4xxxx 3 - CMD: Reset API 21.1 configuration change log
4xxxx+4: 7 through 8	Input Type 1 - 3xxxx Pointers entered in 4x+6 ... 4x+10 2 - Input Values entered in 4x+6 ... 4x+10
4xxxx+4: 9 through 10	Mole % Error Limits 1 - Enable 2 - Disable
4xxxx+4: 11 through 12	Dual Range Differential Pressure Option 1 - Yes 2 - No
4xxxx+4: 13 through 14	Compressible/Incompressible 1 - Compressible 2 - Incompressible
4xxxx+4: 15 through 16	Averaging Methods 0 - Flow Dependent Time Weighted Linear 1 - Flow Dependent Time Weighted Formulaic 2 - Flow Weighted Linear 3 - Flow Weighted Formulaic Note: For most applications you will use 0.
4xxxx+5: 1 through 2	Measurement Units 1 - US 2 - Metric (SI)
4xxxx+5: 3 through 14	Reserved for Future Use (Do not use)
4xxxx+5: 15 through 16	Reserved for API 21.1
4xxxx+6	Temperature 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+7	Pressure (absolute) 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+8	Differential Pressure 1 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+9	Differential Pressure 2 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+10	Analog Input Raw Value Minimum Temperature Data type: Unsigned integer value

Inputs	Description
4xxxx+11	Analog Input Raw Value Maximum Temperature Data type: Unsigned integer value
4xxxx+12	Analog Input Raw Value Minimum Pressure Data type: Signed integer value
4xxxx+13	Analog Input Raw Value Maximum Pressure Data type: Signed integer value
4xxxx+14	Analog Input Raw Value Minimum Differential Pressure 1 Data type: Signed integer value
4xxxx+15	Analog Input Raw Value Maximum Differential Pressure 1 Data type: Signed integer value
4xxxx+16	Analog Input Raw Value Minimum Differential Pressure 2 Data type: Signed integer value
4xxxx+17	Analog Input Raw Value Maximum Differential Pressure 2 Data type: Signed integer value
4xxxx+18 through 19	Engineering Unit Temperature Minimum 14 through 149°F (-10 through 65°C) Data type: Floating point number
4xxxx+20 through 21	Engineering Unit Temperature Maximum 14 through 149°F (-10 through 65°C) Data type: Floating point number
4xxxx+22 through 23	Engineering Unit Pressure Minimum 0 through 1,470psia (0 through 11,996kPa) Data type: Floating point number
4xxxx+24 through 25	Engineering Unit Pressure Maximum 0 through 1,470psia (0 through 11,996kPa) Data type: Floating point number
4xxxx+26 through 27	Engineering Unit Differential Pressure 1 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+28 through 29	Engineering Unit Differential Pressure 1 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+30 through 31	Engineering Unit Differential Pressure 2 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+32 through 33	Engineering Unit Differential Pressure 2 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+34 through 35	Orifice Plate Diameter, d_r (0 < d_r < 100in) (0 < d_r < 2540mm) Data type: Floating point number

Inputs	Description
4xxxx+36 through 37	Orifice Plate Diameter Measurement Temperature, T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+38 through 39	Meter Tube Internal Diameter D_r (0 < D_r < 100in) (0 < D_r < 2540mm) Data type: Floating point number
4xxxx+40 through 41	Measured Meter Tube Internal Diameter Temperature T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+42 through 43	Base Temperature, T_b (32.0 ≤ T_b < 77.0°F) (0 ≤ T_b < 25°C) Data type: Floating point number
4xxxx+44 through 45	Base Pressure, P_b (13.0 ≤ P_b < 16.0PSIA) (89.63 ≤ P_b < 110.32kPa) Data type: Floating point number
4xxxx+46 through 47	Reference Temperature for Relative Density, T_{gr} (32.0 ≤ T_{gr} < 77.0°F) (0 ≤ T_{gr} < 25°C) Data type: Floating point number
4xxxx+48 through 49	Reference Pressure for Relative Density, P_{gr} (13.0 ≤ P_{gr} < 16.0PSIA) (89.63 ≤ P_{gr} < 110.32kPa) Data type: Floating point number
4xxxx+50 through 51	Reference Temperature for Molar Density, T_d (32.0 ≤ T_d < 77.0°F) (0 ≤ T_d < 25°C) Data type: Floating point number
4xxxx+52 through 53	Reference Pressure for Molar Density, P_d (13.0 ≤ P_d < 16.0PSIA) (89.63 ≤ P_d < 110.32kPa) Data type: Floating point number
4xxxx+54 through 55	Reference Temperature fo Heating Value, T_h (32.0 ≤ T_h < 77.0) (0 ≤ T_h < 25°C) Data type: Floating point number
4xxxx+56 through 57	Reserved for Future Use (Do not use)
4xxxx+58 through 59	User Input Correction Factor, F_u (0 < F_u < 2.0) Data type: Floating point number
4xxxx+60 through 61	Absolute Viscosity of Flowing Fluid, μ_c (0.01 ≤ μ_c ≤ 0.1 centipoise) Data type: Floating point number

Inputs	Description
4xxxx+62 through 63	Iisentropic Exponent, k (1.0 <= k < 2.0) Data type: Floating point number
4xxxx+64	Beginning of Day Hour (0 ... 23) Data type: Unsigned integer value
4xxxx+65 through 78	Reserved for API 21.1
4xxxx+79 through 80	Atmospheric Pressure P_{at} (3 <= Pat <30psi) (20.684 <= P _{at} < 206.843kPa) Data type: Floating point number
4xxxx+81 through 82	Low Flow Cut Off Level (>= 0ft ³ /Hr) (>= 0m ³ /Hr) Data type: Floating point number
4xxxx+83 through 84	Reserved for Future Use (Do not use)
4xxxx+85 through 86	Mole % of Nitrogen, x_i *(0.0 <= x _i <= 50) (Required for method 2 ONLY) Data type: Floating point number
4xxxx+87 through 88	Mole % of Carbon Dioxide, x_i *(0.0 <= x _i <= 30) Data type: Floating point number
4xxxx+89 through 90	Mole % of Hydrogen, x_i *(0.0 <= x _i <= 10) Data type: Floating point number
4xxxx+91 through 92	Mole % of Carbon Monoxide, x_i *(0.0 <= x _i <= 3) Data type: Floating point number
4xxxx+93 through 94	Specific Gravity, G_r *(.55 < G _r < 0.87)) Data type: Floating point number
4xxxx+95 through 96	Heating Value, HV *(477 <= HV < 1211BTU/Ft ³) (17.7725 <= HV < 45.1206Kj/dm ³) (Required for Method 1 ONLY) Data type: Floating point number
4xxxx+97 through 124	Reserved for Future Use (Do not use)

*Valid range

Parameter Description - Outputs: GG92 - Gas Flow Function Block

Outputs Results Table

The outputs show the calculation results of the block.

Outputs	Description
4xxxx+0	System Warning/Error Code (Displayed in Hex mode)
4xxxx+1	Program Warning/Error Code
4xxxx+2	Version Number (Displayed in Hex mode)
4xxxx+125 through 126	Temperature at Flowing Conditions (T_f) (F or C)
4xxxx+127 through 128	Pressure (P_f) (psia or kPa)
4xxxx+129 through 130	Differential Pressure (h_w) (in H ₂ O or kPa)
4xxxx+131 through 132	Integral Value (IV)
4xxxx+133 through 134	Integral Multiplier Value (IMV)
4xxxx+135 through 136	Volume Flow Rate at Base Conditions (T_b , P_b), Q_b (ft ³ /hr or m ³ /hr)
4xxxx+137 through 138	Mass Flow Rate (Q_m) (lbm/hr or Kg/hr)
4xxxx+139 through 140	Accumulated Volume Current Day
4xxxx+141 through 142	Accumulated Volume Last Hour
4xxxx+143 through 144	Accumulated Volume Last Day
4xxxx+145 through 152	Reserved for API 21.1
4xxxx+153	User definable warning/error value (Use for API 21.1)
4xxxx+155: 13	4xxxx Table Differs from Actual Configuration
4xxxx+155: 14	Flow Rate Solve Complete Heartbeat
4xxxx+155: 15	Block is Functioning Heartbeat
4xxxx+155: 16	End of Day Flag

Parameter Description - Optional Outputs: GG92 - Gas Flow Function Block

Optional Outputs Configuration Table

The optional outputs show the calculation results of the block. These are only active if 4x+3: 9 ... 10 is 1.

Optional Outputs	Description
4xxxx+156 through 157	Compressibility at Flowing Conditions (T_f, P_f), Z_f
4xxxx+158 through 159	Compressibility at Base Conditions (T_b, P_b), Z_b
4xxxx+160 through 161	Compressibility at Standard Conditions (T_s, P_s), Z_s
4xxxx+162 through 163	Density at Fluid Flowing Conditions ($P_{t,p}$)
4xxxx+164 through 165	Density of Fluid at Base Conditions (ρ)
4xxxx+166 through 167	Supercompressibility (F_{pv})
4xxxx+168 through 169	Gas Relative Density (G_r)
4xxxx+170 through 171	Orifice Plate Coefficient of Discharge (C_d)
4xxxx+172 through 173	Expansion Factor (Y)
4xxxx+174 through 175	Velocity of Approach Factor (E_v)
4xxxx+176 through 177	Volume Flow Rate at Flowing Conditions (T_f, P_f), Q_f
4xxxx+178 through 179	Reserved for Future Use (Do not use)
4xxxx+180	Orifice Plate Coefficient of Discharge Bounds Flag within Iteration Scheme (C_{d-t})

GM92 AGA #3 and #8 1992 Detail Method Gas Flow Function Block

89

At A Glance

Introduction

This chapter describes the instruction GM92 AGA #3 and #8 1992 Detail Method with API 21.1 Audit Trail.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: GM92 - Gas Flow Function Block	562
Representation: GM92 - Gas Flow Function Block	563
Parameter Description - Inputs: GM92 - Gas Flow Function Block	565
Parameter Description - Outputs: GM92 - Gas Flow Function Block	571
Parameter Description - Optional Outputs: GM92 - Gas Flow Function Block	572

Short Description: GM92 - Gas Flow Function Block

Function Description

The GM92 Gas Flow Loadable function block is available only on certain Compact and Micro controllers.

The Gas Flow Loadable Function Block allows you to run AGA 3 (1992) and AGA 8 (1992) equations. The computed flow rates agree within 1 ppm of the published AGA standards.

This function block allows you to run the API 21.1 Audit Trail. The block has 8 meter runs.

Important: LSUP loadable must be installed before GM92.

More Information

For detailed information about the Gas Flow Function Block loadables, especially the:

- System warning/error codes (4x+0) for each instruction
- Program warning/error codes (4x+1) for each instruction
- API 21.1 Audit Trail
- GET_LOGS.EXE utility
- SET_SIZE.EXE utility

please see *Modicon Starling Associates Gas Flow Loadable Function Block User Guide, 890 USE 137 00, Version 2.0*. The *User Guide* is available in PDF format and can be accessed through Schneider Electric's Web site.

The following instructions assume that you are using a Windows-based PC and a mouse with left and right buttons. Browser response should be similar whether you are using *Internet Explorer* or *Navigator*.

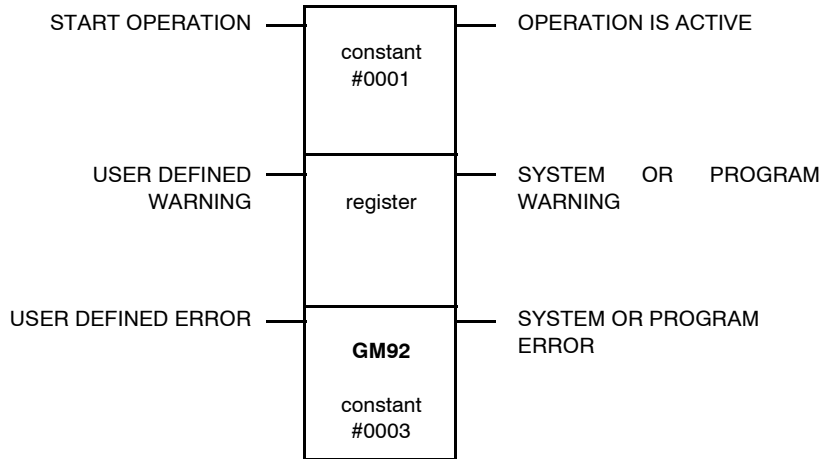
To access the Web site and view the *User Guide* follow these steps:

1. Enter this URL into your browser:
<http://www.schneider-automation.com>
 2. When the home page appears, type the following term in the **Search** field:
"gas loadable"
 3. Execute the search.
 4. The *User Guide*'s link should display in the list that appears in the search results.
 5. Double click on the title/link.
A second window appears.
 6. In the **Attachment:** field of the second, smaller window right-click on **890USE13700.pdf (8036.5K)**
 7. From the shortcut menu select **Save Target As...**
 8. Save the **.pdf** file to your desktop or a folder.
 9. The *User Guide* should open with *Adobe Acrobat Reader*.
-

Representation: GM92 - Gas Flow Function Block

Symbol

Representation of the instruction



**Parameter
Description**

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = solving This input starts the calculation of the gas flow. The calculations are based on your parameters entered into the input registers. Important: Never detach the top input while the block is running. You will generate an error 188 and the data in this block could be corrupted. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 565.</i>)
Middle input	0x, 1x	None	Allows you to set a warning. Allows you to set a warning and log peripheral activities in the audit trail event log without stopping the block. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 565.</i>)
Bottom input	0x, 1x	None	Allows you to set an error and STOP the flow function. Allows you to set an error, log peripheral errors in the audit trail event log, and STOP the flow function. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 565.</i>)
constant #0001 (top node)	4x	INT, UINT	The top node must contain a constant, #0001.
register (middle node)	4x	INT, UINT	The 4x register entered in the middle node is the first in a group of contiguous holding registers that comprise the configuration parameters and values associated with the Gas Flow Block. Important: Do not attempt to change the middle node 4x register while the Gas Flow Block is running. You will lose your data. If you need to change the 4x register, first STOP the PLC.
#0003 (bottom node)		INT, UINT	The bottom node specifies the calculation type and must contain a constant, #0003.
Top output	0x	None	ON = Operation successful
Middle output	0x	None	ON = System or program warning
Bottom output	0x	None	ON = System or program error

Parameter Description - Inputs: GM92 - Gas Flow Function Block

Configuration Table

You MUST fill in all pertinent values in the configuration table using the reference data editor either in ProWORX or Concept, or the DX Zoom screens in Modsoft, or Meter Manager. The following inputs table lists all the configuration parameters that MUST be filled in.

The outputs (Outputs Results Table) and the optional outputs (Optional Outputs Results Table) show the calculation results of the block. Some of those parameters are required.

Important: Only valid entries are allowed. Entries outside the valid ranges are not accepted. Illegal entries result in errors or warnings.

Important: Concept 2.1 or higher may be used to load the Gas blocks. However, Concept and ProWORX do NOT provide help or DX zoom screens for configuration. When using Concept or ProWORX panel software we recommend you use Meter Manager for your configuration needs.

Inputs

The following is a detailed description of configuration variables for the GD92 gas flow function block.

Inputs	Description
4xxxx+3: 1 through 2	Location of Taps 1 - Upstream 2 - Downstream
4xxxx+3: 3 through 4	Meter Tube Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 5 through 6	Orifice Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 7 through 8	Reserved for Future Use (Do not use)
4xxxx+3: 9 through 10	Optional Outputs 1 - Yes 2 - No Note: When using only the standard outputs, the loadable uses 157 4xxxx registers. When using the optional outputs, the loadable uses 181 4xxxx registers.
4xxxx+3: 11 through 16	Reserved for Future Use (Do not use)
4xxxx+4: 1	Absolute/Gauge Pressure 0 - Static Pressure Measured in Absolute Units 1 - Static Pressure Measured in Gauge Units

Inputs	Description
4xxxx+4: 2	Low Flow Cut Off 0 - Do Not Use Flow Cut Off 1 - Use Flow Cut Off
4xxxx+4: 3 through 6	Load Command 0 - Ready to Accept Command 1 - CMD: Send Configuration to Internal Table from 4xxxx 2 - CMD: Read Configuration from Internal Table to 4xxxx 3 - CMD: Reset API 21.1 configuration change log
4xxxx+4: 7 through 8	Input Type 1 - 3xxxx Pointers entered in 4x+6 ... 4x+10 2 - Input Values entered in 4x+6 ... 4x+10
4xxxx+4: 9 through 10	Mole % Error Limits 1 - Enable 2 - Disable
4xxxx+4: 11 through 12	Dual Range Differential Pressure Option 1 - Yes 2 - No
4xxxx+4: 13 through 14	Compressible/Incompressible 1 - Compressible 2 - Incompressible
4xxxx+4: 15 through 16	Averaging Methods 0 - Flow Dependent Time Weighted Linear 1 - Flow Dependent Time Weighted Formulaic 2 - Flow Weighted Linear 3 - Flow Weighted Formulaic Note: For most applications you will use 0.
4xxxx+5: 1 through 2	Measurement Units 1 - US 2 - Metric (SI)
4xxxx+5: 3 through 14	Reserved for Future Use (Do not use)
4xxxx+5: 15 through 16	Reserved for API 21.1
4xxxx+6	Temperature 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+7	Pressure (absolute) 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+8	Differential Pressure 1 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+9	Differential Pressure 2 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+10	Analog Input Raw Value Minimum Temperature Data type: Unsigned integer value

Inputs	Description
4xxxx+11	Analog Input Raw Value Maximum Temperature Data type: Unsigned integer value
4xxxx+12	Analog Input Raw Value Minimum Pressure Data type: Signed integer value
4xxxx+13	Analog Input Raw Value Maximum Pressure Data type: Signed integer value
4xxxx+14	Analog Input Raw Value Minimum Differential Pressure 1 Data type: Signed integer value
4xxxx+15	Analog Input Raw Value Maximum Differential Pressure 1 Data type: Signed integer value
4xxxx+16	Analog Input Raw Value Minimum Differential Pressure 2 Data type: Signed integer value
4xxxx+17	Analog Input Raw Value Maximum Differential Pressure 2 Data type: Signed integer value
4xxxx+18 through 19	Engineering Unit Temperature Minimum -200 through 760°F (-128.89 through 404.4°C) Data type: Floating point number
4xxxx+20 through 21	Engineering Unit Temperature Maximum -200 through 760°F (-128.89 through 404.4°C) Data type: Floating point number
4xxxx+22 through 23	Engineering Unit Pressure Minimum 0 through 40,000psia (0 through 275,790.28kPa) Data type: Floating point number
4xxxx+24 through 25	Engineering Unit Pressure Maximum 0 through 40,000psia (0 through 275,790.28kPa) Data type: Floating point number
4xxxx+26 through 27	Engineering Unit Differential Pressure 1 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+28 through 29	Engineering Unit Differential Pressure 1 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+30 through 31	Engineering Unit Differential Pressure 2 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+32 through 33	Engineering Unit Differential Pressure 2 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+34 through 35	Orifice Plate Diameter, d_r (0 < d_r < 100in) (0 < d_r < 2540mm) Data type: Floating point number

Inputs	Description
4xxxx+36 through 37	Orifice Plate Diameter Measurement Temperature, T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+38 through 39	Meter Tube Internal Diameter D_r (0 < D_r < 100in) (0 < D_r < 2540mm) Data type: Floating point number
4xxxx+40 through 41	Measured Meter Tube Internal Diameter Temperature T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+42 through 43	Base Temperature, T_b (32.0 ≤ T_b < 77.0°F) (0 ≤ T_b < 25°C) Data type: Floating point number
4xxxx+44 through 45	Base Pressure, P_b (13.0 ≤ P_b < 16.0PSIA) (89.63 ≤ P_b < 110.32kPa) Data type: Floating point number
4xxxx+46 through 47	Reference Temperature for Relative Density, T_{gr} (32.0 ≤ T_{gr} < 77.0°F) (0 ≤ T_{gr} < 25°C) Data type: Floating point number
4xxxx+48 through 49	Reference Pressure for Relative Density, P_{gr} (13.0 ≤ P_{gr} < 16.0PSIA) (89.63 ≤ P_{gr} < 110.32kPa) Data type: Floating point number
4xxxx+50 through 57	Reserved for Future Use (Do not use)
4xxxx+58 through 59	User Input Correction Factor, F_u (0 < F_u < 2.0) Data type: Floating point number
4xxxx+60 through 61	Absolute Viscosity of Flowing Fluid, μ_c (0.005 ≤ μ_c ≤ 0.5 centipoise) Data type: Floating point number
4xxxx+62 through 63	Isentropic Exponent, k (1.0 ≤ k < 2.0) Data type: Floating point number
4xxxx+64	Beginning of Day Hour (0 ... 23) Data type: Unsigned integer value
4xxxx+65 through 78	Reserved for API 21.1
4xxxx+79 through 80	Atmospheric Pressure P_{at} (3 ≤ P_{at} < 30psi) (20.684 ≤ P_{at} < 206.843kPa) Data type: Floating point number

Inputs	Description
4xxxx+81 through 82	Low Flow Cut Off Level ($\geq 0\text{ft}^3/\text{Hr}$) ($\geq 0\text{m}^3/\text{Hr}$) Used if enabled in 4x+4: 2. Data type: Floating point number
4xxxx+83 through 84	Mole % of Methane, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+85 through 86	Mole % of Nitrogen, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+87 through 88	Mole % of Carbon Dioxide, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+89 through 90	Mole % of Ethane, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
xxx+91 through 92	Mole % of Propane, x_i $*(0.0 \leq x_i \leq 12)$ Data type: Floating point number
4xxxx+93 through 94	Mole % of Water, x_i $*(0.0 \leq x_i \leq 10)$ Data type: Floating point number
4xxxx+95 through 96	Mole % of Hydrogen Sulfide, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+97 through 98	Mole % of Hydrogen, x_i $*(0.0 \leq x_i \leq 100)$ Data type: Floating point number
4xxxx+99 through 100	Mole % of Carbon Monoxide, x_i $*(0.0 \leq x_i \leq 3)$ Data type: Floating point number
4xxxx+101 through 102	Mole % of Oxygen, x_i $*(0.0 \leq x_i \leq 21)$ Data type: Floating point number
4xxxx+103 through 104	Mole % of I-Butane, x_i $*(0.0 \leq x_i \leq 6)$ for combined butanes Data type: Floating point number

Inputs	Description
4xxxx+105 through 106	Mole % of n-Butane, x_i *(0.0 <= x_i <= 6) for combined butanes Data type: Floating point number
4xxxx+107 through 108	Mole % of I-Pentane, x_i *(0.0 <= x_i <= 4) for combined pentanes Data type: Floating point number
4xxxx+109 through 110	Mole % of n-Pentane, x_i *(0.0 <= x_i <= 4) for combined pentanes Data type: Floating point number
4xxxx+111 through 112	Mole % of Hexane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+113 through 114	Mole % of Heptane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+115 through 116	Mole % of Octane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+117 through 118	Mole % of Nonane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+119 through 120	Mole % of Decane, x_i *(0.0 <= x_i <= 10) for combined hexanes + Data type: Floating point number
4xxxx+121 through 122	Mole % of Helium, x_i *(0.0 <= x_i <= 30) Data type: Floating point number
4xxxx+123 through 124	Mole % of Argon, x_i *(0.0 <= x_i <= 100) Data type: Floating point number

*Valid range

Parameter Description - Outputs: GM92 - Gas Flow Function Block

Outputs Results Table

The outputs show the calculation results of the block.

Outputs	Description
4xxxx+0	System Warning/Error Code (Displayed in Hex mode)
4xxxx+1	Program Warning/Error Code
4xxxx+2	Version Number (Displayed in Hex mode)
4xxxx+125 through 126	Temperature at Flowing Conditions (T_f) (F or C)
4xxxx+127 through 128	Pressure (P_f) (psia or kPa)
4xxxx+129 through 130	Differential Pressure (h_w) (in H ₂ O or kPa)
4xxxx+131 through 132	Integral Value (IV)
4xxxx+133 through 134	Integral Multiplier Value (IMV)
4xxxx+135 through 136	Volume Flow Rate at Base Conditions (T_b , P_b), Q_b (ft ³ /hr or m ³ /hr)
4xxxx+137 through 138	Mass Flow Rate (Q_m) (lbm/hr or Kg/hr)
4xxxx+139 through 140	Accumulated Volume Current Day
4xxxx+141 through 142	Accumulated Volume Last Hour
4xxxx+143 through 144	Accumulated Volume Last Day
4xxxx+145 through 152	Reserved for API 21.1
4xxxx+153	User definable warning/error value (Use for API 21.1)
4xxxx+155: 13	4xxxx Table Differs from Actual Configuration
4xxxx+155: 14	Flow Rate Solve Complete Heartbeat
4xxxx+155: 15	Block is Functioning Heartbeat
4xxxx+155: 16	End of Day Flag

Parameter Description - Optional Outputs: GM92 - Gas Flow Function Block

Optional Outputs Configuration Table

The optional outputs show the calculation results of the block. These are only active if 4x+3: 9 ... 10 is 1.

Optional Outputs	Description
4xxxx+156 through 157	Compressibility at Flowing Conditions (T_f, P_f), Z_f
4xxxx+158 through 159	Compressibility at Base Conditions (T_b, P_b), Z_b
4xxxx+160 through 161	Compressibility at Standard Conditions (T_s, P_s), Z_s
4xxxx+162 through 163	Density at Fluid Flowing Conditions ($P_{t,p}$)
4xxxx+164 through 165	Density of Fluid at Base Conditions (ρ)
4xxxx+166 through 167	Supercompressibility (F_{pv})
4xxxx+168 through 169	Gas Relative Density (G_r)
4xxxx+170 through 171	Orifice Plate Coefficient of Discharge (C_d)
4xxxx+172 through 173	Expansion Factor (Y)
4xxxx+174 through 175	Velocity of Approach Factor (E_v)
4xxxx+176 through 177	Volume Flow Rate at Flowing Conditions (T_f, P_f), Q_f
4xxxx+178 through 179	Reserved for Future Use (Do not use)
4xxxx+180	Orifice Plate Coefficient of Discharge Bounds Flag within Iteration Scheme (C_{d-f})

G392 AGA #3 1992 Gas Flow Function Block

90

At A Glance

Introduction

This chapter describes the instruction G392 AGA #3 1992 Gross Method with API 21.1 Audit Trail.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: G392 - Gas Flow Function Block	574
Representation: G392 - Gas Flow Function Block	575
Parameter Description - Inputs: G392 - Gas Flow Function Block	577
Parameter Description - Outputs: G392 - Gas Flow Function Block	582
Parameter Description - Optional Outputs: G392 - Gas Flow Function Block	583

Short Description: G392 - Gas Flow Function Block

Function Description

The G392 Gas Flow Loadable function block is available only on certain Compact and Micro controllers.

The Gas Flow Loadable Function Block allows you to run AGA 3 (1992) equations. The computed flow rates agree within 1 ppm of the published AGA standards.

Important: LSUP loadable must be installed before G392.

More Information

For detailed information about the Gas Flow Function Block loadables, especially the:

- System warning/error codes (4x+0) for each instruction
- Program warning/error codes (4x+1) for each instruction
- API 21.1 Audit Trail
- GET_LOGS.EXE utility
- SET_SIZE.EXE utility

please see *Modicon Starling Associates Gas Flow Loadable Function Block User Guide, 890 USE 137 00, Version 2.0*. The *User Guide* is available in PDF format and can be accessed through Schneider Electric's Web site.

The following instructions assume that you are using a Windows-based PC and a mouse with left and right buttons. Browser response should be similar whether you are using *Internet Explorer* or *Navigator*.

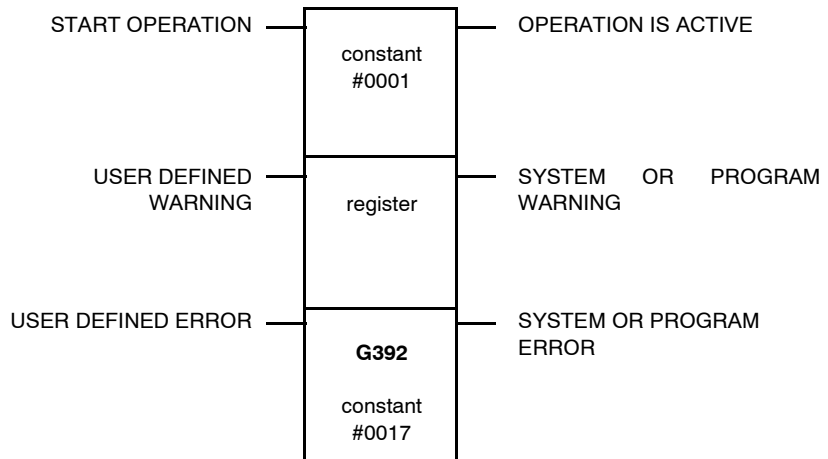
To access the Web site and view the *User Guide* follow these steps:

1. Enter this URL into your browser:
<http://www.schneider-automation.com>
 2. When the home page appears, type the following term in the **Search** field:
"gas loadable"
 3. Execute the search.
 4. The *User Guide*'s link should display in the list that appears in the search results.
 5. Double click on the title/link.
A second window appears.
 6. In the **Attachment:** field of the second, smaller window right-click on
890USE13700.pdf (8036.5K)
 7. From the shortcut menu select **Save Target As...**
 8. Save the **.pdf** file to your desktop or a folder.
 9. The *User Guide* should open with *Adobe Acrobat Reader*.
-

Representation: G392 - Gas Flow Function Block

Symbol

Representation of the instruction



**Parameter
Description**

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = solving This input starts the calculation of the gas flow. The calculations are based on your parameters entered into the input registers. Important: Never detach the top input while the block is running. You will generate an error 188 and the data in this block could be corrupted. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 577.</i>)
Middle input	0x, 1x	None	Allows you to set a warning. Allows you to set a warning and log peripheral activities in the audit trail event log without stopping the block. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 577.</i>)
Bottom input	0x, 1x	None	Allows you to set an error and STOP the flow function. Allows you to set an error, log peripheral errors in the audit trail event log, and STOP the flow function. Important: You MUST fill in all pertinent values in the configuration table. (For information about entering values, see <i>Configuration Table, p. 577.</i>)
constant #0001 (top node)	4x	INT, UINT	The top node must contain a constant, #0001.
register (middle node)	4x	INT, UINT	The 4x register entered in the middle node is the first in a group of contiguous holding registers that comprise the configuration parameters and values associated with the Gas Flow Block. Important: Do not attempt to change the middle node 4x register while the Gas Flow Block is running. You will lose your data. If you need to change the 4x register, first STOP the PLC.
#0017 (bottom node)		INT, UINT	The bottom node specifies the calculation type and must contain a constant, #0017.
Top output	0x	None	ON = Operation successful
Middle output	0x	None	ON = System or program warning
Bottom output	0x	None	ON = System or program error

Parameter Description - Inputs: G392 - Gas Flow Function Block

Configuration Table

You MUST fill in all pertinent values in the configuration table using the reference data editor either in ProWORX or Concept, or the DX Zoom screens in Modsoft, or Meter Manager. The following inputs table lists all the configuration parameters that MUST be filled in.

The outputs (Outputs Results Table) and the optional outputs (Optional Outputs Results Table) show the calculation results of the block. Some of those parameters are required.

Important: Only valid entries are allowed. Entries outside the valid ranges are not accepted. Illegal entries result in errors or warnings.

Important: Concept 2.1 or higher may be used to load the Gas blocks. However, Concept and ProWORX do NOT provide help or DX zoom screens for configuration. When using Concept or ProWORX panel software we recommend you use Meter Manager for your configuration needs.

Inputs

The following is a detailed description of configuration variables for the G392 gas flow function block.

Inputs	Description
4xxxx+3: 1 through 2	Location of Taps 1 - Upstream 2 - Downstream
4xxxx+3: 3 through 4	Meter Tube Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 5 through 6	Orifice Material 1 - Stainless Steel 2 - Monel 3 - Carbon Steel
4xxxx+3: 7 through 8	Compressibility User Input Type 1 - Density at flowing and base condition 2 - Compressibility factor at flowing and base conditions and gas relative density at base conditions
4xxxx+3: 9 through 10	Optional Outputs 1 - Yes 2 - No Note: When using only the standard outputs, the loadable uses 157 4xxxx registers. When using the optional outputs, the loadable uses 181 4xxxx registers.
4xxxx+3: 11 through 16	Reserved for Future Use (Do not use)

Inputs	Description
4xxxx+4: 1	Absolute/Gauge Pressure 0 - Static Pressure Measured in Absolute Units 1 - Static Pressure Measured in Gauge Units
4xxxx+4: 2	Low Flow Cut Off 0 - Do Not Use Flow Cut Off 1 - Use Flow Cut Off
4xxxx+4: 3 through 6	Load Command 0 - Ready to Accept Command 1 - CMD: Send Configuration to Internal Table from 4xxxx 2 - CMD: Read Configuration from Internal Table to 4xxxx 3 - CMD: Reset API 21.1 configuration change log
4xxxx+4: 7 through 8	Input Type 1 - 3xxxx Pointers entered in 4x+6 ... 4x+10 2 - Input Values entered in 4x+6 ... 4x+10
4xxxx+4: 9 through 10	Reserved for Future Use (Do not use)
4xxxx+4: 11 through 12	Dual Range Differential Pressure Option 1 - Yes 2 - No
4xxxx+4: 13 through 14	Compressible/Incompressible 1 - Compressible 2 - Incompressible
4xxxx+4: 15 through 16	Averaging Methods 0 - Flow Dependent Time Weighted Linear 1 - Flow Dependent Time Weighted Formulaic 2 - Flow Weighted Linear 3 - Flow Weighted Formulaic Note: For most applications you will use 0.
4xxxx+5: 1 through 2	Measurement Units 1 - US 2 - Metric (SI)
4xxxx+5: 3 through 14	Reserved for Future Use (Do not use)
4xxxx+5: 15 through 16	Reserved for API 21.1
4xxxx+6	Temperature 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+7	Pressure (absolute) 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+8	Differential Pressure 1 3xxxx Pointer or Input Value Data type: Unsigned integer value
4xxxx+9	Differential Pressure 2 3xxxx Pointer or Input Value Data type: Unsigned integer value

Inputs	Description
4xxxx+10	Analog Input Raw Value Minimum Temperature Data type: Unsigned integer value
4xxxx+11	Analog Input Raw Value Maximum Temperature Data type: Unsigned integer value
4xxxx+12	Analog Input Raw Value Minimum Pressure Data type: Signed integer value
4xxxx+13	Analog Input Raw Value Maximum Pressure Data type: Signed integer value
4xxxx+14	Analog Input Raw Value Minimum Differential Pressure 1 Data type: Signed integer value
4xxxx+15	Analog Input Raw Value Maximum Differential Pressure 1 Data type: Signed integer value
4xxxx+16	Analog Input Raw Value Minimum Differential Pressure 2 Data type: Signed integer value
4xxxx+17	Analog Input Raw Value Maximum Differential Pressure 2 Data type: Signed integer value
4xxxx+18 through 19	Engineering Unit Temperature Minimum -200 through 760°F (-128.89 through 404.4°C) Data type: Floating point number
4xxxx+20 through 21	Engineering Unit Temperature Maximum -200 through 760°F (-128.89 through 404.4°C) Data type: Floating point number
4xxxx+22 through 23	Engineering Unit Pressure Minimum 0 through 40,000psia (0 through 275,790.28kPa) Data type: Floating point number
4xxxx+24 through 25	Engineering Unit Pressure Maximum 0 through 40,000psia (0 through 275,790.28kPa) Data type: Floating point number
4xxxx+26 through 27	Engineering Unit Differential Pressure 1 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+28 through 29	Engineering Unit Differential Pressure 1 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+30 through 31	Engineering Unit Differential Pressure 2 Minimum >= 0 (inches H2O or kPa) Data type: Floating point number

Inputs	Description
4xxxx+32 through 33	Engineering Unit Differential Pressure 2 Maximum > 0 (inches H2O or kPa) Data type: Floating point number
4xxxx+34 through 35	Orifice Plate Diameter, d_r (0 < d_r < 100in) (0 < d_r < 2540mm) Data type: Floating point number
4xxxx+36 through 37	Orifice Plate Diameter Measurement Temperature, T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+38 through 39	Meter Tube Internal Diameter D_r (0 < D_r < 100in) (0 < D_r < 2540mm) Data type: Floating point number
4xxxx+40 through 41	Measured Meter Tube Internal Diameter Temperature T_r (32 ≤ T_r < 77°F) (0 ≤ T_r < 25°C) Data type: Floating point number
4xxxx+42 through 43	Base Temperature, T_b (32.0 ≤ T_b < 77.0°F) (0 ≤ T_b < 25°C) Data type: Floating point number
4xxxx+44 through 45	Base Pressure, P_b (13.0 ≤ P_b < 16.0PSIA) (89.63 ≤ P_b < 110.32kPa) Data type: Floating point number
4xxxx+46 through 57	Reserved for Future Use (Do not use)
4xxxx+58 through 59	User Input Correction Factor, F_u (0 < F_u < 2.0) Data type: Floating point number
4xxxx+60 through 61	Absolute Viscosity of Flowing Fluid, μ_c (0.005 ≤ μ_c ≤ 0.5 centipoise) Data type: Floating point number
4xxxx+62 through 63	Isentropic Exponent, k (1.0 ≤ k < 2.0) Data type: Floating point number
4xxxx+64	Beginning of Day Hour (0 ... 23) Data type: Unsigned integer value
4xxxx+65 through 78	Reserved for API 21.1 configuration
4xxxx+79 through 80	Atmospheric Pressure P_{at} (3 ≤ P_{at} < 30psi) (20.684 ≤ P_{at} < 206.843kPa) Data type: Floating point number

Inputs	Description
4xxxx+81 through 82	Low Flow Cut Off Level ($\geq 0\text{ft}^3/\text{Hr}$) ($\geq 0\text{m}^3/\text{Hr}$) Used if enabled in 4x+4: 2. Data type: Floating point number
4xxxx+83 through 84	Density at Flowing Conditions, ρ_f ($0 < \rho_f < 1000.0\text{lbm}/\text{ft}^3$) ($0 < \rho_f < 1601.846\text{kg}/\text{m}^3$) Data type: Floating point number
4xxxx+85 through 86	Density at Base Conditions, ρ_b ($0 < \rho_b < 100.0\text{lbm}/\text{ft}^3$) ($0 < \rho_b < 1601.846\text{kg}/\text{m}^3$) Data type: Floating point number
4xxxx+87 through 88	Compressibility Factor at Flowing Conditions, Z_f ($0 < Z_f < 3$) Data type: Floating point number
4xxxx+89 through 90	Compressibility Factor at Base Conditions, Z_b ($0 < Z_b < 3$) Data type: Floating point number
xxx+91 through 92	Gas Relative Density at Base Conditions, Gr ($0.07 \leq Gr < 1.52$) Data type: Floating point number
4xxxx+93 through 124	Reserved for Future Use (Do not use)

Parameter Description - Outputs: G392 - Gas Flow Function Block

Outputs Results Table

The outputs show the calculation results of the block.

Outputs	Description
4xxx+0	System Warning/Error Code (Displayed in Hex mode)
4xxx+1	Program Warning/Error Code
4xxx+2	Version Number (Displayed in Hex mode)
4xxx+125 through 126	Temperature at Flowing Conditions (T_f) ((F or C)
4xxx+127 through 128	Pressure (P_f) (psia or kPa)
4xxx+129 through 130	Differential Pressure (h_w) (in H ₂ O or kPa)
4xxx+131 through 132	Integral Value (IV)
4xxx+133 through 134	Integral Multiplier Value (IMV)
4xxx+135 through 136	Volume Flow Rate at Base Conditions (T_b , P_b), Q_b (ft ³ /hr or m ³ /hr)
4xxx+137 through 138	Mass Flow Rate (Q_m)
4xxx+139 through 140	Accumulated Volume Current Day
4xxx+141 through 142	Accumulated Volume Last Hour
4xxx+143 through 144	Accumulated Volume Last Day
4xxx+145 through 152	Reserved for API 21.1
4xxx+153	User definable warning/error value (Use for API 21.1)
4xxx+155: 13	4xxx Table Differs from Actual Configuration
4xxx+155: 14	Flow Rate Solve Complete Heartbeat
4xxx+155: 15	Block is Functioning Heartbeat
4xxx+155: 16	End of Day Flag

Parameter Description - Optional Outputs: G392 - Gas Flow Function Block

Optional Outputs Configuration Table

The optional outputs show the calculation results of the block. These are only active if 4x+3: 9 ... 10 is 1.

Optional Outputs	Description
4xxxx+156 through 157	Compressibility at Flowing Conditions (T_f, P_f), Z_f
4xxxx+158 through 159	Compressibility at Base Conditions (T_b, P_b), Z_b
4xxxx+160 through 161	Reserved for Future Use (Do not use)
4xxxx+162 through 163	Density at Fluid Flowing Conditions ($P_{t,p}$)
4xxxx+164 through 165	Density of Fluid at Base Conditions (ρ)
4xxxx+166 through 167	Supercompressibility (F_{pv})
4xxxx+168 through 169	Gas Relative Density (G_r)
4xxxx+170 through 171	Orifice Plate Coefficient of Discharge (C_d)
4xxxx+172 through 173	Expansion Factor (Y)
4xxxx+174 through 175	Velocity of Approach Factor (E_v)
4xxxx+176 through 177	Volume Flow Rate at Flowing Conditions (T_f, P_f), Q_f
4xxxx+178 through 179	Reserved for Future Use (Do not use)
4xxxx+180	Orifice Plate Coefficient of Discharge Bounds Flag within Iteration Scheme (C_{d-t})

HLTH: History and Status Matrices

91

At a Glance

Introduction

This chapter describes the instruction HLTH.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	586
Representation: HLTH - System Health	587
Parameter Description	588
Parameter Description Top Node (History Matrix)	589
Parameter Description Middle Node (Status Matrix)	594
Parameter Description Bottom Node (Length)	599

Short Description

Function Description

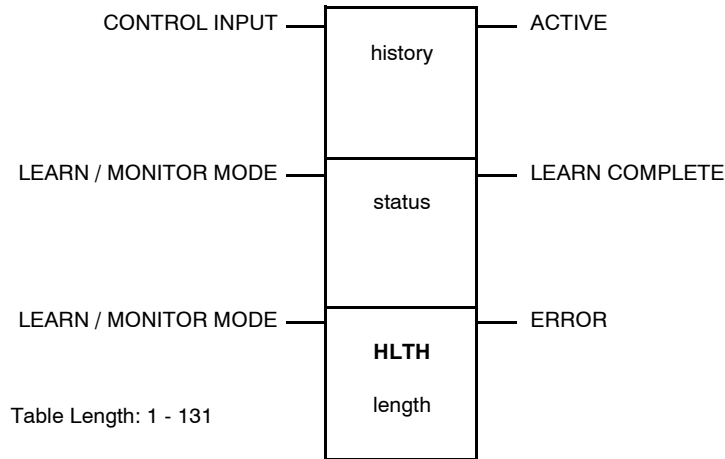
Note: This instruction is only available if you have unpacked and installed the DX Loadables. For further information, see *Installation of DX Loadables*, p. 109.

The HLTH instruction creates history and status matrices from internal memory registers that may be used in ladder logic to detect changes in PLC status and communication capabilities with the I/O. It can also be used to alert the user to changes in a PLC System. HLTH has two modes of operation, (learn) and (monitor).

Representation: HLTH - System Health

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON initiates the designated operation
Middle input	0x, 1x	None	Learn / monitor mode (For detailed information, see <i>Learn / Monitor Mode (Middle and Bottom Input)</i> , p. 588.)
Bottom input	0x, 1x	None	Learn / monitor mode (For expanded and detailed information, see <i>Learn / Monitor Mode (Middle and Bottom Input)</i> , p. 588.)
history (top node)	4x	INT, UINT, WORD	History matrix (first in a block of contiguous registers, range: 6 ... 135)
status (middle node)	4x	INT, UINT, WORD	Status matrix (first in a block of contiguous registers, range: 3 ... 132)
length (bottom node)		INT, UINT	length = (number of RIO drops x 4) + 3
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	Echoes state of the middle input
Bottom output	0x	None	ON = Error

Parameter Description

Modes of operation

The HLTH instruction has two modes of operation:

Type of Mode	Meaning
Learn Mode	<p>HLTH can be initialized to learn the configuration in which it is implemented and save the information as a point-in-time reference called history matrix. This matrix contains:</p> <ul style="list-style-type: none"> ● A user-designated drop number for communications status monitoring ● User logic checksum ● Disabled I/O indicator ● S911 Health ● Choice of single or dual cable system ● I/O Map display
Monitor Mode	<p>Monitor mode enables an operation that checks PLC system conditions. Detected changes are recorded in a status matrix., which monitors the most recent system conditions and sets bit patterns to indicate detected changes. The status matrix contains:</p> <ul style="list-style-type: none"> ● Communication status of the drop designated in the history matrix ● A flag to indicate when there is any disabled I/O ● Flags to indicate the "on/off" status of constant sweep and the Memory protect key switch ● Flags to indicate a battery-low condition and if Hot Standby is functional ● Failed module position data ● Changed user logic checksum flag ● RIO lost-communication flag

Learn / Monitor Mode (Middle and Bottom Input)

The HLTH instruction block has three control inputs and can produce three possible outputs.

The combined states of the middle and bottom inputs control the operating mode:

Middle Input	Bottom Input	Operation
ON	OFF	Learn Mode as Dual Cable System
ON	ON	Learn Mode as Single Cable System
OFF	ON	Monitor Mode
OFF	OFF	Monitor Mode Update Logic Checksum

Parameter Description Top Node (History Matrix)

History Matrix (Top Node)

The 4x register entered in the top node is the first in a block of contiguous registers that comprise the history matrix. The data for the history matrix is gathered by the instruction during a learn mode operation and is set in the matrix when the mode changes to monitor.

The history matrix can range from 6 ... 135 registers in length. Below is a description of the words in the history matrix. The information from word 1 is contained in the displayed register in the top node and the information from words 2 ... 135 is stored in the implied registers.

Word 1

Enter drop number (range 0 ... 32) to be monitored for retries

Word 2

High word of learned checksum

Word 3

Low word of learned checksum

Word 4

The status and a counter for multiplexing the inputs. HLTH processes 16 words of input (256 inputs) per scan. This word holds the last word location of the last scan. The register is overwritten on every scan. The value in the counter portion of the word increases to the maximum number of inputs, then restarts at 0.

Usage of word 4:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	1 = at least one disabled input has been found
2 - 16	Count of the number of word checked for disabled inputs prior to this scan.

Word 5

Status and a counter for multiplexing outputs to detect if one is disabled. HLTH looks at 16 words (256 outputs) per scan to find one that is disabled. It holds the last word location of the last scan. The block is overwritten on every scan. The value in the counter portion increases to maximum outputs then restarts at 0.

Usage of word 5:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	1 = at least one disabled output has been found.
2 - 16	Count of the number of word checked for disabled outputs prior to this scan.

Word 6

Hot Standby cable learned data

Usage of word 6:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	1 = S911 present during learn.
2 - 8	Not used
9	1 = cable A is monitored.
10	1 = cable B is monitored.
11 - 16	Not used

Word 7 ... 134

These words define the learned condition of drop 1 to drop 32 as follows:

Word	Drop No.
7 ... 10	1
11 ... 14	2
15 ... 18	3
:	:
:	:
131 ... 134	32

The structure of the four words allocated to each drop are as follows:

First Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Drop delay bit 1 Note: Drop delay bits are used by the software to delay the monitoring of the drop for four scans after reestablishing communications with a drop. The delay value is for internal use only and needs no user intervention.
2	Drop delay bit 2
3	Drop delay bit 3
4	Drop delay bit 4
5	Drop delay bit 5
6	Rack 1, slot 1, module found
7	Rack 1, slot 2, module found
8	Rack 1, slot 3, module found
9	Rack 1, slot 4, module found
10	Rack 1, slot 5, module found
11	Rack 1, slot 6, module found
12	Rack 1, slot 7, module found
13	Rack 1, slot 8, module found
14	Rack 1, slot 9, module found
15	Rack 1, slot 10, module found
16	Rack 1, slot 11, module found

Second Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 2, slot 1, module found
2	Rack 2, slot 2, module found
3	Rack 2, slot 3, module found
4	Rack 2, slot 4, module found
5	Rack 2, slot 5, module found
6	Rack 2, slot 6, module found
7	Rack 2, slot 7, module found
8	Rack 2, slot 8, module found
9	Rack 2, slot 9, module found
10	Rack 2, slot 10, module found
11	Rack 2, slot 11, module found
12	Rack 3, slot 1, module found
13	Rack 3, slot 2, module found
14	Rack 3, slot 3, module found
15	Rack 3, slot 4, module found
16	Rack 3, slot 5, module found

Third Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 3, slot 6, module found
2	Rack 3, slot 7, module found
3	Rack 3, slot 8, module found
4	Rack 3, slot 9, module found
5	Rack 3, slot 10, module found
6	Rack 3, slot 11, module found
7	Rack 4, slot 1, module found
8	Rack 4, slot 2, module found
9	Rack 4, slot 3, module found
10	Rack 4, slot 4, module found
11	Rack 4, slot 5, module found
12	Rack 4, slot 6, module found
13	Rack 4, slot 7, module found
14	Rack 4, slot 8, module found
15	Rack 4, slot 9, module found
16	Rack 4, slot 10, module found

Fourth Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 4, slot 11, module found
2	Rack 5, slot 1, module found
3	Rack 5, slot 2, module found
4	Rack 5, slot 3, module found
5	Rack 5, slot 4, module found
6	Rack 5, slot 5, module found
7	Rack 5, slot 6, module found
8	Rack 5, slot 7, module found
9	Rack 5, slot 8, module found
10	Rack 5, slot 9, module found
11	Rack 5, slot 10, module found
12	Rack 5, slot 11, module found
13 ... 16	not used

Parameter Description Middle Node (Status Matrix)

Status Matrix (Middle Node)

The 4x register entered in the middle node is the first in a block of contiguous holding registers that will comprise the status matrix. The status matrix is updated by the HLTH instruction during monitor mode (top input is ON and middle input is OFF). The status matrix can range from 3 ... 132 registers in length. Below is a description of the words in the status matrix. The information from word 1 is contained in the displayed register in the middle node and the information from words 2 ... 131 is stored in the implied registers.

Word 1

This word is a counter for lost-communications at the drop being monitored. Usage of word 1:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 8	Indicates the number of the drop being monitored (0 ... 32).
9 - 16	Count of the lost communication incidents (0 ... 15).

Word 2

This word is the cumulative retry counter for the drop being monitored (the drop number is indicated in the high byte of word 1).

Usage of word 2:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 4	Not used
5 - 16	Cumulative retry count (0 ... 255).

Word 3

This word updates PLC status (including Hot Standby health) on every scan.
Usage of word 3:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	ON = all drops are not communicating.
2	Not used
3	ON = logic checksum has changed since last learn.
4	ON = at least one disabled 1x input detected.
5	ON = at least one disabled 0x output detected.
6	ON = constant sweep enabled.
7 - 10	Not used
11	ON = memory protect is OFF.
12	ON = battery is bad.
13	ON = an S911 is bad.
14	ON = Hot Standby not active.
15 - 16	Not used

Word 4 ... 131

These words indicate the status of drop 1 to drop 32 as follows:

Word	Drop No.
4 ... 7	1
8 ... 11	2
12 ... 15	3
:	:
:	:
128 ... 131	32

The structure of the four words allocated to each drop is as follows:

First Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Drop communication fault detected
2	Rack 1, slot 1, module fault
3	Rack 1, slot 2, module fault
4	Rack 1, slot 3, module fault
5	Rack 1, slot 4, module fault
6	Rack 1, slot 5, module fault
7	Rack 1, slot 6, module fault
8	Rack 1, slot 7, module fault
9	Rack 1, slot 8, module fault
10	Rack 1, slot 9, module fault
11	Rack 1, slot 10, module fault
12	Rack 1, slot 11, module fault
13	Rack 2, slot 1, module fault
14	Rack 2, slot 2, module fault
15	Rack 2, slot 3, module fault
16	Rack 2, slot 4, module fault

Second Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 2, slot 5, module fault
2	Rack 2, slot 6, module fault
3	Rack 2, slot 7, module fault
4	Rack 2, slot 8, module fault
5	Rack 2, slot 9, module fault
6	Rack 2, slot 10, module fault
7	Rack 2, slot 11, module fault
8	Rack 3, slot 1, module fault
9	Rack 3, slot 2, module fault
10	Rack 3, slot 3, module fault
11	Rack 3, slot 4, module fault
12	Rack 3, slot 5, module fault
13	Rack 3, slot 6, module fault
14	Rack 3, slot 7, module fault
15	Rack 3, slot 8, module fault
16	Rack 3, slot 9, module fault

Third Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 3, slot 10, module fault
2	Rack 3, slot 11, module fault
3	Rack 4, slot 1, module fault
4	Rack 4, slot 2, module fault
5	Rack 4, slot 3, module fault
6	Rack 4, slot 4, module fault
7	Rack 4, slot 5, module fault
8	Rack 4, slot 6, module fault
9	Rack 4, slot 7, module fault
10	Rack 4, slot 8, module fault
11	Rack 4, slot 9, module fault
12	Rack 4, slot 10, module fault
13	Rack 4, slot 11, module fault
14	Rack 5, slot 1, module fault
15	Rack 5, slot 2, module fault
16	Rack 5, slot 3, module fault

Fourth Word

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1	Rack 5, slot 4, module fault
2	Rack 5, slot 5, module fault
3	Rack 5, slot 6, module fault
4	Rack 5, slot 7, module fault
5	Rack 5, slot 8, module fault
6	Rack 5, slot 9, module fault
7	Rack 5, slot 10, module fault
8	Rack 5, slot 11, module fault
9	Cable A fault
10	Cable B fault
11 ... 16	not used

Parameter Description Bottom Node (Length)

Length (Bottom Node)

The decimal value entered in the bottom node is a function of how many I/O drops you want to monitor. Each drop requires four registers/matrix. The length value is calculated using the following formula:

$$\text{length} = (\# \text{ of I/O drops} \times 4) + 3$$

This value gives you the number of registers in the status matrix. You only need to enter this one value as the length because the length of the history matrix is automatically increased by 3 registers -i.e., the size of the history matrix is length + 3.

HSBY - Hot Standby

92

At A Glance

Introduction

This chapter describes the instruction HSBY.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: HSBY - Hot Standby	602
Representation: HSBY - Hot Standby	603
Parameter Description Top Node: HSBY - Hot Standby	605
Parameter Description Middle Node: HSBY - Hot Standby	606

Short Description: HSBY - Hot Standby

**Function
Description**

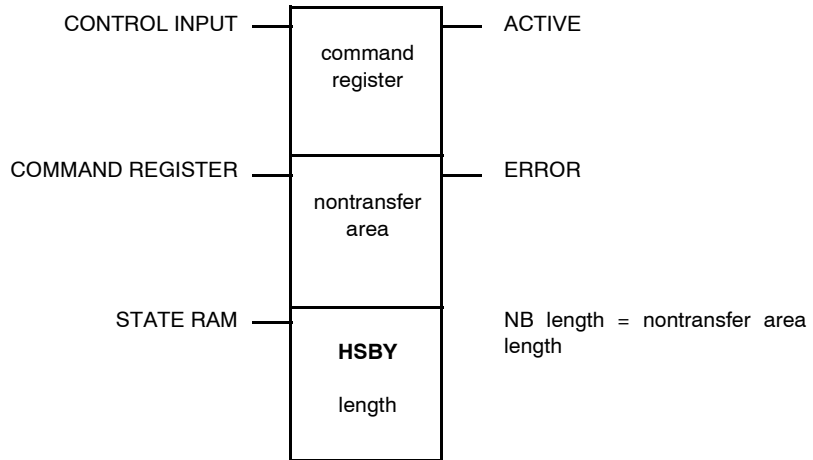
The HSBY loadable instruction manages a 984 Hot Standby control system. This instruction must be placed in network 1 of segment 1 in the application logic for both the primary and standby controllers. It allows you to program a nontransfer area in system state RAM—an area that protects a serial group of registers in the standby controller from being modified by the primary controller.

Through the HSBY instruction you can access two registers—a command register and a status register. Access allows you to monitor and control Hot Standby operations. The status register is the third register in the nontransfer area you specify.

Representation: HSBY - Hot Standby

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Execute HSBY (unconditionally) ON = function enabled
Middle input	0x, 1x	None	Enable command register ON = function enabled
Bottom input	0x, 1x	None	Enable nontransfer area ON = function enabled
command register (top node)	4x	INT, UINT	The 4xxxx register entered in the top node is the HSBY command register; eight bits in this register may be configured and controlled via your panel software. (For more information, see <i>Configuring the Register</i> , p. 605.)
nontransfer area (middle node)	4x	INT, UINT	The 4xxxx register entered in the middle node is the first register reserved for the nontransfer area in state RAM. The first three registers in the nontransfer area are special registers. (For more information, see <i>Nontransfer Area Special Registers</i> , p. 606 or <i>Application Specific Registers</i> , p. 606.)
length (bottom node)		INT, UINT	The integer value entered in the bottom node defines the length (the number of registers) of the HSBY nontransfer area in state RAM. The length must be at least four registers; in the range from 4 through 255 registers in a 16-bit CPU, and in the range of 4 through 8000 registers in a 24-bit CPU.
Top output	0x	None	Hot Standby system ACTIVE
Middle output	0x	None	PLC cannot communicate with its HSBY module

Parameter Description Top Node: HSBY - Hot Standby

Configuring the Register

You may configure bits six through eight and 12 through 16.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Follow these guidelines for configuring those bits.

Bit	Function
6	0 = Swap Modbus port 2 address during switchover 1 = Do not swap Modbus port 3 address during switchover
7	0 = Swap Modbus port 2 address during switchover 1 = Do not swap Modbus port 2 address during switchover
8	0 = Swap Modbus port 1 address during switchover 1 = Do not swap Modbus port q address during switchover
12	0 = Allow exec upgrade only after application stops 1 = Allow exec upgrade without stopping application
13	0 = Force standby offline if there is a logic mismatch 1 = Do not force standby offline if there is a logic mismatch
14	0 = Controller B in OFFLINE mode 1 = Controller B in RUN mode
15	0 = Controller A in OFFLINE mode 1 = Controller A in RUN mode
16	0 = Disable keyswitch override 1 = Enable keyswitch override

Parameter Description Middle Node: HSBY - Hot Standby

Nontransfer Area The first three registers in the nontransfer area are special registers.

Special Registers

Register	Content
Displayed and first implied	These two registers are <i>reverse transfer registers</i> for passing information from the standby to the primary PLC
Second implied	HSBY <i>status register</i>

Application Specific Registers

Bits 11 through 16 are application specific.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

The content of the remaining registers is application specific. The length is defined in the bottom node.

Bit	Function
11	0 = This PLC's switch set to A 1 = This PLC's switch set to B
12	0 = PLCs have matching logic 1 = PLCs do not have matching logic
13	0 1 = The other PLC in OFFLINE mode
14	1 0 = The other PLC running in primary mode 1 1 = The other PLC running in standby mode
15	0 1 = This PLC in OFFLINE mode
16	1 0 = This PLC running in primary mode 1 1 = This PLC running in standby mode

IBKR: Indirect Block Read

93

At a Glance

Introduction

This chapter describes the instruction IBKR.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	608
Representation: IBKR - Indirect Block Read	609

Short Description

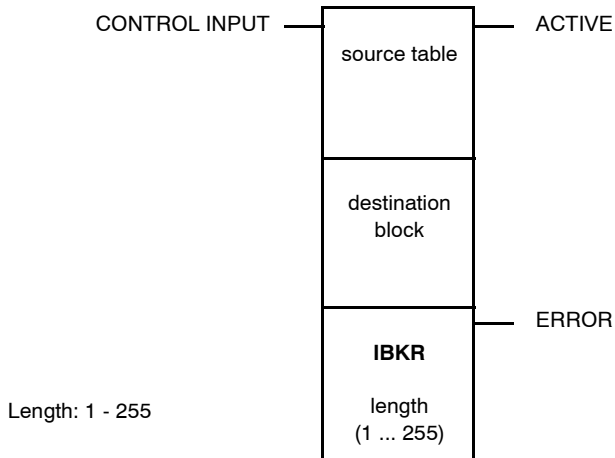
**Function
Description**

The IBKR (indirect block read) instruction lets you access non-contiguous registers dispersed throughout your application and copy the contents into a destination block of contiguous registers. This instruction can be used with subroutines or for streamlining data access by host computers or other PLCs.

Representation: IBKR - Indirect Block Read

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates indirect read operation
source table (top node)	4x	INT, UINT	First holding register in a source table: contain values that are pointers to the non-contiguous registers you want to collect in the operation.
destination block (middle node)	4x	INT, UINT	First in a block of contiguous destination registers, i.e. the block to which the source data will be copied.
length (1 ... 255) (bottom node)		INT, UINT	Number of registers in the source table and the destination block, range: 1 ... 255
Top output	0x	None	Echoes the state of the top input
Bottom output	0x	None	ON = error in source table

IBKW: Indirect Block Write

94

At a Glance

Introduction

This chapter describes the instruction IBKW.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	612
Representation: IBKW - Indirect Block Write	613

Short Description

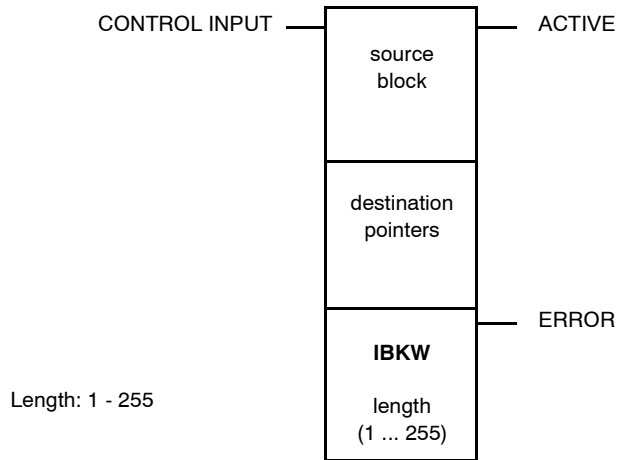
**Function
Description**

The IBKW (indirect block write) instruction lets you copy the data from a table of contiguous registers into several non-contiguous registers dispersed throughout your application.

Representation: IBKW - Indirect Block Write

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates indirect write operation
source block (top node)	4x	INT, UINT	First in a block of source registers: contain values that will be copied to non-contiguous registers dispersed throughout the logic program
destination pointers (middle node)	4x	INT, UINT	First in a block of contiguous destination pointer registers. Each of these registers contains a value that points to the address of a register where the source data will be copied.
length (1 ... 255) (bottom node)		INT, UINT	Number of registers in the source block and the destination pointer block, range: 1 ... 255
Top output	0x	None	Echoes the state of the top input
Bottom output	0x	None	ON = error in destination table

ICMP: Input Compare

95

At a Glance

Introduction

This chapter describes the instruction ICMP.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	616
Representation: ICMP - Input Compare	617
Parameter Description	618
Cascaded DRUM/ICMP Blocks	621

Short Description

Function Description

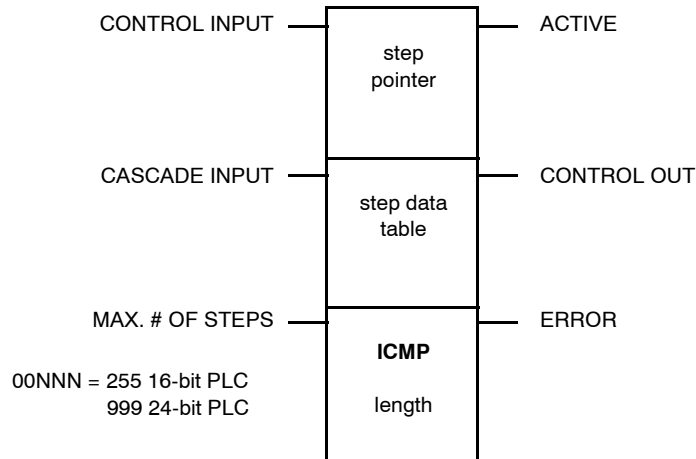
Note: This instruction is only available if you have unpacked and installed the DX Loadables. For further information, see *Installation of DX Loadables*, p. 109.

The ICMP (input compare) instruction provides logic for verifying the correct operation of each step processed by a DRUM instruction. Errors detected by ICMP may be used to trigger additional error-correction logic or to shut down the system. ICMP and DRUM are synchronized through the use of a common step pointer register. As the pointer increments, ICMP moves through its data table in lock step with DRUM. As ICMP moves through each new step, it compares-bit for bit-the live input data to the expected status of each point in its data table.

Representation: ICMP - Input Compare

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates the input comparison
Middle input	0x, 1x	None	A cascading input, telling the block that previous ICMP comparison were all good, ON = compare status is passing to the middle output
step pointer (top node)	4x	INT, UINT	Current step number
step data table (middle node)	4x	INT, UINT	First register in a table of step data information
length (bottom node)		INT, UINT	Number of application-specific registers-used in the step data table, range: 1 .. 999
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	ON =this comparison and all previous cascaded ICMPs are good
Bottom output	0x	None	ON = Error

Parameter Description

**Step Pointer
(Top Node)**

The 4x register entered in the top node stores the step pointer, i.e., the number of the current step in the step data table. This value is referenced by ICMP each time the instruction is solved. The value must be controlled externally by a DRUM instruction or by other user logic. The same register must be used in the top node of all ICMP and DRUM instructions that are solved as a single sequencer.

**Step Data Table
(Middle Node)**

The 4x register entered in the middle node is the first register in a table of step data information. The first eight registers in the table hold constant and variable data required to solve the instruction:

Register	Name	Content
Displayed	raw input data	Loaded by user from a group of sequential inputs to be used by ICMP for current step
First implied	current step data	Loaded by ICMP each time the block is solved; contains a copy of data in the step pointer; causes the block logic to automatically calculate register offsets when accessing step data in the step data table
Second implied	input mask	Loaded by user before using the block; contains a mask to be ANDed with raw input data for each step-masked bits will not be compared; masked data are put in the masked input data register
Third implied	masked input data	Loaded by ICMP each time the block is solved; contains the result of the ANDed input mask and raw input data
Fourth implied	compare status	Loaded by ICMP each time the block is solved; contains the result of an XOR of the masked input data and the current step data; unmasked inputs that are not in the correct logical state cause the associated register bit to go to 1-non-zero bits cause a miscompare, and middle output will not go ON
Fifth implied	machine ID number	Identifies DRUM/ICMP blocks belonging to a specific machine configuration; value range: 0 ... 9999 (0 = block not configured); all blocks belonging to same machine configuration have the same machine ID
Sixth implied	Profile ID Number	Identifies profile data currently loaded to the sequencer; value range: 0... 9999 (0 = block not configured); all blocks with the same machine ID number must have the same profile ID number
Seventh implied	Steps used	Loaded by user before using the block, DRUM will not alter steps used contents during logic solve: contains between 1 ... 999 for 24 bit CPUs, specifying the actual number of steps to be solved; the number must be \leq the table length in the bottom node of the ICMP block

The remaining registers contain data for each step in the sequence.

**Length
(Bottom Node)**

The integer value entered in the bottom node is the length-i.e., the number of application-specific registers-used in the step data table. The length can range from 1 .. 999 in a 24-bit CPU.

The total number of registers required in the step data table is the length + 8. The length must be > the value placed in the steps used register in the middle node.

Cascaded DRUM/ICMP Blocks

Cascaded DRUM/ICMP Blocks

A series of DRUM and/or ICMP blocks may be cascaded to simulate a mechanical drum up to 512 bits wide. Programming the same 4x register reference into the top node of each related block causes them to cascade and step as a grouped unit without the need of any additional application logic.

All DRUM/ICMP blocks with the same register reference in the top node are automatically synchronized. They must also have the same constant value in the bottom node, and must be set to use the same value in the steps used register in the middle node.

ID: Interrupt Disable

96

At a Glance

Introduction

This chapter describes the instruction ID.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: ID - Interrupt Disable	624
Representation: ID - Interrupt Disable	625
Parameter Description: ID - Interrupt Disable	626

Short Description: ID - Interrupt Disable

**Function
Description**

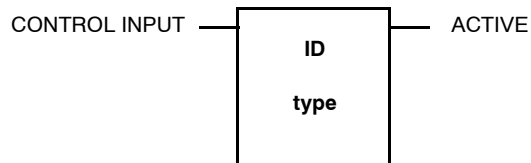
Three interrupt mask/unmask control instructions are available to help protect data in both the normal (scheduled) ladder logic and the (unscheduled) interrupt handling subroutine logic. These are the Interrupt Disable (ID) instruction, the Interrupt Enable (IE) instruction, and the Block Move with Interrupts Disabled (BMDI) instruction.

The ID instruction masks timer-generated and/or local I/O-generated interrupts. An interrupt that is executed in the time frame after an ID instruction has been solved and before the next IE instruction has been solved is buffered. The execution of a buffered interrupt takes place at the time the IE instruction is solved. If two or more interrupts of the same type occur between the ID ... IE solve, the mask interrupt overrun error bit is set, and the subroutine initiated by the interrupts is executed only one time

Representation: ID - Interrupt Disable

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = instruction masks timer-generated and/or local I/O generated interrupts
Type bottom node		INT, UINT	Type of interrupt to be masked (Constant integer) (For expanded and detailed information please see the section <i>Type (Bottom Node)</i> , p. 626.)
Top output	0x	None	Echoes state of the top input

Parameter Description: ID - Interrupt Disable

**Type
(Bottom Node)**

Enter a constant integer in the range 1 ... 3 in the node. The value represents the type of interrupt to be masked by the ID instruction, where:

Integer Value	Interrupt Type
3	Timer interrupt masked
2	Local I/O module interrupt masked
1	Both interrupt types masked

IE: Interrupt Enable

97

At a Glance

Introduction

This chapter describes the instruction IE.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: IE - Interrupt Enable	628
Representation: IE - Interrupt Enable	629
Parameter Description: IE - Interrupt Enable	630

Short Description: IE - Interrupt Enable

Function Description

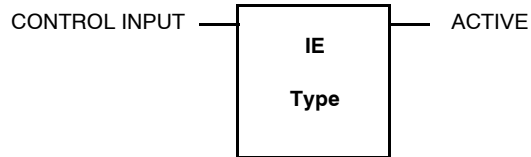
Three interrupt mask/unmask control instructions are available to help protect data in both the normal (scheduled) ladder logic and the (unscheduled) interrupt handling subroutine logic. These are the Interrupt Disable (ID) instruction, the Interrupt Enable (IE) instruction, and the Block Move with Interrupts Disabled (BMDI) instruction.

The IE instruction unmask interrupts from the timer or local I/O module and responds to the pending interrupts by executing the designated subroutines. An interrupt that is executed in the time frame after an ID instruction has been solved and before the next IE instruction has been solved is buffered. The execution of a buffered interrupt takes place at the time the IE instruction is solved. If two or more interrupts of the same type occur between the ID ... IE solve, the mask interrupt overrun error bit is set, and the subroutine initiated by the interrupts is executed only one time.

Representation: IE - Interrupt Enable

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = instruction unmask interrupts and responds pending interrupts
Type bottom node		INT, UINT	Type of interrupt to be unmasked (Constant integer) For more information, see <i>Type (Bottom Node)</i> , p. 630.
Top output	0x	None	Echoes state of the top input

Parameter Description: IE - Interrupt Enable

Top Input

When the input is energized, the IE instruction unmask interrupts from the timer or local I/O module and responds to the pending interrupts by executing the designated subroutines.

**Type
(Bottom Node)**

Enter a constant integer in the range 1 ... 3 in the node. The value represents the type of interrupt to be unmasked by the IE instruction, where:

Integer Value	Interrupt Type
3	Timer interrupt unmasked
2	Local I/O module interrupt unmasked
1	Both interrupt types unmasked

IMIO: Immediate I/O

98

At a Glance

Introduction

This chapter describes the instruction IMIO.

Note: This instruction is only available after configuring a CPU without extension.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: IMIO - Immediate I/O	632
Representation: IMIO - Immediate I/O	633
Parameter Description: IMIO - Immediate I/O	634
Run Time Error Handling: IMIO - Immediate I/O	636

Short Description: IMIO - Immediate I/O

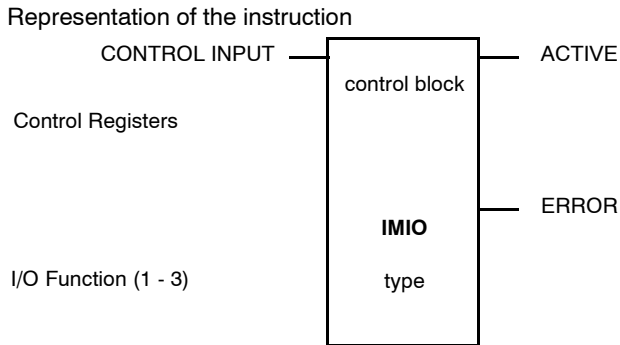
**Function
Description**

The IMIO instruction permits access of specified I/O modules from within ladder logic. This differs from normal I/O processing, where inputs are accessed at the beginning of the logic solve for the segment in which they are used and outputs are updated at the end of the segment's solution. The I/O modules being accessed must reside in the local backplane with the Quantum PLC.

In order to use IMIO instructions, the local I/O modules to be accessed must be designated in the I/O Map in your panel software.

Representation: IMIO - Immediate I/O

Symbol



Note: This IMIO block will not work with the following Compact I/O modules due to hardware design restrictions inherent with these modules

- AS-BADU-204
- AS-BADU-205
- AS-BADU-206
- AS-BADU-216

Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables the immediate I/O access
control block top node	4x	INT, UINT, WORD	Control block (first of two contiguous registers) For more information, see <i>Runtime Errors</i> , p. 636.
type bottom node		INT, UINT	Type of operation (constant integer in the range of 1 ... 3) This is the function to perform <ul style="list-style-type: none"> ● 1 – Input operation: Transfer data from module to state RAM ● 2 – Output operation: Transfer data from state RAM to module ● 3 – Bidirectional or I/O operation: Allows both Input and Output for bidirectional modules
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	Error (indicated by a code in the error status register in the IMIO control block)

Parameter Description: IMIO - Immediate I/O

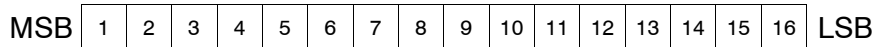
Control Block (Top Node)

The first of two contiguous 4x registers is entered in the top node. The second register is implied.

Register	Content
Displayed	This register specifies the physical address of the I/O module to be accessed.
First implied	This register logs the error status, which is maintained by the instruction.

Physical Address of the I/O Module

The high byte of the displayed register in the control block allows you to specify which rack the I/O module to be accessed resides in, and the low byte allow you to specify slot number within the specified rack where the I/O module resides.
Usage of word:



Bit	Function
1 - 5	Not used Rack 1 only for Quantum Local racks 1 through 4 can be used for 32-bit Compact
6 - 8	Rack number 1 to 4 (only rack 1 is currently supported)
9 - 11	Not used
12 - 16	Slot number

Rack Number

Bit Number			Rack Number
6	7	8	
0	0	1	rack 1 Rack 1 only for Quantum Racks 1 through 4 can be used for 32-bit Compact
0	1	0	rack 2 Racks 1 through 4 can be used for 32-bit Compact
0	1	1	rack 3 Racks 1 through 4 can be used for 32-bit Compact
1	0	0	rack 4 Racks 1 through 4 can be used for 32-bit Compact

Slot Number

Bit Number					Slot Number
12	13	14	15	16	
0	0	0	0	1	slot 1
0	0	0	1	0	slot 2
0	0	0	1	1	slot 3
0	0	1	0	0	slot 4
0	0	1	0	1	slot 5
0	0	1	1	0	slot 6
0	0	1	1	1	slot 7
0	1	0	0	0	slot 8
0	1	0	0	1	slot 9
0	1	0	1	0	slot 10
0	1	0	1	1	slot 11
0	1	1	0	0	slot 12
0	1	1	0	1	slot 13
0	1	1	1	0	slot 14
0	1	1	1	1	slot 15
1	0	0	0	0	slot 16

**Type
(Bottom Node)**

Enter a constant integer in the range 1 ... 3 in the bottom node. The value represents the type of operation to be performed by the IMIO instruction, where:

Integer Value	Type of Immediate Access
1	Input operation: transfers data from the specified module to state RAM
2	Output operation: transfers data from state RAM to the specified module
3	I/O operation: does both input and output if the specified module is bidirectional

Run Time Error Handling: IMIO - Immediate I/O

Runtime Errors

The implied register in the control block will contain the following error code when the instruction detects an error:

Error Code	Meaning
2001	Invalid type specified in the bottom node
2002	Problem with the specified I/O slot, either an invalid slot number entered in the displayed register of the control block or the I/O Map does not contain the correct module definition for this slot
2003	A type 3 operation is specified in the bottom node, and the module is not bidirectional
F001	Specified I/O module is not healthy

IMOD: Interrupt Module Instruction

99

At a Glance

Introduction

This chapter describes the instruction IMOD.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: IMOD - Interrupt Module	638
Representation: IMOD - Interrupt Module	639
Parameter Description: IMOD - Interrupt Module	641

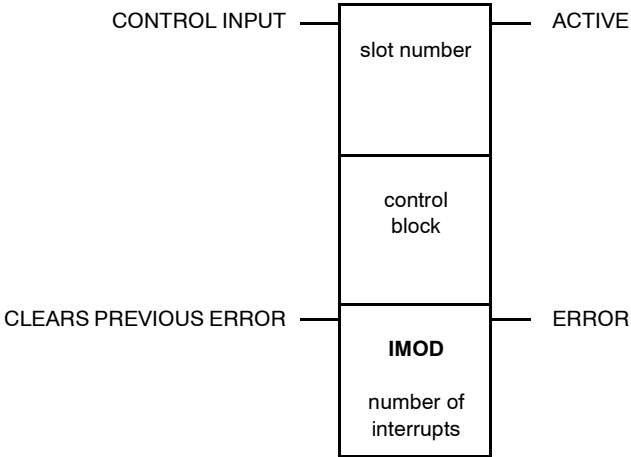
Short Description: IMOD - Interrupt Module

**Function
Description**

The IMOD instruction initiates a ladder logic interrupt handler subroutine when the appropriate interrupt is generated by a local interrupt module and received by the PLC. Each IMOD instruction in an application is set up to correspond to a specific slot in the local backplane where the interrupt module resides. The IMOD instruction can designate the same or a separate interrupt handler subroutine for each interrupt point on the associated interrupt module.

Representation: IMOD - Interrupt Module

Symbol Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates an interrupt
Bottom input	0x, 1x	None	ON = clears a previously detected error
slot number (top node)		INT, UINT	Indicates the slot number where the local interrupt module resides (constant integer in the range of 1 ... 16)
control block (middle node)	4x	INT, UINT, WORD	Control block (first of max. 19 contiguous registers, depending on number of interrupts) The middle node contains the first 4x register in the IMOD control block. The control block contains parameters required to program an IMOD instruction. The size (number of registers) of the control block will equal the total number of programmed interrupt points + 3. The first three registers in the control block contain status information. The remaining registers provide a means for you to specify the label (LAB) number of the interrupt handler subroutine. The interrupt handler subroutine is in the last (unscheduled) segment of the ladder logic program. For expanded and detailed information please see <i>Control Block (Middle Node)</i> , p. 642.)
number of interrupts (bottom node)		INT, UINT	Indicates the number of interrupts that can be generated from the associated interrupt module (constant integer in the range of 1 ... 16) The bottom node contains an integer indicating the number of interrupts that can be generated from the associated interrupt module. The size (number of registers of the control block is the number of interrupts + 3. The PLC is able to be configured for a maximum of 64 module interrupts (from all the interrupt modules residing in the local backplane). If the number you enter in the bottom node of an IMOD instruction causes the total number of module interrupts system wide to exceed 64, an error is logged in bit 7 of the first register in the control block. For example, if you use four interrupt modules in the local backplane and assign 16 interrupts to each of these modules (by entering 16 in the bottom node of each associated 8MOD instruction, the PLC will not be able to handle any more module interrupts. If you attempt to create a fifth IMOD instruction, an error will be logged in the IMOD's control block when you specify a value in the bottom node.
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	ON = error is detected. The source of the error can be from any one of the enabled interrupt points on the interrupt module.

Parameter Description: IMOD - Interrupt Module

General Information to IMOD

Up to 14 IMOD instructions can be programmed in a ladder logic application, one for each possible option slot in a local backplane.

Each interrupting point on each interrupt module can initiate a different interrupt handler subroutine.

A maximum of 64 interrupt points can be defined in a user logic application. It is not necessary that all possible input points on a local interrupt module be defined in the IMOD instruction as interrupts.

Enabling of the Instruction (Top Input)

When the input to the top node is energized, the IMOD instruction is enabled. The PLC will respond to interrupts generated by the local interrupt module in the designated slot number. When the top input is not energized, interrupts from the module in the designated slot are disabled and all previously detected errors are cleared including any pending masked interrupts.

Clear Error (Bottom Input)

This input clears previous errors.

Slot Number (Top Node)

The top node contains a decimal in the range 1 ... 16, indicating the slot number where the local interrupt module resides. This number is used to index into an array of control structures used to implement the instruction.

Note: The slot number in one IMOD instruction must be unique with respect to the slot numbers used in all other IMOD instruction in an application. If not the next IMOD with that particular slot number will have an error.

Note: The slot numbers where the PLC and the power supply reside are illegal entries -i.e., a maximum of 14 of the 16 possible slot numbers can be used as interrupt module slots. If the IMOD slot number is the same as the PLC, the IMOD will have an error.

**Control Block
(Middle Node)**

The middle node contains the first 4x register in the IMOD control block. The control block contains parameters required to program an IMOD instruction. The size (number of registers) of the control block will equal the total number of programmed interrupt points + 3.

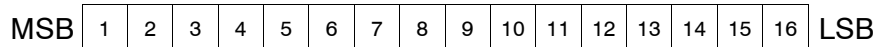
The first three registers in the control block contain status information, of the remaining registers provide means for you to specify the label (LAB) number of the interrupt handler subroutine that is in the last (unscheduled) segment of the ladder logic program.

Control Block for IMOD

Register	Content
Displayed	Function status bits
First implied	State of inputs 1 ... 16 from the interrupt module at the time of the interrupt
Second implied	State of inputs 17 ... 32 from the interrupt module at the time of the interrupt (invalid data for a 16-bit interrupt module)
Third implied	LAB number and status for the first interrupt programmed point on the interrupt module
...	...
Last implied	LAB number and status for the last interrupt programmed point on the interrupt


Function Status Bits

Function Status Bits



Bit	Function
1 - 2	Not used
3	Error: controller slot The slot number given in the top node of the IMOD is the CPU slot number.
4	Error: interrupt lost due to communication error in backplane When reading the interrupting module, a computation error occurred and the data is invalid. Because the interrupting points are cleared on the read, the interrupt(s) are lost.
5	Module not healthy or not in I/O map The I/O module in the slot given in the top node is not healthy (i.e., not working, or missing) or a module has not been specified in the I/O map.
6	Error: interrupt lost because of on-line editing While the operator was editing the ladder logic (this includes requesting a power display of a different network, i.e., page up or page down), two or more interrupts for the same point occurred. Only one is serviced.
7	Error: Maximum number of interrupts exceeded More than 64 interrupts have been specified in the ladder logic and this "IMOD" is the one that causes the count to exceed 64.
8	Error: slot number used in previous network (CAUTION: see <i>Loss of Interrupts</i> , p. 643) The slot number in the top node is used in another "IMOD" block with in the ladder logic. The first block is working, but this one is ignored.
9 - 15	Not used
16	0 = IMOD disabled 1 = IMOD enabled This bit reflects to the state of power in the top node.

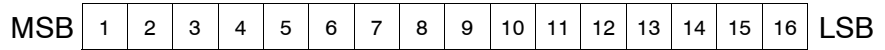
Loss of Interrupts

	CAUTION
	<p>LOSS OF INTERRUPTS: WORKING IMOD INSTRUCTION</p> <p>An error is indicated in bit 8 when two IMOD instructions are assigned the same slot number. When this happens, it is possible to lose interrupts from the working IMOD instruction without an indication if the number specified in the bottom node of the two instructions is different.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

Status Bits and LAB Number for each Interrupt Point

Bits 1 ... 5 of the third implied through last implied registers are status bits for each interrupt point. Bits 7 ... 16 are used to specify the LAB number for the interrupt handler subroutine. The LAB number is a decimal value in the range 1 ... 1023

Function Status Bits



Bit	Function
Interrupt Point Status	
1	Execution delayed because of interrupt mask This is not an error, but an indication that interrupts are disabled and at least one interrupt for this point has occurred and will be serviced when interrupts are enabled.
2	Error: invalid block in the interrupt handler subroutine An invalid DX block has been used in the interrupt handler subroutine for this input point (see Instructions that Cannot be Used in an Interrupt Handler for details).
3	Error: Mask interrupt overrun Two or more interrupts for this point have occurred while the interrupt was disabled: i.e., Use of the Interrupt Disable (ID) block without using the Interrupt Enable (IE) block or during online editing.
4	Error: execution overrun A second interrupt (or more) has occurred while the interrupt handler subroutine was still running.
5	Error: invalid LAB number The LAB number specified in bits 7...16, zero, or that LAB number is not used in the last segment of the user logic. This error will Auto Clear.
6	not used
LAB number	
7 - 16	LAB number for the associated interrupt handler Value in the range 1 ... 1023

Whenever the input to the bottom node of the IMOD instruction is enabled, the status bits (bits 1 ... 5) are cleared. If a LAB number is specified (in bits 7 ... 16) as 0 or an invalid number, any interrupts generated from that point are ignored by the PLC.

**Number of
Interrupts
(Bottom Node)**

The bottom node contains an integer indicating the number of interrupts that can be generated from the associated interrupt module. The size (number of registers) of the control block is this number + 3.

The PLC is able to be configured for a maximum of 64 module interrupts (from all the interrupt modules residing in the local backplane). If the number you enter in the bottom node of an IMOD instruction causes the total number of module interrupts system wide to exceed 64, an error is logged in bit 7 of the first register in the control block.

For example, if you use four interrupt modules in the local backplane and assign 16 interrupts to each of these modules (by entering 16 in the bottom node of each associated IMOD instruction, the PLC will not be able to handle any more module interrupts. If you attempt to create a fifth IMOD instruction, an error will be logged in that IMOD's control block when you specify a value in the bottom node.

ITMR: Interrupt Timer

100

At a Glance

Introduction

This chapter describes the instruction ITMR.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: ITMR - Interval Timer Interrupt	648
Representation: ITMR - Interval Timer Interrupt	649
Parameter Description: ITMR - Interval Timer Interrupt	651

Short Description: ITMR - Interval Timer Interrupt

Function Description

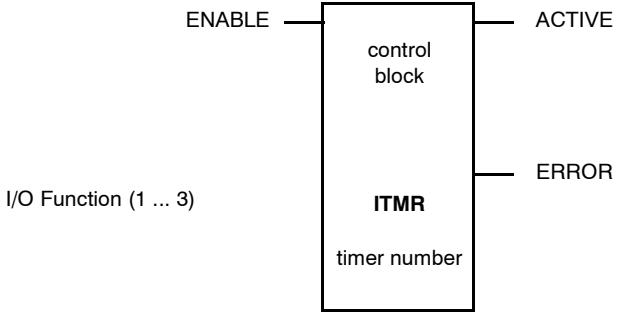
The ITMR instruction allows you to define an interval timer that generates interrupts into the normal ladder logic scan and initiates the execution of an interrupt handling subroutine. The user-defined interrupt handler is a ladder logic subroutine created in the last, unscheduled segment of ladder logic with its first network marked by a LAB instruction. Subroutine execution is asynchronous to the normal scan cycle. Up to 16 ITMR instructions can be programmed in an application. Each interval timer can be programmed to initiate the same or different interrupt handler subroutines, controlled by the JSR/LAB method described in the chapter *General*.

Each instance of the interval timer is delayed for a programmed interval while the PLC is running, then generates a processor interrupt when the interval has elapsed. An interval timer can execute at any time during normal logic scan, including system I/O updating or other system housekeeping operations. The resolution of each interval timer is 1 ms. An interval can be programmed in units of 1 ms, 10 ms, 100 ms, or 1 s. An internal counter increments at the specified resolution.

You should be aware that if the ITMR time is less than the L/L edit time slice, there will be no power flow display or user logic edit allowed.

Representation: ITMR - Interval Timer Interrupt

Symbol Representation of the instruction



**Parameter
Description**

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables instruction (For expanded and detailed information please see the section Top Input.)
control block (top node)	4x	INT, UINT, WORD	Control block (first of three contiguous registers) The top node contains the first of three contiguous 4xxx registers in the ITMR control block. These registers are used to specify the parameters required to program each ITMR instruction. The lower eight bits of the first (displayed) register in the control block allow you to specify function control parameters, and the upper eight bits are used to display function status. In the second register of the control block, specify a value representing the interval at which the ITRM instruction will generate interrupts and initiate the execution of the interrupt handler. The interval will be incremented in the units specified by bits 12 and 13 of the first control block register - i.e., 1 ms, 10 ms, 100 ms, or 1 s units. In the third register of the control block, specify a value indicating the label (LAB) number that will start the interrupt handler subroutine. The number must be in the range of 1 through 1023. Note: We recommend that the size of the logic subroutine associated with the LAB be minimized so that the application does not become interrupt-driven. (For more information, see <i>Control Block (Top Node)</i> , p. 651.)
timer number (bottom node)		INT, UINT	Timer number assigned to this ITMR instruction (must be unique with respect to all other ITMR instructions in the application); range: 1 ... 16
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	Error (source of the error may be in the programmed parameters or a runtime execution error)

Parameter Description: ITMR - Interval Timer Interrupt

Top Input

When the top input is energized, the ITMR instruction is enabled. It begins counting the programmed time interval. When that interval has expired the counter is reset and the designated error handler logic executes.

When the top input is not energized, the following events occur:

- All indicated errors are cleared
- The timer is stopped
- The time count is either reset or held, depending on the state of bit 15 of the first register in the control block (the displayed register in the top node)
- Any pending masked interrupt is cleared for this timer

Control Block (Top Node)

The top node contains the first of three contiguous 4x registers in the ITMR control block. These registers are used to specify the parameters required to program each ITMR instruction.

Control Block for ITMR

Register	Content
Displayed	Function status and function control bits
First implied	In this register specify a value representing the interval at which the ITMR instruction will generate interrupts and initiate the execution of the interrupt handler. The interval will be incremented in the units specified by bits 12 and 13 of the first control block register, i.e. 1 ms, 10 ms, 100 ms, or 1 s units.
Second implied	In this register specify a value indicating the label (LAB) number that will start the interrupt handler subroutine. The number must be in the range 1 ... 1023.

Note: We recommend that the size of the logic subroutine associated with the LAB be minimized so that the application does not become interrupt-driven.

Function Status and Function Control Bits

The lower eight bits of the displayed register in the control block allow you to specify function control parameters, and the upper eight bits are used to display function status:

MSB

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

 LSB

Bit	Function
Function Status	
1	Execution delayed because of interrupt mask.
2	Invalid block in the interrupt handler subroutine.
3	Not used
4	Time = 0
5	Mask interrupt overrun.
6	Execution overrun.
7	No LAB or invalid LAB.
8	Timer number used in previous network.
Function Control	
9 - 11	Not used
12 - 13	0 0 = 1 ms time base 0 1 = 10 ms time base 1 0 = 100 ms time base 1 1 = 1 s time base
14	1 = PLC stop holds counter. 0 = PLC stop resets counter.
15	1 = enable OFF holds counter. 0 = enable OFF resets counter.
16	1 = instruction enabled 0 = instruction disabled

Timer Number (Bottom Node)

Up to 16 ITMR instructions can be programmed in an application. The interrupts are distinguished from one another by a unique number between 1 ... 16, which you assign to each instruction in the bottom node. The lowest interrupt number has the highest execution priority.

For example, if ITMR 4 and ITMR 5 occur at the same time, ITMR 4 is executed first. After ITMR 4 has finished, ITMR 5 generally will begin executing.

An exception would be when another ITMR interrupt with a higher priority occurs during ITMR 4's execution. For example, suppose that ITMR 3 occurs while ITMR 5 is waiting for ITMR 4 to finish executing. In this case, ITMR 3 begins executing when ITMR 4 finishes, and ITMR 5 continues to wait.

ITOF: Integer to Floating Point

101

At a Glance

Introduction

This chapter describes the instruction ITOF.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	654
Representation: ITOF - integer to Floating Point Conversion	655

Short Description

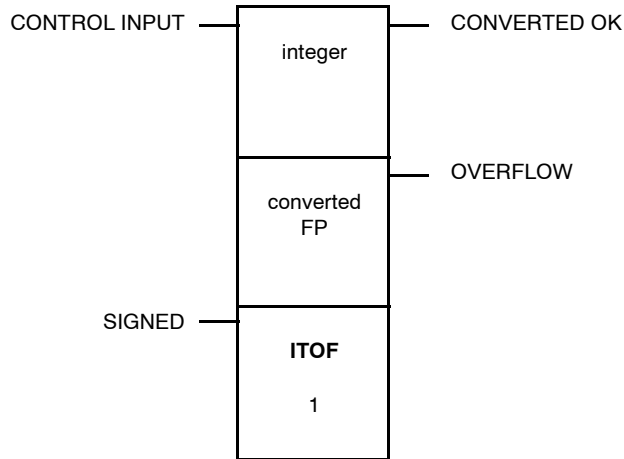
**Function
Description**

The ITOF instruction performs the conversion of a signed or unsigned integer value (its top node) to a floating point (FP) value, and stores the FP value in two contiguous 4x registers in the middle node.

Representation: ITOF - integer to Floating Point Conversion

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables conversion
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
integer (top node)	3x, 4x	INT, UINT	Integer value, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
converted FP (middle node)	4x	REAL	Converted FP value (first of two contiguous holding registers)
1 (bottom node)		INT, UINT	Constant value of 1, can not be changed
Top output	0x	None	ON = FP conversion completed successfully

JSR: Jump to Subroutine

102

At a Glance

Introduction

This chapter describes the instruction JSR.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	658
Representation: JSR - Jump to Subroutine	659

Short Description

Function Description

When the logic scan encounters an enabled JSR instruction, it stops the normal logic scan and jumps to the specified source subroutine in the last (unscheduled) segment of ladder logic.

You can use a JSR instruction anywhere in user logic, even within the subroutine segment. The process of calling one subroutine from another subroutine is called nesting. The system allows you to nest up to 100 subroutines; however, we recommend that you use no more than three nesting levels. You may also perform a recursive form of nesting called looping, whereby a JSR call within the subroutine recalls the same subroutine.

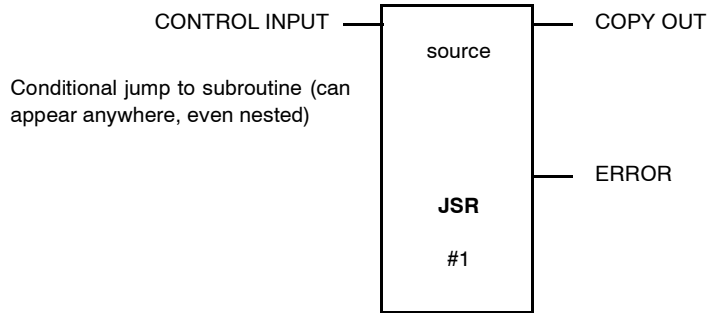
Example to Subroutine Handling

For an example of subroutine handling, see *Subroutine Handling*, p. 107.

Representation: JSR - Jump to Subroutine

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Enables the source subroutine
source (top node)	4x	INT, UINT	Source pointer (indicator of the subroutine to which the logic scan will jump), entered explicitly as an integer or stored in a register; range: 1 ... 1 023
#1 (bottom node)		INT, UINT	Always enter the constant value 1
Top output	0x	None	Echoes state of the top input
Bottom output	0x	None	Error in subroutine jump On if jump cannot be executed Label does not exist or Nesting level > 100

LAB: Label for a Subroutine

103

At a Glance

Introduction

This chapter describes the instruction LAB.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	662
Representation: LAB - Label	663
Parameter Description	664

Short Description

Function Description

The LAB instruction is used to label the starting point of a subroutine in the last (unscheduled) segment of user logic. This instruction must be programmed in row 1, column 1 of a network in the last (unscheduled) segment of user logic. LAB is a one-node function block

LAB also serves as a default return from the subroutine in the preceding networks. If you are executing a series of subroutine networks and you find a network that begins with LAB, the system knows that the previous subroutine is finished, and it returns the logic scan to the node immediately following the most recently executed JSR block.

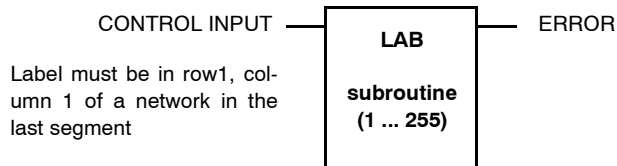
Note: If you need real world I/O serviced while you are in the interrupt subroutine, you **MUST** use the IMIO (read/write) function block in the same subroutine. If you do not, the real world I/O referenced in that subroutine will **NOT** get serviced until the appropriate segment is solved.

Example to Subroutine Handling

For an example of subroutine handling, see *Subroutine Handling*, p. 107.

Representation: LAB - Label

Symbol Representation of the instruction



Parameter Description Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Initiates the subroutine specified by the number in the bottom node
subroutine (top node)		INT, UINT	Integer value, identifies the subroutine you are about to execute Range: 1 ... 255 16-bit PLC. Range: 1 ... 1023 24-bit PLC. Size = constant 1 - 255 or Size = constant 1-1023 for 785L Subroutine number error ON if return cannot be executed If more than one network begins with a LAB instruction with the same subroutine value, the lowest-numbered network is used as the starting point for the subroutine.
Top output	0x	None	ON = error in the specified subroutine's initiation

Parameter Description

**Subroutine
(Bottom Node)**

The integer value entered in the node identifies the subroutine you are about to execute. The value can range from 1 ... 255. If more than one subroutine network has the same LAB value, the network with the lowest number is used as the starting point for the subroutine.

LOAD: Load Flash

104

At a Glance

Introduction

This chapter describes the instruction LOAD.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	666
Representation: LOAD - Load	667
Parameter Description	668

Short Description

Function Description

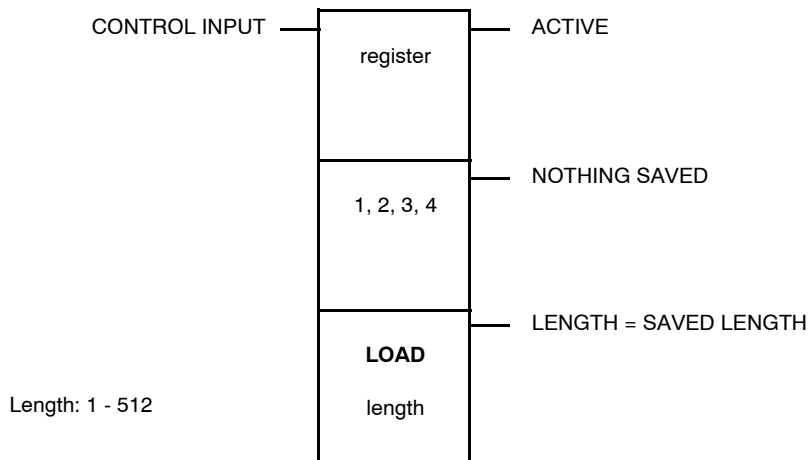
Note: This instruction is available with the PLC family TSX Compact, with Quantum CPUs 434 12/ 534 14 and Momentum CPUs CCC 960 x0/ 980 x0.

The LOAD instruction loads a block of 4x registers (previously saved) from state RAM where they are protected from unauthorized modification.

Representation: LOAD - Load

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Start LOAD operation: it should remain ON until the operation has completed successfully or an error has occurred.
register (top node)	4x	INT, UINT, WORD	First of max. 512 contiguous 4x registers to be loaded from state RAM
1, 2, 3, 4 (middle node)		INT	Integer value, which defines the specific buffer where the block of data is to be loaded
length (bottom node)		INT	Number of words to be loaded, range: 1 ... 512
Top output	0x	None	ON = LOAD is active
Middle output	0x	None	ON = a LOAD is requested from a buffer where no data has been saved.
Bottom output	0x	None	ON = Length not equal to SAVED length

Parameter Description

1, 2, 3, 4

(Middle Node)

The middle node defines the specific buffer where the block of data is to be loaded. Four 512 word buffers are allowed. Each buffer is defined by placing its corresponding value in the middle node, that is, the value 1 represents the first buffer, value 2 represents the second buffer and so on. The legal values are 1, 2, 3, and 4. When the PLC is started all four buffers are zeroed. Therefore, you may not load data from the same buffer without first saving it with the instruction SAVE. When this is attempted the middle output goes ON. In other words, once a buffer is used, it may not be used again until the data has been removed.

Bottom Output

The output from the bottom node goes ON when a LOAD request is not equal to the registers that were SAVEd. This kind of transaction is allowed, however, it is your responsibility to ensure this does not create a problem in your application.

MAP 3: MAP Transaction

105

At a Glance

Introduction

This chapter describes the instruction MAP 3.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	670
Representation: MAP 3 - Map Transaction	671
Parameter Description	672

Short Description

Function Description

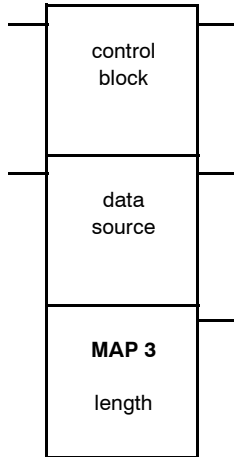
Note: This instruction is only available if you have unpacked and installed the DX Loadables. For further information, see *Installation of DX Loadables, p. 109*.

Ladder logic applications running in the controller initiate communication with MAP network nodes through the MAP3 instruction.

Representation: MAP 3 - Map Transaction

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = initiates a transaction
Middle input	0x, 1x	None	ON = new transaction to be initiated in the same scan
control block (top node)	4x	INT, UINT, WORD	Control Block (first register of a block)
data source (middle node)	4x	INT, UINT, WORD	Data source (starting register)
length (bottom node)		INT, UINT	Length of local data area, range: 1 ... 255)
Top output	0x	None	Transaction completes successfully
Middle output	0x	None	Transaction is in progress
Bottom output	0x	None	Error

Parameter Description

Top Input

This input initiates a transaction. To start a transaction the input must be held ON (HIGH) for at least one scan. If the S980 has resources to process the transaction, the middle output passes power. If resources are not available, no outputs pass power.

Once a transaction is started, it will run until a reply is received, a communications error is detected, or a timeout occurs. The values in the Control Block, Data Source and Length must not be altered, or the transaction will not be completed and the bottom output will pass power. A second transaction cannot be started by the same block until the first one is complete.

Middle Input

If the top input is also HIGH, the middle input going ON allows a new transaction to be initiated in the same scan, following the completion of a previous one. A new transaction begins when the top output passes power from the first transaction.

**Control Block
(Top Node)**

The top node is the starting 4x register of a block of registers that control the block's operation.

The contents of each register is determined by the kind of operation to be performed by the MAP3 block:

- Read or Write
- Information Report
- Unsolicited Status
- Conclude
- Abort

Registers of the Control Block:

Word	Meaning
1	Destination Device
2	Qualifier / Function Code
3	Network Mode / Network Type
4	Function Status
5	Register A Reference Type This word is labeled Register A* and contains the reference type for 4 types of Read (0x, 1x, 3x, and 4x registers) and 2 types of Write (0X or 4x).
6	Register B Reference Number This word is labeled Register B* and contains the starting reference number in the range 1 to 99999.
7	Register C Reference Length This word is labeled Register C* and contains the Quantity of references requested.
8	Register D Timeout This word is labeled Register D* and contains the Timeout parameter. This value sets the maximum length of time used to complete a transaction, including retries.

**Destination
Device**

Word 1 contains the destination device in bit position 9 through 16. The computer works with this byte as the LSB and will accept a range of 1 to 255.

Usage of word 1:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 8	Not used
9 - 16	Destination device

**Qualifier /
Function Code**

Word 2 contains two bytes of information. The qualifier bits 1 to 8 and the function code in bits 9 to 16.

Usage of word 2:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
Qualifier	
1 - 8	0 = addressed >0 = named
Function Code	
9 - 16	4 = read 5 = write

**Network Mode /
Network Type**

Word 3 contains two bytes of information. The mode is in bits 5 through 8 and the type is in bits 9 through 16.

Usage of word 3:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function
1 - 4	Not used
Mode	
5 - 8	1 = association
Type	
9 - 12	7 = 7 layer MAP network
13 - 16	1 = type 1 service

Function Status

Word 4 is the function status. An error code is returned if an error occurs in a block initiated function.

The decimal codes are:

Code	Meaning
1	Association request rejected
4	Message timeout application response
5	Invalid destination device
6	Message size exceeded
8	Invalid function code
17	Device not available
19	Unsupported network type
22	No channel available
23	MMS message not sent
24	Control block changed
25	Initiate failed
26	System download in progress
28	Channel not ready
99	Undetermined error
103	Access denied
105	Invalid address
110	Object nonexistent

Function summary

The network controlling device may issue a function code that alters the control block register assignment as given above for Read/Write. Those differences for Information, Status, Conclude and Abort are identified in this summary on the bottom of your screen

Refer to *Modicon S980 Map 3.0 Network Interface User Guide* that describes the register contents for each operation.

Data Source (Middle Node)

The middle node is the starting 4x register of the local data source (for a write request) or local data destination (for a read).

Length (Bottom Node)

The bottom node defines the maximum size of the local data area (the quantity of registers) starting at 4x register of data source, in the range of 1 to 255 decimal. The quantity of data to be actually transferred in the operation is determined by a Reference Length parameter in one of the control registers.

Top Output	The top output passes power for one scan when a transaction completes successfully.
Middle Output	The middle output passes power when a transaction is in progress. If the top input is ON and the middle input is OFF, then the middle output will go OFF on the same scan that the top output goes ON. If both top input and middle input are ON, then the middle output will remain ON
Bottom Output	The bottom output passes power for one scan when a transaction cannot be completed. An error code is returned to the Function Status Word (register 4x+3) in the function's control block.

MATH - Integer Operations

106

At A Glance

Introduction

This chapter describes the four integer operations executed by the instruction MATH. The four operations are Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10).

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: MATH - Integer Operations - Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10)	678
Representation: MATH - Integer Operations - Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10)	679

Short Description: MATH - Integer Operations - Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10)

Function Description

The MATH instruction performs any one of four integer math operations, which is called by entering a function code in the range 1 ... 4 in the bottom node. Table with two columns

Code	MATH Function
1	Decimal square root
2	Process square root
3	Logarithm (base 10)
4	Antilogarithm (base 10)

Each MATH function operates on the contents of the top node registers and places a result in the middle node registers.

For example, the normal square root uses registers 3/4xxx and 3/4xxx+1 as an 8 digit operand and stores the result in 4yyy and 4yyy+1. The result storage format is XXXX.XX00 where there are 2 places of precision following an implied decimal point.

Math performs the function indicated by the bottom node:

Code	Function	Operand Registers	Range	Result Registers	Range
1	Normal	3/4x, 3/4x + 1	8 Digits	4y, 4y + 1	xxxx.xx00
2	Process	3/4x	4 Digits	4y, 4y + 1	xxxx.xx00
3	Log (x)	3/4x, 3/4x + 1	8 Digits	4y	1 to 7,999
4	Antilog (x)	3/4x	1 to 7,999	4y, 4y + 1	8 Digits

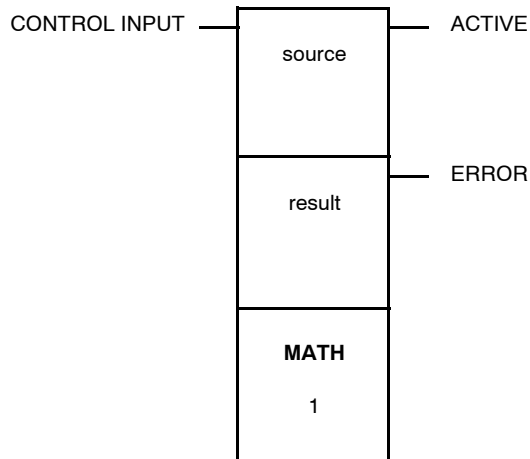
Representation: MATH - Integer Operations - Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10)

Explanation of This Section

This section describes the Decimal Square Root, Process Square Root, Logarithm (base 10), and Antilogarithm (base 10) operations, which are the four operations performed by the instruction MATH. Each operation has two sections describing that operation. The first section is called Symbol, which is a graphical representation of the operation. The second section is called Parameter Description, which is a table-format representation of the operation.

Symbol - Decimal Square Root

Representation of the instruction for the Decimal Square Root operation



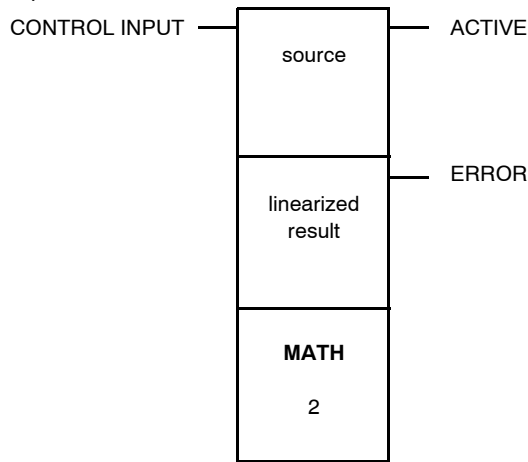
**Parameter
Description -
Decimal Square
Root**

Description of the instruction's parameters for the Decimal Square Root operation

Parameters	State RAM Reference	Data Type	Meaning
Top Input	0x, 1x	None	ON initiates a standard square root operation.
source (top node)	3x, 4x	INT, UINT	<p>The first of two contiguous 3xxxx or 4xxxx registers is entered in the top node. The second register is implied. The source value (the value for which the square root will be derived) is stored here.</p> <p>If you specify a 4xxxx register, the source value may be in the range 0 through 99,999,99. The low-order half of the value is stored in the implied register, and the high-order half is stored in the displayed register.</p> <p>If you specify a 3xxxx register, the source value may be in the range 0 through 9,999. The square root calculation is done on only the value in the displayed register; the implied register is required but not used.</p>
result (middle node)	4x	INT, UINT	<p>Enter the first of two contiguous 4xxxx registers in the middle node. The second register is implied. The result of the standard square root operation is stored here.</p> <p>The result is stored in the fixed-decimal format: 1234.5600. where the displayed register stores the four-digit value to the left of the first decimal point and the implied register stores the four-digit value to the right of the first decimal point. Numbers after the second decimal point are truncated; no round-off calculations are performed.</p>
Top output	0x	None	ON = operation successful
Bottom output	0x	None	ON = top-node value out of range

**Symbol -
Process Square
Root**

Representation of the instruction for the Process Square Root operation



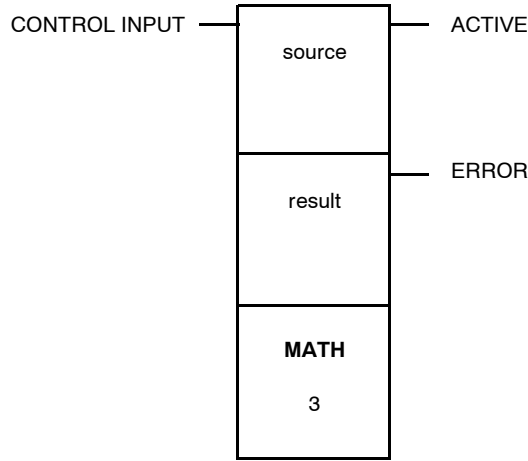
**Parameter
Description -
Process Square
Root**

The process square root function tailors the standard square root function for closed loop analog control applications. It takes the result of the standard square root result, multiplies it by 63.9922 (the square root of 4095) and stores that linearized result in the middle-node registers.

Parameters	State RAM Reference	Data Type	Meaning
Top Input	0x, 1x	None	ON initiates process square root operation
source (top node)	3x, 4x	INT, UINT	The first of two contiguous 3xxxx or 4xxxx registers is entered in the top node. The second register is implied. The source value (the value for which the square root will be derived) is stored in these two registers. In order to generate values that have meaning, the source value must not exceed 4095. In a 4xxxx register group the source value will therefore be stored in the implied register, and in a 3xxxx register group the source value will be stored in the displayed register.
result (middle node)	4x	INT, UINT	The first of two contiguous 4xxxx registers is entered in the middle node. The second register is implied. The linearized result of the process square root operation is stored here. The result is stored in the fixed-decimal format: 1234.5600. where the displayed register stores the four-digit value to the left of the first decimal point and the implied register stores the four-digit value to the right of the first decimal point. Numbers after the second decimal point are truncated; no round-off calculations are performed.
Top output	0x	None	ON = Operation successful
Bottom output	0x	None	ON = Source value out of range

**Symbol -
Logarithm
(base 10)**

Representation of the instruction for the Logarithm (base 10) operation



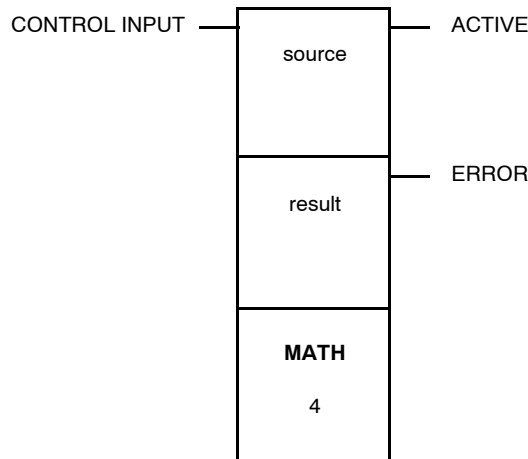
**Parameter
Description -
Logarithm (base
10 logarithm)**

Description of the instruction's parameters for the Logarithm (base 10) operation

Parameters	State RAM Reference	Data Type	Meaning
Top Input	0x, 1x	None	ON enables log(x) operation
source (top node)	3x, 4x	INT, UINT	The first of two contiguous 3xxxx or 4xxxx registers is entered in the top node. The second register is implied. The source value upon which the log calculation will be performed is stored in these registers. If you specify a 4xxxx register, the source value may be in the range 0 through 99,999,99. The low-order half of the value is stored in the implied register, and the high-order half is stored in the displayed register. log calculation is done on only the value in the displayed register; the implied register is required but not used.
result (middle node)	4x	INT, UINT	The middle node contains a single 4xxxx holding register where the result of the base 10 log calculation is posted. The result is expressed in the fixed decimal format 1.234 , and is truncated after the third decimal position. The largest result that can be calculated is 7.999, which would be posted in the middle register as 7999.
Top output	0x	None	ON = Operation Successful
Bottom output	0x	None	ON = an error ar a value out of range

**Symbol -
Antilogarithm
(base 10)**

Representation of the instruction for the Antilogarithm (base 10) operation



**Parameter
Description -
Antilogarithm
(base 10)**

Description of the instruction's parameters for the Antilogarithm (base 10) operation

Parameters	State RAM Reference	Data Type	Meaning
Top Input	0x, 1x	None	ON enables antilog(x) operation
source (top node)	3x, 4x	INT, UINT	The top node is a single 4xxxx holding register or 3xxxx input register. The source value (the value on which the antilog calculation will be performed) is stored here in the fixed decimal format 1.234 . It must be in the range 0 through 7999, representing a source value up to a maximum of 7.999.
result (middle node)	4x	INT, UINT	The first of two contiguous 4xxxx registers is entered in the middle node. The second register is implied. The result of the antilog calculation is posted here in the fixed decimal format 12345678. posted in the implied register. The largest antilog value that can be calculated is 99770006 (9977 posted in the displayed register and 0006 posted in the implied register).
Top output	0x	None	ON = Operation successful
Bottom output	0x	None	ON = An error or a value out of range

MBIT: Modify Bit

107

At a Glance

Introduction

This chapter describes the instruction MBIT.

What's in this Chapter?


This chapter contains the following topics:

Topic	Page
Short Description	686
Representation: MBIT - Logical Bit Modify	687
Parameter Description	688

Short Description

**Function
Description**

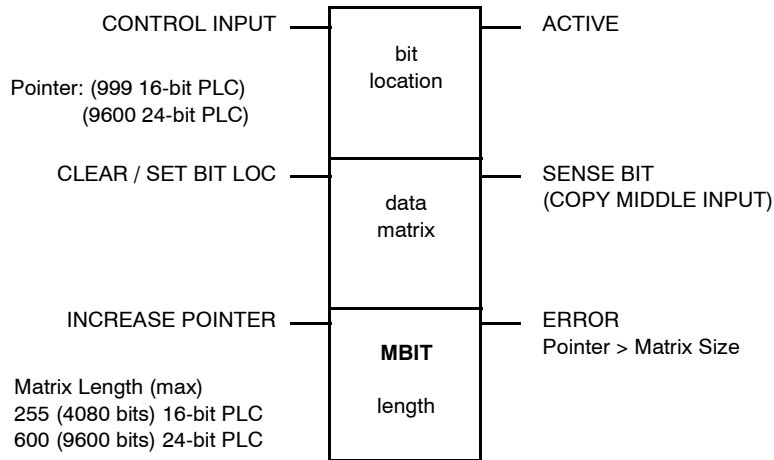
The MBIT instruction modifies bit locations within a data matrix, i.e. it sets the bit(s) to 1 or clears the bit(s) to 0. One bit location may be modified per scan.

	WARNING
	<p>Overriding of disabled coils without enabling them</p> <p>MBIT will override any disabled coils within a destination group without enabling them. This can cause injury if a coil has been disabled for repair or maintenance because the coil's state can change as a result of the MBIT instruction.</p> <p>Failure to follow this precaution can result in death, serious injury, or equipment damage.</p>

Representation: MBIT - Logical Bit Modify

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = implements bit modification
Middle input	0x, 1x	None	OFF = clear bit locations to 0 ON = set bit locations to 1
Bottom input	0x, 1x	None	Increment bit location by one after modification
bit location (top node)	3x, 4x	INT, UINT, WORD	Specific bit location to be set or clear in the data matrix; entered explicitly as an integer value or stored in a register (range 1 ... 9 600)
data matrix (middle node)	0x, 4x	INT, UINT, WORD	First word or register in the data matrix
length (bottom node)		INT, UINT	Matrix length; range: 1 ... 600
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	Echoes state of the middle input
Bottom output	0x	None	ON = error: bit location > matrix length

Parameter Description

Bit Location (Top Node)

Note: If the bit location is entered as an integer or in a 3x register, the instruction will ignore the state of the bottom input.

Matrix Length (Bottom Node)

The integer value entered in the bottom node specifies a matrix length, i.e, the number of 16-bit words or registers in the data matrix. The length can range from 1 ... 600 in a 24-bit CPU, e.g, a matrix length of 200 indicates 3200 bit locations.

At a Glance

Introduction

This chapter describes the instruction MBUS.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	690
Representation: MBUS - Modbus II Transfer	691
Parameter Description	692
The MBUS Get Statistics Function	694

Short Description

Function Description

Note: This instruction is only available if you have unpacked and installed the DX Loadables. For further information, see *Installation of DX Loadables*, p. 109.

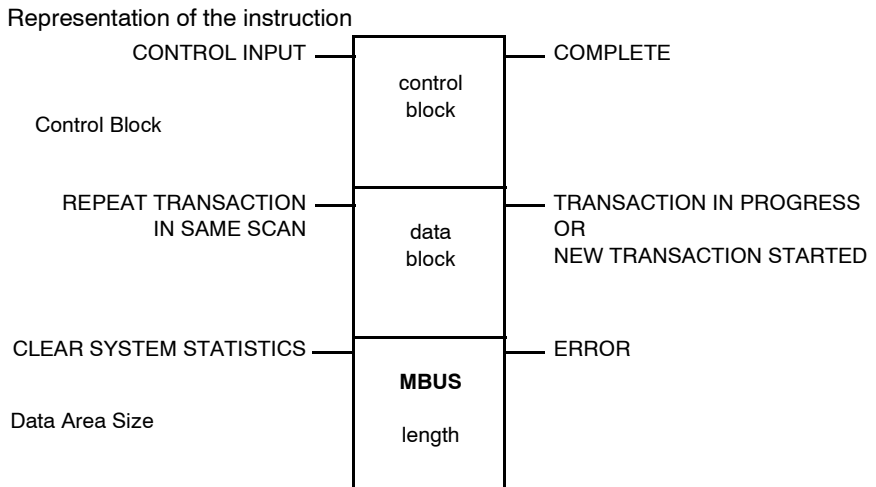
The S975 Modbus II Interface option modules use two loadable function blocks: MBUS and PEER. MBUS is used to initiate a single transaction with another device on the Modbus II network. In an MBUS transaction, you are able to read or write discrete or register data.

PLCs on a Modbus II network can handle up to 16 transactions simultaneously. Transactions include incoming (unsolicited) messages as well as outgoing messages. Thus, the number of message initiations a PLC can manage at any time is 16 - # of incoming messages.

A transaction cannot be initiated unless the S975 has enough resources for the entire transaction to be performed. Once a transaction has been initiated, it runs until a reply is received, an error is detected, or a timeout occurs. A second transaction cannot be started in the same scan that the previous transaction completes unless the middle input is ON. A second transaction cannot be initiated by the same MBUS instruction until the first transaction has completed.

Representation: MBUS - Modbus II Transfer

Symbol



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	Enable MBUS transaction
Middle input	0x, 1x	None	Repeat transaction in same scan
Bottom input	0x, 1x	None	Clears system statistics
control block (top node)	4x	INT, UINT, WORD	First of seven contiguous registers in the MBUS control block (For more information, see <i>Control Block (Top Node)</i> , p. 692.)
data block (middle node)	4x	INT, UINT, WORD	First 4x register in a data block to be transmitted or received in the MBUS transaction.
length (bottom node)		INT, UINT	Number of words reserved for the data block is entered as a constant value (For more information, see <i>Length (Bottom Node)</i> , p. 693.)
Top output	0x	None	Transaction complete
Middle output	0x	None	Transaction in progress or new transaction starting
Bottom output	0x	None	Error detected in transaction

Parameter Description

Control Block (Top Node)

The 4x register entered in the top node is the first of seven contiguous registers in the MBUS control block:

Register	Content
Displayed	Address of destination device (range: 0 ... 246)
First implied	not used
Second implied	Function code
Third implied	Reference type
Fourth implied	Reference number, e.g., if you placed a 4 in the third implied register and you place a 23 in this register, the reference will be holding register 400023
Fifth implied	Number of words of discrete or register references to be read or written
Sixth implied	Time allowed for a transaction to be completed before an error is declared; expressed as a multiple of 10 ms, e.g., 100 indicates 1 000 ms; the default timeout is 250 ms.

Function Code

This register contains the function code for requested action:

Value	Meaning
01	Read discrettes
02	Read registers
03	Write discrete outputs
04	Write register outputs
255	Get system statistics

Reference Type

This register contains one of 4 possible discrete or register reference types:

Value	Reference type
0	Discrete output (0x)
1	Discrete input (1x)
2	Input register (3x)
3	Holding register (4x)

Number of Words to Read or Write

Number of words of discrete or register references to be read or written; the length limits are:

Read register	251 registers
Write register	249 registers
Read coils	7.848 discrettes
Write coils	7.800 discrettes

Length (Bottom Node)

The number of words reserved for the data block is entered as a constant value in the bottom node. This number does not imply a data transaction length, but it can restrict the maximum allowable number of register or discrete references to be read or written in the transaction.

The maximum number of words that may be used in the specified transaction is:

Max. Number of Words	Transaction
251	Reading registers (one register/word)
249	Writing registers (one register/word)
490	Reading discrettes using 24-bit CPUs (up to 16 discrettes/word)
487	Writing discrettes using 24-bit CPUs (up to 16 discrettes/word)

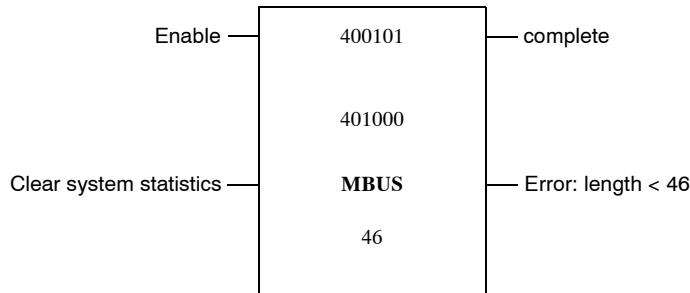
The MBUS Get Statistics Function

General

Issuing function code 255 in the second implied register of the MBUS control block obtains a copy of the Modbus II local statistics, a series of 46 contiguous register locations where data describing error and system conditions is stored. To use MBUS for a get statistics operation, set the length in the bottom node to 46, a length < 46 returns an error (the bottom output will go ON), and a length > 46 reserves extra registers that cannot be used.

Example

Parameterizing of the instruction



Register 400101 is the first register in the MBUS control block, making register 400103 the control register that defines the MBUS function code. By entering a value of 255 in register 400103, you implement a get statistics function. Registers 401000 ... 401045 are then filled with the system statistics.

System Statistics Overview

The following system statistics are available.

- Token bus controller (TBC)
 - Software-maintained receive statistics
 - TBC-maintained error counters
 - Software-maintained transmit errors
 - Software-maintained receive errors
 - User logic transaction errors
 - Manufacturing message format standard
 - (MMFS) errors
 - Background statistics
 - Software revision
-

**Token Bus
Controller (TBC)**

Registers 401000 ... 401003 are then filled with the following:

Register	Content
401000	Number of tokens passed by this station
401001	Number of tokens sent by this station
401002	Number of time the TBC has failed to pass token and has not found a successor
401003	Number of times the station has had to look for a new successor

**Software-
maintained
Receive
Statistics**

Registers 401004 ... 401010 are then filled with the following:

Register	Content
401004	TBC-detected error frames
401005	Invalid request with response frames
401006	Applications message too long
401007	Media access control (MAC) address out of range
401008	Duplicate application frames
401009	Unsupported logical link control (LLC) message types
401010	Unsupported LLC address

**TBC-maintained
Error Counters**

Registers 401011 ... 401018 are then filled with the following:

Register	Content
401011	Receive noise bursts (no start delimiter)
401012	Frame check sequence errors
401013	E-bit error in end delimiter
401014	Fragmented frames received (start delimiter not followed by end delimiter)
401015	Receive frames too long
401016	Discarded frames because there is no receive buffer
401017	Receive overruns
401018	Token pass failures

Software-maintained Transmit Errors

Registers 401019 ... 401020 are then filled with the following:

Register	Content
401019	Retries on request with response frames
401020	All retries performed and no response received from unit

Software-maintained Receive Errors

Registers 401021... 401022 are then filled with the following:

Register	Content
401021	Bad transmit request
401022	Negative transmit confirmation

User Logic Transaction Errors

Registers 401023... 401024 are then filled with the following:

Register	Content
401023	Message sent but no application response
401024	Invalid MBUS/PEER logic

Manufacturing Message Format Standard

Registers 401025... 401026 are then filled with the following:

Register	Content
401025	Command not executable
401026	Data not available

(MMFS) Errors

Registers 401027... 401035 are then filled with the following:

Register	Content
401027	Device not available
401028	Function not implemented
401029	Request not recognized
401030	Syntax error
401031	Unspecified error
401032	Data request out of bounds
401033	Request contains invalid controller address
401034	Request contains invalid data type
401035	None of the above

**Background
Statistics**

Registers 401036... 401043 are then filled with the following:

Register	Content
401036	Invalid MBUS/PEER request
401037	Number of unsupported MMFS message types received
401038	Unexpected response or response received after timeout
401039	Duplicate application responses received
401040	Response from unspecified device
401041	Number of responses buffered to be processed (in the least significant byte); number of MBUS/PEER requests to be processed (in the most significant byte)
401042	Number of received requests to be processed (in the least significant byte); number of transactions in process (in the most significant byte)
401043	S975 scan time in 10 ms increments

**Software
Revision**

Registers 401044... 401045 are then filled with the following:

Register	Content
401044	Version level of fixed software (PROMs): major version number in most significant byte; minor version number in least significant byte
401045	Version of loadable software (EEPROMs): major version number in most significant byte; minor version number in least significant byte

MRTM: Multi-Register Transfer Module

109

At a Glance

Introduction

This chapter describes the instruction MRTM.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	700
Representation: MRTM - Multi-Register Transfer Module	701
Parameter Description	702

Short Description

Function Description

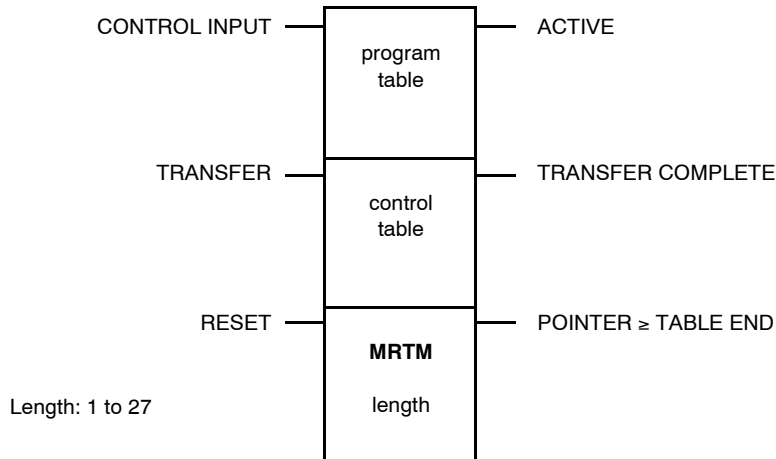
Note: This instruction is only available if you have unpacked and installed the DX Loadables. For further information, see *Installation of DX Loadables*, p. 109.

The MRTM instruction is used to transfer blocks of holding registers from the program table to the command block, a group of output registers. To verify each block transfer, an echo of the data contained in the first holding register is returned to an input register.

Representation: MRTM - Multi-Register Transfer Module

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables the operation
Middle input	0x, 1x	None	ON = one instruction block is transferred, table pointer of control table is increased by the value of "length"
Bottom input	0x, 1x	None	ON =reset
program table (top node)	0x, 1x, 3x, 4x	INT, UINT, WORD	First register of the program table. The digit 4 is assumed as the most significant digit
control table (middle node)	3x, 4x	INT, UINT, WORD	First register of the control table. The digit 4 is assumed as the most significant digit.
length (bottom node)		INT, UINT	Number of registers moved from the program table during each transfer, range: 1 to 127.
Top output	0x	None	Echoes state of the top input
Middle output	0x	None	Instruction block is transferred to the command block (stays on only for the remainder of the current scan)
Bottom output	0x	None	ON = pointer value ≥ table end

Parameter Description

Mode of Functioning

The MRTM transfers contiguous blocks of up to 127 registers from a table of register blocks to a block size holding register area. The MRTM function block controls the operation of the module in the following manner:

If power is applied to the...	Then ...
Top input	The function block is enabled for data transfers. Note: On initial startup, power must be applied to the bottom input.
Middle input	The function block attempts to transfer one instruction block. Before a transfer can occur, the echo register is evaluated. The most significant bit (MSB) of the echo register is not evaluated just bits 0 through 14. Echo mismatch is a condition that prohibits a transfer. If a transfer is permitted, one instruction block is transferred from the table starting at the table pointer. The table pointer in the control table is then advanced. If the pointer's new value is equal to or greater than the table end, the bottom output is turned on. A table pointer value less than the table end turns off the output.
Bottom input	The function block resets. The table pointer in the control table is reloaded with the start of commands value from the header of the program table

**Parameter
Description
Increment Step
(Middle Input)**

When power is applied, this input attempts to transfer one instruction block. Before a transfer can occur, the echo register is evaluated. The most significant bit (MSB) of the echo register is not evaluated, just bits 0 through 14. Echo mismatch is a condition that prohibits a transfer. If a transfer is permitted, one instruction block is transferred from the program table starting at the table pointer. The table pointer in the control table is then incremented by the value "Length" (displayed in the bottom node).

Note: The MRTM function block is designed to accept fault indications from I/O modules, which echo valid commands to the controller, but set a bit to indicate the occurrence of a fault. This method of fault indication is common for motion products and for most other I/O modules. If using a module that reports a fault condition in any other way, especially if the echo involved is not an echo of a valid command, special care must be taken when writing the error handler for the ladder logic to ensure the fault is detected. Failure to do so may result in a lockup or some other undesirable performance of the MRTM.

**Parameter
Description
Reset Pointer
(Bottom Input)**

When power is applied to this input, the function block is reset. The table pointer in the control table is reloaded with the start of commands value from the header of the program table.

MSPX (Seriplex)

110

At A Glance

Introduction

This chapter describes the instruction MSPX.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description: MSPX (Seriplex)	706
Representation: MSPX (Seriplex)	707

Short Description: MSPX (Seriplex)

Function

Description

The MSPX reads and writes bits within the base unit's registers.

The top node of the MSPX instruction represents the internal sub-function number. This node can be assigned a decimal constant value of 32 or a 4xxxx register containing the value of 32.

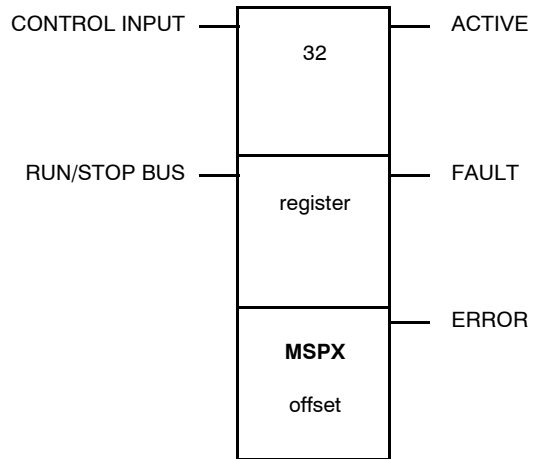
The middle node represents the starting 4xxxx register location for the SERIPLEX-MOMENTUM interface base unit.

The bottom node is interpreted as a numeric offset from 3000 indicating the first 3xxxx input register assigned to the interface base unit. The bottom node value specifies the location of the base unit's status register.

Representation: MSPX (Seriplex)

Symbol

Representation of the instruction



**Parameter
Description**

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	The Block Enable/Disable input enables and disables the operation of the MSPX block. When the associated logic is TRUE the top input is turned ON and the block's instructions are executed. The values of the base unit's input and output registers are not affected by the enabling or disabling of the block.
Middle input	0x, 1x	None	The Run/Stop Bus input regulates the operation of the Seriplex bus through the run/halt bit within the base unit's control register. The run/halt bit is set to 1 when the associated logic is TRUE, and cleared to 0 when the associated logic is FALSE. The parameters of this input are not to be altered while it is enabled or the run/halt bit will result in a configuration fault.
32 (top node)		INT, UINT	Represents the internal sub-function number. This node can be assigned a decimal constant value of 32 or a 4xxxx register containing the value of 32.
register (middle node)	4x	INT, UINT	Represents the starting 4xxxx register location for the SERIPLEX-MOMENTUM interface base unit.
offset (bottom node)	3x	INT, UINT	Interpreted as a numeric offset from 3000 indicating the first 3xxxx input register assigned to the interface base unit. The bottom node value specifies the location of the base unit's status register.
Top output	0x	None	The Bus Running Indicator output reports whether or not the Seriplex bus is running. If the bus running bit is ON, the output is TRUE and the Seriplex bus is operating normally, but, if the bus running bit is OFF, the output will be FALSE.
Middle output	0x	None	The Fault output reports if the MSPX instruction has experienced a fault condition other than a configuration fault. This will occur if any of the following status registers are ON: Bus fault (bit 3); MOMENTUM fault (bit 4); CDR error (bit 5). The detailed description of the detected fault can be determined by reading the base unit's status register.
Bottom output	0x	None	The Config Error output indicates that a configuration error has occurred, and its state is explained in the base unit's status register. When the config fault bit is ON the output becomes TRUE indicating that an improper attempt was made while writing to the base units control register.

MSTR: Master

111

At a Glance

Introduction

This chapter describes the instruction MSTR.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	711
Representation: MSTR - Master Instruction	712
Parameter Description	713
Write MSTR Operation	717
READ MSTR Operation	719
Get Local Statistics MSTR Operation	721
Clear Local Statistics MSTR Operation	723
Write Global Data MSTR Operation	725
Read Global Data MSTR Operation	726
Get Remote Statistics MSTR Operation	727
Clear Remote Statistics MSTR Operation	729
Peer Cop Health MSTR Operation	731
Reset Option Module MSTR Operation	734
Read CTE (Config Extension Table) MSTR Operation	736
Write CTE (Config Extension Table) MSTR Operation	738
Modbus Plus Network Statistics	740
TCP/IP Ethernet Statistics	745
Run Time Errors	746
Modbus Plus and SY/MAX Ethernet Error Codes	747
SY/MAX-specific Error Codes	749
TCP/IP Ethernet Error Codes	751
CTE Error Codes for SY/MAX and TCP/IP Ethernet	754

Short Description

**Function
Description**

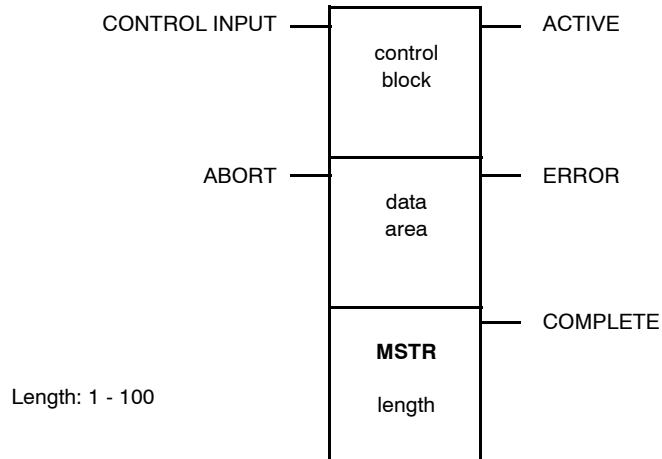
PLCs that support networking communications capabilities over Modbus Plus and Ethernet have a special MSTR (master) instruction with which nodes on the network can initiate message transactions.

The MSTR instruction allows you to initiate one of 12 possible network communications operations over the network.

- Read MSTR Operation
 - Write MSTR Operation
 - Get Local Statistics MSTR Operation
 - Clear Local Statistics MSTR Operation
 - Write Global Data MSTR Operation
 - Read Global Data MSTR Operation
 - Get Remote Statistics MSTR Operation
 - Clear Remote Statistics MSTR Operation
 - Peer Cop Health MSTR Operation
 - Reset Option Module MSTR Operation
 - Read CTE (Config Extension) MSTR Operation
 - Write CTE (Config Extension) MSTR Operation
-

Representation: MSTR - Master Instruction

Symbol Representation of the instruction



Parameter Description

For more information, see *Parameter Description*, p. 713.

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables selected MSTR operation
Middle input	0x, 1x	None	ON = terminates active MSTR operation
control block (top node)	4x	INT, UINT	Control block (first of several (network-dependant) contiguous holding registers)
data area (middle node)	4x	INT, UINT	Data area (source or destination depending on selected operation)
length (bottom node)		INT	Length of data area (maximum number of registers), range: 1 ... 100
Top output	0x	None	ON while the instruction is active (echoes state of the top input)
Middle output	0x	None	ON if the MSTR operation is terminated prior to completion (echoes state of the middle input)
Bottom output	0x	None	ON = operation successful

Parameter Description

Mode of Functioning

The MSTR instruction allows you to initiate one of 12 possible network communications operations over the network. Each operation is designated by a code.

Up to four MSTR instructions can be simultaneously active in a ladder logic program. More than four MSTRs may be programmed to be enabled by the logic flow; as one active MSTR block releases the resources it has been using and becomes deactivated, the next MSTR operation encountered in logic can be activated.

Master Operations

Certain MSTR operations are supported on some networks and not on others.

Code	Type of Operation	Modbus Plus	TCP/IP Ethernet	SY/MAX Ethernet
1	Write Data	x	x	x
2	Read Data	x	x	x
3	Get Local Statistics	x	x	-
4	Clear Local Statistics	x	x	-
5	Write Global Database	x	-	-
6	Read Global Database	x	-	-
7	Get Remote Statistics	x	x	-
8	Clear Remote Statistics	x	x	-
9	Peer Cop Health	x	-	-
10	Reset Option Module	-	x	x
11	Read CTE (config extension)	-	x	x
12	Write CTE (config extension)	-	x	x

Legend

x	supported
-	not supported

**Control Block
(Top Node)**

The 4x register entered in the top node is the first of several (network-dependant) holding registers that comprise the network control block.

The control block structure differs according to the network in use.

- Modbus Plus
- TCP/IP Ethernet
- SY/MAX Ethernet

Note: You need to understand the routing procedures used by the network you are using when you program an MSTR instruction. A full discussion of Modbus Plus routing path structures is given in Modbus Plus Planning and Installation Guide. If TCP/IP or SY/MAX Ethernet routing is being implemented, it must be accomplished via standard third-party Ethernet IP router products.

**Control Block for
Modbus Plus**

The first of twelve contiguous 4x registers is entered in the top node. The remaining eleven registers are implied:

Register	Content
Displayed	Identifies one of the nine MSTR operations legal for Modbus Plus (1 ... 9)
First implied	Displays error status
Second implied	Displays length (number of registers transferred)
Third implied	Displays MSTR operation-dependent information
Fourth implied	The Routing 1 register, used to designate the address of the destination node for a network transaction. The register display is implemented physically for the Quantum PLCs
Fifth implied	The Routing 2 register
Sixth implied	The Routing 3 register
Seventh implied	The Routing 4 register
Eighth implied	The Routing 5 register
Ninth implied	not applicable
Tenth implied	not applicable
Eleventh implied	not applicable

Routing 1 Register for Quantum Automation Series PLCs (Fourth Implied Register)

To target a Modbus Plus Network Option module (NOM) in a Quantum PLC backplane as the destination of an MSTR instruction, the value in the high byte represents the physical slot location of the NOM, e.g. if the NOM resides in slot 7 in the backplane, the high byte of routing register 1 would look like this:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Bit	Function								
1... 8	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td> </tr> </table> <p>High byte: indicating physical location (range 1 ... 16)</p>	0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1		
9 ... 16	<table border="1"> <tr> <td>0</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td> </tr> </table> <p>Destination address: binary value between 1 ... 64</p>	0	x	x	x	x	x	x	x
0	x	x	x	x	x	x	x		

Note: If you have created a logic program using an MSTR instruction for a 984 PLC and want to port it to a Quantum Automation Series PLC without having to edit the routing 1 register value, make sure that NOM #1 is installed in slot 1 of the Quantum backplane (and if a NOM #2 is used, that it is installed in slot 2 of the backplane). If you try to run the ported application with the NOMs in other slots without modifying the register, an F001 status error will appear, indicating the wrong destination node.

Control Block for TCP/IP Ethernet

The first of nine contiguous 4x registers is entered in the top node. The remaining eight registers are implied.

Register	Content
Displayed	Identifies one of the nine MSTR operations legal for TCP/IP (1 ... 4, 7, 8, 10 ... 12)
First implied	Displays error status
Second implied	Displays length (number of registers transferred)
Third implied	Displays MSTR operation-dependent information
Fourth implied	Low byte: slot address of the NOE module High byte: MBP-to-Ethernet Transporter (MET) Map index
Fifth implied	Byte 4 of the 32-bit destination IP Address
Sixth implied	Byte 3 of the 32-bit destination IP Address
Seventh implied	Byte 2 of the 32-bit destination IP Address
Eighth implied	Byte 1 of the 32-bit destination IP Address

**Control Block for
SY/MAX
EthernetEthernet**

The first of seven contiguous 4x registers is entered in the top node. The remaining six registers are implied.

Register	Content
Displayed	Identifies one of the nine MSTR operations legal for SY/MAX (1, 2, 10 ... 12)
First implied	Displays error status
Second implied	Displays Read/Write length (number of registers transferred)
Third implied	Displays Read/Write base address
Fourth implied	Low byte: slot address of the NOE module (e.g., slot 10 = 0A00, slot 6 = 0600) High byte: MBP-to-Ethernet Transporter (MET) Map index
Fifth implied	Destination drop number (or set to FF hex)
Sixth implied	Terminator (set to FF hex)

**Data Area
(Middle Node)**

The 4x register entered in the middle node is the first in a group of contiguous holding registers that comprise the data area. For operations that provide the communication processor with data, such as a Write operation, the data area is the source of the data. For operations that acquire data from the communication processor, such as a Read operation, the data area is the destination for the data.

In the case of the Ethernet Read and Write CTE operations, the middle node stores the contents of the Ethernet configuration extension table in a series of registers.

Write MSTR Operation

Short Description

An MSTR Write operation transfers data from a master source device to a specified slave destination device on the network. Read and Write use one data master transaction path and may be completed over multiple scans.

If you attempt to program the MSTR to Write its own station address, an error will be generated in the first implied register of the MSTR control block. It is possible to attempt a Write operation to a nonexistent register in the slave device. The slave will detect this condition and report it, this may take several scans.

Network Implementation

The MSTR Write operation can be implemented on the Modbus Plus, TCP/IP Ethernet, and SY/MAX Ethernet networks.

Control Block Utilization

In a Write operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using.

- Modbus Plus
- TCP/IP Ethernet
- SY/MAX Ethernet

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	1 = Write
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be sent to slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be written to (1 = 40001, 49 = 40049)
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device

**Control Block for
TCP/IP Ethernet**

Register	Function	Content
Displayed	Operation type	1 = Write
First implied	Error status	Displays a hex value indicating an MSTR error: Exception code + 3000 : Exception response, where response size is correct 4001 : Exception response, where response size is incorrect 4001 : Read/Write
Second implied	Length	Number of registers to be sent to slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be written to (1 = 40001, 49 = 40049)
Fourth implied	Low byte	Slot address of the network adapter module
Fifth ... eighth implied	Destination	Each register contains one byte of the 32-bit IP address

**Control Block for
SY/MAX Ethernet**

Register	Function	Content
Displayed	Operation type	1 = Write
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be sent to slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be written to (1 = 40001, 49 = 40049)
Fourth implied	Slot ID	Low byte: slot address of the network adapter module
Fourth implied	Slot ID	High byte: Destination drop number
Fifth ... eighth implied	Terminator	FF hex

READ MSTR Operation

Short Description

An MSTR Read operation transfers data from a specified slave source device to a master destination device on the network. Read and Write use one data master transaction path and may be completed over multiple scans.

If you attempt to program the MSTR to Read its own station address, an error will be generated in the first implied register of the MSTR control block. It is possible to attempt a Read operation to a nonexistent register in the slave device. The slave will detect this condition and report it, this may take several scans.

Network Implementation

The MSTR Read operation can be implemented on the Modbus Plus, TCP/IP Ethernet, and SY/MAX Ethernet networks.

Control Block Utilization

In a Read operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using.

- Modbus Plus
- TCP/IP Ethernet
- SY/MAX Ethernet

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	2 = Read
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be read from slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be read from (1 = 40001, 49 = 40049)
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device

Control Block for TCP/IP EthernetEthernet

Register	Function	Content
Displayed	Operation type	2 = Read
First implied	Error status	Displays a hex value indicating an MSTR error: Exception code + 3000 : Exception response, where response size is correct 4001 : Exception response, where response size is incorrect 4001 : Read/Write
Second implied	Length	Number of registers to be read from slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be read from (1 = 40001, 49 = 40049)
Fourth implied	High byte	Slot address of the network adapter module
Fifth ... eighth implied	Destination	Each register contains one byte of the 32-bit IP address

Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	2 = Read
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Number of registers to be read from slave
Third implied	Slave device data area	Specifies starting 4x register in the slave to be read from (1 = 40001, 49 = 40049)
Fourth implied	Slot ID	Low byte: slot address of the network adapter module
Fourth implied	Slot ID	High byte: Destination drop number
Fifth ... eighth implied	Terminator	FF hex

Get Local Statistics MSTR Operation

Short Description

The Get Local Statistics operation obtains information related to the local node, where the MSTR has been programmed. This operation takes one scan to complete and does not require a data master transaction path.

Network Implementation

The Get Local Statistics operation (type 3 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

The following network statistics are available.

- Modbus Plus network statistics
- TCP/IP Ethernet network statistics

Control Block Utilization

In a Get local statistics operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using.

- Modbus Plus
- TCP/IP Ethernet

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	3
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from offset, the number of words of statistics from the local processor's statistics table ; the length must be $> 0 \leq \text{data area}$
Third implied	Offset	An offset value relative to the first available word in the local processor's statistics table; if the offset is specified as 1, the function obtains statistics starting with the second word in the table
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 Note: If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

**Control Block for
TCP/IP Ethernet**

Register	Function	Content
Displayed	Operation type	3
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from offset, the number of words of statistics from the local processor's statistics table ; the length must be $> 0 \leq$ data area
Third implied	Offset	An offset value relative to the first available word in the local processor's statistics table, if the offset is specified as 1, the function obtains statistics starting with the second word in the table
Fourth implied	Slot ID	Low byte: Slot address of the network adapter module
Fifth ... Eighth implied	Not applicable	

Clear Local Statistics MSTR Operation

Short Description

The Clear local statistics operation clears statistics relative to the local node (where the MSTR has been programmed). This operation takes one scan to complete and does not require a data master transaction path.

Note: When you issue the Clear Local Statistics operation, only words 13 ... 22 in the statistics table are cleared.

Network Implementation

The Clear Local Statistics operation (type 4 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

The following network statistics are available.

- Modbus Plus network statistics
- TCP/IP Ethernet network statistics

Control Block Utilization

In a Clear local statistics operation, the registers in the MSTR control block (the top node) differ according to the type of network in use.

- Modbus Plus
- TCP/IP Ethernet

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	4
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied		Reserved
Third implied		Reserved
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 Note: If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

**Control Block for
TCP/IP Ethernet**

Register	Function	Content
Displayed	Operation type	4
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied		Reserved
Third implied		Reserved
Fourth implied	Slot ID	Low byte: Slot address of the network adapter module
Fifth ... Eighth implied		Reserved

Write Global Data MSTR Operation

Short Description

The Write global data operation transfers data to the communications processor in the current node so that it can be sent over the network when the node gets the token. All nodes on the local network link can receive this data. This operation takes one scan to complete and does not require a data master transaction path.

Network Implementation

The Write global data operation (type 5 in the displayed register of the top node) can be implemented only for Modbus Plus networks.

Control Block Utilization

The registers in the MSTR control block (the top node) are used in a Write global data operation

Register	Function	Content
Displayed	Operation type	5
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Specifies the number of registers from the data area to be sent to the comm processor; the value of the length must be ≤ 32 and must not exceed the size of the data area
Third implied		Reserved
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 Note: If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

Read Global Data MSTR Operation

Short Description

The Read global data operation gets data from the communications processor in any node on the local network link that is providing global data. This operation may require multiple scans to complete if global data is not currently available from the requested node. If global data is available, the operation completes in a single scan. No master transaction path is required.

Network Implementation

The Read global data operation (type 6 in the displayed register of the top node) can be implemented only for Modbus Plus networks.

Control Block Utilization

The registers in the MSTR control block (the top node) are used in a Read global data operation

Register	Function	Content
Displayed	Operation type	6
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Specifies the number of words of global data to be requested from the comm processor designated by the routing 1 parameter; the value of the length must be $> 0 \leq 32$ and must not exceed the size of the data area
Third implied	Available words	Contains the number of words available from the requested node; the value is automatically updated by internal software
Fourth implied	Routing 1	The low byte specifies the address of the node whose global data are to be returned (a value between 1 ... 64); if this is the second of two local nodes, set the high byte to a value of 1 Note: If your PLC does not support Modbus Plus option modules (S985s or NOMs), the high byte of the fourth implied register is not used and the highbyte bits must all be set to 0.

Get Remote Statistics MSTR Operation

Short Description

The Get Remote Statistics operation obtains information relative to remote nodes on the network. This operation may require multiple scans to complete and does not require a master data transaction path.

Network Implementation

The Get Remote Statistics operation (type 7 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

Control Block Utilization

In a Get remote statistics operation, the registers in the MSTR control block (the top node) contain the information that differs depending on the type of network you are using.

- Modbus Plus
- TCP/IP Ethernet

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	7
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from an offset, the number of words of statistics to be obtained from a remote node; the length must be $> 0 \leq$ total number of statistics available (54) and must not exceed the size of the data area
Third implied	Offset	Specifies an offset value relative to the first available word in the statistics table, the value must not exceed the number of statistic words available.
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device.

The remote comm processor always returns its complete statistics table when a request is made, even if the request is for less than the full table. The MSTR instruction then copies only the amount of words you have requested to the designated 4x registers.

**Control Block for
TCP/IP Ethernet**

Register	Function	Content
Displayed	Operation type	7
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Length	Starting from offset, the number of words of statistics from the local processor's statistics table; the length must be $> 0 \leq$ data area
Third implied	Offset	An offset value relative to the first available word in the local processor's statistics table, if the offset is specified as 1, the function obtains statistics starting with the second word in the table
Fourth implied	Low byte	Slot address of the network adapter module
Fifth ... Eighth implied	Destination	Each register contains one byte of the 32-bit IP address

Clear Remote Statistics MSTR Operation

Short Description

The Clear remote statistics operation clears statistics related to a remote network node from the data area in the local node. This operation may require multiple scans to complete and uses a single data master transaction path.

Note: When you issue the Clear Remote Statistics operation, only words 13 ... 22 in the statistics table are cleared

Network Implementation

The Clear remote statistics operation (type 8 in the displayed register of the top node) can be implemented for Modbus Plus and TCP/IP Ethernet networks. It is not used for SY/MAX Ethernet.

The following network statistics are available.

- Modbus Plus network statistics
- TCP/IP Ethernet network statistics

Control Block Utilization

In a Clear remote statistics operation, the registers in the MSTR control block (the top node) contain information that differs according to the type of network in use.

- Modbus Plus
- TCP/IP Ethernet

Control Block for Modbus Plus

Register	Function	Content
Displayed	Operation type	8
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied		Reserved
Third implied		Reserved
Fourth ... Eighth implied	Routing 1 ... 5	Designates the first ... fifth routing path addresses, respectively; the last nonzero byte in the routing path is the destination device

**Control Block for
TCP/IP Ethernet**

Register	Function	Content
Displayed	Operation type	8
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Low byte	Slot address of the network adapter module
Fifth ... Eighth implied	Destination	Each register contains one byte of the 32-bit IP address

Peer Cop Health MSTR Operation

Short Description

The peer cop health operation reads selected data from the peer cop communications health table and loads that data to specified 4x registers in state RAM. The peer cop communications health table is 12 words long, and the words are indexed via this MSTR operation as words 0 ... 11.

Network Implementation

The peer cop health operation (type 9) in the displayed register of the top node) can be implemented only for Modbus Plus networks.

Control Block Utilization

The registers in the MSTR control block (the top node) are used in a Peer cop health operation:

Register	Function	Content
Displayed	Operation type	9
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Data size	Number of words requested from peer cop table (range 1 ... 12)
Third implied	Index	First word from the table to be read (range 0 ... 11, where 0 = the first word in the peer cop table and 11 = the last word in the table)
Fourth implied	Routing 1	If this is the second of two local nodes, set the high byte to a value of 1 Note: If your PLC does not support Modbus Plus option modules (S985s or NOMs), the fourth implied register is not used.

Peer Cop Communications Health Status Information

The peer cop communications health table comprises 12 contiguous registers that can be indexed in an MSTR operation as words 0 ... 11. Each bit in each of the table words is used to represent an aspect of communications health relative to a specific node on the Modbus Plus network.

**Bit-to-Network
Node
Relationship**

The bits in words 0 ... 3 represent the health of the global input communication expected from nodes 1 ... 64. The bits in words 4 ... 7 represent the health of the output from a specific node. The bits in words 8 ... 11 represent the health of the input to a specific node:

Type of Status	Word Index	Bit-to-network Node Relationship
Global Input	0	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	1	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	2	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	3	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
Specific Output	4	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	5	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	6	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	7	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
Specific Input	8	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	9	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	10	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	11	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49

**State of a Peer
Cop Health Bit**

The state of a peer cop health bit reflects the current communication status of its associated node. A health bit is set when its associated node accepts inputs for its peer copped input data group or hears that another node has accepted specific output data from the its peer copped output data group. A health bit is cleared when no communication has occurred for its associated data group within the configured peer cop health time-out period.

All health bits are cleared when the Put Peer Cop interface command is executed at PLC start-up time. Table values are not valid until at least one full token rotation cycle has been completed after execution of the Put Peer Cop interface command. The health bit for a given node is always zero when its associated peer cop entry is null.

Reset Option Module MSTR Operation

Short Description

The Reset option module operation causes a Quantum NOE option module to enter a reset cycle to reset its operational environment.

Network Implementation

The Reset option module operation (type 10 in the displayed register of the top node) can be implemented for TCP/IP and SY/MAX Ethernet networks, accessed via the appropriate network adapter. Modbus Plus networks do not use this operation.

Control Block Utilization

In a Reset option module operation, the registers in the MSTR control block (the top node) differ according to the type of network in use.

- TCP/IP Ethernet
 - SY/MAX Ethernet
-

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	10
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Slot ID	Number displayed in the low byte, in the range 1 ... 16 indicating the slot in the local backplane where the option module resides
Fifth ... Eighth implied	Not applicable	

**Control Block for
SY/MAX Ethernet**

Register	Function	Content
Displayed	Operation type	10
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Slot ID	Low byte: slot address of the network adapter module
Fifth ... Eighth implied	Not applicable	

Read CTE (Config Extension Table) MSTR Operation

Short Description

The Read CTE operation reads a given number of bytes from the Ethernet configuration extension table to the indicated buffer in PLC memory. The bytes to be read begin at a byte offset from the beginning of the CTE. The content of the Ethernet CTE table is displayed in the middle node of the MSTR block.

Network Implementation

The Read CTE operation (type 11 in the displayed register of the top node) can be implemented for TCP/IP and SY/MAX Ethernet networks, accessed via the appropriate network adapter. Modbus Plus networks do not use this operation.

Control Block Utilization

In a Read CTE operation, the registers in the MSTR control block (the top node) differ according to the type of network in use.

- TCP/IP Ethernet
- SY/MAX Ethernet

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	11
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Map index	Either a value displayed in the high byte of the register or not used
	Slot ID	Number displayed in the low byte, in the range 1 ... 16 indicating the slot in the local backplane where the option module resides
Fifth ... Eighth implied	Not applicable	

Control Block for SY/MAX Ethernet

Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	11
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Data Size	Number of words transferred
Third implied	Base Address	Byte offset in PLC register structure indicating where the CTE bytes will be written
Fourth implied	Low byte	Slot address of the NOE module
	High byte	Terminator (FF hex)
Fifth ... Eighth implied	Not applicable	

CTE Display Implementation (Middle Node)

The values in the Ethernet configuration extension table (CTE) are displayed in a series of registers in the middle node of the MSTR instruction when a Read CTE operation is implemented. The middle node contains the first of 11 contiguous 4x registers.

The registers display the following CTE data.

Parameter	Register	Content
Frame type	Displayed	1 = 802.3 2 = Ethernet
IP address	First implied	First byte of the IP address
	Second implied	Second byte of the IP address
	Third implied	Third byte of the IP address
	Fourth implied	Fourth byte of the IP address
Subnetwork mask	Fifth implied	Hi word
	Sixth implied	Low word
Gateway	Seventh implied	First byte of the gateway
	Eighth implied	Second byte of the gateway
	Ninth implied	Third byte of the gateway
	Tenth implied	Fourth byte of the gateway

Write CTE (Config Extension Table) MSTR Operation

Short Description

The Write CTE operation writes the configuration CTE table from the data specified in the middle node to an indicated Ethernet configuration extension table or a specified slot.

Network Implementation

The Write CTE operation (type 12 in the displayed register of the top node) can be implemented for TCP/IP and SY/MAX Ethernet networks, via the appropriate network adapter. Modbus Plus networks do not use this operation.

Control Block Utilization

In a Write CTE operation, the registers in the MSTR control block (the top node) differ according to the type of network in use.

- TCP/IP Ethernet
- SY/MAX Ethernet

Control Block for TCP/IP Ethernet

Register	Function	Content
Displayed	Operation type	12
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Not applicable	
Third implied		
Fourth implied	Map index	Either a value displayed in the high byte of the register or not used
	Slot ID	Number displayed in the low byte, in the range 1 ... 16 indicating the slot in the local backplane where the option module resides
Fifth ... Eighth implied	Not applicable	

Control Block for SY/MAX Ethernet

Register	Function	Content
Displayed	Operation type	12
First implied	Error status	Displays a hex value indicating an MSTR error, when relevant
Second implied	Data Size	Number of words transferred
Third implied	Base Address	Byte offset in PLC register structure indicating where the CTE bytes will be written
Fourth implied	Low byte	Slot address of the NOE module
	High byte	Destination drop number
Fifth implied	Terminator	FF hex
Sixth ... Eighth implied	Not applicable	

CTE Display Implementation (Middle Node)

The values in the Ethernet configuration extension table (CTE) are displayed in a series of registers in the middle node of the MSTR instruction when a Write CTE operation is implemented. The middle node contains the first of 11 contiguous 4x registers.

The registers are used to transfer the following CTE data.

Parameter	Register	Content
Frame type	Displayed	1 = 802.3 2 = Ethernet
IP address	First implied	First byte of the IP address
	Second implied	Second byte of the IP address
	Third implied	Third byte of the IP address
	Fourth implied	Fourth byte of the IP address
Subnetwork mask	Fifth implied	Hi word
	Sixth implied	Low word
Gateway	Seventh implied	First byte of the gateway
	Eighth implied	Second byte of the gateway
	Ninth implied	Third byte of the gateway
	Tenth implied	Fourth byte of the gateway

Modbus Plus Network Statistics

Modbus Plus Network Statistics

The following table shows the statistics available on the Modbus Plus network. You may acquire this information by using the appropriate MSTR operation or by using Modbus function code 8.

Note: When you issue the Clear local or Clear remote statistics operations, only words 13 ... 22 are cleared.

Modbus Plus Network Statistics

Word	Bits	Meaning																															
00		Node type ID																															
	0	Unknown node type																															
	1	PLC node																															
	2	Modbus bridge node																															
	3	Host computer node																															
	4	Bridge Plus node																															
	5	Peer I/O node																															
01	0 ... 11	Software version number in hex (to read, strip bits 12-15 from word)																															
	12 ... 14	Reserved																															
	15	Defines Word 15 error counters (see Word 15) Most significant bit defines use of error counters in Word 15. Least significant half of upper byte, plus lower byte, contain software version: <div style="margin-left: 40px;"> <table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <tr> <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><td>0</td> </tr> <tr> <td colspan="11" style="border: none;"></td> <td colspan="5" style="border: none;">Software version number (in HEX)</td> </tr> </table> </div> Word 15 error counter (see word 15)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												Software version number (in HEX)			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
											Software version number (in HEX)																						
02		Network address for this station																															

Word	Bits	Meaning
03		MAC state variable:
	0	Power up state
	1	Monitor offline state
	2	Duplicate offline state
	3	Idle state
	4	Use token state
	5	Work response state
	6	Pass token state
	7	Solicit response state
	8	Check pass state
	9	Claim token state
10	Claim response state	
04		Peer status (LED code); provides status of this unit relative to the network:
	0	Monitor link operation
	32	Normal link operation
	64	Never getting token
	96	Sole station
128	Duplicate station	
05		Token pass counter; increments each time this station gets the token
06		Token rotation time in ms
07	LO	Data master failed during token ownership bit map
	HI	Program master failed during token ownership bit map
08	LO	Data master token owner work bit map
	HI	Program master token owner work bit map
09	LO	Data slave token owner work bit map
	HI	Program slave token owner work bit map
10	HI	Data slave/get slave command transfer request bit map
11	LO	Program master/get master rsp transfer request bit map
	HI	Program slave/get slave command transfer request bit map
12	LO	Program master connect status bit map
	HI	Program slave automatic logout request bit map
13	LO	Pretransmit deferral error counter
	HI	Receive buffer DMA overrun error counter

Word	Bits	Meaning
14	LO	Repeated command received counter
	HI	Frame size error counter
15		If Word 1 bit 15 is not set , Word 15 has the following meaning:
	LO	Receiver collision-abort error counter
	HI	Receiver alignment error counter
		If Word 1 bit 15 is set , Word 15 has the following meaning:
	LO	Cable A framing error
	HI	Cable B framing error
16	LO	Receiver CRC error counter
	HI	Bad packet-length error counter
17	LO	Bad link-address error counter
	HI	Transmit buffer DMA-underrun error counter
18	LO	Bad internal packet length error counter
	HI	Bad MAC function code error counter
19	LO	Communication retry counter
	HI	Communication failed error counter
20	LO	Good receive packet success counter
	HI	No response received error counter
21	LO	Exception response received error counter
	HI	Unexpected path error counter
22	LO	Unexpected response error counter
	HI	Forgotten transaction error counter
23	LO	Active station table bit map, nodes 1 ... 8
	HI	Active station table bit map, nodes 9 ...16
24	LO	Active station table bit map, nodes 17 ... 24
	HI	Active station table bit map, nodes 25 ... 32
25	LO	Active station table bit map, nodes 33 ... 40
	HI	Active station table bit map, nodes 41 ... 48
26	LO	Active station table bit map, nodes 49 ... 56
	HI	Active station table bit map, nodes 57 ... 64
27	LO	Token station table bit map, nodes 1 ... 8
	HI	Token station table bit map, nodes 9 ... 16
28	LO	Token station table bit map, nodes 17 ... 24
	HI	Token station table bit map, nodes 25 ... 32

Word	Bits	Meaning
29	LO	Token station table bit map, nodes 33 ... 40
	HI	Token station table bit map, nodes 41 ... 48
30	LO	Token station table bit map, nodes 49 ... 56
	HI	Token station table bit map, nodes 57 ... 64
31	LO	Global data present table bit map, nodes 1 ... 8
	HI	Global data present table bit map, nodes 9 ... 16
32	LO	Global data present table bit map, nodes 17 ... 24
	HI	Global data present table bit map, nodes 25 ... 32
33	LO	Global data present table bit map, nodes 33 ... 40
	HI	Global data present table bit map, nodes 41 ... 48
34	LO	Global data present table map, nodes 49 ... 56
	HI	Global data present table bit map, nodes 57 ... 64
35	LO	Receive buffer in use bit map, buffer 1-8
	HI	Receive buffer in use bit map, buffer 9 ... 16
36	LO	Receive buffer in use bit map, buffer 17 ... 24
	HI	Receive buffer in use bit map, buffer 25 ... 32
37	LO	Receive buffer in use bit map, buffer 33 ... 40
	HI	Station management command processed initiation counter
38	LO	Data master output path 1 command initiation counter
	HI	Data master output path 2 command initiation counter
39	LO	Data master output path 3 command initiation counter
	HI	Data master output path 4 command initiation counter
40	LO	Data master output path 5 command initiation counter
	HI	Data master output path 6 command initiation counter
41	LO	Data master output path 7 command initiation counter
	HI	Data master output path 8 command initiation counter
42	LO	Data slave input path 41 command processed counter
	HI	Data slave input path 42 command processed counter
43	LO	Data slave input path 43 command processed counter
	HI	Data slave input path 44 command processed counter
44	LO	Data slave input path 45 command processed counter
	HI	Data slave input path 46 command processed counter
45	LO	Data slave input path 47 command processed counter
	HI	Data slave input path 48 command processed counter

Word	Bits	Meaning
46	LO	Program master output path 81 command initiation counter
	HI	Program master output path 82 command initiation counter
47	LO	Program master output path 83 command initiation counter
	HI	Program master output path 84 command initiation counter
48	LO	Program master command initiation counter
	HI	Program master output path 86 command initiation counter
49	LO	Program master output path 87 command initiation counter
	HI	Program master output path 88 command initiation counter
50	LO	Program slave input path C1 command processed counter
	HI	Program slave input path C2 command processed counter
51	LO	Program slave input path C3 command processed counter
	HI	Program slave input path C4 command processed counter
52	LO	Program slave input path C5 command processed counter
	HI	Program slave input path C6 command processed counter
53	LO	Program slave input path C7 command processed counter
	HI	Program slave input path C8 command processed counter

TCP/IP Ethernet Statistics

TCP/IP Ethernet Statistics

A TCP/IP Ethernet board responds to Get Local Statistics and Set Local Statistics commands with the following information:

Word	Meaning	
00 ... 02	MAC address, e.g., if the MAC address is 00 00 54 00 12 34, it is displayed as follows:	
	Word	Content
	00	00 00
	01	00 54
	02	34 12
03	Board status	Meaning
	0x0001	Running
	0x4000	APPI LED (1=ON, 0 = OFF)
	0x8000	Link LED
04 and 05	Number of receiver interrupts	
06 and 07	Number of transmitter interrupts	
08 and 09	Transmit-timeout error count	
10 and 11	Collision-detect error count	
12 and 13	Missed packets	
14 and 15	Memory error count	
16 and 17	Number of times driver has restarted lance	
18 and 19	Receive framing error count	
20 and 21	Receiver overflow error count	
22 and 23	Receive CRC error count	
24 and 25	Receive buffer error count	
26 and 27	Transmit buffer error count	
28 and 29	Transmit silo underflow count	
30 and 31	Late collision count	
32 and 33	Lost carrier count	
34 and 35	Number of retries	
36 and 37	IP address, e.g., if the IP address is 198.202.137.113 (or c6 CA 89 71), it is displayed as follows:	
	Word	Content
	36	89 71
	37	C6 CA

Run Time Errors

Runtime Errors

If an error occurs during a MSTR operation, a hexadecimal error code will be displayed in the first implied register in the control block (the top node).

Function error codes are network-specific.

- Modbus Plus and SY/MAX Ethernet Error Codes
 - SY/MAX-specific Error Codes
 - TCP/IP Ethernet Error Codes
 - CTE Error Codes for SY/MAX and TCP/IP Ethernet
-

Modbus Plus and SY/MAX Ethernet Error Codes

Form of the Function Error Code

The form of the function error code for Modbus Plus and SY/MAX Ethernet transactions is **Mmss**, where

- **M** represents the major code
- **m** represents the minor code
- **ss** represents a subcode

Hexadecimal Error Code

Hex Error Code	Meaning
1001	User has aborted the MSTR element
2001	An unsupported operation type has been specified in the control block
2002	One or more control block parameter has been changed while the MSTR element is active (applies only to operations that take multiple scans to complete). Control block parameters may be changed only when the MSTR element is not active.
2003	Invalid value in the length field of the control block
2004	Invalid value in the offset field of the control block
2005	Invalid values in the length and offset fields of the control block
2006	Invalid slave device data area
2007	Invalid slave device network area
2008	Invalid slave device network routing
2009	Route equal to your own address
200A	Attempting to obtain more global data words than available
30ss	Modbus slave exception response
4001	Inconsistent Modbus slave response
5001	Inconsistent network response
6mss)	Routing failure

ss HEX Value in Error Code 30ss

The ss subfield in error code 30ss is:

ss Hex Value	Meaning
01	Slave device does not support the requested operation
02	Nonexistent slave device registers requested
03	Invalid data value requested
04	Reserved
05	Slave has accepted long-duration program command
06	Function can't be performed now: a long-duration command in effect
07	Slave rejected long-duration program command
08 ... 255	Reserved

ss Hex Value in Error Code 6mss

The m subfield in error code 6mss is an index into the routing information indicating where an error has been detected (a value of 0 indicates the local node, a 2 the second device on the route, etc.).

The ss subfield in error code 6mss is:

ss Hex Value	Meaning
01	No response received
02	Program access denied
03	Node off-line and unable to communicate
04	Exception response received
05	Router node data paths busy
06	Slave device down
07	Bad destination address
08	Invalid node type in routing path
10	Slave has rejected the command
20	Initiated transaction forgotten by slave device
40	Unexpected master output path received
80	Unexpected response received
F001	Wrong destination node specified for the MSTR operation

SY/MAX-specific Error Codes

Types or Errors Three additional types of errors may be reported in the MSTR instruction when SY/MAX Ethernet is being used.

The error codes have the following designations:

- 71xx errors: Errors detected by the remote SY/MAX device
- 72xx errors: Errors detected by the serve
- 73xx errors: Errors detected by the Quantum translator

Hexadecimal Error Code SY/MAX-specific

HEX Error Code SY/MAX-specific:

Hex Error Code	Meaning
7101	Illegal opcode detected by the remote SY/MAX device
7103	Illegal address detected by the remote SY/MAX device
7109	Attempt to write a read-only register detected by the remote SY/MAX device
710F	Receiver overflow detected by the remote SY/MAX device
7110	Invalid length detected by the remote SY/MAX device
7111	Remote device inactive, not communicating (occurs after retries and time-out have been exhausted) detected by the remote SY/MAX device
7113	Invalid parameter on a read operation detected by the remote SY/MAX device
711D	Invalid route detected by the remote SY/MAX device
7149	Invalid parameter on a write operation detected by the remote SY/MAX device
714B	Illegal drop number detected by the remote SY/MAX device
7201	Illegal opcode detected by the SY/MAX server
7203	Illegal address detected by the SY/MAX server
7209	Attempt to write to a read-only register detected by the SY/MAX server
720F	Receiver overflow detected by the SY/MAX server
7210	Invalid length detected by the SY/MAX server
7211	Remote device inactive, not communicating (occurs after retries and time-out have been exhausted) detected by the SY/MAX server
7213	Invalid parameter on a read operation detected by the SY/MAX server
721D	Invalid route detected by the SY/MAX server
7249	Invalid parameter on a write operation detected by the SY/MAX server
724B	Illegal drop number detected by the SY/MAX server
7301	Illegal opcode in an MSTR block request by the Quantum translator
7303	Read/Write QSE module status (200 route address out of range)

Hex Error Code	Meaning
7309	Attempt to write to a read-only register when performing a status write (200 route)
731D	Invalid rout detected by Quantum translator Valid routes are: <ul style="list-style-type: none">● dest_drop, 0xFF● 200, dest_drop, 0xFF● 100+drop, dest_drop, 0xFF All other routing values generate an error
734B	One of the following errors has occurred: <ul style="list-style-type: none">● No CTE (configuration extension) table was configured● No CTE table entry was created for the QSE Module slot number● No valid drop was specified● The QSE Module was not reset after the CTE was created Note: After writing and configuring the CTE and downloading it to the QSE Module, you must reset the QSE Module to make the changes take effect. <ul style="list-style-type: none">● When using an MSTR instruction, no valid slot or drop was specified

TCP/IP Ethernet Error Codes

Error in an MSTR Routine

An error in an MSTR routine over TCP/IP Ethernet may produce one of the following errors in the MSTR control block.

The form of the code is **Mmss**, where

- **M** represents the major code
- **m** represents the minor code
- **ss** represents a subcode

Hexadecimal Error Code for MSTR Routine over TCP/IP Ethernet

Hex Error Code	Meaning
1001	User has aborted the MSTR element
2001	An unsupported operation type has been specified in the control block
2002	One or more control block parameter has been changed while the MSTR element is active (applies only to operations that take multiple scans to complete). Control block parameters may be changed only when the MSTR element is not active
2003	Invalid value in the length field of the control block
2004	Invalid value in the offset field of the control block
2005	Invalid values in the length and offset fields of the control block
2006	Invalid slave device data area
3000	Generic Modbus fail code
30ss	Modbus slave exception response
4001	Inconsistent Modbus slave response

ss Hex Value in Error Code 30ss

The ss subfield in error code 30ss is:

ss Hex Value	Meaning
01	Slave device does not support the requested operation
02	Nonexistent slave device registers requested
03	Invalid data value requested
04	Reserved
05	Slave has accepted long-duration program command
06	Function can't be performed now: a long-duration command in effect
07	Slave rejected long-duration program command

**HEX Error Code
TCP/IP Ethernet
Network**

An error on the TCP/IP Ethernet network itself may produce one of the following errors in the MSTR control block.

Hex Error Code	Meaning
5004	Interrupted system call
5005	I/O error
5006	No such address
5009	The socket descriptor is invalid
500C	Not enough memory
500D	Permission denied
5011	Entry exists
5016	An argument is invalid
5017	An internal table has run out of space
5020	The connection is broken
5023	This operation would block and the socket is nonblocking
5024	The socket is nonblocking and the connection cannot be completed
5025	The socket is nonblocking and a previous connection attempt has not yet completed
5026	Socket operation on a nonsocket
5027	The destination address is invalid
5028	Message too long
5029	Protocol wrong type for socket
502A	Protocol not available
502B	Protocol not supported
502C	Socket type not supported
502D	Operation not supported on socket
502E	Protocol family not supported
502F	Address family not supported
5030	Address is already in use
5031	Address not available
5032	Network is down
5033	Network is unreachable
5034	Network dropped connection on reset
5035	The connection has been aborted by the peer
5036	The connection has been reset by the peer
5037	An internal buffer is required, but cannot be allocated
5038	The socket is already connected

Hex Error Code	Meaning
5039	The socket is not connected
503A	Can't send after socket shutdown
503B	Too many references; can't splice
503C	Connection timed out
503D	The attempt to connect was refused
5040	Host is down
5041	The destination host could not be reached from this node
5042	Directory not empty
5046	NI_INIT returned -1
5047	The MTU is invalid
5048	The hardware length is invalid
5049	The route specified cannot be found
504A	Collision in select call; these conditions have already been selected by another task
504B	The task id is invalid
F001	In Reset mode

CTE Error Codes for SY/MAX and TCP/IP Ethernet

CTE Error Codes for SY/MAX and TCP/IP Ethernet

HEX Error Code MSTR routine over TCP/IP Ethernet:

Hex Error Code	Meaning
7001	The is no Ethernet configuration extension
7002	The CTE is not available for access
7003	The offset is invalid
7004	The offset + length is invalid
7005	Bad data field in the CTE

MU16: Multiply 16 Bit

112

At a Glance

Introduction

This chapter describes the instruction MU16.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	756
Representation: MU16 - 16-Bit Multiplication	757

Short Description

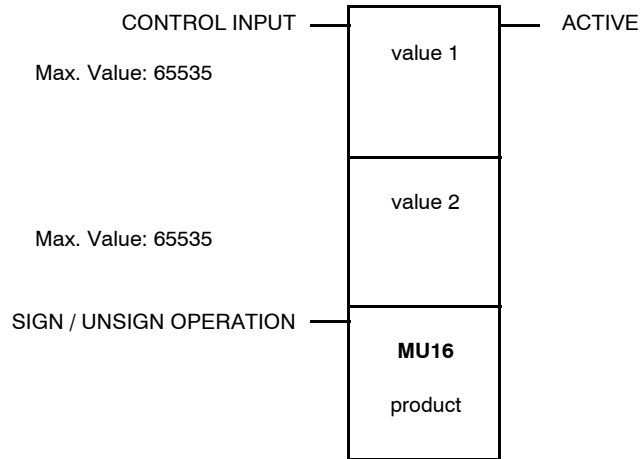
**Function
Description**

The MU16 instruction performs signed or unsigned multiplication on the 16-bit values in the top and middle nodes, then posts the product in two contiguous holding registers in the bottom node.

Representation: MU16 - 16-Bit Multiplication

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables value 1 x value 2
Bottom input	0x, 1x	None	ON = signed operation OFF = unsigned operation
value 1 (top node)	3x, 4x	INT, UINT	Multiplicand, can be displayed explicitly as an integer (range 1 ... 65 535, enter e.g. #65535) or stored in a register
value 2 (middle node)	3x, 4x	INT, UINT	Multiplier, can be displayed explicitly as an integer (range 1 ... 65 535) or stored in a register
product (bottom node)	4x	INT, UINT	First of two contiguous holding registers: displayed register contains half of the product and the implied register contains the other half
Top output	0x	None	Echoes the state of the top input

MUL: Multiply

113

At a Glance

Introduction

This chapter describes the instruction MUL.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	760
Representation: MUL - Single Precision Multiplication	761
Example	762

Short Description

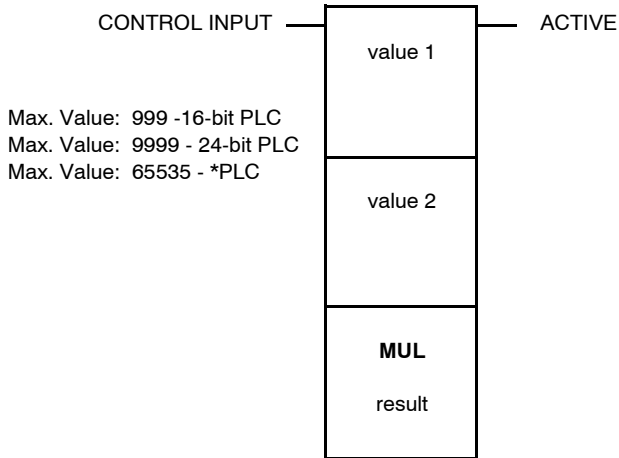
**Function
Description**

The MUL instruction multiplies unsigned value 1 (its top node) by unsigned value 2 (its middle node) and stores the product in two contiguous holding registers in the bottom node.

Representation: MUL - Single Precision Multiplication

Symbol

Representation of the instruction



*Available on the following

- E685/785 PLCs
- L785 PLCs
- Quantum Series PLCs

Parameter Description

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = value 1 multiplied by value 2
value 1 (top node)	3x, 4x	UINT	Multiplicand, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register Max. Value: 999 -16-bit PLC Max. Value: 9999 - 24-bit PLC Max. Value: 65535 - *PLC
value 2 (middle node)	3x, 4x	UINT	Multiplier, can be displayed explicitly as an integer (range 1 ... 9 999) or stored in a register Max. Value: 999 -16-bit PLC Max. Value: 9999 - 24-bit PLC Max. Value: 65535 - *PLC
result (bottom node)	4x	UINT	Product (first of two contiguous holding registers; displayed: high-order digits; implied: low-order digits)
Top output	0x	None	Echoes the state of the top input

Example

Product of Instruction MUL

For example, if value 1 = 8 000 and value 2 = 2, the product is 16 000. The displayed register contains the value 0001 (the high-order half of the product), and implied register contains the value 6 000 (the low-order half of the product).

NBIT: Bit Control

114

At a Glance

Introduction

This chapter describes the instruction NBIT.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	764
Representation: NBIT - Normal Bit	765

Short Description

Function Description

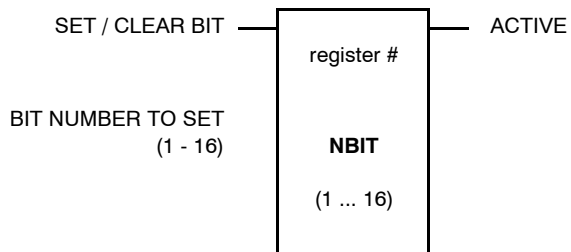
The normal bit (NBIT) instruction lets you control the state of a bit from a register by specifying its associated bit number in the bottom node. The bits being controlled are similar to coils, when a bit is turned ON, it stays ON until a control signal turns it OFF.

Note: The NBIT instruction does not follow the same rules of network placement as 0x-referenced coils do. An NBIT instruction cannot be placed in column 11 of a network and it can be placed to the left of other logic nodes on the same rungs of the ladder.

Representation: NBIT - Normal Bit

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = sets the specified bit to 1 OFF = clears the specified bit to 0
register # (top node)	4x	WORD	Holding register whose bit pattern is being controlled
bit # (bottom node)		INT, UINT	Indicates which one of the 16 bits is being controlled
Top output	0x	None	Echoes the state of the top input: ON = top input ON and specified bit set to 1 OFF = top input OFF and specified bit set to 0

NCBT: Normally Closed Bit

115

At a Glance

Introduction

This chapter describes the instruction NCBT.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	768
Representation: NCBT - Bit Normally Closed	769

Short Description

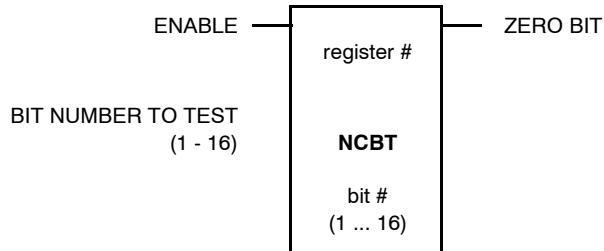
**Function
Description**

The normally closed bit (NCBT) instruction lets you sense the logic state of a bit in a register by specifying its associated bit number in the bottom node. The bit is representative of an N.C contact. It passes power from the top output when the specified bit is OFF and the top input is ON.

Representation: NCBT - Bit Normally Closed

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables bit sensing
register # (top node)	3x, 4x	WORD	Register whose bit pattern is being used to represent N.C. contacts
bit # (bottom node)		INT, UINT	(Indicates which one of the 16 bits is being sensed)
Top output	0x	None	ON = top input is ON and specified bit is OFF (logic state 0)

NOBT: Normally Open Bit

116

At a Glance

Introduction

This chapter describes the instruction NOBT.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Short Description	772
Representation: NOBT - Bit Normally Open	773

Short Description

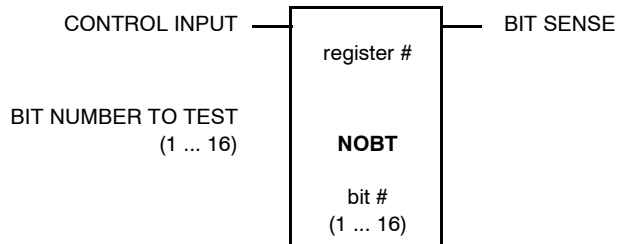
**Function
Description**

The normally open bit (NOBT) instruction lets you sense the logic state of a bit in a register by specifying its associated bit number in the bottom node. The bit is representative of an N.O contact.

Representation: NOBT - Bit Normally Open

Symbol

Representation of the instruction



Parameter Description

Description of the instruction's parameters

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = enables bit sensing
register # (top node)	3x, 4x	WORD	Register whose bit pattern is being used to represent N.O. contacts
bit # (bottom node)		INT, UINT	(Indicates which one of the 16 bits is being sensed)
Top output	0x	None	ON = top input is ON and specified bit is ON (logic state 1)

NOL: Network Option Module for Lonworks

117

At a Glance

Introduction

This chapter describes the instruction NOL.

What's in this Chapter?

This chapter contains the following topics:


Topic	Page
Short Description	776
Representation: NOL - Network Option Module for Lonworks	777
Detailed Description	778

Short Description

Function Requirements

The following steps are necessary before using this instruction:

Step	Action
1	Add loadable NSUP.exe to the controller's configuration Note: This loadable needs only be loaded once to support multiple loadables, such as ECS.exe and XMIT.exe.

	CAUTION
	<p>The outputs of the instruction turn on, regardless of the input states</p> <p>When the NSUP loadable is not installed or is installed after the NOL loadable or is installed in a Quantum PLC with an executive < V2.0, all three outputs turn on, regardless of the input states.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

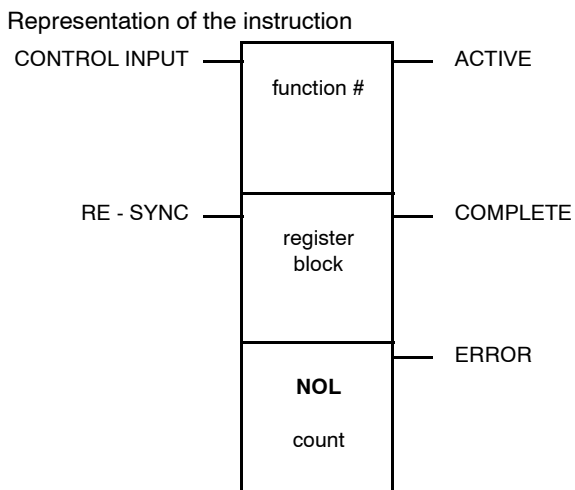
Step	Action
2	Unpack and install the DX Loadable NOL. For more information, see <i>Installation of DX Loadables</i> , p. 109.

Function Description

The NOL instruction is provided to facilitate the movement of the large amount of data between the NOL module and the controller register space. The NOL Module is mapped for 16 input registers (3X) and 16 output registers (4X). Of these registers, two input and two output registers are for handshaking between the NOL Module and the instruction. The remaining fourteen input and fourteen output registers are used to transport the data.

Representation: NOL - Network Option Module for Lonworks

Symbol



Parameter Description

Parameters	State RAM Reference	Data Type	Meaning
Top input	0x, 1x	None	ON = Enables the NOL function
Middle input	0x, 1x	None	ON = Initialize: causes the instruction to re-sync with the module
function # (top node)	4x	INT, UINT, WORD	Function number selects the function of the NOL block Function 0 transfers data to/from the module. Any other function number yields an error.
register block (middle node)	4x	INT, UINT, WORD	Register block (first of 16 contiguous registers)
count (bottom node)		INT, UINT	Total number of registers required by the instruction
Top output	0x	None	ON = instruction enabled and no error
Middle output	0x	None	New data Set for one sweep when the entire data block from the module has been written to the register area.
Bottom output	0x	None	ON = Error

Detailed Description

Register Block (Middle Node)

This block provides the registers for configuration and status information, the registers for the health status bits and the registers for the actual data of the Standard Network Variable Types (SNVTs).

Register Block

	Register	Content
Configuration and Status information	Displayed and first implied	I/O Map input base (3x)
	Second and third implied	I/O Map output base (4x)
	Fourth implied	Enable health bits
	Fifth implied	Number of input registers
	Sixth implied	Number of output registers
	Seventh implied	Number of discrete input registers
	Eighth implied	Number of discrete output registers
	Ninth implied	Config checksum (CRC)
	10th implied	NOL version
	11th implied	Module firmware version
	12th implied	NOL DX version
	13th implied	Module DX version
		14th to 15th implied
SNVTs Health Bit Status (if enabled in DX-Zoom screen)	16th to 31st implied	Health bits of each programmable network variable
SNVTs Actual Data	Enable Health Bit = NO: from 16th implied up	Data is stored in 4 groups: <ul style="list-style-type: none"> ● Discrete inputs ● Register inputs ● Discrete outputs ● Register outputs These groups of data are set up consecutively and start on word boundaries.
	Enable Health Bit = YES: from 32nd implied up	

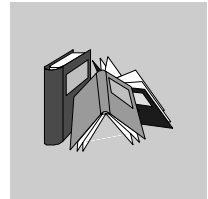
The first 16 registers with configuration and status information can be programmed and monitored via the NOL DX Zoom screen. For setting up the link to the NOL module the only parameters that need to be entered are the beginning 3x and 4x registers used when I/O mapping the NOL module.

Further information you will find in the documentation **Network Option Module for LonWorks**.

**Count
(Bottom Node)**

Defines the total number of registers required by the function block. This value must be set to a value equal to or greater than the number of data registers required to transfer and store the network data being used by the NOL module. If the count value is not large enough for the required data, the error output will be set.

Glossary



A

- active window** The window, which is currently selected. Only one window can be active at any one given time. When a window is active, the heading changes color, in order to distinguish it from other windows. Unselected windows are inactive.
- Actual parameter** Currently connected Input/Output parameters.
- Addresses** (Direct) addresses are memory areas on the PLC. These are found in the State RAM and can be assigned input/output modules.
The display/input of direct addresses is possible in the following formats:
- Standard format (400001)
 - Separator format (4:00001)
 - Compact format (4:1)
 - IEC format (QW1)
- ANL_IN** ANL_IN stands for the data type "Analog Input" and is used for processing analog values. The 3x-References of the configured analog input module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.
- ANL_OUT** ANL_OUT stands for the data type "Analog Output" and is used for processing analog values. The 4x-References of the configured analog output module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.
- ANY** In the existing version "ANY" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD and therefore derived data types.

ANY_BIT	In the existing version, "ANY_BIT" covers the data types BOOL, BYTE and WORD.
ANY_ELEM	In the existing version "ANY_ELEM" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD.
ANY_INT	In the existing version, "ANY_INT" covers the data types DINT, INT, UDINT and UINT.
ANY_NUM	In the existing version, "ANY_NUM" covers the data types DINT, INT, REAL, UDINT and UINT.
ANY_REAL	In the existing version "ANY_REAL" covers the data type REAL.
Application window	The window, which contains the working area, the menu bar and the tool bar for the application. The name of the application appears in the heading. An application window can contain several document windows. In Concept the application window corresponds to a Project.
Argument	Synonymous with Actual parameters.
ASCII mode	American Standard Code for Information Interchange. The ASCII mode is used for communication with various host devices. ASCII works with 7 data bits.
Atrium	The PC based controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module occupies a motherboard (requires SA85 driver) with two slots for PC104 daughter boards. From this, a PC104 daughter board is used as a CPU and the others for INTERBUS control.

B

Back up data file (Concept EFB)	The back up file is a copy of the last Source files. The name of this back up file is "backup??.c" (it is accepted that there are no more than 100 copies of the source files. The first back up file is called "backup00.c". If changes have been made on the Definition file, which do not create any changes to the interface in the EFB, there is no need to create a back up file by editing the source files (Objects → Source). If a back up file can be assigned, the name of the source file can be given.
Base 16 literals	Base 16 literals function as the input of whole number values in the hexadecimal system. The base must be denoted by the prefix 16#. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
16#F_F or 16#FF (decimal 255)
16#E_0 or 16#E0 (decimal 224)

Base 8 literal Base 8 literals function as the input of whole number values in the octal system. The base must be denoted by the prefix 8. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
8#3_1111 or 8#377 (decimal 255)
8#34_1111 or 8#340 (decimal 224)

Basis 2 literals Base 2 literals function as the input of whole number values in the dual system. The base must be denoted by the prefix 2. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
2#1111_1111 or 2#11111111 (decimal 255)
2#1110_1111 or 2#11100000 (decimal 224)

Binary connections Connections between outputs and inputs of FFBs of data type BOOL.

Bit sequence A data element, which is made up from one or more bits.

BOOL BOOL stands for the data type "Boolean". The length of the data elements is 1 bit (in the memory contained in 1 byte). The range of values for variables of this type is 0 (FALSE) and 1 (TRUE).

Bridge A bridge serves to connect networks. It enables communication between nodes on the two networks. Each network has its own token rotation sequence – the token is not deployed via bridges.

BYTE BYTE stands for the data type "Bit sequence 8". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 8 bit. A numerical range of values cannot be assigned to this data type.

C

Cache The cache is a temporary memory for cut or copied objects. These objects can be inserted into sections. The old content in the cache is overwritten for each new Cut or Copy.

Call up	The operation, by which the execution of an operation is initiated.
Coil	A coil is a LD element, which transfers (without alteration) the status of the horizontal link on the left side to the horizontal link on the right side. In this way, the status is saved in the associated Variable/ direct address.
Compact format (4:1)	The first figure (the Reference) is separated from the following address with a colon (:), where the leading zero are not entered in the address.
Connection	A check or flow of data connection between graphic objects (e.g. steps in the SFC editor, Function blocks in the FBD editor) within a section, is graphically shown as a line.
Constants	Constants are Unlocated variables, which are assigned a value that cannot be altered from the program logic (write protected).
Contact	A contact is a LD element, which transfers a horizontal connection status onto the right side. This status is from the Boolean AND- operation of the horizontal connection status on the left side with the status of the associated Variables/direct Address. A contact does not alter the value of the associated variables/direct address.

D

Data transfer settings	Settings, which determine how information from the programming device is transferred to the PLC.
Data types	<p>The overview shows the hierarchy of data types, as they are used with inputs and outputs of Functions and Function blocks. Generic data types are denoted by the prefix "ANY".</p> <ul style="list-style-type: none">● ANY_ELEM<ul style="list-style-type: none">● ANY_NUM<ul style="list-style-type: none">● ANY_REAL (REAL)● ANY_INT (DINT, INT, UDINT, UINT)● ANY_BIT (BOOL, BYTE, WORD)● TIME● System data types (IEC extensions)● Derived (from "ANY" data types)

DCP I/O station	With a Distributed Control Processor (D908) a remote network can be set up with a parent PLC. When using a D908 with remote PLC, the parent PLC views the remote PLC as a remote I/O station. The D908 and the remote PLC communicate via the system bus, which results in high performance, with minimum effect on the cycle time. The data exchange between the D908 and the parent PLC takes place at 1.5 Megabits per second via the remote I/O bus. A parent PLC can support up to 31 (Address 2-32) D908 processors.
DDE (Dynamic Data Exchange)	The DDE interface enables a dynamic data exchange between two programs under Windows. The DDE interface can be used in the extended monitor to call up its own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data onto the PLC via the server. Data can therefore be altered directly in the PLC, while it monitors and analyzes the results. When using this interface, the user is able to make their own "Graphic-Tool", "Face Plate" or "Tuning Tool", and integrate this into the system. The tools can be written in any DDE supporting language, e.g. Visual Basic and Visual-C++. The tools are called up, when the one of the buttons in the dialog box extended monitor uses Concept Graphic Tool: Signals of a projection can be displayed as timing diagrams via the DDE connection between Concept and Concept Graphic Tool.
Decentral Network (DIO)	A remote programming in Modbus Plus network enables maximum data transfer performance and no specific requests on the links. The programming of a remote net is easy. To set up the net, no additional ladder diagram logic is needed. Via corresponding entries into the Peer Cop processor all data transfer requests are met.
Declaration	Mechanism for determining the definition of a Language element. A declaration normally covers the connection of an Identifier with a language element and the assignment of attributes such as Data types and algorithms.
Definition data file (Concept EFB)	The definition file contains general descriptive information about the selected FFB and its formal parameters.
Derived data type	Derived data types are types of data, which are derived from the Elementary data types and/or other derived data types. The definition of the derived data types appears in the data type editor in Concept. Distinctions are made between global data types and local data types.
Derived Function Block (DFB)	A derived function block represents the Call up of a derived function block type. Details of the graphic form of call up can be found in the definition " Function block (Item)". Contrary to calling up EFB types, calling up DFB types is denoted by double vertical lines on the left and right side of the rectangular block symbol.

The body of a derived function block type is designed using FBD language, but only in the current version of the programming system. Other IEC languages cannot yet be used for defining DFB types, nor can derived functions be defined in the current version.

Distinctions are made between local and global DFBs.

- DINT** DINT stands for the data type "double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The range of values for variables of this data type is from $-2 \exp(31)$ to $2 \exp(31) - 1$.
- Direct display** A method of displaying variables in the PLC program, from which the assignment of configured memory can be directly and indirectly derived from the physical memory.
- Document window** A window within an Application window. Several document windows can be opened at the same time in an application window. However, only one document window can be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.
- Dummy** An empty data file, which consists of a text header with general file information, i.e. author, date of creation, EFB identifier etc. The user must complete this dummy file with additional entries.
- DX Zoom** This property enables connection to a programming object to observe and, if necessary, change its data value.
-

E

- Elementary functions/
function blocks
(EFB)** Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose bodies, for example, cannot be modified with the DFB Editor (Concept-DFB). EFB types are programmed in "C" and mounted via Libraries in precompiled form.

EN / ENO (Enable / Error display)	If the value of EN is "0" when the FFB is called up, the algorithms defined by the FFB are not executed and all outputs contain the previous value. The value of ENO is automatically set to "0" in this case. If the value of EN is "1" when the FFB is called up, the algorithms defined by the FFB are executed. After the error free execution of the algorithms, the ENO value is automatically set to "1". If an error occurs during the execution of the algorithm, ENO is automatically set to "0". The output behavior of the FFB depends whether the FFBs are called up without EN/ENO or with EN=1. If the EN/ENO display is enabled, the EN input must be active. Otherwise, the FFB is not executed. The projection of EN and ENO is enabled/disabled in the block properties dialog box. The dialog box is called up via the menu commands Objects → Properties... or via a double click on the FFB.
Error	When processing a FFB or a Step an error is detected (e.g. unauthorized input value or a time error), an error message appears, which can be viewed with the menu command Online → Event display... . With FFBs the ENO output is set to "0".
Evaluation	The process, by which a value for a Function or for the outputs of a Function block during the Program execution is transmitted.
Expression	Expressions consist of operators and operands.

F

FFB (functions/ function blocks)	Collective term for EFB (elementary functions/function blocks) and DFB (derived function blocks)
Field variables	Variables, one of which is assigned, with the assistance of the key word ARRAY (field), a defined Derived data type. A field is a collection of data elements of the same Data type.
FIR filter	Finite Impulse Response Filter
Formal parameters	Input/Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.

- Function (FUNC)** A Program organization unit, which exactly supplies a data element when executing. A function has no internal status information. Multiple call ups of the same function with the same input parameter values always supply the same output values. Details of the graphic form of function call up can be found in the definition " Function block (Item)". In contrast to the call up of function blocks, the function call ups only have one unnamed output, whose name is the name of the function itself. In FBD each call up is denoted by a unique number over the graphic block; this number is automatically generated and cannot be altered.
- Function block (item) (FB)** A function block is a Program organization unit, which correspondingly calculates the functionality values, defined in the function block type description, for the output and internal variables, when it is called up as a certain item. All output values and internal variables of a certain function block item remain as a call up of the function block until the next. Multiple call up of the same function block item with the same arguments (Input parameter values) supply generally supply the same output value(s). Each function block item is displayed graphically by a rectangular block symbol. The name of the function block type is located on the top center within the rectangle. The name of the function block item is located also at the top, but on the outside of the rectangle. An instance is automatically generated when creating, which can however be altered manually, if required. Inputs are displayed on the left side and outputs on the right of the block. The names of the formal input/output parameters are displayed within the rectangle in the corresponding places. The above description of the graphic presentation is principally applicable to Function call ups and to DFB call ups. Differences are described in the corresponding definitions.
- Function block dialog (FBD)** One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.
- Function block type** A language element, consisting of: 1. the definition of a data structure, subdivided into input, output and internal variables, 2. A set of operations, which is used with the elements of the data structure, when a function block type instance is called up. This set of operations can be formulated either in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (called up) several times.
- Function counter** The function counter serves as a unique identifier for the function in a Program or DFB. The function counter cannot be edited and is automatically assigned. The function counter always has the structure: .n.m
- n = Section number (number running)
m = Number of the FFB object in the section (number running)
-

G

Generic data type	A Data type, which stands in for several other data types.
Generic literal	If the Data type of a literal is not relevant, simply enter the value for the literal. In this case Concept automatically assigns the literal to a suitable data type.
Global derived data types	Global Derived data types are available in every Concept project and are contained in the DFB directory directly under the Concept directory.
Global DFBs	Global DFBs are available in every Concept project and are contained in the DFB directory directly under the Concept directory.
Global macros	Global Macros are available in every Concept project and are contained in the DFB directory directly under the Concept directory.
Groups (EFBs)	Some EFB libraries (e.g. the IEC library) are subdivided into groups. This facilitates the search for FFBS, especially in extensive libraries.

I

I/O component list	The I/O and expert assemblies of the various CPUs are configured in the I/O component list.
IEC 61131-3	International norm: Programmable controllers – part 3: Programming languages.
IEC format (QW1)	In the place of the address stands an IEC identifier, followed by a five figure address: <ul style="list-style-type: none">● %0x12345 = %Q12345● %1x12345 = %I12345● %3x12345 = %IW12345● %4x12345 = %QW12345

IEC name conventions (identifier)

An identifier is a sequence of letters, figures, and underscores, which must start with a letter or underscores (e.g. name of a function block type, of an item or section). Letters from national sets of characters (e.g. ö, ü, é, ò) can be used, taken from project and DFB names.

Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as different identifiers. Several leading and multiple underscores are not authorized consecutively.

Identifiers are not permitted to contain space characters. Upper and/or lower case is not significant; e.g. "ABCD" and "abcd" are interpreted as the same identifier.

Identifiers are not permitted to be Key words.

IIR filter

Infinite Impulse Response Filter

Initial step (starting step)

The first step in a chain. In each chain, an initial step must be defined. The chain is started with the initial step when first called up.

Initial value

The allocated value of one of the variables when starting the program. The value assignment appears in the form of a Literal.

Input bits (1x references)

The 1/0 status of input bits is controlled via the process data, which reaches the CPU from an entry device.

Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 100201 signifies an input bit in the address 201 of the State RAM.

Input parameters (Input)

When calling up a FFB the associated Argument is transferred.

Input words (3x references)

An input word contains information, which come from an external source and are represented by a 16 bit figure. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format.

Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the user data store, i.e. if the reference 300201 signifies a 16 bit input word in the address 201 of the State RAM.

Instantiation

The generation of an Item.

Instruction (IL)

Instructions are "commands" of the IL programming language. Each operation begins on a new line and is succeeded by an operator (with modifier if needed) and, if necessary for each relevant operation, by one or more operands. If several operands are used, they are separated by commas. A tag can stand before the instruction, which is followed by a colon. The commentary must, if available, be the last element in the line.

Instruction (LL984)	When programming electric controllers, the task of implementing operational coded instructions in the form of picture objects, which are divided into recognizable contact forms, must be executed. The designed program objects are, on the user level, converted to computer useable OP codes during the loading process. The OP codes are deciphered in the CPU and processed by the controller's firmware functions so that the desired controller is implemented.
Instruction list (IL)	IL is a text language according to IEC 1131, in which operations, e.g. conditional/unconditional call up of Function blocks and Functions, conditional/unconditional jumps etc. are displayed through instructions.
INT	INT stands for the data type "whole number". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The range of values for variables of this data type is from $-2 \exp(15)$ to $2 \exp(15) - 1$.
Integer literals	Integer literals function as the input of whole number values in the decimal system. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant. Example -12, 0, 123_456, +986
INTERBUS (PCP)	To use the INTERBUS PCP channel and the INTERBUS process data preprocessing (PDP), the new I/O station type INTERBUS (PCP) is led into the Concept configurator. This I/O station type is assigned fixed to the INTERBUS connection module 180-CRP-660-01. The 180-CRP-660-01 differs from the 180-CRP-660-00 only by a clearly larger I/O area in the state RAM of the controller.
Item name	An Identifier, which belongs to a certain Function block item. The item name serves as a unique identifier for the function block in a program organization unit. The item name is automatically generated, but can be edited. The item name must be unique throughout the Program organization unit, and no distinction is made between upper/lower case. If the given name already exists, a warning is given and another name must be selected. The item name must conform to the IEC name conventions, otherwise an error message appears. The automatically generated instance name always has the structure: FBI_n_m FBI = Function block item n = Section number (number running) m = Number of the FFB object in the section (number running)

J

Jump Element of the SFC language. Jumps are used to jump over areas of the chain.

K

Key words Key words are unique combinations of figures, which are used as special syntactic elements, as is defined in appendix B of the IEC 1131-3. All key words, which are used in the IEC 1131-3 and in Concept, are listed in appendix C of the IEC 1131-3. These listed keywords cannot be used for any other purpose, i.e. not as variable names, section names, item names etc.

L

Ladder Diagram (LD) Ladder Diagram is a graphic programming language according to IEC1131, which optically orientates itself to the "rung" of a relay ladder diagram.

Ladder Logic 984 (LL) In the terms Ladder Logic and Ladder Diagram, the word Ladder refers to execution. In contrast to a diagram, a ladder logic is used by engineers to draw up a circuit (with assistance from electrical symbols), which should chart the cycle of events and not the existing wires, which connect the parts together. A usual user interface for controlling the action by automated devices permits ladder logic interfaces, so that when implementing a control system, engineers do not have to learn any new programming languages, with which they are not conversant. The structure of the actual ladder logic enables electrical elements to be linked in a way that generates a control output, which is dependant upon a configured flow of power through the electrical objects used, which displays the previously demanded condition of a physical electric appliance. In simple form, the user interface is one of the video displays used by the PLC programming application, which establishes a vertical and horizontal grid, in which the programming objects are arranged. The logic is powered from the left side of the grid, and by connecting activated objects the electricity flows from left to right.

Landscape format Landscape format means that the page is wider than it is long when looking at the printed text.

Language element	Each basic element in one of the IEC programming languages, e.g. a Step in SFC, a Function block item in FBD or the Start value of a variable.
Library	Collection of software objects, which are provided for reuse when programming new projects, or even when building new libraries. Examples are the Elementary function block types libraries. EFB libraries can be subdivided into Groups.
Literals	Literals serve to directly supply values to inputs of FFBS, transition conditions etc. These values cannot be overwritten by the program logic (write protected). In this way, generic and standardized literals are differentiated. Furthermore literals serve to assign a Constant a value or a Variable an Initial value. The input appears as Base 2 literal, Base 8 literal, Base 16 literal, Integer literal, Real literal or Real literal with exponent.
Local derived data types	Local derived data types are only available in a single Concept project and its local DFBs and are contained in the DFB directory under the project directory.
Local DFBs	Local DFBs are only available in a single Concept project and are contained in the DFB directory under the project directory.
Local link	The local network link is the network, which links the local nodes with other nodes either directly or via a bus amplifier.
Local macros	Local Macros are only available in a single Concept project and are contained in the DFB directory under the project directory.
Local network nodes	The local node is the one, which is projected evenly.
Located variable	Located variables are assigned a state RAM address (reference addresses 0x, 1x, 3x, 4x). The value of these variables is saved in the state RAM and can be altered online with the reference data editor. These variables can be addressed by symbolic names or the reference addresses. Collective PLC inputs and outputs are connected to the state RAM. The program access to the peripheral signals, which are connected to the PLC, appears only via located variables. PLC access from external sides via Modbus or Modbus plus interfaces, i.e. from visualizing systems, are likewise possible via located variables.

M

Macro

Macros are created with help from the software Concept DFB.

Macros function to duplicate frequently used sections and networks (including the logic, variables, and variable declaration).

Distinctions are made between local and global macros.

Macros have the following properties:

- Macros can only be created in the programming languages FBD and LD.
- Macros only contain one single section.
- Macros can contain any complex section.
- From a program technical point of view, there is no differentiation between an instanced macro, i.e. a macro inserted into a section, and a conventionally created macro.
- Calling up DFBs in a macro
- Variable declaration
- Use of macro-own data structures
- Automatic acceptance of the variables declared in the macro
- Initial value for variables
- Multiple instancing of a macro in the whole program with different variables
- The section name, the variable name and the data structure name can contain up to 10 different exchange markings (@0 to @9).

MMI

Man Machine Interface

Multi element variables

Variables, one of which is assigned a Derived data type defined with STRUCT or ARRAY.

Distinctions are made between Field variables and structured variables.

N

Network

A network is the connection of devices to a common data path, which communicate with each other via a common protocol.

Network node

A node is a device with an address (164) on the Modbus Plus network.

Node address The node address serves a unique identifier for the network in the routing path. The address is set directly on the node, e.g. with a rotary switch on the back of the module.

O

Operand An operand is a Literal, a Variable, a Function call up or an Expression.

Operator An operator is a symbol for an arithmetic or Boolean operation to be executed.

Output parameters (Output) A parameter, with which the result(s) of the Evaluation of a FFB are returned.

Output/discretes (0x references) An output/marker bit can be used to control real output data via an output unit of the control system, or to define one or more outputs in the state RAM. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 000201 signifies an output or marker bit in the address 201 of the State RAM.

Output/marker words (4x references) An output/marker word can be used to save numerical data (binary or decimal) in the State RAM, or also to send data from the CPU to an output unit in the control system. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

P

Peer processor The peer processor processes the token run and the flow of data between the Modbus Plus network and the PLC application logic.

PLC Programmable controller

Program The uppermost Program organization unit. A program is closed and loaded onto a single PLC.

Program cycle A program cycle consists of reading in the inputs, processing the program logic and the output of the outputs.

Program organization unit	A Function, a Function block, or a Program. This term can refer to either a Type or an Item.
Programming device	Hardware and software, which supports programming, configuring, testing, implementing and error searching in PLC applications as well as in remote system applications, to enable source documentation and archiving. The programming device could also be used for process visualization.
Programming redundancy system (Hot Standby)	A redundancy system consists of two identically configured PLC devices, which communicate with each other via redundancy processors. In the case of the primary PLC failing, the secondary PLC takes over the control checks. Under normal conditions the secondary PLC does not take over any controlling functions, but instead checks the status information, to detect mistakes.
Project	General identification of the uppermost level of a software tree structure, which specifies the parent project name of a PLC application. After specifying the project name, the system configuration and control program can be saved under this name. All data, which results during the creation of the configuration and the program, belongs to this parent project for this special automation. General identification for the complete set of programming and configuring information in the Project data bank, which displays the source code that describes the automation of a system.
Project data bank	The data bank in the Programming device, which contains the projection information for a Project.
Prototype data file (Concept EFB)	The prototype data file contains all prototypes of the assigned functions. Further, if available, a type definition of the internal status structure is given.

R

REAL REAL stands for the data type "real". The input appears as Real literal or as Real literal with exponent. The length of the data element is 32 bit. The value range for variables of this data type reaches from 8.43E-37 to 3.36E+38.

Note: Depending on the mathematic processor type of the CPU, various areas within this valid value range cannot be represented. This is valid for values nearing ZERO and for values nearing INFINITY. In these cases, a number value is not shown in animation, instead NAN (**N**ot **A** Number) oder INF (**I**N**F**inite).

Real literal Real literals function as the input of real values in the decimal system. Real literals are denoted by the input of the decimal point. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example

-12.0, 0.0, +0.456, 3.14159_26

Real literal with exponent Real literals with exponent function as the input of real values in the decimal system. Real literals with exponent are denoted by the input of the decimal point. The exponent sets the key potency, by which the preceding number is multiplied to get to the value to be displayed. The basis may be preceded by a negative sign (-). The exponent may be preceded by a positive or negative sign (+/-). Single underline signs (_) between figures are not significant. (Only between numbers, not before or after the decimal point and not before or after "E", "E+" or "E-")

Example

-1.34E-12 or -1.34e-12

1.0E+6 or 1.0e+6

1.234E6 or 1.234e6

Reference Each direct address is a reference, which starts with an ID, specifying whether it concerns an input or an output and whether it concerns a bit or a word. References, which start with the code 6, display the register in the extended memory of the state RAM.

0x area = Discrete outputs

1x area = Input bits

3x area = Input words

4x area = Output bits/Marker words

6x area = Register in the extended memory

Note: The x, which comes after the first figure of each reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

Register in the extended memory (6x reference) 6x references are marker words in the extended memory of the PLC. Only LL984 user programs and CPU 213 04 or CPU 424 02 can be used.

RIO (Remote I/O) Remote I/O provides a physical location of the I/O coordinate setting device in relation to the processor to be controlled. Remote inputs/outputs are connected to the consumer control via a wired communication cable.

RP (PROFIBUS) RP = Remote Peripheral

- RTU mode** Remote Terminal Unit
The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.
- Rum-time error** Error, which occurs during program processing on the PLC, with SFC objects (i.e. steps) or FFBs. These are, for example, over-runs of value ranges with figures, or time errors with steps.
-

S

- SA85 module** The SA85 module is a Modbus Plus adapter for an IBM-AT or compatible computer.
- Section** A section can be used, for example, to describe the functioning method of a technological unit, such as a motor.
A Program or DFB consist of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages can be used within a section.
Each section has its own Document window in Concept. For reasons of clarity, it is recommended to subdivide a very large section into several small ones. The scroll bar serves to assist scrolling in a section.
- Separator format (4:00001)** The first figure (the Reference) is separated from the ensuing five figure address by a colon (:).
- Sequence language (SFC)** The SFC Language elements enable the subdivision of a PLC program organizational unit in a number of Steps and Transitions, which are connected horizontally by aligned Connections. A number of actions belong to each step, and a transition condition is linked to a transition.
- Serial ports** With serial ports (COM) the information is transferred bit by bit.
- Source code data file (Concept EFB)** The source code data file is a usual C++ source file. After execution of the menu command **Library** → **Generate data files** this file contains an EFB code framework, in which a specific code must be entered for the selected EFB. To do this, click on the menu command **Objects** → **Source**.
- Standard format (400001)** The five figure address is located directly after the first figure (the reference).

Standardized literals	<p>If the data type for the literal is to be automatically determined, use the following construction: 'Data type name' #'Literal value'.</p> <p>Example</p> <p>INT#15 (Data type: Integer, value: 15), BYTE#00001111 (data type: Byte, value: 00001111) REAL#23.0 (Data type: Real, value: 23.0)</p> <p>For the assignment of REAL data types, there is also the possibility to enter the value in the following way: 23.0. Entering a comma will automatically assign the data type REAL.</p>
State RAM	<p>The state RAM is the storage for all sizes, which are addressed in the user program via References (Direct display). For example, input bits, discretets, input words, and discrete words are located in the state RAM.</p>
Statement (ST)	<p>Instructions are "commands" of the ST programming language. Instructions must be terminated with semicolons. Several instructions (separated by semi-colons) can occupy the same line.</p>
Status bits	<p>There is a status bit for every node with a global input or specific input/output of Peer Cop data. If a defined group of data was successfully transferred within the set time out, the corresponding status bit is set to 1. Alternatively, this bit is set to 0 and all data belonging to this group (of 0) is deleted.</p>
Step	<p>SFC Language element: Situations, in which the Program behavior follows in relation to the inputs and outputs of the same operations, which are defined by the associated actions of the step.</p>
Step name	<p>The step name functions as the unique flag of a step in a Program organization unit. The step name is automatically generated, but can be edited. The step name must be unique throughout the whole program organization unit, otherwise an Error message appears.</p> <p>The automatically generated step name always has the structure: S_n_m</p> <p>S = Step n = Section number (number running) m = Number of steps in the section (number running)</p>
Structured text (ST)	<p>ST is a text language according to IEC 1131, in which operations, e.g. call up of Function blocks and Functions, conditional execution of instructions, repetition of instructions etc. are displayed through instructions.</p>

Structured variables	Variables, one of which is assigned a Derived data type defined with STRUCT (structure). A structure is a collection of data elements with generally differing data types (Elementary data types and/or derived data types).
SY/MAX	In Quantum control devices, Concept closes the mounting on the I/O population SY/MAX I/O modules for RIO control via the Quantum PLC with on. The SY/MAX remote subrack has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O system. The SY/MAX I/O modules are performed when highlighting and including in the I/O population of the Concept configuration.
Symbol (Icon)	Graphic display of various objects in Windows, e.g. drives, user programs and Document windows.

T

Template data file (Concept EFB)	The template data file is an ASCII data file with a layout information for the Concept FBD editor, and the parameters for code generation.
TIME	TIME stands for the data type "Time span". The input appears as Time span literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to $2^{\text{exp}(32)-1}$. The unit for the data type TIME is 1 ms.
Time span literals	Permitted units for time spans (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or a combination thereof. The time span must be denoted by the prefix t#, T#, time# or TIME#. An "overrun" of the highest ranking unit is permitted, i.e. the input T#25H15M is permitted. Example t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS
Token	The network "Token" controls the temporary property of the transfer rights via a single node. The token runs through the node in a circulating (rising) address sequence. All nodes track the Token run through and can contain all possible data sent with it.
Traffic Cop	The Traffic Cop is a component list, which is compiled from the user component list. The Traffic Cop is managed in the PLC and in addition contains the user component list e.g. Status information of the I/O stations and modules.

Transition The condition with which the control of one or more Previous steps transfers to one or more ensuing steps along a directional Link.

U

UDEFB User defined elementary functions/function blocks
Functions or Function blocks, which were created in the programming language C, and are available in Concept Libraries.

UDINT UDINT stands for the data type "unsigned double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to $2^{\text{exp}(32)}-1$.

UINT UINT stands for the data type "unsigned integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The value range for variables of this type stretches from 0 to $(2^{\text{exp}16})-1$.

Unlocated variable Unlocated variables are not assigned any state RAM addresses. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the system and can be altered with the reference data editor. These variables are only addressed by symbolic names.

Signals requiring no peripheral access, e.g. intermediate results, system tags etc, should primarily be declared as unlocated variables.

V

Variables Variables function as a data exchange within sections between several sections and between the Program and the PLC.
Variables consist of at least a variable name and a Data type.
Should a variable be assigned a direct Address (Reference), it is referred to as a Located variable. Should a variable not be assigned a direct address, it is referred to as an unlocated variable. If the variable is assigned a Derived data type, it is referred to as a Multi-element variable.
Otherwise there are Constants and Literals.

Vertical format Vertical format means that the page is higher than it is wide when looking at the printed text.

W

Warning When processing a FFB or a Step a critical status is detected (e.g. critical input value or a time out), a warning appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output remains at "1".

WORD WORD stands for the data type "Bit sequence 16". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 16 bit. A numerical range of values cannot be assigned to this data type.

Index



Numerics

3x or 4x register
entering in equation network, 62

A

ABS, 69
AD16, 117
ADD, 121
Add 16 Bit, 117
Addition, 121
 AD16, 117
 ADD, 121
Advanced Calculations, 790
algebraic expression
 equation network, 58
algebraic notation
 equation network, 55
Analog Input, 797
Analog Output, 809
Analog Values, 77
AND, 125
ARCCOS, 69
ARCSIN, 69
ARCTAN, 69
argument
 equation network, 70
 limits, 71
arithmetic operator, 64
ASCII Functions
 READ, 945
 WRIT, 1097

assignment operator, 64
Average Weighted Inputs Calculate, 813

B

Base 10 Antilogarithm, 291
Base 10 Logarithm, 395
BCD, 131
benchmark performance
 equation network, 75
Binary to Binary Code, 131
Bit Control, 763
Bit pattern comparison
 CMPR, 179
Bit Rotate, 147
bitwise operator, 64
BLKM, 135
BLKT, 139
Block Move, 135
Block Move with Interrupts Disabled, 143
Block to Table, 139
BMDI, 143
boolean, 61
BROT, 147

C

Calculated preset formula, 819
Central Alarm Handler, 803
Changing the Sign of a Floating Point
Number, 313
Check Sum, 173

CHS, 165
CKSM, 173
Closed Loop Control, 77
CMPR, 179
coil
 equation network, 57
Coils, 99
Communications
 MSTR, 709
COMP, 191
Compare Register, 179
Complement a Matrix, 191
Comprehensive ISA Non Interacting
PID, 839
conditional expression
 equation network, 55, 66
conditional operator, 64
Configure Hot Standby, 165
constant
 equation network, 55
constant data
 entering in equation network, 63
 equation network, 62
 floating point, 62
 long (32-bit), 62
 LSB (least significant byte), 62
Contacts, 99
Conversion
 BCD to binary, 131
 binary to BCD, 131
COS, 69
COSD, 69
Counters / Timers
 T.01 Timer, 1057
 T0.1 Timer, 1061
 T1.0 Timer, 1065
 T1MS Timer, 1069
 UCTR, 1083
Counters/Timers
 DCTR, 215

D

data
 equation network, 61
 variable, 61
data conversions
 equation network, 72
Data Logging for PCMCIA Read/Write
Support, 235
data type
 boolean, 61
 equation network, 60
 floating point variable, 61
 signed 16-bit variable, 61
 signed long (32-bit) variable, 61
 suffix, 60
 unsigned 16-bit variable, 61
 unsigned long (32-bit) variable, 61
DCTR, 215
Derivative Rate Calculation over a Specified
Time, 891
DIOH, 219
discrete reference
 entering in equation network, 62
 equation network, 61
 variable data, 61
Distributed I/O Health, 219
DIV, 229
Divide, 229
Divide 16 Bit, 257
DLOG, 235
Double Precision Addition, 277
Double Precision Division, 359
Double Precision Multiplication, 407
Double Precision Subtraction, 453
Down Counter, 215
DRUM, 251
DRUM Sequencer, 251
DV16, 257

E

EMTH, 271

EMTH Subfunction

EMTH-ADDDP, 277

EMTH-ADDFP, 283, 287

EMTH-ANLOG, 291

EMTH-ARCOS, 297

EMTH-ARSIN, 303

EMTH-ARTAN, 307

EMTH-CHSIN, 313

EMTH-CMPFP, 319

EMTH-CMPIF, 325

EMTH-CNVDR, 331

EMTH-CNVFI, 337

EMTH-CNVIF, 343

EMTH-CNVRD, 349

EMTH-COS, 355

EMTH-DIVDP, 359

EMTH-DIVFI, 365

EMTH-DIVFP, 369

EMTH-DIVIF, 373

EMTH-ERLOG, 377

EMTH-EXP, 383

EMTH-LNFP, 389

EMTH-LOG, 395

EMTH-LOGFP, 401

EMTH-MULDP, 407

EMTH-MULFP, 413

EMTH-MULIF, 417

EMTH-PI, 423

EMTH-POW, 427

EMTH-SINE, 431

EMTH-SQRFP, 437

EMTH-SQRT, 441

EMTH-SQRTP, 447

EMTH-SUBDP, 453

EMTH-SUBFI, 459

EMTH-SUBFP, 463

EMTH-SUBIF, 467

EMTH-TAN, 471

EMTH-ADDDP, 277

EMTH-ADDFP, 283

EMTH-ADDIF, 287

EMTH-ANLOG, 291

EMTH-ARCOS, 297

EMTH-ARSIN, 303

EMTH-ARTAN, 307

EMTH-CHSIN, 313

EMTH-CMPFP, 319

EMTH-CMPIF, 325

EMTH-CNVDR, 331

EMTH-CNVFI, 337

EMTH-CNVIF, 343

EMTH-CNVRD, 349

EMTH-COS, 355

EMTH-DIVDP, 359

EMTH-DIVFI, 365

EMTH-DIVFP, 369

EMTH-DIVIF, 373

EMTH-ERLOG, 377

EMTH-EXP, 383

EMTH-LNFP, 389

EMTH-LOG, 395

EMTH-LOGFP, 401

EMTHMULDP, 407

EMTH-MULFP, 413

EMTH-MULIF, 417

EMTH-PI, 423

EMTH-POW, 427

EMTH-SINE, 431

EMTH-SQRFP, 437

EMTH-SQRT, 441

EMTH-SQRTP, 447

EMTH-SUBDP, 453

EMTH-SUBFI, 459

EMTH-SUBFP, 463

EMTH-SUBIF, 467

EMTH-TAN, 471

enable contact

equation network, 57

horizontal open, 57

horizontal short, 57

normally closed, 57

normally open, 57

Engineering Unit Conversion and Alarms, 495

equation

exponential notation, 63

- equation network
 - ABS, 69
 - algebraic expression, 58
 - algebraic notation, 55
 - ARCCOS, 69
 - ARCSIN, 69
 - ARCTAN, 69
 - argument, 70
 - argument limits, 71
 - arithmetic operator, 64
 - assignment operator, 64
 - benchmark performance, 75
 - bitwise operator, 64
 - conditional expression, 55, 66
 - conditional operator, 64
 - constant, 55
 - constant data, 62
 - content, 58
 - COS, 69
 - COSD, 69
 - create, 56
 - data conversions, 72
 - data type, 60
 - discrete reference, 61
 - enable contact, 57
 - entering 3x or 4x register, 62
 - entering constant data, 63
 - entering function, 70
 - entering parentheses, 68
 - entering variable data, 62
 - EXP, 69
 - exponentiation operator, 64
 - FIX, 69
 - FLOAT, 69
 - format, 59
 - group expressions in nested layers of
 - parentheses, 55
 - infix notation, 56
 - input offset, 56
 - input type, 56
 - LN, 69
 - LOG, 69
 - logic editor, 55
 - logical expression, 55
 - math operator, 55
 - mathematical function, 69
 - mathematical operation, 64
 - nested parentheses, 68
 - operator precedence, 67
 - output coil, 57
 - overview, 55, 56
 - parentheses, 64, 68
 - registers consumed, 61
 - relational operator, 64
 - result, 58
 - roundoff differences, 74
 - SIN, 69
 - SIND, 69
 - single expression, 66
 - size, 58
 - SQRT, 69
 - suffix, 60
 - TAN, 69
 - TAND, 69
 - unary operator, 64
 - use, 56
 - value, 60
 - variable, 55
 - variable data, 61
 - words consumed, 58, 61, 62
- ESI, 475
- EUCA, 495
- Exclusive OR, 1151
- EXP, 69
- exponentiation operator, 64
- Extended Math, 271
- Extended Memory Read, 1139
- Extended Memory Write, 1145

F

Fast I/O Instructions
 BMDI, 143
 ID, 623
 IE, 627
 IMIO, 631
 IMOD, 637
 ITMR, 647
FIN, 509
First In, 509
First Out, 513
First-order Lead/Lag Filter, 859
FIX, 69
FLOAT, 69
Floating Point - Integer Subtraction, 459
Floating Point Addition, 283
Floating Point Arc Cosine of an Angle
(in Radians), 297
Floating Point Arc Tangent of an Angle
(in Radians), 307
Floating Point Arcsine of an Angle
(in Radians), 303
Floating Point Common Logarithm, 401
Floating Point Comparison, 319
Floating Point Conversion of Degrees to
Radians, 331
Floating Point Conversion of Radians to
Degrees, 349
Floating Point Cosine of an Angle
(in Radians), 355
Floating Point Divided by Integer, 365
Floating Point Division, 369
Floating Point Error Report Log, 377
Floating Point Exponential Function, 383
Floating Point Multiplication, 413
Floating Point Natural Logarithm, 389
Floating Point Sine of an Angle
(in Radians), 431
Floating Point Square Root, 437, 441
Floating Point Subtraction, 463
Floating Point Tangent of an Angle
(in Radians), 471
Floating Point to Integer, 519
Floating Point to Integer Conversion, 337
floating point variable, 61

Formatted Equation Calculator, 829
Formatting Messages, 91
Four Station Ratio Controller, 895
FOUT, 513
FTOI, 519
function
 ABS, 69
 ARCCOS, 69
 ARCSIN, 69
 ARCTAN, 69
 argument, 70
 argument limits, 71
 COS, 69
 COSD, 69
 entering in equation network, 70
 EXP, 69
 FIX, 69
 FLOAT, 69
 LN, 69
 LOG, 69
 SIN, 69
 SIND, 69
 SQRT, 69
 TAN, 69
 TAND, 69

G

group expressions in nested layers of
parentheses
 equation network, 55

H

History and Status Matrices, 585
HLTH, 585
horizontal open
 equation network, 57
horizontal short
 equation network, 57
Hot standby
 CHS, 165

I

IBKR, 607
IBKW, 611
ICMP, 615
ID, 623
IE, 627
IMIO, 631
Immediate I/O, 631
IMOD, 637
Indirect Block Read, 607
Indirect Block Write, 611
infix notation
 equation network, 56
Input Compare, 615
input offset
 equation network, 56
Input Selection, 905
input type
 equation network, 56
Installation of DX Loadables, 109
Instruction
 Coils, Contacts and Interconnects, 99
Instruction Groups, 41
 ASCII Communication Instructions, 43
 Coils, Contacts and Interconnects, 54
 Counters and Timers Instructions, 44
 Fast I/O Instructions, 45
 Loadable DX, 46
 Math Instructions, 47
 Matrix Instructions, 49
 Miscellaneous, 50
 Move Instructions, 51
 Overview, 42
 Skips/Specials, 52
 Special Instructions, 53
Integer - Floating Point Subtraction, 467
Integer + Floating Point Addition, 287
Integer Divided by Floating Point, 373
Integer to Floating Point, 653
Integer x Floating Point Multiplication, 417
Integer-Floating Point Comparison, 325
Integer-to-Floating Point Conversion, 343
Integrate Input at Specified Interval, 835
Interconnects, 99
Interrupt Disable, 623

Interrupt Enable, 627
Interrupt Handling, 105
Interrupt Module Instruction, 637
Interrupt Timer, 647
ISA Non Interacting PI, 873
ITMR, 647
ITOF, 653

J

JSR, 657
Jump to Subroutine, 657

L

LAB, 661
Label for a Subroutine, 661
Limiter for the Pv, 845

LL984

AD16, 117
ADD, 121
AND, 125
BCD, 131
BLKM, 135
BLKT, 139
BMDI, 143
BROT, 147
CHS, 165
CKSM, 173
Closed Loop Control / Analog Values, 77
CMPR, 179
Coils, Contacts and Interconnects, 99
COMP, 191
DCTR, 215
DIOH, 219
DIV, 229
DLOG, 235
DRUM, 251
DV16, 257
EMTH, 271
EMTH-ADDDP, 277
EMTH-ADDFP, 283
EMTH-ADDIF, 287
EMTH-ANLOG, 291
EMTH-ARCOS, 297
EMTH-ARSIN, 303
EMTH-ARTAN, 307
EMTH-CHSIN, 313
EMTH-CMPFP, 319
EMTH-CMPIF, 325
EMTH-CNVDR, 331
EMTH-CNVFI, 337
EMTH-CNVIF, 343
EMTH-CNVRD, 349
EMTH-COS, 355
EMTH-DIVDP, 359
EMTH-DIVFI, 365
EMTH-DIVFP, 369
EMTH-DIVIF, 373
EMTH-ERLOG, 377
EMTH-EXP, 383
EMTH-LNFP, 389
EMTH-LOG, 395
EMTH-LOGFP, 401

EMTH-MULDP, 407
EMTH-MULFP, 413
EMTH-MULIF, 417
EMTH-PI, 423
EMTH-POW, 427
EMTH-SINE, 431
EMTH-SQRFP, 437
EMTH-SQRT, 441
EMTH-SQRTP, 447
EMTH-SUBDP, 453
EMTH-SUBFI, 459
EMTH-SUBFP, 463
EMTH-SUBIF, 467
EMTH-TAN, 471
ESI, 475
EUCA, 495
FIN, 509
Formatting Messages for ASCII

READ/WRIT Operations, 91
FOUT, 513
FTOI, 519
HLTH, 585
IBKR, 607
IBKW, 611
ICMP, 615
ID, 623
IE, 627
IMIO, 631
IMOD, 637
Interrupt Handling, 105
ITMR, 647
ITOF, 653
JSR, 657
LAB, 661
LOAD, 665
MAP 3, 669
MBIT, 685
MBUS, 689
MRTM, 699
MSTR, 709
MU16, 755
MUL, 759
NBIT, 763
NCBT, 767
NOBT, 771
NOL, 775
OR, 783
PCFL, 789
PCFL-AIN, 797
PCFL-ALARM, 803
PCFL-AOUT, 809
PCFL-AVER, 813
PCFL-CALC, 819
PCFL-DELAY, 825
PCFL-EQN, 829
PCFL-INTEG, 835
PCFL-KPID, 839
PCFL-LIMIT, 845
PCFL-LIMV, 849
PCFL-LKUP, 853
PCFL-LLAG, 859
PCFL-MODE, 863
PCFL-ONOFF, 867
PCFL-PI, 873
PCFL-PID, 879
PCFL-RAMP, 885
PCFL-RATE, 891
PCFL-RATIO, 895
PCFL-RMPLN, 901
PCFL-SEL, 905
PCFL-TOTAL, 911
PEER, 917
PID2, 921
R --> T, 937
RBIT, 941
READ, 945
RET, 951
SAVE, 969
SBIT, 973
SCIF, 977
SENS, 983
SRCH, 995
STAT, 1001
SU16, 1029
SUB, 1033
Subroutine Handling, 107
T.01 Timer, 1057
T-->R, 1045
T-->T, 1051
T0.1 Timer, 1061
T1.0 Timer, 1065
T1MS Timer, 1069
TBLK, 1073
TEST, 1079
UCTR, 1083
WRIT, 1097
XMRD, 1139
XMWT, 1145
XOR, 1151
LN, 69
LOAD, 665
Load Flash, 665
Load the Floating Point Value of "Pi", 423

- Loadable DX
 - CHS, 165
 - DRUM, 251
 - ESI, 475
 - EUCA, 495
 - HLTH, 585
 - ICMP, 615
 - Installation, 109
 - MAP 3, 669
 - MBUS, 689
 - MRTM, 699
 - NOL, 775
 - PEER, 917
 - LOG, 69
 - Logarithmic Ramp to Set Point, 901
 - logic editor
 - equation network, 55, 56
 - Logical And, 125
 - logical expression
 - equation network, 55
 - Logical OR, 783
 - Look-up Table, 853
 - LSB (least significant byte)
 - constant data, 62
- M**
- MAP 3, 669
 - MAP Transaction, 669
 - Master, 709
 - Math
 - AD16, 117
 - ADD, 121
 - BCD, 131
 - DIV, 229
 - DV16, 257
 - FTOI, 519
 - ITOF, 653
 - MU16, 755
 - MUL, 759
 - SU16, 1029
 - SUB, 1033
 - TEST, 1079
 - math coprocessor
 - roundoff differences, 74
 - math operator
 - equation network, 55
 - mathematical function
 - ABS, 69
 - ARCCOS, 69
 - ARCSIN, 69
 - ARCTAN, 69
 - argument, 70
 - argument limits, 71
 - COS, 69
 - COSD, 69
 - entering in equation network, 70
 - equation network, 69
 - EXP, 69
 - FIX, 69
 - FLOAT, 69
 - LN, 69
 - LOG, 69
 - SIN, 69
 - SIND, 69
 - SQRT, 69
 - TAN, 69
 - TAND, 69
 - mathematical operation
 - arithmetic operator, 64
 - assignment operator, 64
 - bitwise operator, 64
 - conditional operator, 64
 - equation network, 64
 - exponentiation operator, 64
 - parentheses, 64
 - relational operator, 64
 - unary operator, 64
 - Matrix
 - AND, 125
 - BROT, 147
 - CMPR, 179
 - COMP, 191
 - MBIT, 685
 - NBIT, 763
 - NCBT, 767, 771
 - OR, 783
 - RBIT, 941
 - SBIT, 973
 - SENS, 983
 - XOR, 1151

MBIT, 685
MBUS, 689
MBUS Transaction, 689

Miscellaneous

CKSM, 173
DLOG, 235
EMTH, 271
EMTH-ADDDP, 277
EMTH-ADDFP, 283
EMTH-ADDIF, 287
EMTH-ANLOG, 291
EMTH-ARCOS, 297, 355
EMTH-ARSIN, 303
EMTH-ARTAN, 307
EMTH-CHSIN, 313
EMTH-CMPFP, 319
EMTH-CMPIF, 325
EMTH-CNVDR, 331
EMTH-CNVFI, 337
EMTH-CNVIF, 343
EMTH-CNVRD, 349
EMTH-DIVDP, 359
EMTH-DIVFI, 365
EMTH-DIVFP, 369
EMTH-DIVIF, 373
EMTH-ERLOG, 377
EMTH-EXP, 383
EMTH-LNFP, 389
EMTH-LOG, 395
EMTH-LOGFP, 401
EMTH-MULDP, 407
EMTH-MULFP, 413
EMTH-MULIF, 417
EMTH-PI, 423
EMTH-POW, 427
EMTH-SINE, 431
EMTH-SQRFP, 437
EMTH-SQRT, 441
EMTH-SQRTP, 447
EMTH-SUBDP, 453
EMTH-SUBFI, 459
EMTH-SUBFP, 463
EMTH-SUBIF, 467
EMTH-TAN, 471
LOAD, 665
MSTR, 709
SAVE, 969
SCIF, 977
XMRD, 1139

- XMWT, 1145
- mixed data types
 - equation network, 72
- Modbus Functions, 1105
- Modbus Plus
 - MSTR, 709
- Modbus Plus Network Statistics
 - MSTR, 740
- Modify Bit, 685
- Move
 - BLKM, 135
 - BLKT, 139
 - FIN, 509
 - FOUT, 513
 - IBKR, 607
 - IBKW, 611
 - R --> T, 937
 - SRCH, 995
 - T-->R, 1045
 - T-->T, 1051
 - TBLK, 1073
- MRTM, 699
- MSTR, 709
 - Clear Local Statistics, 723
 - Clear Remote Statistics, 729
 - CTE Error Codes for SY/MAX and TCP/IP Ethernet, 754
 - Get Local Statistics, 721
 - Get Remote Statistics, 727
 - Modbus Plus and SY/MAX Ethernet Error Codes, 747
 - Modbus Plus Network Statistics, 740
 - Peer Cop Health, 731
 - Read CTE (Config Extension Table), 736
 - Read Global Data, 726
 - Reset Option Module, 734
 - SY/MAX-specific Error Codes, 749
 - TCP/IP Ethernet Error Codes, 751
 - TCP/IP Ethernet Statistics, 745
 - Write CTE (Config Extension Table), 738
 - Write Global Data, 725
- MU16, 755
- MUL, 759
- Multiply, 759
- Multiply 16 Bit, 755
- Multi-Register Transfer Module, 699

N

- NBIT, 763
- NCBT, 767
- nested layer
 - parentheses, 55
- nested parentheses
 - equation network, 68
- Network Option Module for Lonworks, 775
- NOBT, 771
- NOL, 775
- Normally Closed Bit, 767
- normally closed contact
 - equation network, 57
- Normally Open Bit, 771
- normally open contact
 - equation network, 57

O

- ON/OFF Values for Deadband, 867
- One Hundredth Second Timer, 1057
- One Millisecond Timer, 1069
- One Second Timer, 1065
- One Tenth Second Timer, 1061
- operator combinations
 - equation network, 72
- operator precedence
 - equation network, 67
- OR, 783
- output coil
 - equation network, 57

P

- parentheses
 - entering in equation network, 68
 - equation network, 55
 - nested, 68
 - nested layer, 55
 - using in equation network, 68
- PCFL, 789
- PCFL Subfunctions
 - General, 79
- PCFL-AIN, 797
- PCFL-ALARM, 803

- PCFL-AOUT, 809
 - PCFL-AVER, 813
 - PCFL-CALC, 819
 - PCFL-DELAY, 825
 - PCFL-EQN, 829
 - PCFL-INTEG, 835
 - PCFL-KPID, 839
 - PCFL-LIMIT, 845
 - PCFL-LIMV, 849
 - PCFL-LKUP, 853
 - PCFL-LLAG, 859
 - PCFL-MODE, 863
 - PCFL-ONOFF, 867
 - PCFL-PI, 873
 - PCFL-PID, 879
 - PCFL-RAMP, 885
 - PCFL-RATE, 891
 - PCFL-RATIO, 895
 - PCFL-RMPLN, 901
 - PCFL-SEL, 905
 - PCFL-Subfunction
 - PCFL-AIN, 797
 - PCFL-ALARM, 803
 - PCFL-AOUT, 809
 - PCFL-AVER, 813
 - PCFL-CALC, 819
 - PCFL-DELAY, 825
 - PCFL-EQN, 829
 - PCFL-INTEG, 835
 - PCFL-KPID, 839
 - PCFL-LIMIT, 845
 - PCFL-LIMV, 849
 - PCFL-LKUP, 853
 - PCFL-LLAG, 859
 - PCFL-MODE, 863
 - PCFL-ONOFF, 867
 - PCFL-PI, 873
 - PCFL-PID, 879
 - PCFL-RAMP, 885
 - PCFL-RATE, 891
 - PCFL-RATIO, 895
 - PCFL-RMPLN, 901
 - PCFL-SEL, 905
 - PCFL-TOTAL, 911
 - PCFL-TOTAL, 911
 - PEER, 917
 - PEER Transaction, 917
 - PID Algorithms, 879
 - PID Example, 83
 - PID2, 921
 - PID2 Level Control Example, 87
 - PLCs
 - roundoff differences, 74
 - scan time, 75
 - precedence
 - equation network, 67
 - Process Control Function Library, 789
 - Process Square Root, 447
 - Process Variable, 78
 - Proportional Integral Derivative, 921
 - Put Input in Auto or Manual Mode, 863
- ## Q
- Quantum PLCs
 - roundoff differences, 74
- ## R
- R --> T, 937
 - Raising a Floating Point Number to an Integer Power, 427
 - Ramp to Set Point at a Constant Rate, 885
 - RBIT, 941
 - READ, 945
 - MSTR, 719
 - Read, 945
 - READ/WRITE Operations, 91
 - Register to Table, 937
 - registers consumed
 - equation network, 61
 - variable data, 61
 - Regulatory Control, 790
 - relational operator, 64
 - Reset Bit, 941
 - result
 - equation network, 58
 - RET, 951
 - Return from a Subroutine, 951
 - roundoff differences
 - equation network, 74

S

SAVE, 969
Save Flash, 969
SBIT, 973
SCIF, 977
Search, 995
SENS, 983
Sense, 983
Sequential Control Interfaces, 977
Set Bit, 973
Set Point Variable, 78
signed 16-bit variable, 61
signed long (32-bit) variable, 61
SIN, 69
SIND, 69
single expression
 equation network, 66
Skips / Specials
 RET, 951
Skips/Specials
 JSR, 657
 LAB, 661

Special

DIOH, 219
PCFL, 789
PCFL-, 809
PCFL-AIN, 797
PCFL-ALARM, 803
PCFL-AVER, 813
PCFL-CALC, 819
PCFL-DELAY, 825
PCFL-EQN, 829
PCFL-KPID, 839
PCFL-LIMIT, 845
PCFL-LIMV, 849
PCFL-LKUP, 853
PCFL-LLAG, 859
PCFL-MODE, 863
PCFL-ONOFF, 867
PCFL-PI, 873
PCFL-PID, 879
PCFL-RAMP, 885
PCFL-RATE, 891
PCFL-RATIO, 895
PCFL-RMPLN, 901
PCFL-SEL, 905
PCFL-TOTAL, 911
PCPCFL-INTEGFL, 835
PID2, 921
STAT, 1001
SQRT, 69
SRCH, 995
STAT, 1001
Status, 1001
SU16, 1029
SUB, 1033
Subroutine Handling, 107
Subtract 16 Bit, 1029
Subtraction, 1033
suffix
 data type, 60
 equation network, 60
Support of the ESI Module, 475

T

- T.01 Timer, 1057
- T-->R, 1045
- T-->T, 1051
- T0.1 Timer, 1061
- T1.0 Timer, 1065
- T1MS Timer, 1069
- Table to Block, 1073
- Table to Register, 1045
- Table to Table, 1051
- TAN, 69
- TAND, 69
- TBLK, 1073
- TCP/IP Ethernet Statistics
 - MSTR, 745
- TEST, 1079
- Test of 2 Values, 1079
- Time Delay Queue, 825
- Totalizer for Metering Flow, 911

U

- UCTR, 1083
- unary operator, 64
- unsigned 16-bit variable, 61
- unsigned long (32-bit) variable, 61
- Up Counter, 1083

V

- value
 - equation network, 60
- variable
 - equation network, 55

variable data

- boolean, 61
- discrete reference, 61
- entering in equation network, 62
- equation network, 61
- floating point variable, 61
- registers consumed, 61
- signed 16-bit variable, 61
- signed long (32-bit) variable, 61
- unsigned 16-bit variable, 61
- unsigned long (32-bit) variable, 61
- words consumed, 61

Velocity Limiter for Changes in the Pv, 849

W

word

- maximum in an equation network, 58

words consumed

- constant data, 62
- equation network, 61
- variable data, 61

WRIT, 1097

Write, 1097

- MSTR, 717

X

XMRD, 1139

XMWT, 1145

XOR, 1151