Host Command Reference

Q and SCL commands for servo and stepper drives

Includes RS-232, RS-485,
Ethernet UDP, Ethernet TCP/IP, EtherNet/IP,
Modbus RTU and Modbus TCP/IP

Applied Motion Products
A MOONS' COMPANY

APPLIED MOTION PRODUCTS

920-0002 Rev. W
7/6/2021

Contents

Getting Started 11
Servo Drives 11
StepSERVO 11
Stepper Drives 11
Commands 13
Buffered Commands 13
Stored Programs in Q Drives 13
Multi-tasking in Q Drives 13
Immediate Commands 13
Using Commands 13
Commands in Q drives 14
SCL Utility software 15
Command Summary 16
Motion Commands 17
Servo Commands 18
Configuration Commands 19
Communications Commands 21
Register Commands 21
I/O Commands 22
Q Program Commands 23
Command Listing 24
AC - Acceleration Rate 25
AD - Analog Deadband 26
AD - Analog Deadband (SV200 Drives) 27
AF - Analog Filter 28
AG - Analog Velocity Gain 29
AI - Alarm Reset Input 30
AL - Alarm Code 33
AM - Max Acceleration 36
AN - Analog Torque Gain 37
AO - Alarm Output 38
AP - Analog Position Gain 40
AR - Alarm Reset (Immediate) 41
AS - Analog Scaling 42
AT - Analog Threshold 43
AV - Analog Offset Value 44
AV - Analog Offset Value - SV200 45
AX - Alarm Reset (Buffered) 46
AZ - Analog Zero 47

BD - Brake Disengage Delay 48
BE - Brake Engage Delay 49
BO - Brake Output 50
BR - Baud Rate 52
BS - Buffer Status 53
CA - Change Acceleration Current 54
CB - CANopen Baudrate....55
CC - Change Current....56
CD - Idle Current Delay Time....58
CE - Communication Error....59
CF - Anti-resonance Filter Frequency....60
CG - Anti-resonance Filter Gain....61
CI - Change Idle Current....62
CJ - Commence Jogging 64
CM - Command Mode (AKA Control Mode) 65
CN - Secondary Control Mode....67
CO - Node ID/ IP address....68
CP - Change Peak Current 69
CR - Compare Registers 70
CS - Change Speed....71
CT - Continue....72
DA - Define Address 73
DC - Change Distance....74
DD - Default Display Item of LEDs....75
DE - Deceleration....76
DI - Distance/Position 77
DL - Define Limits 78
DL - Define Limits (StepSERVO and SV200 drives)....80
DR - Data Register for Capture....81
DS - Switching Electronic Gearing....82
DW - Dumping Voltage Setting....83
ED - Encoder Direction 84
EF - Encoder Function....85
EG - Electronic Gearing....87
EH - Extended Homing 88
EI - Input Noise Filter 90
EN - Numerator of Electronic Gearing Ratio....91
EP - Encoder Position....92
ER - Encoder Resolution 93

ES - Single-Ended Encoder Usage....94
ES - Absolute Encoder Mode....95
EU - Denominator of Electronic Gearing Ratio....96
FA - Function of the Single-ended Analog Input....97
FC - Feed to Length with Speed Change....98
FD - Feed to Double Sensor....100
FE - Follow Encoder....101
FH - Find Home....102
FI - Filter Input....105
FL - Feed to Length....109
FM - Feed to Sensor with Mask Distance....110
FO - Feed to Length and Set Output....111
FP - Feed to Position....112
FS - Feed to Sensor....113
FX - Filter select inputs....114
FY - Feed to Sensor with Safety Distance....115
GC - Current Command....116
GG - Controller Global Gain Selection....117
HA - Homing Acceleration....118
HC – Hard Stop Current....119
HD - Hard Stop Fault Delay....120
HG - 4th Harmonic Filter Gain....121
HL - Homing Deceleration....122
HO – Home Offset....123
HP - 4th Harmonic Filter Phase....124
HS - Hard Stop Homing....125
HV - Homing Velocity....127
HW - Hand Wheel....128
Immediate Status Commands....129
IA - Immediate Analog....130
IC - Immediate Current (Commanded)....132
ID - Immediate Distance....133
IE - Immediate Encoder....134
IF - Immediate Format....135
IH - Immediate High Output....136
IL - Immediate Low Output....137
IO - Output Status....138
IP - Immediate Position....140
IQ - Immediate Current (Actual)....141

IS - Input Status 142
IT - Immediate Temperature 145
IU - Immediate Voltage 147
IV - Immediate Velocity 148
IX - Immediate Position Error 149
JA - Jog Acceleration 150
JC - Velocity (Oscillator) mode second speed 151
JC - 8 Jog Velocities (SV200 drives) 152
JD - Jog Disable 153
JE - Jog Enable 154
JL - Jog Decel 155
JM - Jog Mode 156
JS - Jog Speed 157
KC - Overall Servo Filter 158
KD - Differential Constant 159
KE - Differential Filter 160
KF - Velocity Feedforward Constant 161
KG – Secondary Global Gain 162
KI - Integrator Constant 163
KJ - Jerk Filter Frequency 164
KK - Inertia Feedforward Constant 165
KP - Proportional Constant 166
KV - Velocity Feedback Constant 167
LA - Lead Angle Max Value 168
LM - Software Limit CCW 170
LP - Software Limit CW 171
LS - Lead Angle Speed 172
LV - Low Voltage threshold 173
MC - Motor Current, Rated 174
MD - Motor Disable 175
ME - Motor Enable 176
MN - Model Number 177
MO - Motion Output 178
MR - Microstep Resolution 181
MS - Control Mode Selection 182
MT - Multi-Tasking 183
MV - Model & Revision 184
NO - No Operation 187
OF - On Fault 188

OI - On Input 189
OP - Option board 190
PA - Power-up Acceleration Current 191
PB - Power-up Baud Rate 193
PC - Power-up Current 194
PD - In-Position Counts 195
PE - In-Position Timing 196
PF - Position Fault 197
PH - Inhibit Pulse Command 198
PI - Power-up Idle Current 199
PK - Parameter Lock 200
PL - Position Limit 201
PM - Power-up Mode 202
PN - Probe On Demand 203
PP - Power-up Peak current 204
PR - Protocol 205
PS - Pause 206
PT - Pulse Type 207
PV - Secondary Electronic Gearing 208
PW - Password 209
QC - Queue Call 210
QD - Queue Delete 211
QE - Queue Execute 212
QG - Queue Goto 213
QJ - Queue Jump 214
QK - Queue Kill 215
QL - Queue Load 216
QR - Queue Repeat 217
QS - Queue Save 218
QU - Queue Upload 219
QX - Queue Load & Execute 220
RC - Register Counter 221
RD - Register Decrement 223
RE - Restart or Reset 224
RI - Register Increment 225
RL - Register Load - immediate 226
RM - Register Move 227
RO - Anti-Resonance ON 228
RR - Register Read 229
RS - Request Status 230

RU - Register Upload 231
RV - Revision Level 232
RW - Register Write 233
RX - Register Load - buffered 234
R+ - Register Add 235
R- - Register Subtract 236
R* - Register Multiply 237
R/ - Register Divide 238
R& - Register AND 239
RI - Register OR 240
SA - Save Parameters 241
SC - Status Code 242
SD - Set Direction 243
SF - Step Filter Frequency 244
SH - Seek Home 245
SI - Enable Input Usage 246
SJ - Stop Jogging 248
SK - Stop & Kill 249
SM - Stop Move 250
SO - Set Output 251
SP - Set Position 252
SS - Send String 253
ST - Stop 254
TD - Transmit Delay 255
TI - Test Input 256
TO - Tach Output 257
TR - Test Register 259
TS - Time Stamp 260
TT - Pulse Complete Timing 261
TV - Torque Ripple 262
VC - Velocity Change 263
VE - Velocity 264
VI - Velocity Integrator Constant 265
VL - Voltage Limit 266
VM - Maximum Velocity 267
VP - Velocity Mode Proportional Constant 268
VR - Velocity Ripple 269
WD - Wait Delay 270
WI - Wait for Input 271

WM - Wait on Move 272
WP - Wait Position 273
WT - Wait Time 274
ZA – Network Communication Time-out (Watchdog) Action 274
ZC - Regen Resistor Continuous Wattage 275
ZE - Network Communication Time-Out (Watchdog) Enable 276
ZR - Regen Resistor Value 277
ZS- Network Communication Time-out (Watchdog) Delay 278
ZT - Regen Resistor Peak Time 279
Data Registers 280
Read-Only data registers 280
Read/Write data registers 280
User-Defined data registers 280
Storage data registers 280
Using Data Registers 281
Loading (RL, RX) 281
Uploading (RL, RU) 281
Writing Storage registers (RW) *(Q drives only)* *282*
Reading Storage registers (RR) *(Q drives only)* *282*
Moving data registers (RM) *(Q drives only)* *282*
Incrementing/Decrementing (RI, RD) *(Q drives only)* *282*
Counting (RC, “I” register) *(Q drives only)* *282*
Math & Logic (R+, R-, R*, R/, R&, RI) *(Q drives only)* *282*
Conditional Testing (CR, TR) *(Q drives only)* *283*
Data Register Assignments 283
Read-Only data registers: a - z 283
Read/Write data registers: A - Z 288
User-Defined data registers: 0 - 9, other characters 292
Appendices 293
Appendix A: Non-Volatile Memory in Q drives 294
Appendix B: Host Serial Communications 295
Appendix C: Host Serial Connections 299
Appendix D: The PR Command 303
Appendix E: Alarm and Status Codes 312
Appendix F: Working with Inputs and Outputs 320
Appendix G: eSCL (SCL over Ethernet) Reference 328
Appendix H: EtherNet/IP 342
Input Assembly (0x64) 344
Input Assembly (0x65) 346

Input Status Details 346
Output Assembly (0x70) 347
Explicit Messaging 354
Type 2 Message Format 360
Table 1: Message Type 1 Command List 364
Table 2: Message Type 2 Commands 369
Table 3: Parameter read/write operands 370
IO Encoding Table 373
Register Encoding Table 374
EtherNet/IP And Q Programs 376
EtherNet/IP on large networks 378
Appendix I: Troubleshooting 379
Appendix J: List of Supported Drives 381
Appendix K: Modbus appendix 388
What is Modbus? 390
Wiring 390
Drive Behavior 391
Monitoring 391
Sending Commands 391
Examples 392
SCL Command Mode Table 393
IO Encoding Table 394
Register Encoding Table 395
Modbus Register Table for Step Drives 398
Modbus Register Table for Servo Drives 406
Modbus Register Table for StepSERVO Drives 416

Available SCL commands

Note: command-specific “data” units and ranges can be found on the dedicated information page for each command in the main part of this manual. This table spans two pages.

Command	Data/SCL Register 1	Data/SCL Register 2	SSM, TSM, TXM	ST, STAC5	STM, SWM	SV200	STF
AD - Analog Deadband	Data	0	x	x	x	x	
AF - Analog Filter Gain	Data	0	x	x	x	x	
AG - Analog Velocity Gain	Data	0	x	x	x	x	
AI - Alarm Reset Input	Data	0	x	x	x	x	x
AM - Max Accel	Data	0	x	x	x	x	x
AN - Analog Torque Gain	Data	0	x			x	
AO - Fault Output	Data	0	x	x	x	x	x
AP - Analog Position Gain	Data	0	x	x	x	x	
AS - Analog Scaling	Data	0	x	x	x	x	
AT - Analog Threshold	Data	0	x	x	x	x	
AV - Analog Offset	Data	0	x	x	x	x	
AZ - Analog Zero	0	0	x	x	x	x	
BD - Brake Release Delay	Data	0	x	x	x	x	x
BE - Brake Engage Delay	Data	0	x	x	x	x	x
BO - Brake Output	Data	0	x	x	x	x	x
CA - Accel Current	Data	0			x		
CC - Continuous / Running Current	Data	0	x	x	x	x	x
CD - Idle Current Delay	Data	0		x	x		
CI - Idle Current	Data	0		x	x		x
CM - Control Mode	Data	0	x	x	x	x	x
CP - Peak Current	Data	0	x	x	x	x	
DC - Set Change Distance	Data	0	x	x	x	x	x
DL - Define Limits	Data	0				x	x
ED - Encoder Direction	0 or 1	0		x	x		
EF - Encoder Function	Data	0	x	x	x	x	
EG - Electronic Gearing	Data	0	x			x	x
EH - Extended Homing	See IO Encoding Table	0	x	x	x	x	x
EP - Encoder Position	Data	0		x		x	
ER - Encoder Resolution	Data	0		STAC5			
FX - Filter Select Inputs	Data	0	x			x	
GC - Current Command	Data	0	x			x	
HA - Homing Accel	‘1’, ‘2’, or ‘3’	Data	x	x	x		x
HC - Hardstop Current Limit	Data	0	x	x	x		
HD - Hard Stop Fault Delay	Data	0		x	x		
HG - Harmonic Smoothing Gain	Data	0		x	x		x

Command	Data/SCL Register 1	Data/SCL Register 2	SSM, TSM, TXM	ST, STAC5	STM, SWM	SV200	STF
HL - Homing Decel	'1', '2', or '3'	Data	x			x	x
HO - Homing Offset	Data	0	x			x	x
HP - Harmonic Smoothing Phase	Data	0		x	x		x
HS - Hard Stop Homing	0 or 1	0	x			x	
HV - Homing Velocity	'1', '2', or '3'	Data	x			x	x
LP - Software Limit CW	Data	0	x			x	x
LM - Software Limit CCW	Data	0	x			x	x
MO - Motion Output	Data	0	x	x	x	x	
MO - Motion Output	'1', '2', or '3'	Data	TXM/TSM34				x
PA - Powerup Accel Current	Data	0			x		
PC - Powerup Max Current	Data	0	x	x	x	x	x
PD - In Position Counts	Data	0	x			x	
PE - In Position Timing	Data	0	x			x	
PF - Position Fault	Data	0	x	x	x	x	
PI - Powerup Idle Current	Data	0		x	x		x
PM - Powerup Mode	Data	0	x	x	x	x	x
PP - Powerup Peak Current	Data	0	x			x	
RL - Register Load (immediate)	Register (ASCII character)	Data	x	x	x	x	x
SA - Save Settings	0	0	x	x	x	x	x
SF - Step Filter Frequency	Data	0	x	x	x		x
SI - Enable Input	Data	0	x	x	x	x	x
SO - Set Output	See IO Encoding Table	0	x	x	x	x	x
SP - Set Position	Data	0	x	x	x	x	x
VC - Change Velocity	Data	0	x	x	x	x	x
ZA – Network Communication Time-out (Watchdog) Action	Data	0	TSM14POE/ TSM23X/ TSM34/ TXM24X/ TXM34/ TXM34X		STM23X		x
ZS – Network Communication Time-out (Watchdog) Delay	Data	0	TSM14POE/ TSM23X/ TSM34/ TXM24X/ TXM34/ TXM34X		STM23X		x

Additional Implicit SCL Commands for SV200

Command	Data/SCL Register 1	Data/SCL Register 2
AD - Analog Deadband (SV200 Drives)	'1', '2', or '3'	Data
AV - Analog Offset Value - SV200	'1', '2', or '3'	Data
CN - Secondary Control Mode		
DS - Switching Electronic Gearing		
EU - Denominator of Electronic Gearing Ratio		
FA - Function of the Single-ended Analog Input		
GG - Controller Global Gain Selection		
JC - 8 Job Velocities (SV200 drives)	'1', '2', '3', '4', '5', '6', '7', or '8'	Data
KC - Overall Servo Filter		
KD - Differential Constant		
KE - Differential Filter		
KF - Velocity Feedforward Constant		
KG – Secondary Global Gain		
KI - Integrator Constant		
KJ - Jerk Filter Frequency		
KK - Inertia Feedforward Constant		
KP - Proportional Constant		
KV - Velocity Feedback Constant		
MS - Control Mode Selection		
PH - Inhibit Pulse Command		
PK - Parameter Lock		
PL - Position Limit		
PV - Secondary Electronic Gearing		
TO - Tach Output		
TT - Pulse Complete Timing		
TV - Torque Ripple		
VI - Velocity Integrator Constant		
VP - Velocity Mode Proportional Constant		
VR - Velocity Ripple		

Explicit Messaging

The AMP EtherNet/IP implementation allows for Explicit Messaging using either a Class 3 connection or the Unconnected Message Manager (UCMM). The service code of this custom profile is 0x3C and the class code is 0x64.

In addition to the custom profile, the following standard objects and services are implemented:

- Message Router Object (Volume 1, Section 5-3)
- Connection Manager (Volume 1, Section 5-7)
- Connection Configuration (Volume 1, Section 5-50)
- Port (Volume 1, Section 3-7)
- Ethernet Link Object (Volume 2, Chapter 5)
- TCP/IP Object (Volume 2, Chapter 5)
- Assembly (Volume 1, Section 5-37)
- CIP Sync Object (Volume 1, Section 5-47.7)

Documentation can be found in the following ODVA specifications (specific sections are noted above next to each object name):

Volume One: Common Industrial Protocol (CIP™), edition 3.8

Volume Two: EtherNet/IP™ Adaptation of CIP, edition 1.9

Services

Name	Service	Class	Instance	Attribute
Profile Code & ARM Firmware Version Example response message: 00.75.00.07.41.5F.00.01.03.4A 0x00 = ARM (Ethernet board) firmware major revision, most significant byte 0x75 = ARM (Ethernet board) firmware major revision, least significant byte 0x00 = ARM (Ethernet board) firmware minor revision, most significant byte 0x07 = ARM (Ethernet board) firmware minor revision, least significant byte 0x41 = ASCII 'A', the profile code 0x5F = 95, Model Number (see table below) 0x00 = Sub-model Number (see table below) 0x01 = 1, Drive firmware major revision number (1.xx) 0x03 = 3, Drive firmware minor revision number (x.03) 0x4A = ASCII 'J', Drive firmware revision letter (x.xxJ) ARM firmware : [major rev] . [minor rev] = [0x0075] . [0x0007] = 117.07	0x0E *("Get Attribute Single")*	0x64	0x00	0x01
Vendor-Specific Device Profile A	0x3C	0x64	0x01	0x01

Explicit Message Types

Two types of explicit messages can be sent to Applied Motion EtherNet/IP drives. Type 1 messages include most of the buffered SCL and Q commands. However, unlike SCL and Q commands that are sent over RS-232, RS-485 and standard Ethernet, Type 1 messages do not support queries. “Immediate” SCL commands cannot be encapsuated in Type 1 messages.

Type 2 messages provide additional functionality not available with Type 1 messages, including the ability to read back settings and registers. Both types can be sent over a Class 3 connection, or they can be sent to the Unconnected Message Manager (UCMM).

Both command message types result in a response message even when no data is requested.

All numerical values are in two’s complement. Integers are sent big endian (most significant byte first).

For detailed SCL and Q command descriptions, please see the main section of this manual. When reading the command descriptions in the main part of this manual, please be advised that the EtherNet/IP encapsulation often requires that different units, and a different range of acceptable values, be used.

Type 1 Message Format

See Table 1 for the complete list of commands. The response message will always echo back the opcode and register code (if present). Also contained in the response message is the drive’s status code, a bit pattern that indicates useful information such as whether there is a fault or if the motor is in motion. For more information, please see the section on the SC command earlier in this manual.

Note: All numerical values are in two’s complement. Integers are sent big endian (most significant byte first).

Command Message Format:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0
B1	Command Axis Number = 0x0			Command Message Type = 0x1				
B2	Register, I/O or other code here for some commands (see Table 1), 0 for all others.							
B3	Opcode							
B4	Parameter1							
B5	Parameter2							
B6	Parameter3							
B7	Parameter4							

Response Message Format:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0
B1	Response Axis Number = 0x0			Response Message Type = 0x1				
B2	Register code for commands QR, RR, RW and RX, 0 for all others							
B3	Opcode							
B4	Status Code MSB							
B5	Status Code LSB							
B6	Unused = 0							
B7	Unused = 0							

Type 1 Message Examples

Example 1: SCL commands required for Point to Point move

AC100 set acceleration rate to 100 rev/sec/sec (6000 rpm/sec)

opcode 0x001E from Table 1

operand 0x258 units are 10 rpm/sec, so 6000 rpm/sec is represented by 600 decimal = 258 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1E	opcode
byte 4	0	unused
byte 5	0	unused
byte 6	2	operand MSB
byte 7	58	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1E	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

DE100 set deceleration rate to 100 rev/sec/sec (6000 rpm/sec)

opcode 0x001F from Table 1

operand 0x258 units are 10 rpm/sec, so 6000/sec is represented by 600 decimal = 258 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1F	opcode
byte 4	0	unused
byte 5	0	unused
byte 6	2	operand MSB
byte 7	58	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1F	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

VE5 set velocity to 5 rev/sec (300 rpm)

opcode 0x001D from Table 1

operand 0x4B0 units are 0.25 rpm, so 300 rpm is represented by 1200 decimal = 4B0 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1D	opcode
byte 4	0	unused
byte 5	0	unused
byte 6	4	operand MSB
byte 7	B0	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1D	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

DI100000 set move distance to 100,000 steps

opcode 0x00B6 from Table 1

operand 0x186A0 units are steps, so 100000 is represented by 186A0 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	not used
byte 3	B6	opcode
byte 4	0	not used
byte 5	1	operand MSB
byte 6	86	operand 2nd LSB
byte 7	A0	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	not used
byte 3	B6	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

FL start the “feed to length” move

opcode 0x0066 from Table 1

operand 0 no operand

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	66	opcode
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	66	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

Example 2: setting an output

SO2L set output 2 low (closed)

opcode 0x008B from Table 1

operand 0x4CB2 LSB is “2” = 0xB2. MSB is “L” = 0x4C (see IO Encoding Table)

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	not used
byte 3	8B	opcode
byte 4	0	not used
byte 5	0	not used
byte 6	4C	operand MSB
byte 7	B2	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	not used
byte 3	8B	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

Example 3: enabling the motor

ME motor enable

opcode 0x009F from Table 1

operand 0 no operand

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	9F	opcode
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	9F	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

Example 4: SCL commands required for Feed to Sensor move

AC200 set acceleration rate to 200 rev/sec/sec (12000 rpm/sec)

opcode 0x001E from Table 1

operand 0x4B0 units are 10 rpm/sec, so 12000 rpm/sec is represented by 1200 decimal = 4B0 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1E	opcode
byte 4	0	unused
byte 5	0	unused
byte 6	4	operand MSB
byte 7	B0	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1E	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

DE150 set deceleration rate to 150 rev/sec/sec (9000 rpm/sec)

opcode 0x001F from Table 1

operand 0x384 units are 10 rpm/sec, so 9000/sec is represented by 900 decimal = 384 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1F	opcode
byte 4	0	unused
byte 5	0	unused
byte 6	3	operand MSB
byte 7	84	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1F	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

VE3 set velocity to 3 rev/sec (180 rpm)

opcode 0x001D from Table 1

operand 0x2D0 units are 0.25 rpm, so 180 rpm is represented by 720 decimal = 2D0 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1D	opcode
byte 4	0	unused
byte 5	0	unused
byte 6	2	operand MSB
byte 7	D0	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	1D	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

DI5000 set move distance to 5,000 steps (this is the distance beyond the sensor where motor will stop)

opcode 0x00B6 from Table 1

operand 0x1388 units are steps, so 5000 is represented by 1388 hex

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	B6	opcode
byte 4	0	operand MSB
byte 5	0	operand 2nd MSB
byte 6	13	operand 2nd LSB
byte 7	88	operand LSB

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	B6	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

FS2R start the “feed to sensor” move, stop 5000 steps after input 2 rising edge

opcode 0x006B from Table 1

operand 0x52B2 LSB is “2” = 0xB2. MSB is “R” = 0x52 (see IO Encoding Table)

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	6B	opcode
byte 4	0	not used
byte 5	0	not used
byte 6	52	condition (R)
byte 7	B2	ionum (2)

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	1	message type
byte 2	0	unused
byte 3	6B	opcode
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

Type 2 Message Format

Message Type 2 commands provide functionality that is not available with Type 1 commands. This is the only way to read back information from the drive. All Type 2 commands require and 8 bit opcode and an 8 bit operand. Return values include a 16 or 32 bit response, as appropriate.

The response message will always echo back the opcode and operand from the command message. Also contained in the response message is the drive's status code, unless other information is requested (e.g. parameter read command). The status code is a bit pattern that indicates useful information such as whether there is a fault or if the motor is in motion. For more information, please see the section on the SC command earlier in this manual.

Command Message Format:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0
B1	Command Axis Number = 0x0			Command Message Type = 0x2				
B2	Opcode (see Table 2)							
B3	Operand (see Table 2)							
B4	Data MSB							
B5	Data LSB [Data 2nd MSB for opcode 0x9E]							
B6	Unused = 0 [Data 2nd LSB for opcode 0x9E]							
B7	Unused = 0 [Data LSB for opcode 0x9E]							

Response Message Format:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0	Reserved = 0
B1	Response Axis Number = 0x0			Response Message Type = 0x2				
B2	Opcode (see Table 2)							
B3	Operand (see Table 2)							
B4	Status MSB [Data MSB for opcodes 0x84, 0x88, 0x89, 0x9F]							
B5	Status LSB [Data LSB for opcodes 0x84, 0x88, 0x89] [Data 2nd MSB for opcode 0x9F]							
B6	Unused = 0 [Data 2nd LSB for opcode 0x9F]							
B7	Unused = 0 [Data LSB for opcode 0x9F]							

Type 2 Message Examples

Example 1: parameter write

AC100 set acceleration rate to 100 rev/sec/sec (6000 rpm/sec)

opcode 0x83 parameter write, from Table 2

operand 0x1E from Table 3

data 0x258 units are 10 rpm/sec, so 6000/sec is represented by 600 decimal = 258 hex

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	83	opcode
byte 3	1E	operand
byte 4	2	data MSB
byte 5	58	data LSB
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	83	opcode
byte 3	1E	operand
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

Example 2: parameter read

AC read back the acceleration rate

opcode 0x84 parameter read, from Table 2

operand 0x1E from Table 3

return value 0x258 units are 10 rpm/sec, so 6000/sec is represented by 600 decimal = 258 hex

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	84	opcode
byte 3	1E	operand
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	84	opcode
byte 3	1E	operand
byte 4	2	read data MSB
byte 5	58	read data LSB
byte 6	0	not used
byte 7	0	not used

Example 3: read absolute position

opcode 0x88 read 32 bit abs posn/enc posn, from Table 2

operand 1 from Table 2, indicates abs posn

return value 0x87654321

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	88	opcode
byte 3	1	operand
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	88	opcode
byte 3	1	operand
byte 4	87	read data MSB
byte 5	65	read data 2nd MSB
byte 6	43	read data 2nd LSB
byte 7	21	read data LSB

Example 4: read encoder position

opcode 0x88 read 32 bit abs posn/enc posn, from Table 2

operand 0 from Table 2, indicates enc posn

return value 0x12345678

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	88	opcode
byte 3	0	operand
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	88	opcode
byte 3	0	operand
byte 4	12	read data MSB
byte 5	34	read data 2nd MSB
byte 6	56	read data 2nd LSB
byte 7	78	read data LSB

Example 5: read Q user register 3

opcode 0x9F read 32 bit Q register, from Table 2

operand 0x33 from Reg Code Table, indicates register ‘3’

return value 0x12345678

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	9F	opcode
byte 3	33	operand
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	9F	opcode
byte 3	33	operand
byte 4	12	read data MSB
byte 5	34	read data 2nd MSB
byte 6	56	read data 2nd LSB
byte 7	78	read data LSB

Example 6: read Q register D

opcode 0x9F read 32 bit Q register, from Table 2

operand 0x44 from Reg Code Table, indicates register 'D'

return value 0x12345678

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	9F	opcode
byte 3	44	operand
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	9F	opcode
byte 3	44	operand
byte 4	12	read data MSB
byte 5	34	read data 2nd MSB
byte 6	56	read data 2nd LSB
byte 7	78	read data LSB

Example 7: write Q register D

opcode 0x9E read 32 bit Q register, from Table 2

operand 0x44 from Reg Code Table, indicates register 'D'

data 0x12345678

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	9E	opcode
byte 3	44	operand
byte 4	12	data MSB
byte 5	34	data 2nd MSB
byte 6	56	data 2nd LSB
byte 7	78	data LSB

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	9E	opcode
byte 3	44	operand
byte 4	?	status code MSB
byte 5	?	status code LSB
byte 6	0	not used
byte 7	0	not used

Table 1: Message Type 1 Command List

For detailed SCL and Q command descriptions, please see the main section of this manual. When reading the command descriptions in the main part of this manual, please be advised that the EtherNet/IP encapsulation often requires that different units, and a different range of acceptable values, be used.

Command	Description	opcode (hex)	Reg code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Range ST5 / ST10 / STAC5	Units
Motion Commands									
AC	P_TO_P_ACCEL,	1E	0			accel rate		1..32000	10 rpm/sec
AM	MAX_ACCEL,	16	0			accel rate		1..32000	10 rpm/sec
AN	analog torque gain	5F	0			gain factor		100	amps*100
AX	ALARM_RESET	BA	0						
CJ	START_JOGGING	96	0						
DC	SET_CHNG_DISTANCE	B7	0	32 bit distance or position				+/-2,147,483,647	steps
DE	P_TO_P_DECEL,	1F	0			decel rate		1..32000	10 rpm/sec
DI	SET_REL_DISTANCE	B6	0	32 bit distance or position				+/-2,147,483,647	steps
EF	ENCODER_FUNCTION	D6	0			function		0,1,2 or 6	0 = Encoder function off 1 = Stall Detection 2 = Stall Prevention 6 = Stall prevention w/ time-out
EG	Steps/rev / 2	26	0			steps/rev		100..25600	steps/rev divided by 2
EI	input noise filter	43	0				filter value	0..255	filter freq = 15,000,000/EI
EP	ENCODER_POSITION	98	0	32 bit encoder position				+/-2,147,483,647	counts
EN	electronic gearing numerator	2A	0			numerator		0..1000	
EU	electronic gearing denomintor	2B	0			denominator		0..1000	set to 0 to turn off electronic gearing
FC	P_TO_P_CHANGE	6D	0						
FD	feed to double sensor	69	0	cond 2	io2	cond 1	io1	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table
FE	FOLLOW ENCODER	CC	0			cond	io	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table
FL	feed to length (relative move)	66	0						
FM	Feed to Sensor with mask distance	6A	0			cond	io	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table
FO	feed and set output	68	0			cond	io	ST: Y1..Y4, L or H STAC5: 1..4, Y1,Y2. L or H	see IO Encoding Table
FP	feed to position (absolute move)	67	0						

Command	Description	opcode (hex)	Reg code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Range ST5 / ST10 / STAC5	Units
FS	Feed to Sensor	6B	0			cond	io	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table
FY	Feed to Sensor with safety distance	6C	0			cond	io	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table
GG	global gain select	EF	0				gain select	1..3	see HCR GG command
HW	Hand wheel	AB	0			cond	io	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table
JA	VM_ACCEL,	1B	0			jog accel rate		1..32000	10 rpm/sec
JD	JOG_DISABLE	A3	0						
JE	JOG_ENABLE	A2	0			direction		1=cw enable, 2=ccw enable, 3=both	
JL	VM_DECEL,	1C	0			jog decel rate		1..32000	10 rpm/sec
JS	VM_VELOCITY,	1A	0			jog speed		0..32000	.25 rpm
LM	CCW software limit	EA	0	ccw position limit				+/-2,147,483,647	steps
LP	CW software limit	E9	0	cw position limit				+/-2,147,483,647	steps
MD	MOTOR_DISABLE	9E	0						
ME	MOTOR_ENABLE	9F	0						
PD	in position counts	71	0			in position counts		0..32767	encoder counts
PE	in position time	72	0			in position time		0..30000	msec*4
SH	SEEK_HOME, ionum+cond	6E	0			cond	io	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table
SM	STOP_MOVING	B5	0			decel code		D (DE rate) or M (AM rate)	'D' or 'M'
SP	SET_ABS_POSITION	A5	0	32 bit abs position				+/-2,147,483,647	steps
TO	tach output	64	0				tach code	0..7	see HCR TO command
TV	torque ripple	65	0			torque ripple		100	amps*100
VC	CHANGE_VELOCITY,	4A	0			speed		1..32000	.25 rpm
VE	P_TO_P_VELOCITY,	1D	0			speed		1..32000	.25 rpm
VR	velocity ripple	63	0				velocity ripple	0..136	rev/sec

Command	Description	opcode (hex)	Reg code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Range ST5 / ST10 / STAC5	Units
Configuration Commands									
AS	Analog Scaling	D1	0			scale code		0..7	0 = single-ended +/- 10 volts 1 = single-ended 0 - 10 volts 2 = single-ended +/- 5 volts 3 = single-ended 0 - 5 volts 4 = differential +/- 10 volts 5 = differential 0 - 10 volts 6 = differential +/- 5 volts 7 = differential 0 - 5 volts
BD	BRAKE RELEASE DELAY	40	0			brake release delay		1..32000	msec
BE	BRAKE ENGAGE DELAY	41	0			brake engage delay		1..32000	msec
CA	ACCEL_CURRENT,	61	0			accel current		not supported	10 rpm/sec
CC	Running CURRENT	18	0			motor current when running		500 / 1000 / 500	.01 amps
CD	IDLE_CURRENT_DELAY,	4F	0			delay time		1..32000	msec
CI	IDLE CURRENT	19	0			motor current when idle		500 / 1000 / 500	.01 amps
CM	CONTROL_MODE,	10	0			mode code		7, 10..18, 21, 22	
EF	Encoder Function	D6	0			function code		0,1,2 or 6	0 = Encoder function off 1 = Stall detection 2 = Stall prevention 6 = Stall prevention w/ time-out
ER	ENCODER_RESOLUTION,	20	0			encoder line count		50..32000	lines/rev (counts/rev/4)
FI	Filter Input	C0	io			filter value		0..32767	CPU cycles
FX	Filter Select Inputs	D3	0			input bank		0 or 1	1=IN/OUT1, 0=IN/OUT2
HG	harmonic smoothing gain	4	0			gain		0..32000	
HP	harmonic smoothing phase	5	0			phase		+/-255	
PF	POSITION_FAULT,	21	0			posn fault limit		1..32000	encoder counts
PM	OPERATION_MODE,	44	0			mode code		2 or 7	
SF	STEP_FILTER_FREQUENCY,	6	0			freq		100..25000	0.1 Hz
I/O Commands									
AD	ANALOG_DEADBAND	D2	0			deadband		0..255	mV
AF	ANALOG_FILTER_GAIN,	4C	0			freq		0..32000	Filter value = 72090 / [(1400 / Hz) + 2.2]. 0=no filter
AG	ANALOG_VELOCITY_GAIN,	3B	0			speed at full scale		+/-32000	.25 rpm
AI	ALARM_RESET INPUT	46	0			state		1..3	
AO	FAULT OUTPUT	47	0			state		1..3	

Command	Description	opcode (hex)	Reg code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Range ST5 / ST10 / STAC5	Units
AP	ANALOG_POSITION_GAIN,	4B	0			posn at full scale		1..32000	steps
AS	ANALOG_SCALING	D1	0			input range		0..7	0 = single-ended +/- 10 volts 1 = single-ended 0 - 10 volts 2 = single-ended +/- 5 volts 3 = single-ended 0 - 5 volts 4 = differential +/- 10 volts 5 = differential 0 - 10 volts 6 = differential +/- 5 volts 7 = differential 0 - 5 volts
RE	DSP RESET	A4	0						
AT	ANALOG_THRESHOLD,	4D	0			threshold voltage		+/-32767	ADC Counts 32767 = +10 volts -32767 = -10 volts
AV	ANALOG_OFFSET,	3C	0			offset		+/-32000	ADC counts
AZ	AUTO_OFFSET	A1	0						
BD	BRAKE RELEASE DELAY	40	0			brake release delay		1..32000	msec
BE	BRAKE ENGAGE DELAY	41	0			brake engage delay		1..32000	msec
BO	BRAKE_OUTPUT,	48	0			state		1..3	
DL	DEFINE_LIMITS,	42	0			state		1..3	
FI	FILTER_INPUT	C0	io			filter value		0..32767	CPU cycles
FX	FILTER_SELECT_INPUTS	D3	0			input bank		0=extended, 1 = main board	
JD	JOG_DISABLE	A3	0						
JE	JOG_ENABLE	A2	0			direction		1=cw enable, 2=ccw enable, 3=both	
MO	MOVE_OUTPUT,	49	0			state		1..3	
MO	MOVE OUTPUT (SV200, STF, TSM/TXM34)	49	io			state		1..3	
SI	ENABLE INPUT	45	0			state		1..3	
SO	Set Output	8B	0			cond	io	ST: Y1..Y4, L or H STAC5: 1..4, Y1,Y2. L or H	see IO Encoding Table
TI	Test Input	A8	0			cond	io	ST: X0..X8, L/H/F/R STAC5: X0..X4, 1..8. L/H/F/R	see IO Encoding Table

Command	Description	opcode (hex)	Reg code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Range ST5 / ST10 / STAC5	Units
Register Commands									
RX	REGISTER_LOAD	AE	reg	value (16 or 32 bits, depending on register type)				reg: A..Z or 0..9 value: +/- 2147483647 (long data registers) +/- 32767 (short data registers)	see register code table
TR	Test Register Immediate	AC	reg	value (16 or 32 bits, depending on register type)				reg: a..z or A..Z or 0..9 value: +/- 2147483647 (long data registers) +/- 32767 (short data registers)	see register code table
Q Program Commands									
AX	ALARM_RESET	BA	0						
QX	Queue Load Execute	78	0			segment		1..12	
WD	WAIT_DELAY_REGISTER	BF	reg					a..z or A..Z or 0..9	see register code table

Table 2: Message Type 2 Commands

Opcode	Definition	Operand	Action
83	Parameter Write	see Table 3	write a 16 bit parameter to a register. Add 128 (0x80) to operand for non-volatile (flash) write
84	Parameter read	see Table 3	Returns the 16 bit parameter indicated by operand
87	Read alarm code	0	Returns alarm history value indicated by operand
88	Read Encoder/Abs Posn	0	Returns the 32 bit encoder position
		1	Returns the 32 bit absolute position
8B*	Set Output (immediate)	bit 7 state, bits 0-6 output	Set the given output to given state.
8E	Clear Fault (AR)	0	Clear the drive fault. A motor enable must be sent to re-enable the motor
98	Stop Motion, Kill Buffer (SK)	decel rate	stops a move, purge all commands from buffer. 0=use quick decel (AM), 1=use normal decel (DE or JL)
9E**	Write Q Register (RL)	see Reg Encoding table	write a 16 or 32 bit parameter to a Q register (A..Z or 0..9, etc)
9F	Read Q Register (RL)	see Reg Encoding table	read a 16 or 32 bit Q register (a..z, A..Z or 0..9, etc)
A1	Queue Load (QL)	0	load incoming Type 1 commands into Q buffer
A2	Queue Save (QS)	segment number 1..12	saves Q buffer as a Q segment
A3	Stop Motion (ST)	decel rate	stops a move. 0=use quick decel (AM), 1=use normal decel (DE or JL)
FF	UDP port reset	0	Opens UDP port 7775 and listens for a new connection
		1	Closes and resets UDP port 7775

*Type 2 Set Output Immediate (opcode 8B) operand table

Operand	00	01	02	03	04	05	80	81	82	83	84	85
ST, STF	OUT1 high	OUT2 high	OUT3 high	OUT4 high			OUT1 low	OUT2 low	OUT3 low	OUT4 low		
STAC5	Y1 high	Y2 high	OUT1 high	OUT2 high	OUT3 high	OUT4 high	Y1 low	Y2 low	OUT1 low	OUT2 low	OUT3 low	OUT4 low
STM23, STM24, SWM24, TXM24	Y1 high						Y1 low					
SV200	Y1 high	Y2 high	Y3 high	Y4 high	Y5 high	Y6 high	Y1 low	Y2 low	Y3 low	Y4 low	Y5 low	Y6 low

***Q register writes are not range checked, so be careful before you write.*

Table 3: Parameter read/write operands

All values are HEX

Command	Description	Index	Q Register Char
Read/write			
--	MISC_FLAGS	5B	F
--	ENCODER_ATTEMPTS,	62	
AC	P_TO_P_ACCEL,	1E	A
AD	ANALOG_DEADBAND,	22	
AF	ANALOG_FILTER_GAIN,	4C	
AG	ANALOG_VELOCITY_GAIN,	3B	H
AI	ALARM_RESET,	46	
AM	MAX_ACCEL,	16	
AN	ANALOG_TORQUE_GAIN	3D	
AO	ALARM_OUTPUT,	47	
AP	ANALOG_POSITION_GAIN,	4B	X
AT	ANALOG_THRESHOLD,	4D	Y
AV	ANALOG_OFFSET,	3C	Z
BD	BRAKE_DELAY,	40	
BE	BRAKE_DELAY_2,	41	
BO	BRAKE_OUTPUT,	48	
CA	ACCEL_CURRENT [STM only]	61	
CC	MAX_CURRENT	18	N
CD	IDLE CURRENT DELAY	4F	
CF	Anti-resonance Frequency	50	
CG	Anti-resonance Gain	51	
CI	IDLE CURRENT	19	O
CM	CONTROL_MODE,	10	
CN	CONTROL_MODE_1,	27	
CP	PEAK_CURRENT,	19	0
DD	DEFAULT_DISPLAY,	5E	
DE	P_TO_P_DECEL,	1F	B
DL	DEFINE_LIMITS,	42	
DO	DBC_RATIO_N,	23	
ED	ENC_DIRECTION,	5F	
EE	FULL_ENCODER_RESOLUTION,//	28	
EG	Steps/rev divided by 2	26	R
EN	ELEC_RATIO_N,	2A	
ER	ENCODER_RESOLUTION,	20	
EU	ELEC_RATIO_D,	2B	
FA	ANALOG_FUNCTION	2D	
GC	CURRENT_COMMAND,	12	

Command	Description	Index	Q Register Char
GV	VOLTAGE_COMMAND,	13	
HG	HYPERBOLIC_GAIN,	4	
HP	HYPERBOLIC_PHASE,	5	
JA	VM_ACCEL,	1B	K
JL	VM_DECEL,	1C	L
JM	JOG_MODE,	55	M
JS	VM_VELOCITY,	1A	M
KC	CURRENT_FILTER_GAIN,	51	
KD	DAMPING_GAIN,	D	
KE	DAMPING_FILTER_GAIN,	4F	
KF	VELOCITY_DAMPING, PROPORTIONAL GAIN	A	
KG	GLOBAL_GAIN,	24	
KI	INTEG_GAIN,	8	
KJ	JERK,	B	
KK	INERTIA_FF_GAIN,	C	
KL	VELOCITY_DAMPING,	5C	
KP	GLOBAL_GAIN,	7	
KV	VELOCITY_GAIN,	9	
MA	MASK_ALARM,	60	
MO	MOVE_OUTPUT,	49	
MV	ModelNum:F/W version	1	
PD	POSERR_RANGE,	71	
PE	INRANGE_COUNT,	72	
PF	POSITION_FAULT,	21	
PK	PARAMETERS_LOCK,	5D	
PL	IN_POSITION_COUNT,	14	
PM	OPERATION_MODE,	44	
PR	PROTOCOL,	59	
PT	Pulse Type (STF only)	3F	
PV	STEPS_REV_FC,	29	
SF	STEP_FILTER_FREQUENCY,	6	
SI	SERVO_ENABLE,	45	
SZ	STEP_MODE,	3F	
TD	ACK_DELAY,	5A	
TO	TACH_OUT_COUNT,	64	
TT	STEP_COMPLETE_CHECK_PER,	C	
TV	TORQUE_VALVE,	65	
VC	CHANGE_VELOCITY,	4A	U
VE	P_TO_P_VELOCITY,	1D	V

Command	Description	Index	Q Register Char
VF	VELOCITY_VOLTAGE_FF,	22	
VI	VELOCITY_MODE_INTEG_GAIN,//	F	
VP	VELOCITY_MODE_GAIN,	E	
VR	VELOCITY_VALVE,	63	
ZC	CLAMP_CONT,	57	
ZR	CLAMP_RESISTANCE,	56	
ZT	CLAMP_TIME,	58	
Read Only			
--	DSP firmware letter	8E	
--	Hall Pattern (SV7 only)	8F	
--	Sub Model (STM only)	90	
--	IsServo (ST/SV only: 1=servo, 0=stepper). Can be used to tell if drive is servo or stepper	91	
AL	alarm code	81	f
BS	Buffer Status	94	
EP	encoder count upper	84	
EP	encoder count lower	85	
IA	command voltage (Ain)	83	a
IC	command current	88	c
IO	Output Status (reads back outputs)	95	
IQ	actual current	89	q
IS	IN/OUT 2 input status [STAC5 only]	8D	y
ISX	IN/OUT 1 input status	82	i
IT	drive temp	87	t
IU	supply voltage	86	u
IV	actual speed	8B	v
IV1	target speed	8C	w
IX	position error	8A	x
OP	DriveOptions – bit pattern indicating presence of option boards. Bit 0 = Encoder Bit 1 = RS-485 Bit 2 = CANopen Bit 3 = reserved Bit 4 = Resolver Bit 5 = MCF (encoder in and out – SV7 only) Bit 6 = Ethernet	92	
SC	status word	80	s

IO Encoding Table

Useful ASCII values for IO commands

On STAC5, inputs X1-X4 and outputs Y1 & Y2 are on the DB15 (IN/OUT 1) connector. Input X0 is the encoder index signal. Inputs 1-8 and outputs 1-4 are on the DB25 (IN/OUT 2) connector.

Character	hex code	Signifies				
		ST5 & ST10, STF, TSM34	**STAC5**	**SSM23, STM23/24, SWM24, TXM24**	**TXM34**	**SV200**
n/a	0xB0	encoder index signal	encoder index signal	encoder index signal	encoder index signal	index
n/a	0xB1	input X1 or output Y1	input X1 or output Y1	IN1/STEP or OUT	input X1 or output Y1	X1/Y1
n/a	0xB2	input X2 or output Y2	input X2 or output Y2	IN2/DIR	input X2 or output Y2	X2/Y2
n/a	0xB3	input X3 or output Y3	input X3	IN3/EN	input X3 or output Y3	X3/Y3
n/a	0xB4	input X4 or output Y4	input X4	n/a	input X4	X4/Y4
n/a	0xB5	input X5	n/a	n/a	input X5	X5/Y5
n/a	0xB6	input X6	n/a	n/a	n/a	X6/Y6
n/a	0xB7	input X7	n/a	n/a	n/a	X7
n/a	0xB8	input X8	n/a	n/a	n/a	X8
n/a	-	input X9	-	-	-	X9
n/a	-	input X10	-	-	-	X10
n/a	-	input X11	-	-	-	X11
n/a	-	input X12	-	-	-	X12
1	0x31	n/a	input 1 or output 1	IN1/STEP - FI only	n/a	X09
2	0x32	n/a	input 2 or output 2	IN2/DIR - FI only	n/a	X10
3	0x33	n/a	input 3 or output 3	IN3/EN - FI only	X3 - FI only	X11
4	0x34	n/a	input 4 or output 4	n/a	X4 - FI only	X12
5	0x35	n/a	input 5	n/a	X5 - FI only	
6	0x36	n/a	input 6	n/a	n/a	
7	0x37	n/a	input 7	n/a	n/a	
8	0x38	n/a	input 8	n/a	n/a	
L	0x4C	low state (closed)	low state (closed)	low state (closed)	low state (closed)	low
H	0x48	high state (open)	high state (open)	high state (open)	high state (open)	high
R	0x52	rising edge	rising edge	rising edge	rising edge	rising
F	0x46	falling edge	falling edge	falling edge	falling edge	falling

Register Encoding Table

Register Name	Use	equivalent SCL command	Code	Size	Read Only
0	Accumulator		0x30	long	
1	user defined		0x31	long	
2	user defined		0x32	long	
3	user defined		0x33	long	
4	user defined		0x34	long	
5	user defined		0x35	long	
6	user defined		0x36	long	
7	user defined		0x37	long	
8	user defined		0x38	long	
9	user defined		0x39	long	
:	user defined		0x3A	long	
;	user defined		0x3B	long	
<	user defined		0x3C	long	
=	user defined		0x3D	long	
>	user defined		0x3E	long	
?	user defined		0x3F	long	
@	user defined		0x40	long	
[	user defined		0x5B	long	
\	user defined		0x5C	long	
]	user defined		0x5D	long	
^	user defined		0x5E	long	
_	user defined		0x5F	long	
`	user defined		0x60	long	
a	analog command	IA	0x61	short	yes
b	Q line number		0x62	short	yes
c	current command	IC	0x63	short	yes
d	relative distance	ID	0x64	long	yes
e	encoder position	IE, EP	0x65	long	yes
f	alarm code	AL	0x66	long	yes
g	sensor position		0x67	short	yes
h	condition code		0x68	short	yes
i	X inputs (IN/OUT 1)	ISX	0x69	short	yes
j	analog IN1	IA1	0x6A	short	yes
k	analog IN2	IA2	0x6B	short	yes
l	absolute position		0x6C	long	yes
m	control mode	CM	0x6D	short	yes
n	velocity mode state		0x6E	short	yes
o	point to point state		0x6F	short	yes

Register Name	Use	equivalent SCL command	Code	Size	Read Only
p	Q segment		0x70	short	yes
q	actual current	IQ	0x71	short	yes
r	average regen power		0x72	short	yes
s	status code	SC	0x73	short	yes
t	drive temperature	IT	0x74	short	yes
u	bus voltage	IU	0x75	short	yes
v	actual velocity	IV0	0x76	short	yes
w	target velocity	IV1	0x77	short	yes
x	position error	IX	0x78	long	yes
y	IN/OUT 2 inputs	IS	0x79	short	yes
z	phase error		0x7A	short	yes
A	accel rate	AC	0x41	short	
B	decel rate	DE	0x42	short	
C	change distance	DC	0x43	long	
D	distance	DI	0x44	long	
E	position offset		0x45	long	
F	other (misc) flags		0x46	long	
G	current command	GC	0x47	short	
H	analog velocity gain		0x48	short	
I	input counter		0x49	long	
J	jog speed		0x4A	short	
K	jog accel rate		0x4B	short	
L	jog decel rate		0x4C	short	
M	max velocity	JS	0x4D	short	
N	continuous current	CC	0x4E	short	
O	idle current	CI	0x4F	short	
P	absolute position command		0x50	long	
Q	reserved		0x51		
R	steps/rev	EG	0x52	short	
S	pulse count		0x53	long	
T	total count		0x54	long	
U	change speed	VC	0x55	short	
V	velocity	VE	0x56	short	
W	time stamp		0x57	short	
X	analog position gain	AP	0x58	short	
Y	analog threshold	AT	0x59	short	
Z	analog offset	AV	0x5A	short	

EtherNet/IP And Q Programs

To provide additional functionality and autonomy, Q programs can be stored in EtherNet/IP drives. These programs can be started and stopped “on demand” using explicit messaging. The *Q Programmer* application is used to compose, download and test Q programs. Please avoid sending EtherNet/IP messages to the drive while the *Q Programmer* software is running.

To start a Q program from an EtherNet/IP message, you must send a Type 1 message with opcode 0x78 (the QX command) or the “QX” command through the Output Assembly with command “0x0a”. You’ll need to specifiy the Q segment number, as shown in the example. This allows you to store up to 12 Q segments, or subprograms, and operate them independently. Q segments can also call each other once one has been started.

Example: Starting Q Segment 1

QX1 start Q segment 1

opcode 0x0078 from Table 1

operand 0x1 segment 1 (up to 12 segments are allowed in a Q program)

Type 1 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	0	not used
byte 3	78	operand
byte 4	0	unused
byte 5	0	unused
byte 6	0	unused
byte 7	1	segment number

Type 1 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	0	not used
byte 3	78	operand
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

Once a Q segment has begun, Type 1 messages are no longer permitted, because the CPU is busy executing the commands in the Q segment. To stop a Q program, you must use a Type 2 “SK” message (opcode 98, as shown in the next example). Q programs also stop running if they encounter a blank line in the segment. This makes it possible to launch a segment, have it complete a task, and stop by itself.

Example: Stopping a Q Program

SK stop the Q program

opcode 0x98 from Table 2

operand decel rate (0 = use quick decel rate from AM, 1 = use normal decel rate from DE or JL)

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	98	opcode
byte 3	0	operand
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	98	opcode
byte 3	0	operand
byte 4	?	status code MSB
byte 5	?	status code LSB
byte 6	0	not used
byte 7	0	not used

Communicating with a Q Program While It's Running

You can use Type 2 commands to read and write registers while a Q program is running. The Q program can send information to the host by changing a register that the host is polling. Registers 0 - 9 can be polled using the Type 2 User Register Read command (opcode 9A).

The host can make changes to the Q program operation by writing to parameters that the program uses. For example, you could change the motor speed sending a parameter write message that alters VE (Type 2 message, opcode 83, operand 1D). The speed change will take effect on the next move.

Changes that affect a Q program immediately can be made using the Write Q Register command (message type 2, opcode 9E). For example, if the motor is jogging after having been sent a CJ command, writing to register J will result in an immediate speed change. *Please note that Q register writes are not range checked, so be careful before you write.*

How to Know if a Q Program Has Stopped

Since a Q program can be launched and allowed to stop itself when it encounters a blank line, you may want to know when it stops. You can do this by polling for the status word and observing bit 14. This bit is a one if the program is executing. To fetch the status word, use the Type 2 Parameter Read command with operand 0x80 as shown below.

Example: Checking Status While a Q Program is Running

opcode 0x84 parameter read, from Table 2

operand 0x80 status code, from Table 3

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	84	opcode
byte 3	80	operand
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	0	not used

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	84	opcode
byte 3	80	operand
byte 4	?	status code MSB
byte 5	?	status code LSB
byte 6	0	not used
byte 7	0	not used

Typical return values:

0001 Motor enabled, Q program not running

4001 Motor enabled, Q program running

4801 Motor enabled, Q running, Wait Time command executing

4019 Motor enabled, motor moving, Q running

For more information about the status code, please read about the SC command in the main part of this manual.

EtherNet/IP on large networks

Once a computer connects to an Applied Motion EtherNet/IP drive with Applied Motion software such as ST Configurator, STAC Configurator or Q Programmer, that connection is maintained until power is cycled. In most cases this will be acceptable because only one computer will ever need to connect to the drive for monitoring or Q program download. In large complex installations however, it may simply not be feasible to cycle power to the machine every time a new technician connects to the drive.

To address this, we have implemented opcode 0xFF. Using an operand of 1 will allow the user to forcibly reset the maintenance port (UDP port 7775), effectively yielding control of the drive. Once reset, the port must be reinitialized, which requires opcode 0xFF to be sent again, this time with an operand of 0. This will instruct the drive to accept a new connection from the next computer that tries to connect using Applied Motion software.

It is important to understand that only one host computer may be connected to the drive at any given time. To change hosts again, simply repeat the sequence.

Example: Close and reset UDP port for access by another host

opcode 0xFF from Table 2

operand 0x1 Close and reset UDP port 7775.

Type 2 Command Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	FF	opcode
byte 3	0	not used
byte 4	0	not used
byte 5	0	not used
byte 6	0	not used
byte 7	1	operand

Type 2 Response Message Payload		
byte 0	0	reserved
byte 1	2	message type
byte 2	FF	opcode
byte 3	0	not used
byte 4	?	Status Code MSB
byte 5	?	Status Code LSB
byte 6	0	not used
byte 7	0	not used

Remember, this is a two step process. First the port must be closed and reset, as shown above. Once reset, the port must be opened for new connections, which may be accomplished by sending opcode FF again, but this time with an operand of 0.

Appendix I: Troubleshooting

This Appendix addresses potential issues that may occur while using AMP equipment.

NOTE: Every drive must be configured with AMP software prior to operation. For stepper systems, use the appropriate Configurator utility, while QuickTuner should be used for servos. It is never safe to assume that the configuration state of the drive is known when it is received. This step should not be considered optional.

Error Message / Indication	Explanation	Solution
While streaming commands to the drive, it behaves erratically or does not send legible ACK / NACK responses.	The drive's command buffer may be full, which may cause unpredictable behavior.	It is recommended that the user receive and process the drive's ACK / NACK character before sending the next command. This will ensure that the drive's command buffer never overflows and the drive behaves normally. If this is not possible, a delay should be introduced between commands that are streamed to the drive. A delay of approximately 10ms should be sufficient for all commands that do not cause motion.
"The drive is not responding. Is it connected to the right port and turned on?"	The software is unable to communicate to the drive. There are four common causes for this error: 1 - The drive is not powered. 2 - The software is using the wrong COM port. 3 - The drive was already running before the software was launched. (wrong power-up sequence) 4 - The USB/Serial converter is faulty or not supported by AMP. If an onboard 9-pin COM port is not available, use a USB/Serial converter based on the FTDI chipset. The chipset used will be shown on the converter's documentation. Contact AMP for specific device recommendations. Hint: If communications have been established, AMP software will display the drive's firmware revision along with the model number. If this box is empty, communications have not been established.	1 - Apply power to the drive. 2 - Physical 9-pin COM ports are typically assigned COM1 or COM2. USB Adaptors are often assigned arbitrary COM port identifiers. Check your computer's hardware settings in the Control Panel to verify which COM port your device is using. 3 - Ensure that the software is running and using the correct COM port. Then, cycle power on the drive. This will allow the software to intercept the drive's power-up packet (as detailed in Appendix B) and initiate communications.
"You have not set the load inertia in the Motor Settings. The electronic damping and anti-resonance will work better if you set the load inertia accurately. Do you want to download your settings anyway?"	The drive is missing important information used to properly configure the anti-resonance features. The motor will run without this information, but it may not be as smooth as it otherwise could be. This is generally acceptable only for initial testing, and should be addressed before normal operation.	Set the load inertia. Depending on the configuration software used, it is either possible to enter the actual calculated load inertia or a best-guess estimate of the inertia ratio (load : motor). For example, if the load inertia is five times that of the motor's rotor, the ratio would be entered as 5 : 1.

Error Message / Indication	Explanation	Solution
Drive's LED blinks red and green	An alarm or fault condition exists. The display consists of a specific number of red and green blinks, and will repeat continuously until resolved.	Fault codes are drive-dependent. Consult Appendix E and your drive's hardware manual for specific information.
Drive's LED shows solid red	A firmware download was interrupted, and the drive is unable to boot properly.	Cycle power on the drive and repeat the firmware download process.

Appendix J: List of Supported Drives

Drive	Description
Integrated Steppers	
STM17S-3AN	NEMA 17 Integrated Stepper, RS-232
STM17S-3AE	NEMA 17 Integrated Stepper, RS-232, Encoder
STM17S-3RN	NEMA 17 Integrated Stepper, RS-485
STM17S-3RE	NEMA 17 Integrated Stepper, RS-485, Encoder
STM17Q-3AN	NEMA 17 Integrated Stepper, Q Programming, RS-232
STM17Q-3AE	NEMA 17 Integrated Stepper, Q Programming, RS-232, Encoder
STM17Q-3RN	NEMA 17 Integrated Stepper, Q Programming, RS-485
STM17Q-3RE	NEMA 17 Integrated Stepper, Q Programming, RS-485, Encoder
STM17C-3CN	NEMA 17 Integrated Stepper, CANOpen
STM17C-3CE	NEMA 17 Integrated Stepper, CANOpen, Encoder
STM23S-2AN	NEMA 23 Integrated Stepper, 2-stack motor, RS-232
STM23S-2AE	NEMA 23 Integrated Stepper, 2-stack motor, RS-232, Encoder
STM23S-2RN	NEMA 23 Integrated Stepper, 2-stack motor, RS-485
STM23S-2RE	NEMA 23 Integrated Stepper, 2-stack motor, RS-485, Encoder
STM23Q-2AN	NEMA 23 Integrated Stepper, 2-stack motor, Q Programming, RS-232
STM23Q-2AE	NEMA 23 Integrated Stepper, 2-stack motor, Q Programming, RS-232, Encoder
STM23Q-2RN	NEMA 23 Integrated Stepper, 2-stack motor, Q Programming, RS-485
STM23Q-2RE	NEMA 23 Integrated Stepper, 2-stack motor, Q Programming, RS-485, Encoder
STM23S-3AN	NEMA 23 Integrated Stepper, 3-stack motor, RS-232
STM23S-3AE	NEMA 23 Integrated Stepper, 3-stack motor, RS-232, Encoder
STM23S-3RN	NEMA 23 Integrated Stepper, 3-stack motor, RS-485
STM23S-3RE	NEMA 23 Integrated Stepper, 3-stack motor, RS-485, Encoder
STM23Q-3AN	NEMA 23 Integrated Stepper, 3-stack motor, Q Programming, RS-232
STM23Q-3AE	NEMA 23 Integrated Stepper, 3-stack motor, Q Programming, RS-232, Encoder
STM23Q-3RN	NEMA 23 Integrated Stepper, 3-stack motor, Q Programming, RS-485
STM23Q-3RE	NEMA 23 Integrated Stepper, 3-stack motor, Q Programming, RS-485, Encoder
STM23C-3CN	NEMA 23 Integrated Stepper, 3-stack motor, Q Programming, CANOpen
STM23C-3CE	NEMA 23 Integrated Stepper, 3-stack motor, Q Programming, CANOpen, Encoder
STM24SF-3AN	NEMA 24 Integrated Stepper, RS-232
STM24SF-3AE	NEMA 24 Integrated Stepper, RS-232, Encoder
STM24SF-3RN	NEMA 24 Integrated Stepper, RS-485
STM24SF-3RE	NEMA 24 Integrated Stepper, RS-485, Encoder
STM24QF-3AN	NEMA 24 Integrated Stepper, Q Programming, RS-232
STM24QF-3AE	NEMA 24 Integrated Stepper, Q Programming, RS-232, Encoder
STM24QF-3RN	NEMA 24 Integrated Stepper, Q Programming, RS-485
STM24QF-3RE	NEMA 24 Integrated Stepper, Q Programming, RS-485, Encoder
STM24C-3CN	NEMA 24 Integrated Stepper, CANOpen
STM24C-3CE	NEMA 24 Integrated Stepper, CANOpen, Encoder

Drive	Description
ST Drives	
ST5-S	5A DC Stepper Drive, RS-232
ST5-Plus	5A DC Stepper Drive, Q Programming, RS-232
ST5-Q-NN	5A DC Stepper Drive, Q Programming, RS-232
ST5-Q-NE	5A DC Stepper Drive, Q Programming, RS-232, Encoder
ST5-Q-RN	5A DC Stepper Drive, Q Programming, RS-485
ST5-Q-RE	5A DC Stepper Drive, Q Programming, RS-485, Encoder
ST5-Q-EN	5A DC Stepper Drive, Q Programming, Ethernet
ST5-Q-EE	5A DC Stepper Drive, Q Programming, Ethernet, Encoder
ST5-IP-EN	5A DC Stepper Drive, Q Programming, EtherNet/IP
ST5-IP-EE	5A DC Stepper Drive, Q Programming, EtherNet/IP, Encoder
ST5-Si-NN	5A DC Stepper Drive, Si Programming, RS-232
ST5-Si-NE	5A DC Stepper Drive, Si Programming, RS-232, Encoder
ST5-C-CN	5A DC Stepper Drive, CANOpen
ST5-C-CE	5A DC Stepper Drive, CANOpen, Encoder
ST10-S	10A DC Stepper Drive, RS-232
ST10-Plus	10A DC Stepper Drive, RS-232
ST10-Q-NN	10A DC Stepper Drive, Q Programming, RS-232
ST10-Q-NE	10A DC Stepper Drive, Q Programming, RS-232, Encoder
ST10-Q-RN	10A DC Stepper Drive, Q Programming, RS-485
ST10-Q-RE	10A DC Stepper Drive, Q Programming, RS-485, Encoder
ST10-Q-EN	10A DC Stepper Drive, Q Programming, Ethernet
ST10-Q-EE	10A DC Stepper Drive, Q Programming, Ethernet, Encoder
ST10-IP-EN	10A DC Stepper Drive, Q Programming, EtherNet/IP
ST10-IP-EE	10A DC Stepper Drive, Q Programming, EtherNet/IP, Encoder
ST10-Si-NN	10A DC Stepper Drive, Si Programming, RS-232
ST10-Si-NE	10A DC Stepper Drive, Si Programming, RS-232, Encoder
ST10-C-CN	10A DC Stepper Drive, CANOpen
ST10-C-CE	10A DC Stepper Drive, CANOpen, Encoder
STF03-Q	3A DC Stepper Drive, Q Programming, Dual Port Ethernet
STF05-Q	5A DC Stepper Drive, Q Programming, Dual Port Ethernet
STF06-Q	6A DC Stepper Drive, Q Programming, Dual Port Ethernet
STF10-Q	10A DC Stepper Drive, Q Programming, Dual Port Ethernet
STF03-IP	3A DC Stepper Drive, Q Programming, Dual Port EtherNet/IP
STF05-IP	5A DC Stepper Drive, Q Programming, Dual Port EtherNet/IP
STF06-IP	6A DC Stepper Drive, Q Programming, Dual Port EtherNet/IP
STF10-IP	10A DC Stepper Drive, Q Programming, Dual Port EtherNet/IP
SV Drives	
SV7-S-AE	7A DC Servo Drive, RS-232, Encoder
SV7-S-AF	7A DC Servo Drive, RS-232, Encoder, MCF Encoder Feedback Board

Drive	Description
SV7-S-RE	7A DC Servo Drive, RS-485, Encoder
SV7-Q-AE	7A DC Servo Drive, Q Programming, RS-232, Encoder
SV7-Q-AF	7A DC Servo Drive, Q Programming, RS-232, Encoder, MCF Encoder Feedback Board
SV7-Q-RE	7A DC Servo Drive, Q Programming, RS-485, Encoder
SV7-Q-EE	7A DC Servo Drive, Q Programming, Ethernet, Encoder
SV7-IP-EE	7A DC Servo Drive, Q Programming, EtherNet/IP, Encoder
SV7-Si-AE	7A DC Servo Drive, Si Programming, RS-232, Encoder
SV7-Si-AF	7A DC Servo Drive, Si Programming, RS-232, Encoder, MCF Encoder Feedback Board
SV7-C-CE	7A DC Servo Drive, CANOpen, Encoder
Blu Servo Drives	
BLuDC4-S	4A DC Servo Drive, RS-232
BLuDC4-SE	4A DC Servo Drive, RS-232, Expanded I/O
BLuDC4-Q	4A DC Servo Drive, RS-232, Q Programming
BLuDC4-QE	4A DC Servo Drive, RS-232, Q Programming, Expanded I/O
BLuDC4-Si	4A DC Servo Drive, RS-232, Si Programming
BLuDC9-S	9A DC Servo Drive, RS-232
BLuDC9-SE	9A DC Servo Drive, RS-232, Expanded I/O
BLuDC9-Q	9A DC Servo Drive, RS-232, Q Programming
BLuDC9-QE	9A DC Servo Drive, RS-232, Q Programming, Expanded I/O
BLuDC9-Si	9A DC Servo Drive, RS-232, Si Programming
BLuAC5-S	5A AC Servo Drive, RS-232
BLuAC5-SE	5A AC Servo Drive, RS-232, Expanded I/O
BLuAC5-Q	5A AC Servo Drive, RS-232, Q Programming
BLuAC5-QE	5A AC Servo Drive, RS-232, Q Programming, Expanded I/O
BLuAC5-Si	5A AC Servo Drive, RS-232, Si Programming
STAC5 Stepper Drives	
STAC5-S-N120	5A 120VAC Stepper Drive, Ethernet
STAC5-S-N220	5A 220VAC Stepper Drive, Ethernet
STAC5-S-E120	5A 120VAC Stepper Drive, Ethernet, Encoder
STAC5-S-E220	5A 220VAC Stepper Drive, Ethernet, Encoder
STAC5-Q-N120	5A 120VAC Stepper Drive, Ethernet, Q Programming
STAC5-Q-N220	5A 220VAC Stepper Drive, Ethernet, Q Programming
STAC5-Q-E120	5A 120VAC Stepper Drive, Ethernet, Q Programming, Encoder
STAC5-Q-E220	5A 220VAC Stepper Drive, Ethernet, Q Programming, Encoder
STAC5-IP-N120	5A 120VAC Stepper Drive, EtherNet/IP
STAC5-IP-N220	5A 220VAC Stepper Drive, EtherNet/IP
STAC5-IP-E120	5A 120VAC Stepper Drive, EtherNet/IP, Encoder
STAC5-IP-E220	5A 220VAC Stepper Drive, EtherNet/IP, Encoder

Drive	Description
STAC6 Stepper Drives	
STAC6-S	6A 120VAC Stepper Drive, RS-232
STAC6-S-220	6A 220VAC Stepper Drive, RS-232
STAC6-SE	6A 120VAC Stepper Drive, RS-232, Expanded I/O
STAC6-SE-220	6A 220VAC Stepper Drive, RS-232, Expanded I/O
STAC6-Q	6A 120VAC Stepper Drive, RS-232, Q Programming
STAC6-Q-220	6A 220VAC Stepper Drive, RS-232, Q Programming
STAC6-QE	6A 120VAC Stepper Drive, RS-232, Q Programming, Expanded I/O
STAC6-QE-220	6A 220VAC Stepper Drive, RS-232, Q Programming, Expanded I/O
STAC6-Si	6A 120VAC Stepper Drive, RS-232, Si Programming
STAC6-Si-220	6A 220VAC Stepper Drive, RS-232, Si Programming
STAC6-C	6A 120VAC Stepper Drive, CANOpen
STAC6-C-220	6A 220VAC Stepper Drive, CANOpen
SVAC3 Servo Drives	
SVAC3-S-E-120	3A 120VAC Servo Drive, Ethernet
SVAC3-S-E-220	3A 220VAC Servo Drive, Ethernet
SVAC3-Q-E-120	3A 120VAC Servo Drive, Ethernet, Q Programming
SVAC3-Q-E-220	3A 220VAC Servo Drive, Ethernet, Q Programming
SVAC3-IP-E-120	3A 120VAC Servo Drive, Ethernet, Q Programming, EtherNet/IP
SVAC3-IP-E-220	3A 220VAC Servo Drive, Ethernet, Q Programming, EtherNet/IP

Drive	Description
StepSERVO integrated motors	
TSM11Q-1RM	NEMA 11 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM11Q-2RM	NEMA 11 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM11Q-3RM	NEMA 11 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM17C-1CG	NEMA 17 Integrated CANopen StepSERVO™ Motor w/ Q Programming
TSM17C-2CG	NEMA 17 Integrated CANopen StepSERVO™ Motor w/ Q Programming
TSM17C-3CG	NEMA 17 Integrated CANopen StepSERVO™ Motor w/ Q Programming
TSM17P-1AG	NEMA 17 Integrated StepSERVO™ Motor w/ Pulse & Direction Control
TSM17P-2AG	NEMA 17 Integrated StepSERVO™ Motor w/ Pulse & Direction Control
TSM17P-3AG	NEMA 17 Integrated StepSERVO™ Motor w/ Pulse & Direction Control
TSM17Q-1AG	NEMA 17 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM17Q-1RG	NEMA 17 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM17Q-2AG	NEMA 17 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM17Q-2RG	NEMA 17 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM17Q-3AG	NEMA 17 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM17Q-3RG	NEMA 17 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM23C-2CG	NEMA 23 Integrated CANopen StepSERVO™ Motor w/ Q Programming
TSM23C-3CG	NEMA 23 Integrated CANopen StepSERVO™ Motor w/ Q Programming
TSM23C-4CG	NEMA 23 Integrated CANopen StepSERVO™ Motor w/ Q Programming
TSM23P-2AG	NEMA 23 Integrated StepSERVO™ Motor w/ Pulse & Direction Control
TSM23P-3AG	NEMA 23 Integrated StepSERVO™ Motor w/ Pulse & Direction Control
TSM23P-4AG	NEMA 23 Integrated StepSERVO™ Motor w/ Pulse & Direction Control
TSM23Q-2AG	NEMA 23 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM23Q-2RG	NEMA 23 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM23Q-3AG	NEMA 23 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM23Q-3RG	NEMA 23 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM23Q-4AG	NEMA 23 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM23Q-4RG	NEMA 23 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TSM23S-2AG	NEMA 23 Integrated StepSERVO™ Motor w/ Streaming Command support
TSM23S-2RG	NEMA 23 Integrated StepSERVO™ Motor w/ Streaming Command support
TSM23S-3AG	NEMA 23 Integrated StepSERVO™ Motor w/ Streaming Command support
TSM23S-3RG	NEMA 23 Integrated StepSERVO™ Motor w/ Streaming Command support
TSM23S-4AG	NEMA 23 Integrated StepSERVO™ Motor w/ Streaming Command support
TSM23S-4RG	NEMA 23 Integrated StepSERVO™ Motor w/ Streaming Command support

Drive	Description
StepSERVO integrated motors	
TXM24C-1CG	IP65 Rated NEMA 24 Integrated CANopen StepSERVO™ Motor
TXM24C-3CG	IP65 Rated NEMA 24 Integrated CANopen StepSERVO™ Motor
TXM24IP-1EG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ EtherNet/IP
TXM24IP-3EG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ EtherNet/IP
TXM24Q-1AG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TXM24Q-3AG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TXM24Q-1EG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ Q Programming
TXM24Q-3EG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ Q Programming
TXM24Q-1RG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TXM24Q-3RG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor w/ Q Programming & Modbus/RTU
TXM24S-1AG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor
TXM24S-3AG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor
TXM24S-1EG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor
TXM24S-3EG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor
TXM24S-1RG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor
TXM24S-3RG	IP65 Rated NEMA 24 Integrated StepSERVO™ Motor

Drive	Description
StepSERVO integrated motors	
SSM23IP-2EG	NEMA 23 Integrated StepSERVO™ Motor w/ EtherNet/IP
SSM23IP-3EG	NEMA 23 Integrated StepSERVO™ Motor w/ EtherNet/IP
SSM23IP-4EG	NEMA 23 Integrated StepSERVO™ Motor w/ EtherNet/IP

Drive	Description
SV200 Servo Drives	
SV2A3-C-CE	78-139 VAC Digital Servo Drive w/ CANopen interface
SV2A3-P-NE	78-139 VAC Digital Servo Drive
SV2A3-Q-AE	78-139 VAC Digital Servo Drive w/ On-board motion control
SV2A3-Q-EE	78-139 VAC Digital Servo Drive w/ On-board motion control
SV2A3-Q-RE	78-139 VAC Digital Servo Drive w/ On-board motion control
SV2A5-C-CE	78-139 VAC Digital Servo Drive w/ CANopen interface
SV2A5-P-NE	78-139 VAC Digital Servo Drive
SV2A5-Q-AE	78-139 VAC Digital Servo Drive w/ On-board motion control
SV2A5-Q-EE	78-139 VAC Digital Servo Drive w/ On-board motion control
SV2A5-Q-RE	78-139 VAC Digital Servo Drive w/ On-board motion control
SV2B3-C-CE	156-264 VAC Digital Servo Drive w/ CANopen interface
SV2B3-P-NE	156-264 VAC Digital Servo Drive
SV2B3-Q-AE	156-264 VAC Digital Servo Drive w/ On-board motion control
SV2B3-Q-EE	156-264 VAC Digital Servo Drive w/ On-board motion control
SV2B3-Q-RE	156-264 VAC Digital Servo Drive w/ On-board motion control
SV2B5-C-CE	156-264 VAC Digital Servo Drive w/ CANopen interface
SV2B5-P-NE	156-264 VAC Digital Servo Drive
SV2B5-Q-AE	156-264 VAC Digital Servo Drive w/ On-board motion control
SV2B5-Q-EE	156-264 VAC Digital Servo Drive w/ On-board motion control
SV2B5-Q-RE	156-264 VAC Digital Servo Drive w/ On-board motion control
SV2D6-Q-DE	DC Input Digital Servo Drive, Q Programming, Dual Port Ethernet
SV2D10-Q-DE	DC Input Digital Servo Drive, Q Programming, Dual Port Ethernet
SV2D6-Q-DE	DC Input Digital Servo Drive, Q Programming, Dual Port EtherNet/IP
SV2D10-Q-DE	DC Input Digital Servo Drive, Q Programming, Dual Port EtherNet/IP

Appendix K: Modbus appendix

Covers the following Modbus® RTU enabled drives:

ST5-PLUS
ST5-Q-NE
ST5-Q-NF
ST5-Q-NN
ST5-Q-RE
ST5-Q-RN
ST5-S

ST10-PLUS
ST10-Q-NE
ST10-Q-NF
ST10-Q-NN
ST10-Q-RE
ST10-Q-RN
ST10-S

STF03-R
STF06-R
STF10-R

STM17Q-1AE
STM17Q-1AN
STM17Q-1RE
STM17Q-1RN
STM17Q-2AE
STM17Q-2AN
STM17Q-2RE
STM17Q-2RN
STM17Q-3AE
STM17Q-3AN
STM17Q-3RE
STM17Q-3RN

STM23Q-2AE
STM23Q-2AN
STM23Q-2RE
STM23Q-2RN
STM23Q-3AE
STM23Q-3AN
STM23Q-3RE
STM23Q-3RN

STM24QF-3AE
STM24QF-3AN
STM24QF-3RE
STM24QF-3RN

SV2A3-Q-RE
SV2B3-Q-RE
SV2A5-Q-RE
SV2B5-Q-RE
SV2D10-Q-RE

SWM24QF-3AE
SWM24QF-3AN

TSM11Q-1RM
TSM11Q-2RM
TSM11Q-3RM
TSM11Q-1RM-H01
TSM11Q-2RM-H01
TSM11Q-3RM-H01

TSM17Q-1AG
TSM17Q-1RG
TSM17Q-2AG
TSM17Q-2RG
TSM17Q-3AG
TSM17Q-3RG
TSM23Q-2AG
TSM23Q-2RG
TSM23Q-3AG
TSM23Q-3RG
TSM23Q-4RG

TXM24Q-1AG
TXM24Q-1RG
TXM24Q-3AG
TXM24Q-3RG

TSM34Q-1AG
TSM34Q-1RG
TSM34Q-3AG
TSM34Q-3RG
TSM34Q-5AG
TSM34Q-5RG
TSM34Q-6AG
TSM34Q-6RG

TXM34Q-1AG
TXM34Q-1RG
TXM34Q-3AG
TXM34Q-3RG
TXM34Q-5AG
TXM34Q-5RG
TXM34Q-6AG
TXM34Q-6RG

Also covers the following Modbus® TCP enabled drives:

ST5-IP-EE
ST5-IP-EN
ST5-Q-EE
ST5-Q-EN
ST10-IP-EE
ST10-IP-EN
ST10-Q-EE
ST10-Q-EN

STF03-D
STF03-IP
STF06-D
STF06-IP
STF10-D
STF10-IP

STM23Q-2EE
STM23Q-2EN
STM23Q-3EE
STM23Q-3EN
STM23IP-2EE
STM23IP-2EN
STM23IP-3EE
STM23IP-3EN

STM24Q-3EE
STM24Q-3EN
STM24IP-3EE
STM24IP-3EN

SWM24Q-3EE
SWM24IP-3EE

SSM23Q-2EG
SSM23Q-3EG
SSM23Q-4EG
SSM23IP-2EG
SSM23IP-3EG
SSM23IP-4EG

TXM24Q-1EG
TXM24Q-3EG
TXM24IP-1EG
TXM24IP-3EG

SV2A3-Q-DE
SV2A3-IP-DE
SV2A5-Q-DE
SV2A5-IP-DE
SV2B3-Q-DE
SV2B3-IP-DE
SV2B5-Q-DE
SV2B5-IP-DE

SV2D10-Q-DE
SV2D10-IP-DE

TSM34Q-1DG
TSM34Q-3DG
TSM34Q-5DG
TSM34Q-6DG

TSM34IP-1DG
TSM34IP-3DG
TSM34IP-5DG
TSM34IP-6DG

TXM34Q-1DG
TXM34Q-3DG
TXM34Q-5DG
TXM34Q-6DG

TXM34IP-1DG
TXM34IP-3DG
TXM34IP-5DG
TXM34IP-6DG

What is Modbus?

Modbus, originally created by Modicon, is a fieldbus that allows a master and one or more slave devices to share data. These data are organized into 16-bit registers, which can also be used to share information single-bit I/O points.

It is a popular protocol with PLC vendors due to its simplicity and the inherent ease of sending PLC register data (often 16-bits in width) over a fieldbus protocol optimized for 16-bit data.

The master may initiate read and write operations on single registers or blocks of registers. While there is no rule to this effect, it is common for the master to read and write on a periodic time base (polling), rather than sending and requesting data only when it is needed. In this manner, PLC register data is ensured to be valid and consistent as a representation of the slave device's status.

Getting help: we offer Modbus application notes and sample code for popular PLCs and HMIs in the support section of our website. Please visit www.applied-motion.com/support/application-notes

Wiring

Modbus/RTU:

Modbus RTU uses the standard RS-232 or RS-485 physical layer.

RS-232 is a point-to-point communications scheme, and as such the largest possible network would consist of a single slave drive. Please note that even though it will be the only device on the "network", it will still require an address. This address may be an integer value from 1-32, and is set through the *ST Configurator™* software during initial configuration.

For drives with RS-485 communications, there are a few things to consider.

It is possible to use 2-wire RS-485 for operational communication over Modbus, however 4-wire RS-485 is required for use with all Applied Motion configuration and programming software (i.e., *ST Configurator™* and *Q Programmer™*). As such, we recommend that all RS-485 networks be constructed using the 4-wire method.

Be sure to consult your drive's hardware manual for specific wiring details.

Modbus/TCP

Modbus TCP uses the Ethernet physical layer. Connect your drive(s) to the host using standard Ethernet cabling and if necessary an Ethernet switch or router. Port 502 is used for communication per the Modbus standard.

Drive configuration

First, download and install the most recent version of the appropriate software from the Applied Motion Products website (www.applied-motion.com). For the ST, STAC, and STM series, use *ST Configurator™* software. For the TSM series, use *Step-Servo Quick Tuner™*. For SV2 models, use *SVX Servo Suite™*. For RS-485 drives you will also need a USB-to-RS-485 converter, such as the USB-COMi-M module available from AMP. Consult your drive's manual for wiring and general communication instructions.

1. Launch software and select the appropriate COM port. For Ethernet drives, enter the IP address of the drive you wish to configure.

2. Apply power to the drive. Note, perform this step with only a single drive on the network. Each drive must be configured individually before it may be used on a network.

3. The Software will automatically identify the drive, but will not upload all parameters. Click "Upload from Drive". This will ensure that the software accurately reflects the current state of the drive's configuration parameters.

4. Click "Motion & I/O" in *ST Configurator™.* In *Step-Servo Quick Tuner™ and SVX Servo Suite™ use the Configuration panel.*

5. for Modbus RTU drives, specify the drive's node address and baud rate.

6. For drives with Flex/IO, confirm the desired I/O settings. Click OK to return to the main screen.

7. Set any desired motor and encoder configurations your application may require.

8. Click Download to Drive to save the settings.

9. Optionally, use *Q Programmer™* to write and download a program for this axis. The drive may be configured to launch the program automatically upon powerup, or to wait for the QX command to be sent from the Modbus master.

10. Remove power from the drive and install into the main machine network.

11. Repeat for all other slave nodes, taking care to ensure each has a unique node address.

Note, the drive must be rebooted to switch between Modbus and configuration modes.

Drive Behavior

An extensive list of registers has been made available, allowing the user to monitor or change every detail of the drive's status It is also possible to send commands to a specific register, mimicking the behavior of our proprietary SCL command set. The capability allows a PLC to have unparalleled control over the drive's behavior at runtime.

The drive will respond to the following Modbus function codes:

3	Read Holding Registers
4	Read Input Registers
6	Write Single Register
16	Write Multiple Registers

Monitoring

See the Register Map table for details on specific data that can be monitored and written in this manner.

Sending Commands

The Command Opcode register, 40125, is designated to receive encoded SCL commands via Modbus. Many SCL commands have been made available in this manner, and will allow the user full control over the motion capabilities of the drive.

SCL command encoding details can be found in the Modbus Register Table.

Examples

Point-to-Point Moves

Four pieces of data are required to fully define a point-to-point move. Acceleration, Deceleration, Distance, and Velocity. Followed of course by the move command itself. To command such a move over Modbus, we must first write to the aforementioned control registers, then send the command.

Function	Register	Value	Notes
Acceleration	40028	100	Units: 10 RPM/sec
Deceleration	40029	100	Units: 10 RPM/sec
Velocity	40030	240	Units: 0.25 RPM
Distance	40031..32	20000	This register is affected by Endianness setting

Now it's time for the command itself. The most common point-to-point move is the Feed to Length which uses the SCL command FL. In Modbus we do the following:

The drive will execute the move command immediately, pulling the relevant parameter data from the above registers. It is immediately ready to accept another command. Note, it is not necessary to send parameter data with each command, unless this data changes between moves. For example, to repeat a move simply send the command again, leaving the relevant parameter data unchanged.

Function	Register	Value	Notes
FL Command	40125	102	0x66 (Hexadecimal) is equal to 102

Launch a Q Program

It is possible to execute a stored Q program over Modbus using the QX command (opcode 0x78). This command requires a value to be written to Parameter 1, the first of 5 registers set aside for command-dependant parameter data.

To launch the program at segment 1, we would use the SCL command QX1. Over Modbus, the following procedure applies.

First, configure the parameter data (only Parameter 1 is used by this command).

Now we can send the QX command itself.

Function	Register	Value	Notes
Parameter 1	40126	1	Q segment to execute. Integer: 1..12

Function	Register	Value	Notes
QX Command	40125	120	0x78 (Hexadecimal) is equal to 120 decimal

Note, to stop a command or interrupt a Q program, either the SK or SKD commands may be used.

Function	Register	Value	Notes
SK Command	40125	225	0xE1 (Hexadecimal) is equal to 225

Function	Register	Value	Notes
SKD Command	40125	226	0xE2 (Hexadecimal) is equal to 226

SCL Command Mode Table

SCL Command Encoding Table							
Function	**SCL**	**Opcode**	**Parameter 1**	**Parameter 2**	**Parameter 3**	**Parameter 4**	**Parameter 5**
Alarm Reset	AX	0xBA	N/A	N/A	N/A	N/A	N/A
Command Mode	CM	0x10	CM (11, 12, etc)	N/A	N/A	N/A	N/A
Start Jogging	CJ	0x96	N/A	N/A	N/A	N/A	N/A
Stop Jogging	SJ	0xD8	N/A	N/A	N/A	N/A	N/A
Encoder Function	EF	0xD6	0,1,2 or 6	N/A	N/A	N/A	N/A
Encoder Position	EP	0x98	Position	N/A	N/A	N/A	N/A
Feed to Double Sensor	FD	0x69	I/O Point 1	Condition 1	I/O Point 2	Condition 2	N/A
Follow Encoder	FE	0xCC	I/O Point	Condition	N/A	N/A	N/A
Feed to Length	FL	0x66	N/A	N/A	N/A	N/A	N/A
Feed to Sensor with Mask Distance	FM	0x6A	I/O Point	Condition	N/A	N/A	N/A
Feed and Set Output	FO	0x68	I/O Point	Condition	N/A	N/A	N/A
Feed to Position	FP	0x67	N/A	N/A	N/A	N/A	N/A
Feed to Sensor	FS	0x6B	I/O Point	Condition	N/A	N/A	N/A
Feed to Sensor with Safety Distance	FY	0x6C	I/O Point	Condition	N/A	N/A	N/A
Jog Disable	JD	0xA3	N/A	N/A	N/A	N/A	N/A
Jog Enable	JE	0xA2	N/A	N/A	N/A	N/A	N/A
Motor Disable	MD	0x9E	N/A	N/A	N/A	N/A	N/A
Motor Enable	ME	0x9F	N/A	N/A	N/A	N/A	N/A
Seek Home	SH	0x6E	I/O Point	Condition	N/A	N/A	N/A
Set Position	SP	0xA5	Position	N/A	N/A	N/A	N/A
Filter Input	FI	0xC0	I/O Point	Filter Time	N/A	N/A	N/A
Filter Select Inputs	FX	0xD3	N/A	N/A	N/A	N/A	N/A
Step Filter Freq	SF	0x06	Freq	N/A	N/A	N/A	N/A
Analog Deadband	AD	0xD2	0.001 V	N/A	N/A	N/A	N/A
Alarm Reset Input	AI	0x46	Function ('1'..'3')	I/O Point	N/A	N/A	N/A
Alarm Output	AO	0x47	Function ('1'..'3')	I/O Point	N/A	N/A	N/A
Analog Scaling	AS	0xD1	N/A	N/A	N/A	N/A	N/A
Define Limits	DL	0x42	1..3	N/A	N/A	N/A	N/A
Set Output	SO	0x8B	I/O Point	Condition	N/A	N/A	N/A
Wait for Input	WI	0x70	N/A	N/A	N/A	N/A	N/A
Queue Load & Execute	QX	0x78	1..12	N/A	N/A	N/A	N/A
Wait Time	WT	0x6F	0.01 sec	N/A	N/A	N/A	N/A
Stop Move, Kill Buffer	SK	0xE1	N/A	N/A	N/A	N/A	N/A
Stop Move, Kill Buffer, Normal Decel	SKD	0xE2	N/A	N/A	N/A	N/A	N/A

N/A = Not Applicable

- 32-bit values may be either big-endian or little-endian. Endianness is determined by Bit 6 of the PR command word. It is recommended that this setting be configured with the configuration software.
- Consult the I/O Encoding Table for details on parameter data for commands referencing I/O.

IO Encoding Table

Useful ASCII values for IO commands. Character values shown are taken from standard SCL command nomenclature. For example, a Feed to Sensor command example may utilize the falling edge of input 1 and appear as FS1F. To encode this command the user would select 0x31 for the character '1' and 0x46 for the character 'F'. These codes must be written to the appropriate parameter register before writing the desired command opcode to register 40125.

Character	hex code	Signifies		
		ST5/10, STF, MDX, TSM/TXM34	**STAC5**	**SSM23, STM23/24, SWM24 & TXM24**
X0	0xB0	encoder index signal	encoder index signal	encoder index signal
X1 or Y1	0xB1	input X1 or output Y1	input X1 or output Y1	IN1/STEP or OUT
X2 or Y2	0xB2	input X2 or output Y2	input X2 or output Y2	IN2/DIR
X3 or Y3	0xB3	input X3 or output Y3	input X3	IN3/EN
X4 or Y4	0xB4	input X4 or output Y4	input X4	n/a
X5	0xB5	input X5	n/a	n/a
X6	0xB6	input X6	n/a	n/a
X7	0xB7	input X7	n/a	n/a
X8	0xB8	input X8	n/a	n/a
1	0x31	n/a	input 1 or output 1	IN1/STEP - FI only
2	0x32	n/a	input 2 or output 2	IN2/DIR - FI only
3	0x33	n/a	input 3 or output 3	IN3/EN - FI only
4	0x34	n/a	input 4 or output 4	n/a
5	0x35	X5 only	input 5	n/a
6	0x36	X6 only	input 6	n/a
7	0x37	X7 only	input 7	n/a
8	0x38	X8 only	input 8	n/a
L	0x4C	low state (closed)	low state (closed)	low state (closed)
H	0x48	high state (open)	high state (open)	high state (open)
R	0x52	rising edge	rising edge	rising edge
F	0x46	falling edge	falling edge	falling edge

Register Encoding Table

Register Name	Use	equivalent SCL command	Code	Size	Read Only
0	Accumulator		0x30	long	
1	user defined		0x31	long	
2	user defined		0x32	long	
3	user defined		0x33	long	
4	user defined		0x34	long	
5	user defined		0x35	long	
6	user defined		0x36	long	
7	user defined		0x37	long	
8	user defined		0x38	long	
9	user defined		0x39	long	
:	user defined		0x3A	long	
;	user defined		0x3B	long	
<	user defined		0x3C	long	
=	user defined		0x3D	long	
>	user defined		0x3E	long	
?	user defined		0x3F	long	
@	user defined		0x40	long	
[	user defined		0x5B	long	
\	user defined		0x5C	long	
]	user defined		0x5D	long	
^	user defined		0x5E	long	
_	user defined		0x5F	long	
`	user defined		0x60	long	
a	analog command	IA	0x61	short	yes
b	Q line number		0x62	short	yes
c	current command	IC	0x63	short	yes
d	relative distance	ID	0x64	long	yes
e	encoder position	IE, EP	0x65	long	yes
f	alarm code	AL	0x66	long	yes
g	sensor position		0x67	short	yes
h	condition code		0x68	short	yes
i	X inputs (IN/OUT 1)	ISX	0x69	short	yes
j	analog IN1	IA1	0x6A	short	yes
k	analog IN2	IA2	0x6B	short	yes
l	absolute position		0x6C	long	yes
m	control mode	CM	0x6D	short	yes
n	velocity mode state		0x6E	short	yes
o	point to point state		0x6F	short	yes

Register Name	Use	equivalent SCL command	Code	Size	Read Only
p	Q segment		0x70	short	yes
q	actual current	IQ	0x71	short	yes
r	average regen power		0x72	short	yes
s	status code	SC	0x73	short	yes
t	drive temperature	IT	0x74	short	yes
u	bus voltage	IU	0x75	short	yes
v	actual velocity	IV0	0x76	short	yes
w	target velocity	IV1	0x77	short	yes
x	position error	IX	0x78	long	yes
y	IN/OUT 2 inputs	IS	0x79	short	yes
z	phase error		0x7A	short	yes
A	accel rate	AC	0x41	short	
B	decel rate	DE	0x42	short	
C	change distance	DC	0x43	long	
D	distance	DI	0x44	long	
E	position offset		0x45	long	
F	other (misc) flags		0x46	long	
G	current command	GC	0x47	short	
H	analog velocity gain		0x48	short	
I	input counter		0x49	long	
J	jog speed		0x4A	short	
K	jog accel rate		0x4B	short	
L	jog decel rate		0x4C	short	
M	max velocity	JS	0x4D	short	
N	continuous current	CC	0x4E	short	
O	idle current	CI	0x4F	short	
P	absolute position command		0x50	long	
Q	reserved		0x51		
R	steps/rev	EG	0x52	short	
S	pulse count		0x53	long	
T	total count		0x54	long	
U	change speed	VC	0x55	short	
V	velocity	VE	0x56	short	
W	time stamp		0x57	short	
X	analog position gain	AP	0x58	short	
Y	analog threshold	AT	0x59	short	
Z	analog offset	AV	0x5A	short	

To execute a feed to sensor routine active on a low (closed) condition for input 3:

FUNCTION	REGISTER	VALUE	NOTES
define input #	40126	51	0x33 (Hexadecimal) equals 51 decimal
define input condition	40127	76	0x4C equals 76 decimal
Feed to Sensor	40125	107	0x6B equals 107 decimal

Modbus Register Table for Step Drives				
Applies to Q and IP drives in these product families: ST, STM, SWM, STF				
Register	**Access**	**Data Type**	**Description**	**SCL Register**
40001	Read Only	SHORT	Alarm Code (AL)	f
40002	Read Only	SHORT	Status Code (SC)	s
40003	Read Only	SHORT	reserved	y
40004	Read Only	SHORT	Input Status	i
40005..6	Read Only	LONG	Encoder Position (IE, EP) [requires encoder feedback]	e
40007..8	Read Only	LONG	Immediate Absolute Position	I
40009..10	Write	LONG	Absolute Position Command	P
40011	Read Only	SHORT	Immediate Actual Velocity (IV0) [requires encoder feedback]	v
40012	Read Only	SHORT	Immediate Target Velocity (IV1)	w
40013	Read Only	SHORT	Immediate Drive Temperature (IT)	t
40014	Read Only	SHORT	Immediate Bus Voltage (IU)	u
40015..16	Read Only	LONG	Immediate Position Error (IX) [requires encoder feedback]	x
40017	Read Only	SHORT	Immediate Analog Input Value (IA) [reserved on STF]	a
40018	Read Only	SHORT	Q Program Line Number	b
40019	Read Only	SHORT	Immediate Current Command (IC)	c
40020..21	Read Only	LONG	Relative Distance (ID)	d
40022..23	Read Only	LONG	Sensor Position	g
40024	Read Only	SHORT	Condition Code	h
40025	Read Only	SHORT	Analog Input 1 (IA1) [reserved on STF]	j
40026	Read Only	SHORT	Analog Input 2 (IA2) [reserved on STF]	k
40027	Read Only	SHORT	Command Mode (CM)	m
40028	R/W	SHORT	Point-to-Point Acceleration (AC)	A
40029	R/W	SHORT	Point-to-Point Deceleration (DE)	B
40030	R/W	SHORT	Velocity (VE)	V
40031..32	R/W	LONG	Point-to-Point Distance (DI)	D
40033..34	R/W	LONG	Change Distance (DC)	C
40035	R/W	SHORT	Change Velocity (VC)	U
40036	Read Only	SHORT	Velocity Move State	n
40037	Read Only	SHORT	Point-to-Point Move State	o
40038	Read Only	SHORT	Q Program Segment Number	p
40039	Read Only	SHORT	reserved	r
40040	Read Only	SHORT	Phase Error	z
40041..42	R/W	LONG	Position Offset	E

Modbus Register Table for Step Drives				
Applies to Q and IP drives in these product families: ST, STM, SWM, STF				
Register	**Access**	**Data Type**	**Description**	**SCL Register**
40043	R/W	SHORT	Miscellaneous Flags	F
40044	Read Only	SHORT	reserved	G
40045..46	R/W	LONG	Input Counter	I
40047	R/W	SHORT	Jog Accel (JA)	
40048	R/W	SHORT	Jog Decel (JL)	
40049	R/W	SHORT	Jog Velocity (JS)	J
40050	R/W	SHORT	Accel/Decel Current (CA) [STM, SWM only]	
40051	R/W	SHORT	Running Current (CC)	N
40052	R/W	SHORT	Idle Current (CI)	O
40053	R/W	SHORT	Steps per Revolution (EG) / 2	R
40054..40055	R/W	LONG	Pulse Counter	S
40056	R/W	SHORT	Analog Position Gain (AP) [reserved on STF]	X
40057	R/W	SHORT	Analog Threshold (AT) [reserved on STF]	Y
40058	R/W	SHORT	Analog Offset (AV) [reserved on STF]	Z
40059..60	R/W	LONG	Accumulator	0
40061..62	R/W	LONG	User Defined Register	1
40063..64	R/W	LONG	User Defined Register	2
40065..66	R/W	LONG	User Defined Register	3
40067..68	R/W	LONG	User Defined Register	4
40069..70	R/W	LONG	User Defined Register	5
40071..72	R/W	LONG	User Defined Register	6
40073..74	R/W	LONG	User Defined Register	7
40075..76	R/W	LONG	User Defined Register	8
40077..78	R/W	LONG	User Defined Register	9
40079..80	R/W	LONG	User Defined Register	:
40081..82	R/W	LONG	User Defined Register	;
40083..84	R/W	LONG	User Defined Register	<
40085..86	R/W	LONG	User Defined Register	=
40087..88	R/W	LONG	User Defined Register	>
40089..90	R/W	LONG	User Defined Register	?
40091..92	R/W	LONG	User Defined Register	@
40093..94	R/W	LONG	User Defined Register	[
40095..96	R/W	LONG	User Defined Register	\
40097..98	R/W	LONG	User Defined Register	]

Modbus Register Table for Step Drives				
Applies to Q and IP drives in these product families: ST, STM, SWM, STF				
Register	**Access**	**Data Type**	**Description**	**SCL Register**
40099..100	R/W	LONG	User Defined Register	^
40101..102	R/W	LONG	User Defined Register	_
40103..104	R/W	LONG	User Defined Register	`
40105	R/W	SHORT	Brake Release Delay	
40106	R/W	SHORT	Brake Engage Delay	
40107	R/W	SHORT	Idle Current Delay (CD)	
40108	Read Only	SHORT	(reserved)	
40109	Read Only	SHORT	(reserved)	
40110	R/W	SHORT	Analog Filter Gain [reserved on STF]	
40111	Read Only	SHORT	(reserved)	
40112	Read Only	SHORT	(reserved)	
40113	R/W	SHORT	(reserved)	
40114	R/W	SHORT	(reserved)	
40115	R/W	SHORT	(reserved)	
40116	R/W	SHORT	(reserved)	
40117	R/W	SHORT	(reserved)	
40118	R/W	SHORT	(reserved)	
40119	R/W	SHORT	(reserved)	
40120	R/W	SHORT	(reserved)	
40121	R/W	SHORT	(reserved)	
40122	R/W	SHORT	(reserved)	
40123	R/W	SHORT	(reserved)	
40124	R/W	SHORT	(reserved)	
40125	R/W	SHORT	Command Opcode	
40126	R/W	SHORT	Parameter 1	
40127	R/W	SHORT	Parameter 2	
40128	R/W	SHORT	Parameter 3	
40129	R/W	SHORT	Parameter 4	
40130	R/W	SHORT	Parameter 5	
40131	R/W	SHORT	Harmonic Smoothing Phase (HP) [STF only]	
40132	R/W	SHORT	Harmonic Smoothing Gain (HG) [STF only]	
40133	R/W	SHORT	Step Smoothing filter (SF) [STF only]	
40134	R/W	SHORT	(reserved)	
40135	R/W	SHORT	Motor Details [STF only]	
40136	R/W	SHORT	Step Input Mode (PM) [STF only]	