EtherNET/IP用户手册

中文版

《鸣志迷》翻译整理

https://www.bjmoons.com

视频观看

抖音号：1130459512

附录 H: EtherNet/IP

EtherNet/IP产品，由型号中的字母“IP”指定，在EtherNet/IP网络上提供对Q和SCL功能的访问。本附录详细介绍了哪些命令是可用的，以及如何将它们封装到EtherNet/IP和CIP包中。假设用户具有EtherNet/IP的工作知识，因为它涉及正在使用的控制器，因为本章将不解释一般的EtherNet/IP实现细节。

对于AB PLC用户, AMP开发了一组附加说明（aoi）和相关的应用说明。这组AOI可以使用隐式和显式消息传递的组合来控制来自Rockwell CompactLogix或ControlLogix PLC的应用运动产品步进、步进伺服和伺服驱动器。zip文件中包括用户指南、示例RSLogix（适用于v20及以上版本）程序和未锁定的AOI文件。AOI可以在这里找到。(https://www.applied-motion.com/sites/default/files/APPN0046A-EtherNet-IP-Add-On-Instructions-for-RSLogix.zip)

AMP提供class1和class3连接，每种类型都适用于特定任务。class1连接对于高带宽任务（如监视驱动器的特定功能）很有用，而class3连接用于发送目标消息以直接控制驱动器。后者用于实现显式消息传递。

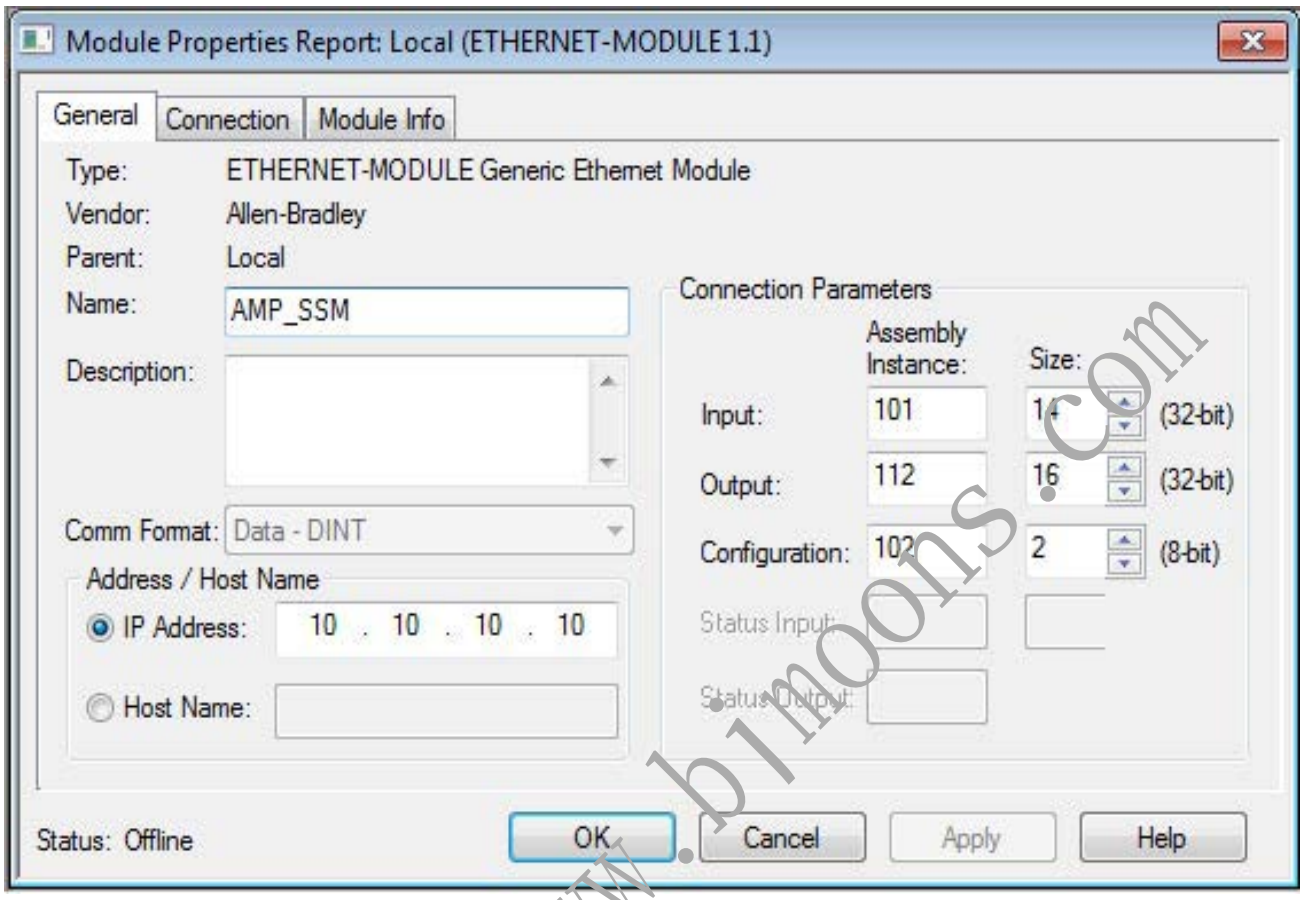
请注意，对于EtherNet/IP，所有数据方向标记均以网络视图的点为准。这样，驱动器发送到控制器的数据被称为输入，而控制器向下发送到驱动器的数据被称为输出。

Class 1 连接

Class 1连接使用连接点，可以将其视为具有预定义函数的地址。若要使用Class 1连接与应用的运动驱动器通信，可使用以下连接点：

目标ID		Size		功能	注释
16进制	十进制	32 bit DINTS	Bytes		
0x64	100	n/a	30	输入组件（经典版本）	用于监视驱动器状态的静态装配对象
0x65	101	14	56	输入组件（增强版）	静态装配对象（版本251.0及更高版本）
0x66	102	n/a	2	配置程序集	指定参数，如分组间隔、数据长度。
0x67	103	0	0	心跳输入只有程序集	告诉控制器驱动器仍然处于活动状态的零长度消息。驱动器将多播指定的输入程序集（100或101）
0x68	104	0	0	心跳只监听程序集	告诉驱动器控制器仍然处于活动状态的零长度消息。
0x70	112	16	64	输出组件	用于控制驱动器的静态装配对象（版本251或更高版本）

典型的PLC连接设置



输入程序集 (0x64)—经典版本

此连接点用于监视驱动器的行为。驱动器发送的30字节数据如下：

字段描述符		长度 (bytes)
IP Address	IP地址（以Internet格式编码）	4
Status Word	状态字	2
Alarm Word	报警字	2
Supply Voltage	电源电压	2
Actual Current	实际电流	2
Drive Temperature	驱动温度	2
Encoder Position	编码器位置（32位有符号）	4
Absolute Position	绝对位置（32位有符号）	4
Actual Velocity	实际速度	2
Input Status	输入状态（主板）	2
Input Status	输入状态（扩展）	2
Output Status	输出状态	2

驱动器传输的数据以小尾数格式发送，因此在使用前可能需要重新排列。

所谓以“互联网格式”存储的IP地址只需编码成十六进制表示法，然后重新排列成小尾数格式。每个八位字节的值为0-255，可以用单个字节表示。

标准IP地址： *192.168.0.40*

转换为十六进制：
192 = 0xC0
168 = 0xA8
0 = 0x00
40 = 0x28

重新排列为小尾数格式： *C0 A8 00 28 -> 28 00 A8 C0*

转换的IP地址： *192.168.0.40 -> 0x2800A8C0*

请注意，所有数字都是以小尾数格式发送的，因此对每一个数据段的转换过程都是相同的。

因此，一个示例信息可以组织如下：

Raw: E0032800A8C019000000E90100003802BAFCFFFFC72A0600C3FFFF40000F0F00

Grouped: [E003] [2800A8C0] [1900] [0000] [E901] [0000] [3802] [BAFCFFFF] [C72A0600] [C3FF] [FF40] [000F] [0F00]

数据解码如下。在可能的情况下，这些值已转换为人类可读的单位。有关详细信息，请参阅相应的命令页。注意，编码器位置、绝对位置和速度是有符号整数，负值将以2的补码形式表示。

序号：	*0xE003*		
IP地址：	*0x2800A8C0*	*= 0xC0A80028*	*= 192.168.0.40*
状态（请参阅SC命令）：	*0x1900*	*= 0x0019*	*= 0000 0000 0001 1001*
警报（见AL命令）：	*0x0000*		
电压：	*0xE901*	*= 0x01E9*	*= 489 (48.9 V)*
电流（见IC命令）：	*0x0000*		
温度（请参阅IT0命令）：	*0x3802*	*= 0x238*	*= 568 (56.8 degrees C)*
编码器位置（参见EP命令）：	*0xBAFCFFFF*	*= 0xFFFFFCBA*	*= -838*
绝对位置（参见SP命令）：	*0xC72A0600*	*= 0x00062AC7*	*= 404167*
速度（见IV命令）：	*0XC3FF*	*= 0xFFC3*	*= -61*
扩展输入（见IS命令）：	*0xFF40*	*= 0x40FF*	
主板输入：（见ISX命令）：	*0x000F*	*= 0x0F00*	
输出（见IO命令）：	*0x0F00*	*= 0x000F*	

输入程序集 (0x65) —增强版本（常用）驱动器反馈→PLC

固件版本251.0或更高版本可用。此连接点用于监视驱动器的行为。驱动器发送的14种数据如下：

元素	字段描述符	长度	单位	注释
0	Status Code 状态码	32 Bits	见SC 命令	见此手册尾部
1	Alarm Code 报警码	32 Bits	见AL命令	见此手册尾部
2	Supply Voltage 电源电压	32 Bits	0.1 VDC	ex: 242 = 24.2 V
3	Actual Current 实际电流（有符号）	32 Bits	mA	ex: 1501 = 1.501A
4	Drive Temperature 驱动器温度	32 Bits	0.1 deg C	ex: 305 = 30.5 deg C
5	Encoder Position 编码器位置（32位有符号）	32 Bits	counts	20,000 counts = 1 圈
6	Absolute Position 绝对位置（32位有符号）	32 Bits	counts	20,000 counts = 1 圈
7	Position Error 位置错误（有符号）	32 Bits	counts	
8	Actual Velocity 实际速度（有符号）	32 Bits	rev/sec * 240	ex: 480 = 2 rev/sec. 如果产品没有编码器，则此字段为零
9	Input Status (extended)输入状态(扩展)	32 Bits	see below	also includes output status
10	Input Status (main board)输入状态(主板)	32 Bits	see below	also includes output status
11	Reserved 保留或输出点状态	32 Bits		输出点状态，参考EDS文件
12	Analog In 1 模拟输入1	32 Bits	ADC counts	0 = min V, 16383 = max V
13	Analog In 2 模拟输入2	32 Bits	ADC counts	0 = min V, 16383 = max V

输入状态详细信息（I/O状态）

当输入关闭时，读数为0。打开时，它们根据下面的位代码读取。当输出打开时，它们的读数为0。当输出关闭时，它们按照下面的代码读取。如果多个输入打开（输出关闭），则代码将相加。

	I/O表-主板输入状态									主板输出			
	位码									位码			
产品	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	enc index	OUT1	OUT2	OUT3	OUT4
SSM23IP	1	2	4	n/a	n/a	n/a	n/a	n/a	n/a	256	n/a	n/a	n/a
STM23/24IP	2	4	6	n/a	n/a	n/a	n/a	n/a	1	256	n/a	n/a	n/a
TSM34IP	1	2	4	8	16	32	64	128	n/a	256	512	1024	2048
ST5/10IP	1	2	4	8	16	32	64	128	16384	256	512	1024	2048
ST[illegible]	1	2	4	8	16	32	64	128	n/a	256	512	1024	2048
TXM34IP	1	2	4	8	16	n/a	n/a	n/a	n/a	256	512	1024	n/a
TXM24IP	1	2	4	n/a	n/a	n/a	n/a	n/a	n/a	256	n/a	n/a	n/a
SWM24IP	2	4	8	n/a	n/a	n/a	n/a	n/a	1	256	n/a	n/a	n/a
STAC5IP*	2	4	8	16	n/a	n/a	n/a	n/a	1	256	512	n/a	n/a
*DB15 IN/OUT1 Connector													
	I/O表-扩展输入状态									使用31...34作为扩展产出			
	位码									位码			
产品	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	enc index				
STAC5IP**	1	2	4	8	16	32	64	128	n/a	256	512	1024	2048

位状态详细信息 - M2伺服

当输入关闭时，读数为0。打开时，它们根据下面的位代码读取。当输出打开时，它们的读数为0。当输出关闭时，它们按照下面的代码读取。如果多个输入打开（输出关闭），则代码将相加。

	位码												位码					
	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	IN9	IN10	IN11	IN12	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6
SV200	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536	131072

Configuration Assembly (0x66) 配置程序集

此连接点被EtherNET/IP Protocol用于配置各种参数，包括接收分组间隔（RPI），数据大小等。该连接点必须由用户具体规定。.

Heartbeat Input Only Assembly (0x67) 心跳输入只有程序集

此连接点表示零长度的程序集对象，其目的不是发送数据，而是简单地通知控制器，驱动器仍处于活动状态并生成数据.

Heartbeat Listen Only Assembly (0x68) 心跳仅侦听程序集

此连接点表示零长度的程序集对象，其目的不是发送数据，而是简单地通知驱动器，控制器仍处于活动状态并接收数据.

Output Assembly (0x70) 输出组件 PLC→控制驱动器

使用固件版本251.0或更高版本

输出程序集中的第一个元素用于指定要执行的命令类型。

装配元件	Description	Length	Units	Notes
0	Command Word 指令字	32 Bits		
1	Jog Speed Jog 点动速度	32 Bits	Jog Speed 0.25 RPM	See JS
2	Jog Accel Jog 点动加速度	32 Bits	10RPS²	See JA
3	Jog Decel Jog 点动减速度	32 Bits	10RPS²	See JL
4	Point-to-Point Velocity 点到点速度	32 Bits	0.25RPM	See VE
5	Point-to-Point Acceleration点到点加速度	32 Bits	10RPS²	See AC
6	Point-to-Point Deceleration点到点减速度	32 Bits	10RPS²	See DE
7	Point-to-point Distance点到点距离	32 Bits	Motor Steps	See DI
8	Current Command 当前电流仅伺服和步进伺服	32 Bits	0.01A	
9	SCL Command Letters SCL 命令字母	32 Bits	2x ASCII 字符	
10	Data/SCL Register 1 Data/SCL 寄存器 1	32 Bits	命令相关	
11	Data/SCL Register 2 Data/SCL 寄存器 2	32 Bits	命令相关	
12	(Reserved)保留	32 Bits		
13	(Reserved)保留	32 Bits		
14	(Reserved)保留	32 Bits		
15	(Reserved)保留	32 Bits		

Command Word：“指令字”可以写入的功能值：

Bit	命令值	描述
0	0x0	空闲状态（在重复命令之间发送此状态）
	0x1	保留
1	0x2	使能
2	0x4	去使能
3	0x8	执行相对定位 - FL
4	0x10	执行绝对定位 - FP
5	0x20	开始移动 - FS (找传感器)
6	0x40	开始移动 - FD (找双传感器)
7	0x80	开始移动 - FY (带安全距离的FS功能)
8	0x100	开始移动 - FM (带屏蔽距离的FS)
9	0x200	开始移动 - FO (置位输出)
10	0x400	开始移动 - FC (运行到设定距离改变速度)
11	0x800	开始移动 - SH (回原点)
12	0x1000	开始移动 - FH(回原点，具体看驱动器是否支持)
13	0x2000	开始移动 - FE(编码器跟随)
14	0x4000	按最大减速度停止/清空缓存 - AM
15	0x8000	按减速度停止/清空缓存 - DE
16	0x10000	开始点动
17	0x20000	更新点动速度，实时改变JOG速度
18	0x40000	发送主机命令(执行SCL指令)
19	0x80000	Q程序加载和执行（QX）
20	0x100000	报警复位 - AR
21	0x200000	电流指令（GC）-仅限伺服和步进伺服，改变力矩电流 CM=1力矩模式下使用

此程序集用于使能/去使能电机、执行速度或位置移动或调用嵌入式SCL处理程序。

注意：除了发送主机命令(0x40000)外，其他命令都是边沿触发的。因此，您必须在连续命令（如重复移动和警报重置）之间发送空闲状态命令（0x0）。

如果命令或相关参数被更改，则可以在没有中间空闲状态的情况下发送主机命令(0x40000)。

1、电机使能

1. 装配元件 0: 将0 -> 0x02 (bit 1). 电机将使能.
2. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

2、电机去使能

1. 装配元件 0: 将 0 -> 0x04 (bit 2). 电机将去使能.
2. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

3、点对点定位移动 (FL/FP）相对运动/绝对运动

要开始移动，用户必须指定适当的移动参数，然后触发移动。

1. 装配元件 4: 速度 (单位: 0.25 RPM)--不要写0，不然电机会不动，但在执行走位置
2. 装配元件 5: 加速度 (单位: 10 rps^2)
3. 装配元件 6: 减速度 (单位: 10 rps^2)
4. 装配元件 7: 距离
5. 装配元件 0: 从0 -> 0x08 (bit 3)开始FL增量移动, 或者从 0 -> 0x10 (bit 4)开始FP绝对移动.
6. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

4、点对点定位移动与I/O (FS/FD/FY/FM/FO/SH/FE)

1. 在装配元件4中指定移动速度 (单位: rev/sec*240)不要写0，不然电机会不动，但在执行走位置
2. 在装配元件5中指定加速度 (单位: rev/sec/sec*6)
3. 在组件元素6中指定减速率 (单位 rev/sec/sec*6)
4. 在部件元素7中指定距离 (单位：steps)
5. 在SCL寄存器1中指定IO点和条件 (见IO 编码表)
6. 在SCL寄存器2中指定第二个io点和条件(仅FD指令，找两个传感器)
7. 通过将程序集元素0设置为命令字表中的适当值，指定开始移动。
8. 将程序集元素0设置为0以准备下一个命令。

5、停止并清空缓存 (AM, “立即停车”)

1. 装配元件 0: 从 0 -> 0x4000 (bit 14). 电机将急速停止.
2. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

6、停止并清空缓存 (DE, “正常制动停车”)

1. 装配元件 0: 从 0 -> 0x8000 (bit 15)电机将正常停止.
2. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

7、点动 (Jog) 运动，速度模式

1. 在部件元素1中指定点动速度 (单位: rev/sec*240) (若要向CCW方向慢跑，请输入负速度或16进制取反加1)
2. 在部件元素2中指定加速度 (单位: rev/sec/sec*6)
3. 在组件元素3中指定减速率 (单位: rev/sec/sec*6)
4. 通过将装配元素0设置为0x10000（位16）指定开始点动
5. 将程序集元素0设置为0以准备下一个命令.

8、更新JOG速度 (速度模式运动时实时改变速度)

1. 装配元件 1: 新的点动速度 (单位: 0.25 RPM) (若要向CCW方向慢跑，请输入负速度)
2. 装配元件 0: 从0x20000 (bit 17) 实时更新慢跑速度.
3. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

注意：当电机转动时，点动加速度和点动减速度值不能改变

9、嵌入式SCL处理程序

(有关命令列表，请参见下面页面上的表)

一般用法如下:

1. 装配元件 9: SCL 命令字符
   a. 注意：总是两个ASCii字符，在下面的16位中
2. 装配元件 10 (Data/SCL 寄存器 1)
3. 装配元件 11 (Data/SCL 寄存器2)
4. 装配元件 0: 从 0x40000 (bit 18) 执行命令.
5. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

10、Q加载并执行 (QX)

1. 装配元件 10: 要执行的段
2. 装配元件 0: 从 0 -> 0x80000 (bit 19) 开始执行指定的Q段.
3. 装配元件 0: 复位为0x0 (空闲状态) 为下一个命令做准备.

11、点对点移动并改变速度 (FC)

1. 要指定要更改电机速度的距离，请使用主机命令DC.
2. 要指定要将电机更改为的速度，请使用主机命令VC.
3. 在装配元件4中指定初始移动速度 (单位: rev/sec*240)
4. 在装配元件5中指定加速度(单位: rev/sec/sec*6)
5. 在装配元件6中指定减速度(单位: rev/sec/sec*6)
6. 在装配元件7中指定距离 (单位: steps)
7. 通过将汇编元素0设置为0x400（bit10）来指定开始移动。
8. 将装配元素0设置为0，以准备下一个命令

12、报警复位

1. 通过设置装配元件请求报警复位0 到0x100000 (bit 20).
2. 将程序集元素0设置为0以准备下一个命令。

对于涉及输入和输出的命令，如FO、FS等，需要在Data/SCL寄存器1中指定IO和条件.

举例1: SO2L

将输出程序集的SCL命令函字段设置为“SO”. 如果您的编程环境不支持ASCII字符串，请使用等效于SO0x534F的十六进制.
将data/scl寄存器1字段设置为2L的十六进制代码，来自下表：0x4CB2
设置主机命令设置命令字位：bit18（0x40000）

举例2: FS3R

将data/scl寄存器1字段设置为3R的十六进制代码, 从下表：0x52B3设置给传感器的命令字位：bit5（0x20）

IO命令的数据/SCL寄存器代码:

	Enc Index	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	IN9	IN10	IN11	IN12
High	0x48B0	0x48B1	0x48B2	0x48B3	0x48B4	0x48B5	0x48B6	0x48B7	0x48B8	0x48B9	0x48BA	0x48BB	0x48BC
Low	0x4CB0	0x4CB1	0x4CB2	0x4CB3	0x4CB4	0x4CB5	0x4CB6	0x4CB7	0x4CB8	0x4CB9	0x4CBA	0x4CBB	0x4CBC
Rising Edge	0x52B0	0x52B1	0x52B2	0x52B3	0x52B4	0x52B5	0x52B6	0x52B7	0x52B8	0x52B9	0x52BA	0x52BB	0x52BC
Falling Edge	0x46B0	0x46B1	0x46B2	0x46B3	0x46B4	0x46B5	0x46B6	0x46B7	0x46B8	0x46B9	0x46BA	0x46BB	0x46BC

可用SCL命令

注：可在本手册主要部分中每个命令的专用信息页上找到命令特定的数据单位和范围。这张表跨两页.

命令	Data/SCL 寄存器 1	Data/SCL 寄存器 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 =1力矩	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 Limit	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-力矩电流	Data	0	x			x	
HA - Homing Accel	'1', '2', or '3'	Data		x	x		x
HC - Hardstop Current Limit	Data	0		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

SV200的附加隐式SCL命令

命令	Data/SCL Register 1	Data/SCL Register 2
AD - Analog Deadband (SV200 Drives)	‘1’, ‘2’, or ‘3’	Data
AV - Analog O□set 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 - Di□erential Constant		
KE - Di□erential 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		

显式消息传递

AMP的EtherNet/IP实现允许使用class 3连接或未连接消息管理器（UCMM）显式消息传递。此自定义配置文件的服务代码为0x3C，类代码为0x64.

除了自定义概要外，还实现了下列标准对象和服务:

- 消息路由器对象 (Volume 1, Section 5-3)
- 连接管理器(Volume 1, Section 5-7)
- 连接配置 (Volume 1, Section 5-50)
- 接口 (Volume 1, Section 3-7)
- 以太网链路对象 (Volume 2, Chapter 5)
- TCP/IP 对象 (Volume 2, Chapter 5)
- 装配 (Volume 1, Section 5-37)
- CIP同步对象 (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

显式消息类型

可以向应用的运动EtherNET/IP驱动器发送两种类型的显式消息。class1消息包括大多数缓冲的SCL和Q命令。然而，与通过RS-232、RS-485和标准以太网发送的SCL和Q命令不同，class1消息不支持查询。“立即”SCL命令不能封装在class1消息中.

class2消息提供了与class1消息不可用的附加功能，包括读取设置和寄存器的能力。这两种类型都可以通过class3连接发送，也可以发送到未连接的消息管理器（ucmm）.

两种命令消息类型都会导致响应消息，即使没有请求数据.

所有数值都是二的补码。整数以大端字节（最高有效字节优先）发送.

有关SCL和Q命令的详细说明，请参阅本手册的主要部分。阅读本手册主要部分中的命令说明时，请注意，EtherNET/IP封装通常需要使用不同的单元和不同范围的可接受值。

类型1消息格式

有关命令的完整列表，请参见表1。响应消息将始终回显操作码和注册码（如果存在）。响应消息中还包含驱动器的状态代码，这是一种指示有用信息的位模式，例如是否存在故障或电机是否处于运动状态。有关更多信息，请参见本手册前面有关sc命令的部分.

注：所有数值均为二补。整数被发送到大端（最高有效字节优先）。

命令消息格式:

	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		..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	in			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/TXM24)	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 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	DBG_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 Configu ator, STAC Configu ator 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.

AL - Alarm Code - 報警代码

Compatibility: All drives
See also: AI, AR, AX commands, Appendix

Reads back an equivalent hexadecimal value of the Alarm Code's 16-bit binary word.

Command Details:

Structure	AL
Command Type	IMMEDIATE
Usage	READ ONLY
Non-Volatile	NO
Register Access	"f" (054) Note: response to AL command is a different format than the response to the RLf command. See Appendix F for details.
Units	Hexadecimal value of 16-bit binary word (see below)

Response Details:

Hex Value	BLu	SV	STAC6	ST	STM
0001	***Position Limit***				
0002	CCW Limit				
0004	CW Limit				
0008	***Over Temp***				
0010	***Excess Regen****	***Internal Voltage***	***Excess Regen***	***Internal Voltage***	***Internal Voltage***
0020	***Over Voltage***				
0040	***Under Voltage****	Under Voltage	***Under Voltage***	Under Voltage	Under Voltage
0080	***Over Current***				
0100	***Bad Hall Sensor***		***Open Motor Winding***		
0200	***Bad Encoder***				(not used)
0400	Comm Error				
0800	Bad Flash				
1000	Wizard Failed		No Move		
2000	Current Foldback		Motor Resistance Out of Range	(not used)	(not used)
4000	Blank Q Segment				
8000	No Move		(not used)		

* BLuAC drives only
NOTE: Items in ***bold italic*** represent Drive Faults, which automatically disable the motor. Use the OF command in a Q Program to branch on a Drive Fault.
NOTE: See Appendix for more detailed information on Alarm Codes.

Examples:

Command	Drive sends	Notes
AL	AL=0000	No alarms
AL	AL=0001	Position limit alarm
AL	AL=0201	Position limit and bad encoder signal alarms

SC - Status Code - 状态码

Compatibility: All drives
See also: RS command

Requests the current drive status as the Hexadecimal equivalent of a binary word. Each bit in the binary word relates to a status condition (see assignments below). The representation of this binary word as a hexadecimal value is called the Status Code. Drives can have multiple status conditions at one time, and host systems can typically interpret a Hexadecimal code very quickly. See Appendix E for more details on the Status Code.

Command Details:

Structure	SC
Type	IMMEDIATE
Usage	READ ONLY
Non-Volatile	NO
Register Access	None
Units	Hexadecimal equivalent of the binary status code word (see bit assignments below)

反馈详情：

Hex Value	状态码位定义
0001	电机已经使能(如果此位为0，则电机禁能)
0002	取样(用于快速调谐器)
0004	驱动故障(检查报警代码)
0008	就位（电机就位）
0010	移动(电机正在运动)
0020	点动(目前处于点动模式)
0040	停止(在停止命令停止的过程中)
0080	等待(因为一个输入点；执行WI命令)
0100	保存(正在保存参数数据)
0200	报警存在（检查报警代码）
0400	正在回原点（正在执行SH命令）
0800	等待(等待时间，执行WD或WT命令)
1000	向导正在运行（定时向导正在运行）
2000	检查编码器(计时向导正在运行)
4000	程序正在运行
8000	初始化(在启动时发生)

例子：

Command	驱动器发送	注释
SC	SC=0009	驱动器已就位并启用(十六进制值0001和0008)
SC	SC=0004	驱动器故障并禁用(十六进制值0004)
SC	SC=0209	驱动器具有报警，处于位置并已启用（十六进制值0001、0008和0200）