EtherNet/IP option board OPTCK for 9000X drives

Effective September 2010

User manual

Powering Business Worldwide

September 2010

Important notice—please read

The product discussed in this literature is subject to terms and conditions outlined in Eaton selling policies. The sole source governing the rights and remedies of any purchaser of this equipment is the relevant Eaton selling policy.

NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE OR MERCHANTABILITY, OR WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE, ARE MADE REGARDING THE INFORMATION, RECOMMENDATIONS AND DESCRIPTIONS CONTAINED HEREIN. In no event will Eaton be responsible to the purchaser or user in contract, in tort (including negligence), strict liability or otherwise for any special, indirect, incidental or consequential damage or loss whatsoever, including but not limited to damage or loss of use of equipment, plant or power system, cost of capital, loss of power, additional expenses in the use of existing power facilities, or claims against the purchaser or user by its customers resulting from the use of the information, recommendations and descriptions contained herein.

The information contained in this manual is subject to change without notice.

Cover Photo: Eaton 9000X AF Drives.

Table of contents

List of figures

List of tables

List of tables, continued

Safety

Definitions and symbols

WARNING

This symbol indicates high voltage. It calls your attention to items or operations that could be dangerous to you and other persons operating this equipment. Read the message and follow the instructions carefully.

This symbol is the "Safety Alert Symbol." It occurs with either of two signal words: CAUTION or WARNING, as described below.

WARNING

Indicates a potentially hazardous situation which, if not avoided, can result in serious injury or death.

CAUTION

Indicates a potentially hazardous situation which, if not avoided, can result in minor to moderate injury, or serious damage to the product. The situation described in the CAUTION may, if not avoided, lead to serious results. Important safety measures are described in CAUTION (as well as WARNING).

Hazardous high voltage

WARNING

Motor control equipment and electronic controllers are connected to hazardous line voltages. When servicing drives and electronic controllers, there may be exposed components with housings or protrusions at or above line potential. Extreme care should be taken to protect against shock.

Stand on an insulating pad and make it a habit to use only one hand when checking components. Always work with another person in case an emergency occurs. Disconnect power before checking controllers or performing maintenance. Be sure equipment is properly grounded. Wear safety glasses whenever working on electronic controllers or rotating machinery.

Cautions and notices

Read this manual thoroughly and make sure you understand the procedures before you attempt to install, set up, or operate Eaton’s 9000X AF Drives.

Cautions

CAUTION

Be ABSOLUTELY sure not to connect two functions to one and same output in order to avoid function overruns and to ensure flawless operation.

CAUTION

The calculated model does not protect the motor if the airflow to the motor is reduced by blocked air intake grill.

Notices

Notice

The *inputs*, unlike the *outputs*, cannot be changed in RUN state.

Chapter 1—General information

Eaton 9000X frequency converters can be connected to an Ethernet using the Eaton Multiprotocol Ethernet module, part number OPTCK. Figure 1 shows the module.

Figure 1-1: EtherNet/IP Module

The OPTCK module can be installed in the card slots D or E.

Every appliance connected to an Ethernet network has two identifiers: a MAC address and an IP address. The MAC address (address format: xx:xx:xx:xx:xx:xx) is unique to the appliance and cannot be changed. The Ethernet board's MAC address can be found on the sticker attached to the board.

In a local network, IP addresses can be defined by the user as long as all units connected to the network are given the same network portion of the address. For more information about IP addresses, contact your network administrator. Overlapping IP addresses cause conflicts between appliances. For more information about setting IP addresses, see **Chapter 2—Installation**.

WARNING

Internal components and circuit boards are at high potential when the frequency converter is connected to the power source. This voltage is extremely dangerous and may cause death or severe injury if you come into contact with it.

Table 1-1: Ethernet Board Technical Data

General	Card Name	OPTCK
Ethernet connections	Interface	RJ-45 connector
	Transfer cable	Foiled CAT5e
Communications	Speed	10/100 Mb
	Duplex	half/full
	Default IP-address	192.168.0.10
Protocols	EtherNet/IP	
Environment	Ambient operating temperature	–10°C to 50°C
	Storing temperature	–40°C to 70°C
	Humidity	<95%, no condensation allowed
	Altitude	Max. 1000m
	Vibration	0.5G at 9 to 200 Hz
Safety	—	Fulfills EN50178 standard

Chapter 2—Installation

⚠ CAUTION

Make sure that the frequency converter is switched off before an option or fieldbus board is changed or added.

Table 2-1: Installing the Ethernet Option Board

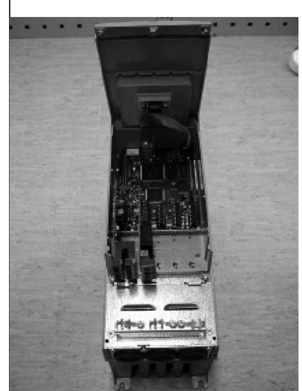
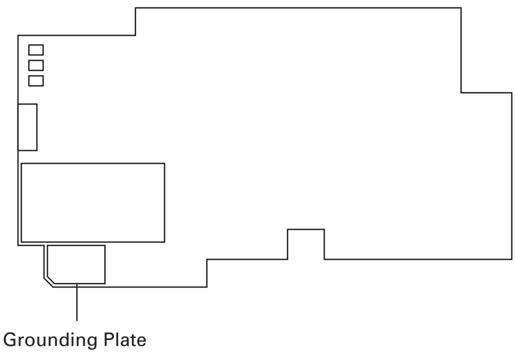
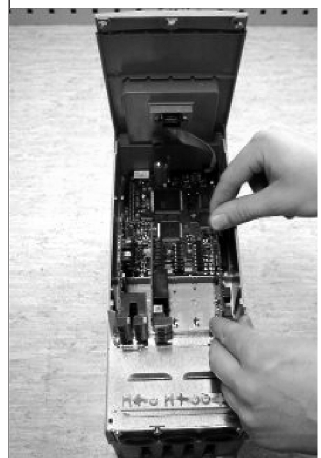
Item	Description	
A	9000X frequency converter.	
B	Remove the cable cover.	
C	Open the cover of the control unit.	

Table 2-1: Installing the Ethernet Option Board (continued)

Item	Description	
D	Install Ethernet option board in slot D or E on the control board of the frequency converter. Make sure that the grounding plate (see below) fits tightly in the clamp.  	
E	Make a sufficiently wide opening for your cable by cutting the grid as wide as necessary.	
F	Close the cover of the control unit and the cable cover.	

Ethernet Configuration

The Ethernet parameters of the OPTCK option board are configured with the control keypad by giving values to appropriate parameters in the Expander board menu.

Expander board menu

The Expander board menu on the control keypad makes it possible for the user to see what expander boards are connected to the control board, and to reach and edit the parameters associated with the expander board. From within the Expander board menu, you can browse through slots A to E with the up/down buttons to see what expander boards are connected. Selecting the OPTCK and the parameters will display the Ethernet parameters for configuration.

Table 2-2: Ethernet Parameters

Number	Name	Default	Range	Description
1	Comm. Time-out	—	—	Not used
2	IP Part 1	192	1…223	IP address Part 1
3	IP Part 2	168	0…255	IP address Part 2
4	IP Part 3	0	0…255	IP address Part 3
5	IP Part 4	10	0…255	IP address Part 4
6	SubNet Part 1	255	0…255	SubNet Mask Part 1
7	SubNet Part 2	255	0…255	SubNet Mask Part 2
8	SubNet Part 3	255	0…255	SubNet Mask Part 3
9	SubNet Part 4	0	0…255	SubNet Mask Part 4
10	DefGW Part 1	0	0…255	Default Gateway Part 1
11	DefGW Part 2	0	0…255	Default Gateway Part 2
12	DefGW Part 3	0	0…255	Default Gateway Part 3
13	DefGW Part 4	0	0…255	Default Gateway Part 4
14	Input Instance	—	—	Not used
15	Output Instance	—	—	Not used

IP address

IP is divided to 4 parts. (Part = Octet) Default IP Address is 192.168.0.10.

Note: Ethernet parameters are saved to the OPTCK option card, not the drive.

Chapter 3—Operation

Status LEDs

Figure 3-1: Led Status

Table 3-1: EtherNet/IP Status LEDs

LED	State	Meaning
Power	Off	Option card not powered
	On	Option card powered
Connection	Off	Not configured
	Blinking	Configured — no EtherNet/IP connections
	On	Configured — EtherNet/IP connection active
Status	Off	No EtherNet/IP faults
	Blinking	Recoverable faults
	On	Unrecoverable fault
Activity	—	Flashes with Ethernet message activity
Link	Off	Not wired or wired to 10 mbps port
	On	Wired to 100 mbps port

Chapter 4—EtherNet/IP protocol connections

CIP defines two types of connection-based messages: I/O messages and explicit messages. In EtherNet/IP, these are implemented by type 1 and type 3 messages, respectively. Type 1 messages (I/O messages) periodically access I/O assemblies. Type 3 messages (explicit messages) are used to access attribute(s) of a specified instance of a specified class. An attribute is defined by a (class, instance, attribute) triplet, with instance 0 used for class attributes applying to an entire class, and instances >1 for instance attributes that are defined separately for each supported instance of the class. The Connection Manager object allocates and manages the internal resources associated with both I/O and explicit message connections.

Both explicit messages and I/O messages can also be sent as unconnected messages. To read input assembly M using an unconnected message, use the "get attribute single" service with attribute (class, instance, attribute) = (4, M, 3). To write/read output assembly N using an unconnected message, use the "set attribute single" / "get attribute single" service with attribute (class, instance, attribute) = (4, N, 3).

EtherNet/IP classes

The EtherNet/IP specification defines the following ranges of classes.

Table 4-1: Class ID Ranges

Range (Decimal)	Range (Hexadecimal)	Meaning	Used by OPTCK
0–99	00–63	CIP common	Yes
100–199	64–C7	Vendor specific	Yes
200–239	C8–EF	Reserved	No
240–255	F0–FF	CIP common	Yes
256–767	100–2FF	CIP common	No
768–1279	300–4FF	Vendor specific	No
1280–65,535	500–FFFF	Reserved	No

By default, the OPTCK module supports the following object classes.

Table 4-2: Default Supported Object Classes

Object Class Code			
Decimal	**Hexadecimal**	**Description**	**Type**
1	1	Identity	CIP common
2	2	Message router	CIP common
4	4	Assembly	CIP common
6	6	Connection manager	CIP common
40	28	Motor data	CIP common
41	29	Control supervisor	CIP common
42	2A	AC/DC drive	CIP common
245	F5	IP network	CIP common
246	F6	Ethernet link	CIP common
160	A0	Window into parameter space	Vendor specific
170	AA	Measurement table	Vendor specific
190	BE	Selectors	Vendor specific

I/O assemblies

I/O assembly instances 20-25 and 70-75 are defined in the ODVA AC drive profile. The range 100-199 is reserved for vendor-specific assemblies. Of this range, six are defined for use.

Table 4-3: Assembly Instance Ranges

Range			
Decimal	**Hexadecimal**	**Meaning**	**Used by OPTCK**
1–99	01–63	Open (static assemblies defined in device profile)	Yes
100–199	64–C7	Vendor-specific static and dynamic assemblies	Yes
200–255	C8–FF	Reserved	No
256–767	100–2FF	Open (static assemblies defined in device profile)	No
768–1279	300–4FF	Vendor-specific static and dynamic assemblies	No
1280–65,535	500–FFFF	Reserved	No

Table 4-4: Supported I/O Assemblies

Number			Size	
Decimal	Hex	Type	Bytes	Name
20	0x14	Output	4	Basic speed control output
21	0x15	Output	4	Extended speed control output
22	0x16	Output	6	Speed and torque control output
23	0x17	Output	6	Extended speed and torque control output
24	0x18	Output	6	Process control output
25	0x19	Output	6	Extended process control output
101	0x65	Output	8	Dynamic process output
111	0x6F	Output	20	EIP process output format 1
121	0x79	Output	12	EIP process output format 2
70	0x46	Input	4	Basic speed control input
71	0x47	Input	4	Extended speed control input
72	0x48	Input	6	Speed and torque control input
73	0x49	Input	6	Extended speed and torque control input
74	0x4A	Input	6	Process control input
75	0x4B	Input	6	Extended process control input
107	0x6B	Input	8	Dynamic process input
117	0x75	Input	34	EIP process input format 1
127	0x7F	Input	20	EIP process input format 2

There are two mechanisms used to pass data between the interface board and the main processor board of the drive (See **Figure 4-1** on following page). Both use a serial peripheral interface (SPI). The drives SPI API provides a fast data access channel that passes 11 process data items in 10 milliseconds, and a slow data access mechanism for general parameter access in 50 to 100 milliseconds.

The first two fast data channels for both output and input are reserved for control and status. The third for speed reference and actual speed. The remaining eight (in each direction) can be mapped to any parameter ID.

CAUTION

In the standard application contained in the 9000X drive's "All in One" software, I/O assemblies 22–25 and 70–75 require certain mappings to work correctly.
Assemblies 22 and 23 requires Torque Reference mapped to Process Data In 1.
Assemblies 72 and 73 requires Actual Torque mapped to Process Data Out 4.
Assemblies 24 and 25 requires Process Reference 1 mapped to Process Data In 1.
Assemblies 74 and 75 requires Process Actual 1 mapped to Process Data Out 1.

Notes on the use of process data variables

A good understanding of the way that data is communicated between the OPTCK and the main processor board of the 9000X drive will help maximize system performance. Please refer to **Figure 4-1** for the following discussion.

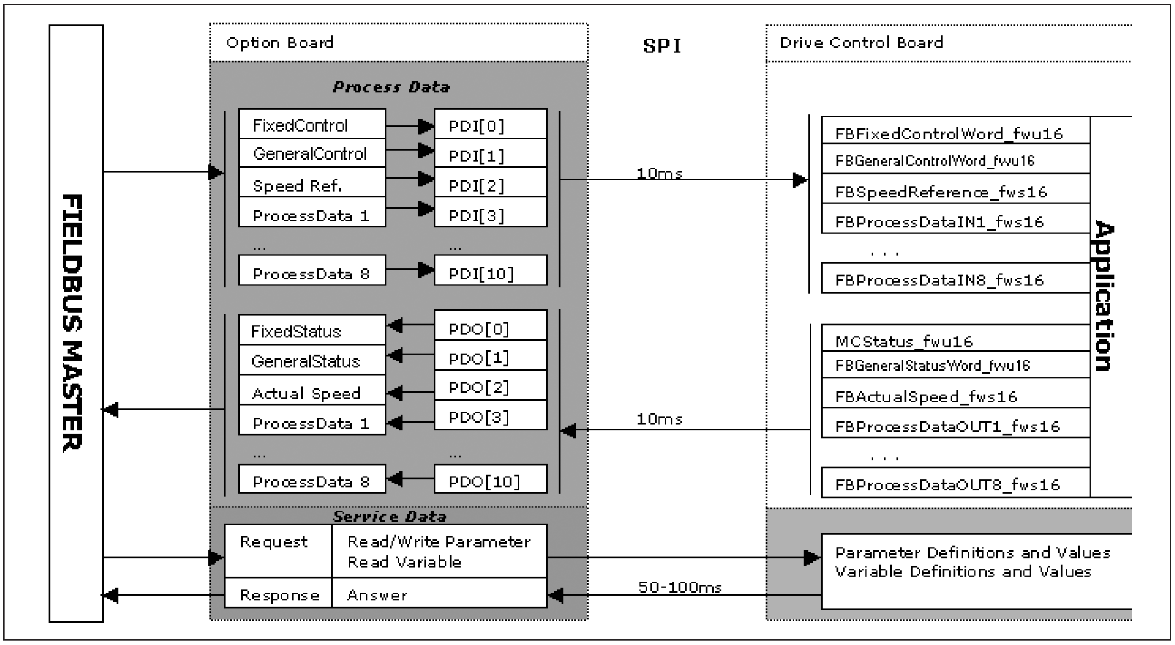


Figure 4-1: Data Channels—High Speed and Service

Every piece of data that is exchanged between the 9000X drive and the OPTCK occurs over a high-speed serial channel (SPI). There are 22 items that are given high priority and that are guaranteed to be updated every 10 milliseconds. There are 11 output words and 11 input words. The first three words in either direction are reserved for control, status, speed reference, and speed feedback. The remaining 16 words are available for the end user. These 16 words are referenced as process data variables throughout all 9000X documentation. The process data outputs are configured from the factory to provide the data listed in **Table 8-13, page 8-70 of SVX User Manual MN04003002E**. The Process Data Inputs vary in usage according to the application (see application-specific documentation for details. Process data variables can be mapped to any parameter ID in the drive. Some all-in-one applications provide keypad support for this feature. These process data selectors are usually found in the field bus group of the keypad. Please reference the SVX/SPX application user manual for details. It is strongly recommended that the multipurpose application be used for new applications, as it supports remapping, as well as others features optimized for network communications. The use of process data variables will allow the OPTCK to transmit any parameter with an ID number at a 10-millisecond update. These parameters are then mapped into the I/O assemblies as process data variables. If the same parameter is read directly by using it ID, it will take 50 to 100 milliseconds to get the parameter. Please note that the I/O assemblies of the OPTCK have been optimized for speed and use process data variables for this reason. The use of explicit messages is also possible and will normally read parameters from their native location at the lower speed. This is usually not an issue, as explicit messages are normally used for configuration and are usually not used for control (although they can be if desired). Note that process data selectors have parameter IDs, so they may also be set over the network. There will be a delay of about 100 milliseconds before the remapped data for the selected parameter starts to update at the 10 millisecond rate, if this is technique is used.

Example

The controller must read the drive's minimum frequency (parameter ID 101). Use the keypad to set the FB Process Sel Out 1 item to a value of 101. This will allow Process Data Out 1 to indirectly reference parameter 101. The controller will then read the value at Process Data Out 1, register number 2104.

Notes on the use of I/O assemblies

The I/O assemblies used in the OPTCK fall into two categories: ODVA predefined, and vendor specific. The ODVA predefined assemblies follow the bit layouts specified by the CIP standard drive profile. In general, the first two words of the input and output assemblies are very similar among the drive profile assemblies. The first word of each assembly contains bits for basic status and control. The second word contains speed feedback and speed reference information. The drive profile assemblies provide bits additional functionality as the instance numbers increase.

The ODVA standard also allows for vendor-specific I/O assemblies. Vendor-specific assemblies can be identified by instance numbers greater than or equal to 100. In Eaton's vendor-specific assemblies, there is a mechanism that allows the status and control words to either mimic those of the drive profiles assemblies, or to reflect the native bit layouts as mapped internally to the drive. In general, the vendor-specific assemblies provide a greater depth of information than the drive profile assemblies.

It is recommended that Eaton's vendor-specific assemblies be used in new applications because of their additional information. Drive profile assemblies may be used in systems were backward compatibility is required.

Status word

Tables 4-6 and **4-7** shows the options for the status in assemblies 117 and 127. Be advised that the content of the drive profile assemblies vary slightly as the instance numbers become larger; please reference the tables that appear in this document's I/O assembly section for details. Also note that assembly 107 is not user configurable as are the other vendor-specific assemblies. Note: The RefFromNet and CtrlFromNet bits in the CIP drive profile status word use the drive status word (parameter ID 43) as shown in **Table 4-5.** Care should be taken to make sure that the drives application program uses the correct version of parameter 43 (this is only an issue with custom applications; the Eaton all-in-one applications all support parameter ID43).

Table 4-5: Definition of CtrlFromNet and RefFromNet

Bit Variable	Value
CtrlFromNet	BusCtrl and FB_ Ctrl_Active = (fixed control word bit 8) and (Parameter[43] bit 10)
RefFromNet	BusRef and FB_ Ref_Active = (fixed control word bit 9) and (Parameter[43] bit 4)

Table 4-6: Assembly 117 Status Word

FB Status Type	Status Word	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0	CIP Drive Profile	Drive State								AtRef	RefFromNet	CtrlFromNet	Ready	Running2	Running1	Warning	Faulted
1	General Status Word	FB WD Pluse	Remote Act	Start Delay	FB Ref Active	DC Brake	UV Fast Stop	Detected Encoder Direction	TC Speed Limit Active	Flux Ready	Zero Speed	At Ref	Warning	Fault	Reverse	Running	Ready

Table 4-7: Assembly 127 Status Word

FB Status Type	Status Word	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0	Fixed Status Word	0	0	Start Delay	0	DC Brake	UV Fast Stop	Detected Encoder Direction	TC Speed Limit Active	Flux Ready	Zero Speed	At Ref	Warning	Fault	Reverse	Running	Ready
1	General Status Word	FB WD Pluse	Remote Act	Start Delay	FB Ref Active	DC Brake	UV Fast Stop	Detected Encoder Direction	TC Speed Limit Active	Flux Ready	Zero Speed	At Ref	Warning	Fault	Reverse	Running	Ready

Control word

The following diagrams contain direct comparisons of the drive profile and vendor-specific control words. Note that the drive profile assemblies vary slightly as the instance numbers become larger. Please reference the diagrams that appear in this document's I/O assembly section for the exact layouts. Control word bits in assemblies 21 through 101 will be a variant of the CIP drive profile. Higher number assemblies will adhere to the Eaton vendor-specific template. When using the drive profile output assemblies, the network controller must set bits 5 and 6 TRUE in order for network control and speed reference to be active. For the drive profile assemblies, the control bit functions are remapped to the equivalent functions in the drive fixed-control word. In the Eaton vendor-specific assemblies, the bits are passed directly to the drive's fixed-control word, exactly as used by the drive. It is possible to use the OPTCK adapter with any application as long as the exact bit layout of the application control word is known, and it is passed by an appropriate assembly with FB Control Type is set properly. For example, the HVX application could be used with assy 121 and FB Control Type = 0. Note that bit 15 of the fixed control word is reserved by the OPTCK firmware, which uses it to indicate an Ethernet connection loss (i.e., any custom application should NOT use this bit).

Table 4-8: Control Word

Control Word	**B15**	**B14**	**B13**	**B12**	**B11**	**B10**	**B9**	**B8**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
CIP drive profile	Varies (see semantics)									NetRef	NetCtrl			Fault Reset	RunRev	RunFwd
Fixed control word	Connection Loss													FaultReset	Dir	Run
Fixed control word (HVX)		PMSetback	FireMode Activate	DigOut1, RO5 or RO8	RelOut2, 4 or 7	RelOut1, 3 or 6			FBDIN6	FBDIN5	FBDIN4	FBDIN3	EnaBypass	FaultReset	Dir	Run

Speed reference word

The following table shows the content of the speed reference. The attribute FBControl type of the selectors class can be used to modify the behavior of the speed reference word in certain assemblies. When FBControlType is set to zero, the speed reference passed from assemblies 20–-25 is scaled to RPMs. When FBControlType is set to one, the same assemblies pass the speed reference to FBSpeedReference unaltered. The standard all-in-one applications will expect to see a value that ranges from 0 to 10,000, representing 0 to 100.00% speed (min. freq. to max. freq.). All remaining assemblies pass the speed reference unaltered.

Table 4-9: Speed Reference Word

Speed Control	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
FB Speed Reference	Speed reference word (signed 16 bit)															

Speed actual word

The speed feedback word follows the same layout as the speed reference word.

Table 4-10: Speed Actual Word

Actual Speed	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
FB Speed Actual	Speed reference word (signed 16 bit)															

Default I/O assemblies

The default I/O assemblies for the OPTCK are 101/127. The tables below show how the data has been organized in these assemblies. The data is displayed in byte-aligned format. Depending on the type of controller you are using, some logic may be required to reorder the bytes to get correct values. If you are using an AB controller such as a Control Logix or Micro Logix PLC, the bytes will be used in correct order when they are defined as INTs in the Logix5000 programming software.

Table 4-11: Input Assembly Instance 107

Instance 107 – Input From Drive – Length = 8 Bytes								
Byte Offset	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	At Ref	RefFromNet	CtrlFromNet	Ready	Running2	Running1	Warning	Faulted
1	Drive State							
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Process Data Out 1 (low byte)							
5	Process Data Out 1 (high byte)							
6	Process Data Out 2 (low byte)							
7	Process Data Out 2 (high byte)							

Input instance 107 details

Faulted—Set when the drive is in a faulted state.

Warning—Set when the drive is in a warning state (drive will still run).

Running1—Set when drive is running forward.

Running2—Set when drive is running in reverse.

Ready—Set when drive is ready. The drive will run when the RUN bit of the OUTPUT assembly is set (or when in local mode, the run button is pushed).

The **CtrlFromNet** and **RefFromNet** bits are on when the drive is in Fieldbus mode and the equivalent bits are set in the output assembly. These bits may be used to determine if the drive is in LOCAL or REMOTE mode (local/remote mode is set from the keypad, or one “force to control” function mapped to digital input has been set).

AtRef—Bit is set when drive is within 2% of requested speed.

Drive State—This byte value indicates the current drive state in accordance with the CIP specification for the control supervisor object (Class 0x29).

Speed Reference—This is a 16-bit unsigned value that controls the speed of the drive. It is scaled from 0 to 10,000 and represents 0 to 100.00% speed. The percent speed is based on the minimum and maximum frequency values programmed into the drive from the basic parameters menu of the keypad.

Process Data Out 1 and **Process Data Out 2**. Both of these are 16-bit unsigned values and can be programmed to provide any item from the drive’s parameter map. Refer to the parameter ID section in the SVX or SPX user manual for a list of parameters. Note that some parameters are application specific (there are tables for application-specific parameters, as well as a table of common parameters in the user manual). In order to use process data outputs, the parameter ID must be entered into the corresponding process data output selector, found in the Fieldbus group of the drive’s keypad. Note that some applications do not have a Fieldbus group, and therefore, will not allow changing the parameters (it is recommended that the multipurpose application is used for this reason). The process data outputs are set to output frequency and motor speed by default.

Table 4-12: Output Assembly Instance 101

Instance 101 – Output to the Drive – Length = 8 Bytes								
Byte Offset	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0		NetRef	NetCtrl			FaultReset	RunRev	RunFwd
1	Process Pointer High Process Pointer Low							
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Process Data In 1 (low byte)							
5	Process Data In 1 (high byte)							
6	Process Data In 2 (low byte)							
7	Process Data In 2 (high byte)							

Output instance 101 details

Run Fwd—Set to command forward run of the drive.

Run Rev—Set to command reverse run of the drive.

Fault Reset—Set to perform a remote reset of the drive after a fault condition. The fault condition must be cleared first.

NetRef—This bit must be set for the drive to accept a network reference.

NetCtrl—This bit must be set in order for the drive to accept a network run command.

Note: These two bits may be constantly forced to an ON state by ladder logic, or actively controlled along with the RUN FWD or RUN REV bits. Note that both of the above bits must be set in order for the controller to determine if the drive is in Remote or Local mode, as the complimentary bits in the input assembly represent the keypad Local/ Remote status AND’ed with these two bits.

Process Pointer High and **Process Pointer Low**—Are two nibbles that allow dynamic selection of PROCESS DATA SEL OUT 1 to PROCESS DATA SEL OUT 2. There are eight process data out selectors. These two nibbles will change which selectors are used *on the fly*.

When the nibbles are changed, a short delay will occur before the data is valid in the input assembly. For this reason, the second byte of the input assembly, normally used to provide drive state, is used as a handshake to indicate the data is valid. The data is valid when the second byte of the input assembly matches the second byte of the output assembly. If this feature is not used, leave this word set to zero (it must be set to zero in order for the drive state in byte 1 of the input assembly to be valid).

RPM Speed Reference—The speed reference used by the drive, in RPM.

Process Data In 1 and **Process Data In 2**. These two 16-bit registers are used to pass application-specific parameters to the drive (refer to the following section on process data variables for a detailed description.

Data Sync field

The Data Sync field only applies to the Selectors object and input assembly 117. After the Selectors object is written with a "set attributes all" service code, there is a short delay while the selectors are being transferred to the drive. After this transfer is complete, the Data Sync field in assembly 117 is updated to match the value written to the Selectors object to indicate that the selectors in the drive have been updated. However, it is still possible for the value-returned input assembly 117 to reflect the earlier selectors for a short time, because the interlock only extends to when the drive receives the write of the selectors, it does not include the time necessary to act upon the modified selectors and start outputting new data.

Output assemblies

Table 4-13: Assembly 20

Output Assembly 20, Length = 4 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0						FaultReset		RunFwd
1								
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							

Table 4-14: Assembly 21

Output Assembly 21, Length = 4 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0		NetRef	NetCtrl			FaultReset	RunRev	RunFwd
1								
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							

Table 4-15: Assembly 22

Output Assembly 22, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0						FaultReset		RunFwd
1								
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Torque Reference (low byte) in % of maximum torque setting							
5	Torque Reference (high byte) in % of maximum torque setting							

CAUTION

Output assemblies 22 and 23 requires Torque Reference mapped to Process Data In 1.

Table 4-16: Assembly 23

Output Assembly 23, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0		NetRef	NetCtrl			FaultReset	RunRev	RunFwd
1								
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Torque Reference (low byte)							
5	Torque Reference (high byte)							

CAUTION

Output assemblies 22 and 23 requires Torque Reference mapped to Process Data In 1.

Table 4-17: Assembly 24

Output Assembly 24, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0						FaultReset		RunFwd
1								
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Process Reference (low byte)							
5	Process Reference (high byte)							

CAUTION

Output assemblies 24 and 25 requires Process Reference 1 mapped to Process Data In 1.

Table 4-18: Assembly 25

Output Assembly 25, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	NetProc ①	NetRef	NetCtrl			FaultReset	RunRev	RunFwd
1	Mode							
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Process Reference (low byte)							
5	Process Reference (high byte)							

① Bit 7 of byte 0, NetProc in the ODVA specification, is not supported in this product.

CAUTION

Output assemblies 24 and 25 requires Process Reference 1 mapped to Process Data In 1.

Table 4-19: Assembly 101

Output Assembly 101, Length = 8 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0		NetRef	NetCtrl			FaultReset	RunRev	RunFwd
1	FB Out A3	FB Out A2	FB Out A1	FB Out A0	FB Out B3	FB Out B2	FB Out B1	FB Out B0
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Process Data In 1 (low byte)							
5	Process Data In 1 (high byte)							
6	Process Data In 2 (low byte)							
7	Process Data In 2 (high byte)							

Assembly 101 semantics

In byte 1 of output assembly 101, the two nibbles, FB Out A and FB Out B, are used as data selectors for Process Data Out A and Process Data Out B in input assembly 107. When FB Out A is set to a value from 1 to 8, Process Data Out A points to Process Data Out 1 through Process Data Out 8. If FBOutA is 0, Process Data Out A defaults to Process Data Out 1. Similarly, FB Out B selects where Process Data Out B points. Except, if FB Out B is 0, Process Data Out B defaults to Process Data Out 2.

If both FB Out A and FB Out B are 0, byte 1 of assembly 107 contains the control supervisor “state.” If at least one is nonzero, they are used to handshake the data selection, such that when byte 1 of assembly 107 equals byte 1 of assembly 101, the data in Process Data Out A and Process Data Out B of assembly 107 is valid.

Table 4-20: Assembly 111

Output Assembly 111, Length = 20 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Control Word (low byte)							
1	Control Word (high byte)							
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Process Data In 1 (low byte)							
5	Process Data In 1 (high byte)							
6	Process Data In 2 (low byte)							
7	Process Data In 2 (high byte)							
8	Process Data In 3 (low byte)							
9	Process Data In 3 (high byte)							
10	Process Data In 4 (low byte)							
11	Process Data In 4 (high byte)							
12	Process Data In 5 (low byte)							
13	Process Data In 5 (high byte)							
14	Process Data In 6 (low byte)							
15	Process Data In 6 (high byte)							
16	Process Data In 7 (low byte)							
17	Process Data In 7 (high byte)							
18	Process Data In 8 (low byte)							
19	Process Data In 8 (high byte)							

Table 4-21: Assembly 121

Output Assembly 121, Length = 12 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Control Word (low byte)							
1	Control Word (high byte)							
2	Speed Reference (low byte) in % of maximum speed							
3	Speed Reference (high byte) in % of maximum speed							
4	Process Data In 1 (low byte)							
5	Process Data In 1 (high byte)							
6	Process Data In 2 (low byte)							
7	Process Data In 2 (high byte)							
8	Process Data In 3 (low byte)							
9	Process Data In 3 (high byte)							
10	Process Data In 4 (low byte)							
11	Process Data In 4 (high byte)							

Assemblies 111 and 121 semantics

Assembly 121 is a shortened version of assembly 111. SpeedReference and Process Data In 1–4 are the same for both assemblies. But the control word is defined differently for assemblies 111 and 121.

If FB Control Type of the selectors class is 0, for assembly 111 the control word is defined the same as bytes 0 and 1 of assembly 23; and for assembly 121, it is written to the fixed control word with one change. Bit 15 is replaced by Type 1 Loss of Connection Fault bit. If FB Control Type is nonzero, for both assemblies the control word is written to the general control word.

Input assemblies

Table 4-22: Assembly 70

Input Assembly 70, Length = 4 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0						Running1		Faulted
1								
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							

Table 4-23: Assembly 71

Input Assembly 71, Length = 4 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	AtReference	RefFromNet	CtrlFromNet	Ready	Running2	Running1	Warning	Faulted
1	Drive State							
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							

Table 4-24: Assembly 72

Input Assembly 72, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0						Running1		Faulted
1								
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Torque Actual (low byte)							
5	Torque Actual (high byte)							

CAUTION

Input assemblies 72 and 73 requires Torque Actual mapped to Process Data Out 4.

Table 4-25: Assembly 73

Input Assembly 73, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	AtReference	RefFromNet	CtrlFromNet	Ready	Running2	Running1	Warning	Faulted
1	Drive State							
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Torque Actual (low byte)							
5	Torque Actual (high byte)							

CAUTION

Input assemblies 72 and 73 requires Torque Actual mapped to Process Data Out 4.

Table 4-26: Assembly 74

Input Assembly 74, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0						Running1		Faulted
1								
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Process Actual (low byte)							
5	Process Actual (high byte)							

CAUTION

Input assemblies 74 and 75 requires Process Actual 1 mapped to Process Data Out 1.

Table 4-27: Assembly 75

Input Assembly 75, Length = 6 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	AtReference	RefFromNet	CtrlFromNet	Ready	Running2	Running1	Warning	Faulted
1	Drive State							
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Process Actual (low byte)							
5	Process Actual (high byte)							

CAUTION

Input assemblies 74 and 75 requires Process Actual 1 mapped to Process Data Out 1.

Table 4-28: Assembly 107

Input Assembly 107, Length = 8 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	AtReference	RefFromNet	CtrlFromNet	Ready	Running2	Running1	Warning	Faulted
1	Drive State (if both FB Out A and FB Out B = 0), or the following if one of them is nonzero							
	FB Out A				FB Out B			
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Process Data Out A (low byte)							
5	Process Data Out A (high byte)							
6	Process Data Out B (low byte)							
7	Process Data Out B (high byte)							

Note: See Assembly 101 for the explanation of bytes 1 and 4–7.

Table 4-29: Assembly 117

Input Assembly 117, Length = 34 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Status Word (low byte)							
1	Status Word (high byte)							
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							
12	Data Selector Sync Word (low byte)							
13	Data Selector Sync Word (high byte)							
14	Reserved							
15	Reserved							
16	Reserved							
17	Reserved							
18	Process Data Out 1 (low byte)							
19	Process Data Out 1 (high byte)							
20	Process Data Out 2 (low byte)							
21	Process Data Out 2 (high byte)							
22	Process Data Out 3 (low byte)							
23	Process Data Out 3 (high byte)							
24	Process Data Out 4 (low byte)							
25	Process Data Out 4 (high byte)							
26	Process Data Out 5 (low byte)							
27	Process Data Out 5 (high byte)							
28	Process Data Out 6 (low byte)							
29	Process Data Out 6 (high byte)							
30	Process Data Out 7 (low byte)							
31	Process Data Out 7 (high byte)							
32	Process Data Out 8 (low byte)							
33	Process Data Out 8 (high byte)							

Table 4-30: Assembly 127

Input Assembly 127, Length = 20 Bytes								
Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Status Word (low byte)							
1	Status Word (high byte)							
2	Speed Actual (low byte) in % of maximum speed							
3	Speed Actual (high byte) in % of maximum speed							
4	Process Data Out 1 (low byte)							
5	Process Data Out 1 (high byte)							
6	Process Data Out 2 (low byte)							
7	Process Data Out 2 (high byte)							
8	Process Data Out 3 (low byte)							
9	Process Data Out 3 (high byte)							
10	Process Data Out 4 (low byte)							
11	Process Data Out 4 (high byte)							
12	Process Data Out 5 (low byte)							
13	Process Data Out 5 (high byte)							
14	Process Data Out 6 (low byte)							
15	Process Data Out 6 (high byte)							
16	Process Data Out 7 (low byte)							
17	Process Data Out 7 (high byte)							
18	Process Data Out 8 (low byte)							
19	Process Data Out 8 (high byte)							

Assemblies 117 and 127 semantics

Assembly 127 is a shortened version of assembly 117. Speed Actual and Process Data Out 1–8 are the same for both assemblies. But the Data Select Sync Word is only present in assembly 117, and the status word is defined differently for assemblies 117 and 127.

If FB Status Type of the Selectors object (0 x BE) is 0, for assembly 117 the status word is defined the same as bytes 0 and 1 of assembly 75; and for assembly 127, it is the fixed status word. See **Tables 4-6** and **4-7**.

Chapter 5—Object class details

See **Appendix A: Table of supported services by object class** for a list services implemented for all objects.

Identity object—Class 1

This object provides basic identification and general information about the device. The identity object is required in all EtherNet/IP products.

Table 5-1: Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the object	1
2		Get	Max. instance	UINT	Maximum instance number	1
3		Get	Number of instances	UINT	Number of instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	
			Number of attributes	UINT	The number of optional attributes implemented	1
			Optional attributes	ARRAY of UINT		8
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	1
			Optional services	ARRAY of UINT		1
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		8

Table 5-2: Instance Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Vendor ID	UINT		44
2		Get	Device type	UINT		2
3		Get	Product code	UINT		200
4			Revision	Struct of:		
		Get	Major	USINT		1.2
		Get	Minor	USINT		
5		Get	Status	WORD		See semantics section
6		Get	Serial number	UDINT		Entered during manufacturing process
7		Get	Product name	SHORT STRING		9000X drive
8		Get	State	USINT	0 = Nonexistent 1 = Device self-testing 2 = Standby 3 = Operational 4 = Major recoverable fault 5 = Major unrecoverable fault 6–254 = Reserved 255 = Default for get attributes all service	See semantics section

Status

This attribute represents the current status of the entire device. Its value changes as the state of the device changes. Release V11 returns a constant value of 0x0074.

Table 5-3: Bit Definitions for Instance #1, Status Attribute of Identity Object

Bit(s)	Called	Definition
0	Owned	Reserved, set to 0
1		Reserved, set to 0
2	Configured	TRUE indicates the application of the device has been configured to do something different than the out-of-box default. This does not include configuration of the communications. This bit shall be set TRUE after a discovery process has resulted in the configuration of known QC Port devices being recorded within the Gateway.
3		reserved, set to 0
4–7	Extended device status	
8	Minor recoverable fault	TRUE indicates the device detected a problem with itself, which is thought to be recoverable. The problem does not cause the device to go into one of the faulted states.
9	Minor unrecoverable fault	TRUE indicates the device detected a problem with itself, which is thought to be unrecoverable. The problem does not cause the device to go into one of the faulted states.
10	Major recoverable fault	TRUE indicates the device detected a problem with itself, which caused the device to go into the major recoverable fault state.
11	Major unrecoverable fault	TRUE indicates the device detected a problem with itself, which caused the device to go into the major unrecoverable fault state. See the Behavior section.
12–15		Reserved, set to 0

Serial number

This attribute is a number used in conjunction with the Vendor ID to form a unique identifier for each device on Ethernet. Each vendor is responsible for guaranteeing the uniqueness of the serial number across all of its devices.

This is not the same serial number reported by ADDaptACC. ADDaptACC reports the full serial number of the power unit of the drive. But this serial number is a text string, and EtherNet/IP serial numbers are a 32-bit unsigned number, so format conversion is necessary.

A new serial number is constructed from three fields, bits 31–27 = manufacturing site code, bits 26–18 = date code, bits 17–0 = a subset of the serial number of the power unit of the drive. The date code is the month and date of manufacture encoded as a Julian month by the following formula: date_code = (12*(year–1996) + month) and 0x01FF. The and is to ensure that the code fits in nine bits. This code supports dates from 1/1996 to 7/2038 without rollover. The serial number of the power unit consists of an eight-digit decimal number represented as an ASCII character string, followed by a letter. The last five digits of the decimal number is converted to a binary number to form the subset used.

State

This attribute is an indication of the present state of the device. Note that the nature of a major unrecoverable fault could be such that it may not be accurately reflected by the state attribute.

This attribute reflects the dynamic status of the gateway. The defined states are:

Table 5-4: Defined States

Value	State Name	Description
0	Non-existent	This state will never be visible from within a device. This state is principally intended for a tool to be able to represent the lack of an instance in a physical device.
1	Device self-testing	Power-up or reset operation. Will not be visible from within a device because communications are not active in this state.
2	Standby	This state is reported while a QC port discovery is in process.
3	Operational	This state is reported when the gateway is powered up, configured, and operating normally.
4	Major recoverable fault	
5	Major unrecoverable fault	

Message router object—Class 2

Table 5-5: Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the object	1
2		Get	Max. instance	UINT	Maximum instance number	1
3		Get	Number of instances	UINT	Number of instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	
			Number of attributes	UINT	The number of optional attributes implemented	4
			Optional attributes	ARRAY of UINT		1, 2, 3, 4
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	1
			Optional Services	ARRAY of UINT		1
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		4

Table 5-6: Instance Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Object_list	STRUCT of	A list of supported objects	—
			Number	UINT	Number of supported classes in the classes array	—
			Classes	ARRAY of UINT	List of supported class codes	—
2		Get	Number available	UINT	Maximum number of connections supported	—
3		Get	Number active	UINT	Number of connections currently used by system components	—
4		Get	Active connections	ARRAY of: UINT	A list of the connection IDs of the currently active connections	—

Assembly object—Class 4

The assembly object binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection. Assembly objects can be used to bind input data or output data. The terms "input" and "output" are defined from the network's point of view. An input will produce data on the network, and an output will consume data from the network. Important: As currently implemented, all instances of the assembly object are static.

Table 5-7: Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the object	2
2		Get	Max. instance	UINT	Maximum instance number	Larger value of attribute is 1, 3 (input assembly) or is 1, 4 (output assembly)
3		Get	Number of instances	UINT	Number of instances	2
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	
			Number of attributes	UINT	The number of optional attributes implemented	1
			Optional attributes	ARRAY of UINT		4
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	3
			Optional services	ARRAY of UINT		0x10, 0x18, 0x19
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		4

Table 5-8: Instance Attributes (for All Implemented Instances N)

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
3		Set	Data	ARRAY of BYTE	The entire I/O assembly fits in this attribute	For the details of an assembly see **Chapter 4**.
4		Get	Assy instance size	UINT	Size of assembly in bytes	

Motor data object—Class 40 (0x28)

This object is defined by the config.ini file. The default configuration is given.

Table 5-9: Default Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Values
1		Get	Revision	UINT	Revision of the object	1
2		Get	Max. instance	UINT	Maximum instance number	1
3		Get	Number of instances	UINT	Number of instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	①
			Number of attributes	UINT	The number of optional attributes implemented	3
			Optional attributes	ARRAY of UINT		9, 12, 15
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	2
			Optional Services	ARRAY of UINT		1, 2
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		15 ①

① Attributes (0x28, 0, 4) and (0x28, 0, 7) will vary depending upon the definition of class 40 in config.ini. For the default configuration, the values for both are as listed here.

Table 5-10: Default Instance Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
3		Get	MotorType	USINT	0 = Nonstandard 1 = PM DC motor 2 = FC DC motor 3 = PM synchronous motor 4 = FC synchronous motor 5 = Switched reluctance motor 6 = Wound rotor induction motor 7 = Squirrel cage induction motor 8 = Stepper motor 9 = Sinusoidal PM BL motor 10 = Trapezoidal PM BL motor	7
6		Get/Set	RatedCurrent	UINT	Rated stator current X 100 mA	FW MotorNomCurrent
7		Get/Set	RatedVoltage	UINT	Rated base voltage volts	FW MotorNomVoltage
9		Get/Set	RatedFrequency	UINT	Rated electrical frequency Hz	FW MotorNomFreq

Control supervisor object—Class 41 (0x29)

This object is defined by the config.ini file. The default configuration is given.

Table 5-11: Default Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the object	1
2		Get	Max. instance	UINT	Maximum instance number	1
3		Get	Number of instances	UINT	Number of instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	①
			Number of attributes	UINT	The number of optional attributes implemented	9
			Optional attributes	ARRAY of UINT		4, 5, 6, 8, 9, 11, 13, 14, 15
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	2
			Optional services	ARRAY of UINT		1, 2
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		15 ①

① Attributes (0x29, 0, 4) and (0x29, 0, 7) will vary depending upon the definition of class 41 in config.ini. For the default configuration, the values for both are as listed here.

Table 5-12: Default Instance Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
3	NV	Get/Set	Run1	BOOL	Run forward request	Output assembly 21, Byte 0, Bit 0
4	NV	Get/Set	Run2	BOOL	Run reverse request	Output assembly 21, Byte 0, Bit 1
5	NV	Get/Set	NetCtrl	BOOL	Requests control from network	Output assembly 21, 101, 127, Byte 0, Bit 5
6		Get	State	USINT	1 = Startup 2 = Not ready 3 = Ready 4 = Enabled 5 = Stopping 6 = Fault stop 7 = Faulted	Output assembly 71, 73, 75, 107, Byte 1
7		Get	Running1	BOOL	0 = Other state 1 = Running forward	Output assembly 71, Byte 0, Bit 2
8		Get	Running2	BOOL	0 = Other state 1 = Running reverse	Output assembly 71, Byte 0, Bit 3
9		Get	Ready	BOOL	0 = Other state 1 = Ready for RUN command	Output assembly 71, Byte 0, Bit 4

AC/DC drive object—Class 42 (0x2A)

This object is defined by the config.ini file. The default configuration is given.

Table 5-13: Default Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the object	1
2		Get	Max. instance	UINT	Maximum instance number	1
3		Get	Number of instances	UINT	Number of instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	①
			Number of attributes	UINT	The number of optional attributes implemented	21
			Optional attributes	ARRAY of UINT		3, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
5		Get	Optional service list	Struct of	A list of optional instance services implemented	2
			Number of services	UINT	The number of optional services implemented	1, 2
			Optional services	ARRAY of UINT		
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		19 ①

① Attributes (0x2A, 0, 4) and (0x2A, 0, 7) will vary depending upon the definition of class 42 in config.ini. For the default configuration, the values for both are as listed here.

Table 5-14: Default Instance Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
3	NV	Get	AtReference	BOOL	Actual speed at reference based on mode	Input assembly 71 Byte 0 Bit 7
4	NV	Get/Set	NetRef	BOOL	Requests speed or torque reference from network 0 = Local 1 = Ethernet	Output assembly 21 Byte 0 Bit 6
6		Get/Set	DriveMode	USINT	1 = Open loop speed 3 = Torque	Class 160 Instance 1 Attribute 600
7		Get	SpeedActual	INT	Best approx. drive speed in RPM/ 2SpeedScale	Input assembly 71 Bytes 2 and 3
8		Get/Set	SpeedRef	INT	Speed ref to drive in RPM/2SpeedScale	Output assembly 21 Bytes 2 and 3
9		Get	CurrentActual	INT	Actual motor phase current x 0.100A/ 2CurrentScale	Process data out 3 scaled to listed units
10		Get/Set	CurrentLimit	INT	Motor limit current x 0.100A/2CurrentScale	(Class 160, instance 1, attribute 129) scaled to listed units
11		Get	TorqueActual	INT	Actual torque in Newton-Meters/ 2TorqueScale	Process data out 4 scaled to listed units
12		Get/Set	TorqueRef	INT	Torq Ref in Newton-Meters/2TorqueScale	Process data in 1 scaled to listed units
13		Get	ProcessActual	INT	Units = % process/ 2ProcessScale	Process data out 1 scaled to listed units
15		Get	PowerActual	INT	Actual output power in Watts/2PowerScale	Process data out 5 scaled to listed units

Table 5-14: Default Instance Attributes (continued)

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
17		Get	OutputVoltage	INT	Output (drive) voltage in V/2VoltageScale	From measurement table, instance 1, attribute 10, scaled to listed units
18		Get/Set	AccelTime	UINT	Accel Time (scaling from attr 28) in msec/2TimeScale	(Class 16, instance 1, attribute 103) scaled to listed units
19		Get/Set	DecelTime	UINT	Decel Time (scaling from attr 28) in msec/2TimeScale	(Class 16, instance 1, attribute 104) scaled to listed units

Window into parameter space—Class 160 (0xA0)

This window provides access to all drive parameters addressable by an ID. By default, all attributes are R/W (get/set access) drive parameters of the UINT data type and are passed without scaling applied.

This can be changed on an attribute-by-attribute basis in the configuration file, config.ini. Also, two mappings from the (class, instance, attribute) triplet to the parameter ID are supported. The default mapping is for controller's that can support two-byte attributes, in which case, the mapping is simply ID = attribute. An alternate mapping for controller's that can only support one-byte attributes can be selected in the configuration file. With this alternate mapping, ID = attribute + (100 * [instance – 1]).

Note: As long as scaling is not applied, UINT and INT are equivalent.

Table 5-15: Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Value with Default Mapping	Value With Alternate Mapping
1		Get	Revision	UINT	Revision of the object	1	1
2		Get	Max. instance	UINT	Maximum instance number	1	20
3		Get	Number of instances	UINT	Number of instances	1	20
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	0	0
			Number of attributes	UINT	The number of optional attributes implemented		
			Optional attributes	ARRAY of UINT			
5		Get	Optional service list	Struct of	A list of optional instance services implemented		
			Number of services	UINT	The number of optional services implemented	1	1
			Optional services	ARRAY of UINT		0x10	0x10
6		Get	Max. class attribute ID	UINT		7	7
7		Get	Max. instance attribute ID	UINT		2000	100

Measurement table object—Class 170 (0xAA)

This object serves two purposes: the first is to provide access to the measurement table. Because this object is defined by the configuration file, it a convenient place to add attributes that are needed to support scaling, one entry more than is supported by Modbus/TCP only boards. Any additional attributes needed to support scaling that do not already exist in another object can be added starting with attribute 27. For example, if masking for CtrlFromNet and RefFromNet is enabled, two attributes need to be added.

Table 5-16: Default Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the object	1
2		Get	Max. instance	UINT	Maximum instance number	1
3		Get	Number of instances	UINT	Number of instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	①
			Number of attributes	UINT	The number of optional attributes implemented	26
			Optional attributes	ARRAY of UINT		1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26
5		Get	Optional service list	Struct of	A list of optional instance services implemented	1
			Number of services	UINT	The number of optional services implemented	1
			Optional services	ARRAY of UINT		
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		19 ①

① Attributes (0xAA, 0, 4) and (0xAA, 0, 7) will vary depending upon the definition of class 170 in config.ini. For the default configuration, the values for both are as listed here.

Table 5-17: Default Instance Attributes

Attribute ID	Access Rule	Name	Ethernet Data Type	Description of Attribute
1	Get	MotorTorque	INT	
2	Get	MotorPower	INT	
3	Get	MotorSpeed	INT	
4	Get	FreqOut	INT	
5	Get	FreqRef	INT	
6	Get	REMOTEIndication	USINT	
7	Get	MotorControlMode	USINT	
8	Get	ActiveFault_1	USINT	
9	Get	MotorCurrent	UINT	
10	Get	MotorVoltage	UINT	
11	Get	FreqMin	UINT	
12	Get	FreqScale	UINT	
13	Get	DCVoltage	UINT	
14	Get	MotorNomCurrent	UINT	
15	Get	MotorNomVoltage	UINT	
16	Get	MotorNomFreq	UINT	
17	Get	MotorNomSpeed	UINT	
18	Get	CurrentScale	UINT	
19	Get	MotorCurrentLimit	UINT	

Selectors object—Class 190 (0xBE)

The Selectors class is used to configure the I/O assemblies.

Table 5-18: Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the selectors class	1
2		Get	Max. instance	UINT	Maximum instance number of selectors instances	1
3		Get	Number of instances	UINT	Number of selectors instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	
			Number of attributes	UINT	The number of optional attributes implemented	6
			Optional attributes	ARRAY of UINT		3, 4, 5, 6, 7, 8
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	3
			Optional services	ARRAY of UINT		1, 2, 0x10
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		8

Table 5-19: Instance Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
3	*	Get/Set	InputAssemblySelector	UINT	Selects active input assembly	
4	*	Get/Set	OutputAssemblySelector	UINT	Selects active output assembly	
5	*	Get/Set	FBStatusType	UINT	Selects status source for input assemblies 111 and 121	See sections 7.1, and 7.6.18.1
6	*	Get/Set	FBControlType	UINT	Selects control destination for output assemblies 117 and 127	See sections 7.3, 7.6.7.1, and 7.6.9.1
7	*	Get/Set	FBSpeedActualType	UINT	Selects actual speed source for all input assemblies. Enables scaling of ActualSpeed for assemblies 20–25 if zero	See sections 7.2, and 7.6.18.1
8	*	Get/Set	FBSpeedRefType	UINT	Selects speed reference destination for all output assemblies. Enables scaling of SpeedRef for assemblies 70–75 if zero	See sections 7.4, 7.6.7.1, and 7.6.9.1

IP interface object—Class 245 (0xF5)

The IP interface object provides the mechanism to configure a device's IP network interface. Examples of configurable items include the device's IP address, network mask, and gateway address.

The physical port associated with the IP interface object shall be any port supporting the IP protocol. For example, a IP interface object may be associated with any of the following: an Ethernet 802.3 port, an ATM port, a serial port running SLIP, a serial port running PPP, and so on. The IP interface object provides an attribute that identifies the link-specific object for the associated physical port. The link-specific object is generally expected to provide link-specific counters, as well as any link-specific configuration attributes.

Each device shall support exactly one instance of the IP interface object for each IP-capable port on the module. A request to access instance 1 of the IP interface object shall always refer to the instance associated with the port over which the request was received.

Table 5-20: Class Attributes

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the IP interface class	1
2		Get	Max. instance	UINT	Maximum instance number of IP interface instances	1
3		Get	Number of instances	UINT	Number of TC/IP interface instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	0
			Number of attributes	UINT	The number of optional attributes implemented	
			Optional attributes	ARRAY of UINT		
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	2
			Optional services	ARRAY of UINT		1, 2
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		6

Table 5-21: Instance Attributes

Attribute ID	Access Rule	Name	Data Type	Description of Attribute	Semantics or Value
1	Get	Status	DWORD	Interface status	See "Status Instance Attribute"
2	Get	Configuration capability	DWORD	Interface capability flags	Bit map of capability flags. See "Configuration Capability Instance Attribute"
3	Get/Set	Configuration control	DWORD	Interface control flags	Bit map of control flags. See "Configuration Control Instance Attribute"
4	Get	Physical link object	STRUCT of:	Path to physical link object	See "Physical Link Object"
		Path size	UINT	Size of path	Number of UINTs in path
		Path	ARRAY of UINT	Logical segments identifying the physical link object	Class segment and instance segment. Maximum length is six UINTs (if 32-bit logical segments are used)
5	Get/Set	Interface configuration	STRUCT of:	Network interface configuration	See "Interface Configuration"
		IP address	UDINT	IP address	Value of 0 indicates no IP address has been configured. Otherwise, the IP address shall be set to a valid Class A, B, or C address and shall not be set to the loopback address (127.0.0.1)
		Network mask	UDINT	Network mask	Value of 0 indicates no network mask address has been configured
		Gateway address	UDINT	Gateway address	Value of 0 indicates no IP address has been configured. Otherwise, the IP address shall be set to a valid Class A, B, or C address and shall not be set to the loopback address (127.0.0.1)
		Name server	UDINT	Primary name server	Value of 0 indicates no name server address has been configured. Otherwise, the name server address shall be set to a valid Class A, B, or C address
		Name server 2	UDINT	Secondary name server	Value of 0 indicates no secondary name server address has been configured. Otherwise, the name server address shall be set to a valid Class A, B, or C address
		Domain name	STRING	Default domain name	ASCII characters. Maximum length is 48 characters. Must be padded to an even number of characters (pad not included in length)
6	Get/Set	Host name	STRING	Host name	ASCII characters. Maximum length is 64 characters. Must be padded to an even number of characters (pad not included in length). See clause 0

Table 5-22: Status Instance Attribute

The status attribute shall indicate the status of the network interface.

Value	Meaning
0x00000000	Network interface not configured
0x00000001	Network interface configured
0x00000002–0xFFFFFFFF	Reserved for future use

Table 5-23: Configuration Capability Instance Attribute

The configuration capability attribute is a bitmap that indicates the device's support for optional network configuration capability.

Bit(s)	Called	Definition
0	BOOTP client	1 (TRUE) shall indicate the device is capable of obtaining its network configuration via BOOTP
1	DNS client	1 (TRUE) shall indicate the device is capable of resolving host names by querying a DNS server
2	DHCP client	1 (TRUE) shall indicate the device is capable of obtaining its network configuration via DHCP
3	DHCP-DNS update	1 (TRUE) shall indicate the device is capable of sending its host name in the DHCP request as documented in Internet draft <draft-ietf-dhc-dhcp-dns-12.txt>
4	Configuration settable	1 (TRUE) shall indicate the interface configuration attribute is settable. Some devices, for example a PC or workstation, may not allow the interface configuration to be set via the IP interface object
5–31	Reserved	Reserved for future use and shall be set to zero

Table 5-24: Configuration Control Instance Attribute

The configuration control attribute is a bitmap used to control network configuration options.

Bit(s)	Called	Definition	
0–3	Startup configuration	Determines how the device shall obtain its initial configuration at start up.	0 = The device shall use the network configuration previously stored in nonvolatile memory
			1 = The device shall obtain its network configuration via BOOTP.
			2 = The device shall obtain its network configuration via DHCP upon startup
			3–15 = Reserved for future use
4	DNS enable	If 1 (TRUE), the device shall resolve host names by querying a DNS server	
5–31	Reserved	Reserved for future use and shall be set to zero	

When the value of the startup configuration bits is 0, a request to set the interface configuration attribute shall cause the device to store the contents of the interface configuration attribute in nonvolatile storage if supported by the device. Nonvolatile storage is not required; some low-end devices may choose to obtain network interface configuration via BOOTP or DHCP only.

The Startup Configuration bits shall not be set to 0 unless the Interface Configuration attribute has previously been set. Otherwise, the device could be rendered unable to communicate on the network.

Physical link object

This attribute identifies the object associated with the underlying physical port. There are two components to the attribute: a path size (in UINTs) and a path. The path shall contain a class segment and an instance segment that identifies the physical port object. The maximum path size is six (assuming a 32-bit logical segment for each of the class and instance).

The physical link object itself would typically maintain link-specific counters as well as any link-specific configuration attributes.

Interface configuration

This attribute contains the configuration parameters required to operate as a IP node. In order to prevent incomplete or incompatible configuration, the parameters making up the Interface configuration attribute cannot be set individually. To modify the Interface configuration attribute, the user should first get the interface configuration attribute, change the desired parameters then set the attribute.

The IP interface object shall apply the new configuration upon completion of the set service. If the value of the startup configuration bits (configuration control attribute) is 0, the new configuration shall be stored in nonvolatile memory. The device shall not reply to the set service until the values are safely stored to nonvolatile storage.

If initial configuration is to be obtained via BOOTP or DHCP, the interface configuration attribute components shall be all zeros until the BOOTP or DHCP reply is received. Upon receipt of the BOOTP or DHCP reply, the interface configuration attribute shall show the configuration obtained via BOOTP/DHCP.

Devices are not required to support the set service. Some implementations, for example those running on a PC or workstation, need not support setting the network interface configuration via the IP interface object.

Host name

The host name attribute contains the device's host name. The host name attribute is used when the device supports the DHCP-DNS update capability and has been configured to use DHCP upon start up. The DHCP-DNS update mechanism is specified Internet draft <draft-ietf-dhc-dhcp-dns-12.txt>, and is supported in Windows 2000. The mechanism allows the DHCP client to transmit its host name to the DHCP server. The DHCP server then updates the DNS records on behalf of the client. The host name attribute does not need to be set for the device to operate normally.

The value of the host name attribute, if it is configured, shall be used for the value of the FQDN option in the DHCP request. If the host name attribute has not been configured then the device shall not include the FQDN option in the DHCP request.

For devices that do not support the DHCP-DNS capability, or are not configured to use DHCP, then the host name can be used for informational purposes.

Get_Attribute_All Response

For class attributes, attributes are returned in numerical order, up to the last implemented attribute. This is an extension of the standard that is used because optional attributes 2–7 have been implemented. The standard is written for the case that only attribute 1 exists. “For class attributes, (because there is only one class attribute) class Attribute ID1 shall be returned.”

For instance attributes, attributes shall be returned in numerical order. The Get_Attribute_All reply shall be as follows:

Attribute ID	Size in Bytes	Contents
1	4	Status
2	4	Configuration capability
3	4	Configuration control
4	2	Physical link object, path size
	Variable, 12 bytes max.	Physical link object, path (if path size is non-zero)
5	4	IP address
	4	Network mask
	4	Gateway address
	4	Name server
	4	Secondary name server
	2	Domain name length
	Variable, equal to domain name length	Domain name
	1	Pad byte only if domain name length is odd
6	2	Host name length
	Variable, equal to host name length	Host name
	1	Pad byte only if host name length is odd

The lengths of the physical port object path, domain name, and host name are not known before issuing the Get_Attribute_All service request. Implementers shall be prepared to accept a response containing the maximum sizes of the physical link object path (6 UINTs), the domain name (48 USINTs), and host name (64 USINTs).

Set_Attribute_All Request

The instance Set_Attribute_All request contains the configuration control attribute, followed by the Interface configuration attribute.

Behavior

The behavior of the IP interface object shall be as illustrated in the state transition diagram below.

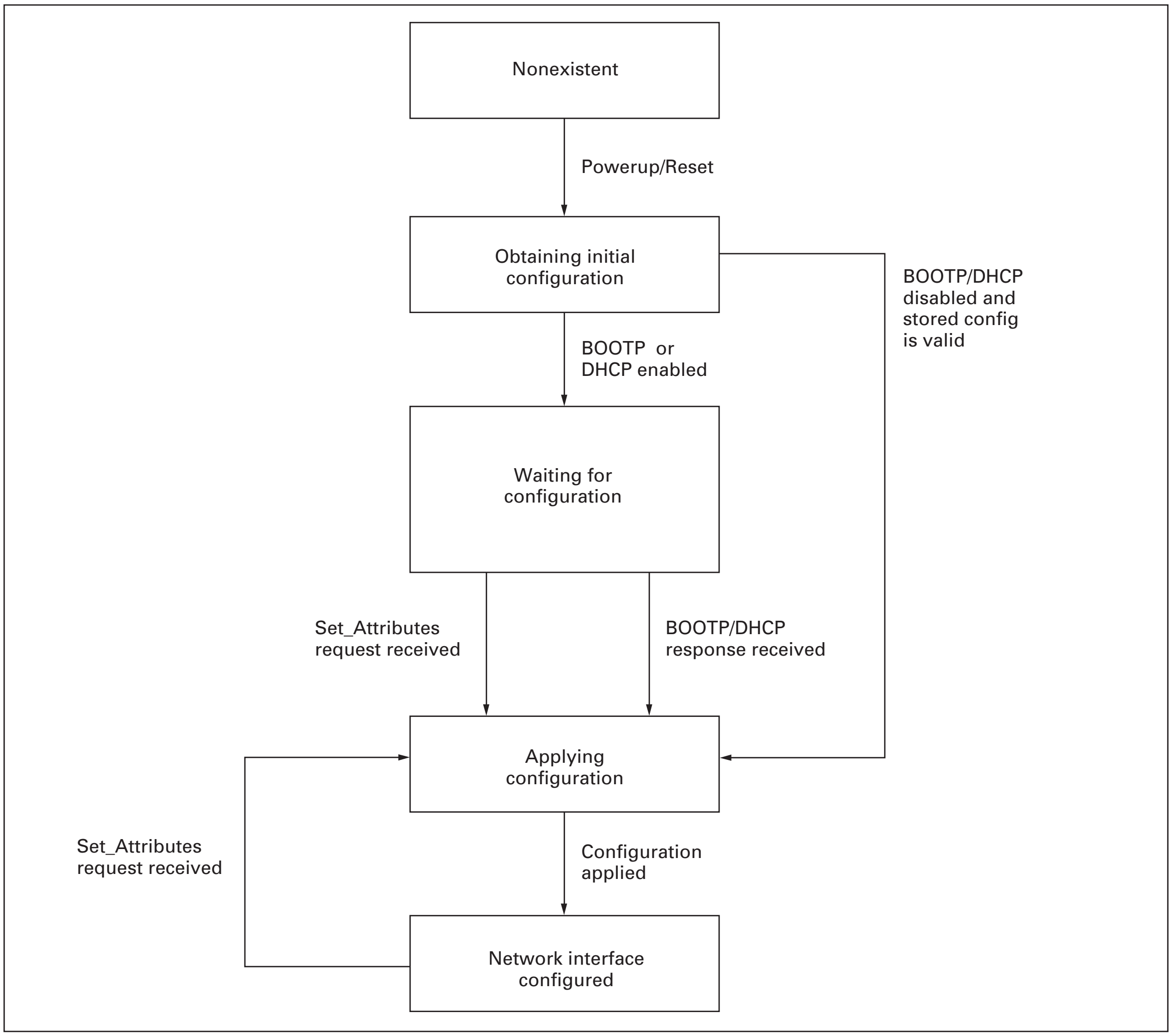


Figure 5-1: State Transition Diagram

Ethernet link object—Class 246 (0xF6)

Scope

The Ethernet link object maintains link-specific counters and status information for a physical Ethernet 802.3 port. Each device shall support exactly one instance of the Ethernet link object for each Ethernet port on the module. A request to access instance 1 of the Ethernet link object shall always refer to the instance associated with the port over which the request was received.

Table 5-25: Class Attributes

The Ethernet link object shall support the following class attributes.

Attribute ID	NV	Access Rule	Name	Ethernet Data Type	Description of Attribute	Semantics or Value
1		Get	Revision	UINT	Revision of the Ethernet link object class	2
2		Get	Max. instance	UINT	Maximum instance number of Ethernet link object instances	1
3		Get	Number of instances	UINT	Number of Ethernet link object instances	1
4		Get	Optional attribute list	Struct of	A list of optional instance attributes implemented	
			Number of attributes	UINT	The number of optional attributes implemented	2
			Optional attributes	ARRAY of UINT		4, 5
5		Get	Optional service list	Struct of	A list of optional instance services implemented	
			Number of services	UINT	The number of optional services implemented	1
			Optional services	ARRAY of UINT		1
6		Get	Max. class attribute ID	UINT		7
7		Get	Max. instance attribute ID	UINT		5

Table 5-26: Instance Attributes

The Ethernet link object shall support the following instance attributes.

Attribute ID	Access Rule	Name	Data Type	Description of Attribute	Semantics or Value
1	Get	Interface speed	UDINT	Speed of the interface	Speed in megabits per second (e.g., 10, 100, 1000, etc.)
2	Get	Interface flags	DWORD	Interface status flags	Bit map of interface flags. See "Interface Flags"
3	Get	Physical address	ARRAY of 6 USINTs	MAC layer address	See "Physical Address"
4	Get/ Get_and_Clear	Interface counters	STRUCT of:		See "Interface Counters"
		In octets	UDINT	Octets received on the interface	
		In ucast packets	UDINT	Unicast packets received on the interface	
		In NUcast packets	UDINT	Non-unicast packets received on the interface	
		In discards	UDINT	Inbound packets received on the interface but discarded	
		In errors	UDINT	Inbound packets that contain errors (does not include In discards)	
		In unknown protos	UDINT	Inbound packets with unknown protocol	
		Out octets	UDINT	Octets sent on the interface	
		Out ucast packets	UDINT	Unicast packets sent on the interface	
		Out NUcast packets	UDINT	Non-unicast packets sent on the interface	
		Out discards	UDINT	Outbound packets discarded	
		Out errors	UDINT	Outbound packets that contain errors	
6	Get/Get_and_Char	Media counters	STRUCT of:	Media-specific counters	See "Media Counters"

Note: The interface counters are an optional attribute; however, they shall be implemented if the media counters attribute is implemented.

Table 5-26: Instance Attributes (continued)

Attribute ID	Access Rule	Name	Data Type	Description of Attribute	Semantics or Value
6	Get/Get_and_Char	Alignment errors	UDINT	Frames received that are not an integral number of octets in length	See “Media Counters”
		FCS errors	UDINT	Frames received that do not pass the FCS check	
		Single collisions	UDINT	Successfully transmitted frames that experienced exactly one collision	
		Multiple collisions	UDINT	Successfully transmitted frames that experienced more than one collision	
		SQE test errors	UDINT	Number of times SQE test error message is generated	
		Deferred transmissions	UDINT	Frames for which first transmission attempt is delayed because the medium is busy	
		Late collisions	UDINT	Number of times a collision is detected later than 512 bit-times into the transmission of a packet	
		Excessive collisions	UDINT	Frames for which transmission fails due to excessive collisions	
		MAC transmit errors	UDINT	Frames for which transmission fails due to an internal MAC sublayer transmit error	
		Carrier sense errors	UDINT	Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame	
		Frame too long	UDINT	Frames received that exceed the maximum permitted frame size	
		MAC receive errors	UDINT	Frames for which reception on an interface fails due to an internal MAC sublayer receive error	

Note: The interface counters are an optional attribute; however, they shall be implemented if the media counters attribute is implemented.

Interface speed

The interface speed attribute shall indicate whether the interface is running at 10 Mbps, 100 Mbps, 1 Gbps, and so forth. The scale of the attribute is in Mbps, so if the interface is running at 100 Mbps then the value of interface speed attribute shall be 100. The interface speed is intended to represent the media bandwidth; the attribute shall not be doubled if the interface is running in full-duplex mode.

Interface flags

The interface flags attribute contains status and configuration information about the physical interface and shall be as follows:

Bit(s)	Called	Definition
0	Link status	Indicates whether or not the port is connected to an active network. 0 indicates an inactive link; 1 indicates an active link. The determination of link status is implementation specific. In some cases, devices can tell whether the link is active via hardware/driver support. In other cases, the device may only be able to tell whether the link is active by the presence of incoming packets.
1	Half/full duplex	0 indicates the port is running half duplex; 1 indicates full duplex. Note that if the link status flag is 0, then the value of the half/full duplex flag is indeterminate.
2-31	Reserved	Shall be set to zero

Physical address

The physical address attribute contains the interface's MAC layer address. The physical address is an array of octets. The recommended display format is "XX-XX-XX-XX-XX-XX," starting with the first octet.

Interface counters

The interface counters attribute contains counters relevant to the receipt of packets on the interface. These counters shall be as defined in RFC 1213 "MIB-II Management Information Base." The interface counters are an optional attribute; however, they shall be implemented if the media counters attribute is implemented.

Media counters

The media counters attribute contains counters specific to Ethernet media. These counters shall be as defined by RFC 1643, "Definitions of Managed Objects for Ethernet-Like Interface Types." The media counters are an optional attribute; however, if they are implemented the interface counters shall also be implemented.

Common services

All services

The Ethernet link object shall provide the following common services.

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0x01	Optional	Optional	Get_Attribute_All	Returns a predefined listing of this objects attributes (See the Get_Attribute_All response definition below.)
0x0E	Conditional	Required	Get_Attribute_Single	Returns the contents of the specified attribute

The Get_Attribute_Single shall be implemented for the class attribute if the class attribute is implemented.

Get_Attribute_All Response

For class attributes, attributes are returned in numerical order, up to the last implemented attribute. This is an extension of the standard that is used because optional attributes 2–7 have been implemented. The standard is written for the case that at most attribute 1 exists. “For class attributes, because there is only one possible attribute, the Get_Attribute_All response is the same as the Get_Attribute_Single response. If no class attributes are implemented, then no data is returned in the data portion of the reply.”

For instance attributes, attributes shall be returned in numerical order, up to the last implemented attribute.

Class-specific services

The Ethernet link object shall support the following class-specific services.

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0x4C	n/a	Conditional	Get_and_Clear	Gets then clears the specified attribute (interface counters or media counters)

The Get_and_Clear service shall only be implement if the interface counters and media counters are implemented.

Get_and_Clear Service

The Get_and_Clear service is a class-specific service. It is only supported for the interface counters and media counters attributes. The Get_and_Clear response shall be the same as the Get_Attribute_Single response for the specified attribute. After the response is built, the value of the attribute shall be set to zero.

September 2010

Appendix A—Table of supported services by object class

Table A-1: Supported Services by Object Class

		1	2	4	6	40	41	42	160	170	190	245	246
		1	2	4	6	27	29	2A	A0	AA	BE	F5	F6
Name	**#**	**Identity**	**Router**	**Assembly**	**Con Mgr**	**Motor**	**Control Supr**	**Drive**	**ID Window**	**Measurement**	**Selectors**	**TCP/IP**	**Ethernet**
Class services													
SVC_GET_ATTR_ALL	01	*	*		*	*	*	*	*	*	*	*	*
SVC_GET_ATTR_SINGLE	0E	*	*	*	*	*	*	*	*	*	*	*	*
SVC_CREATE	08												
SVC_DELETE	09												
SVC_RESET	05												
FIND_NEXT_OBJECT_INSTANCE	11												
Instance services													
SVC_GET_ATTR_ALL	01	*	*		*	*	*	*		*	*	*	*
SVC_GET_ATTR_SINGLE	0E	*	*	*	*	*	*	*	*	*	*	*	*
SVC_SET_ATTR_ALL	02					*	*	*			*	*	
SVC_SET_ATTR_SING10	10	*		*	*	*	*	*	*		*	*	*
SVC_GET_MEMBER	16												
SVC_SET_MEMBER	19												
SVC_INSERT_MEMBER	1A												
SVC_REMOVE_MEMBER	1B												
SVC_DELETE	09												
SVC_RESET	05	*					*						
APPLY_ATTRIBUTES	0D												
FWD_OPEN_CMD_CODE	54				*								
FWD_CLOSE_CMD_CODE	4E				*								
UNCONNECTED_SEND_CMD_CODE	52				*								
GET CONNECTION DATA	56												
SEARCH CONNECTION DATA	57												
GET CONNECTION OWNER	5A												
RESTORE	15												
SAVE	16												
ENETLINK_GET_AND_CLEAR	4C												*

Appendix B—Default Get All responses

Table B-1: Default Get All Responses

Name	#	Identity	Router	Assembly	Con Mgr	Motor	Control Supr	Drive	ID Window	Measurement	Selectors	TCP/IP	Ethernet
		1	2	4	6	40	41	42	160	170	190	245	246
		1	2	4	6	27	29	2A	A0	AA	BE	F5	F6
Class attributes—GET_ATTR_ALL_RESPONSE													
Revision	1	1		1	1	1	1	1	1	1	1	1	1
Max. instance	2			2	2	2	2	2	2	2	2	2	2
Number of instances	3				3	3	3	3	3	3	3	3	3
Optional attribute list	4		4			4	4	4	4	4	4	4	4
Optional service list	5		5			5	5	5	5	5	5	5	5
Maximum ID # class attributes	6	6	6		6	6	6	6	6	6	6	6	6
Maximum ID # instance attributes	7	7	7		7	7	7	7	7	7	7	7	7
Instance attributes—GET_ATTR_ALL_RESPONSE													
	1	1	1		1					1	1	1	1
	2	2	2		2					2	2	2	2
	3	3	3		3	3	3	3		3	3	3	3
	4	4	4		4		4	4		4	4	4	4
	5	5			5					5	5	5	5
	6	6			6	6	6	6		6	6	6	
	7	7			7	7	7	7		7	7		
	8	8			8		8	8		8	8		
	9	9			9	9	9	9		9	9		
	A	A					A	A		A	A		
	B						B	B		B	B		
	C					C	C	C		C	C		
	D						D	D		D	D		
	E						E			E	E		
	F					F	F	F		F			
	10							10		10			
	11							11		11			
	12							12		12			
	13							13		13			
	14							14		14			
	15							15		15			
	16							16		16			
	17							17		17			
	18							18		18			
	19							19		19			
	1A							1A		1A			
	1B							1B					
	1C							1C					
	1D							1D					

Appendix C—Process data variables for all-in-one application

This appendix lists how process data variables are defined for the all-in-one application. Other applications may define the process data variables differently.

Process data out (slave to master)

The fieldbus master can read the frequency converter's actual values using process data variables. All software applications use process data as follows:

Table C-1: Process Data Out Variables

ID	Data	Value	Unit	Scale
2104	Process data OUT 1	Output frequency	Hz	0.01 Hz
2105	Process data OUT 2	Motor speed	rpm	1 rpm
2106	Process data OUT 3	Motor current	A	0.1A
2107	Process data OUT 4	Motor torque	%	0.1%
2108	Process data OUT 5	Motor power	%	0.1%
2109	Process data OUT 6	Motor voltage	V	0.1V
2110	Process data OUT 7	DC link voltage	V	1V
2111	Process data OUT 8	Active fault code	—	—

The multipurpose control application has a selector parameter for every process data. The monitoring values and drive parameters can be selected using the ID number. Default selections are as in the table above.

Process data in (master to slave)

ControlWord, reference and process data are used with all-in-one applications as follows.

Table C-2: Basic, Standard, Local/Remote Control and Multistep Speed Control Applications

ID	Data	Value	Unit	Scale
2003	Reference	Speed reference	%	0.01%
2001	ControlWord	Start/stop command fault reset command	—	—
2004–2011	PD1–PD8	Not used	—	—

Table C-3: Multipurpose Control Application

ID	Data	Value	Unit	Scale
2003	Reference	Speed reference	%	0.01%
2001	ControlWord	Start/stop command fault reset command	—	—
2004	Process Data In 1	Torque reference	%	0.1%
2005	Process Data In 2	Free analogia INPUT	%	0.01%
2006–2011	PD3–PD8	Not used	—	—

Table C-4: PID Control and Pump and Fan Control Applications

ID	Data	Value	Unit	Scale
2003	Reference	Speed reference	%	0.01%
2001	ControlWord	Start/stop command fault reset command	—	—
2004	Process Data In 1	Reference for PID controller	%	0.01%
2005	Process Data In 2	Actual value 1 to PID controller	%	0.01%
2006	Process Data In 3	Actual value 2 to PID controller	%	0.01%
2007–2011	PD4–PD8	Not used	—	—

PLC programming

ControlLogix 5000

When using a ControlLogix PLC as an OPTCK master, you must first configure a compatible EtherNet/IP scanner, and then map ladder logic variables to the scanner. The following example is for a ControlLogix5550 with an ENET/B Ethernet bridge module. The ENET/B supports polled messaging. Some PLCs do not support polled messaging for EtherNet/IP. For example, the SLC500 only supports explicit messaging.

Right-click on I/O configuration and select "New Module." Select the 1756-ENET/B Ethernet Bridge (**Figure C-1**).

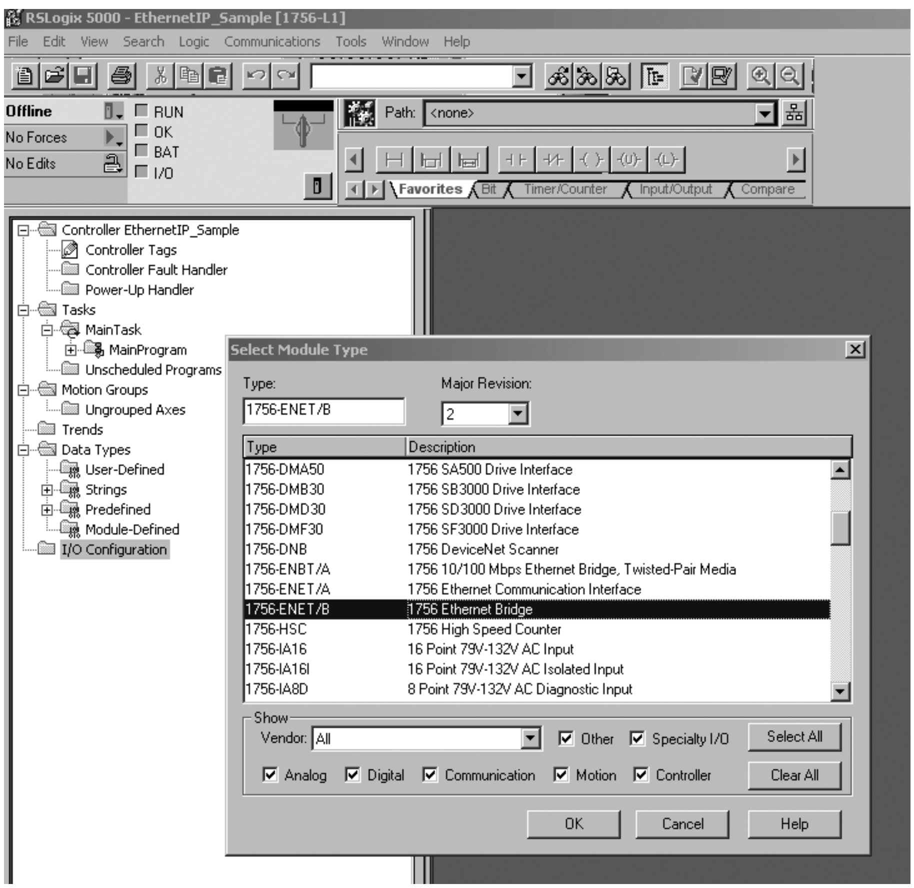


Figure C-1: 1756-ENET/B Ethernet Bridge

After the bridge module is added, a dialog box will appear requesting the configuration of the bridge module parameters. Enter a name and the IP address used by the bridge module on the first tab (**Figure C-2**). Select next and enter a polling interval for the bridge. A polling interval of 200 ms to 1000 ms is recommended.

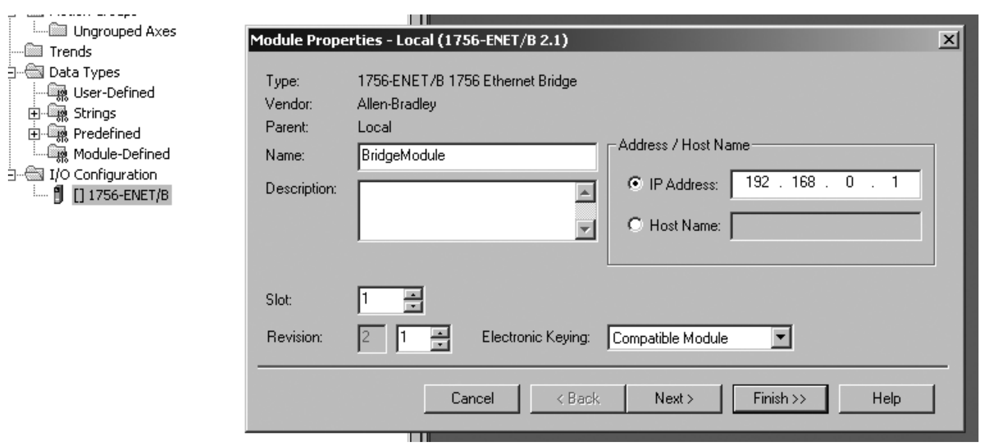


Figure C-2: Module Properties

The next step is to add a drive to the bridge module. Right click on the bridge module, and add a new Generic Ethernet Module (**Figure C-3**). Fill in the drive specific information. Be sure to select comm. Format INT. Do this before entering the connection parameters. In this example, the input and output assemblies match the default assembly numbers used by the OPTCK. Use a configuration assembly value of 1 with a length of zero (**Figure C-4**).

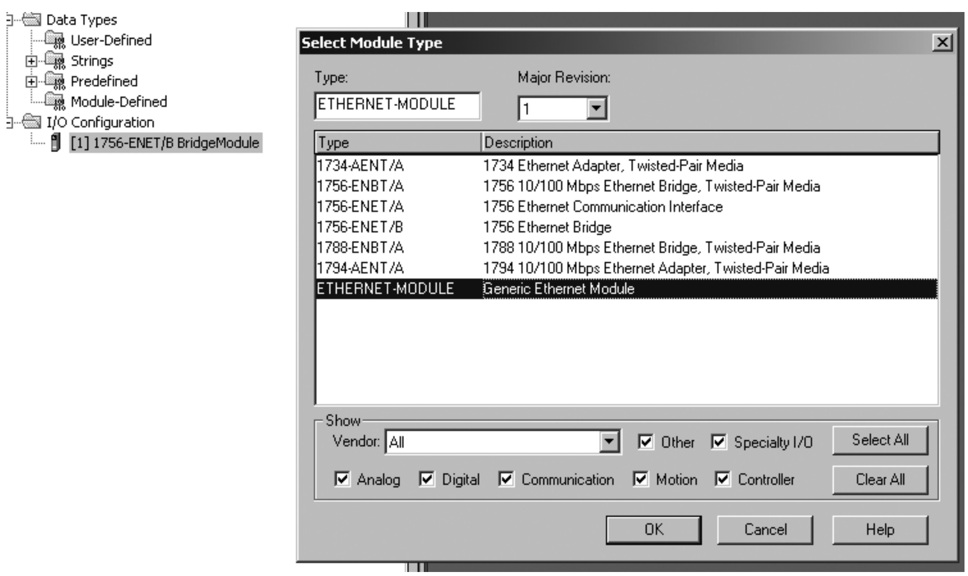


Figure C-3: Select Module Type

Add additional drive modules as needed, remembering to assign unique names and IP addresses to each module. Variable tags may then be viewed from the controller tags item in the property tree.

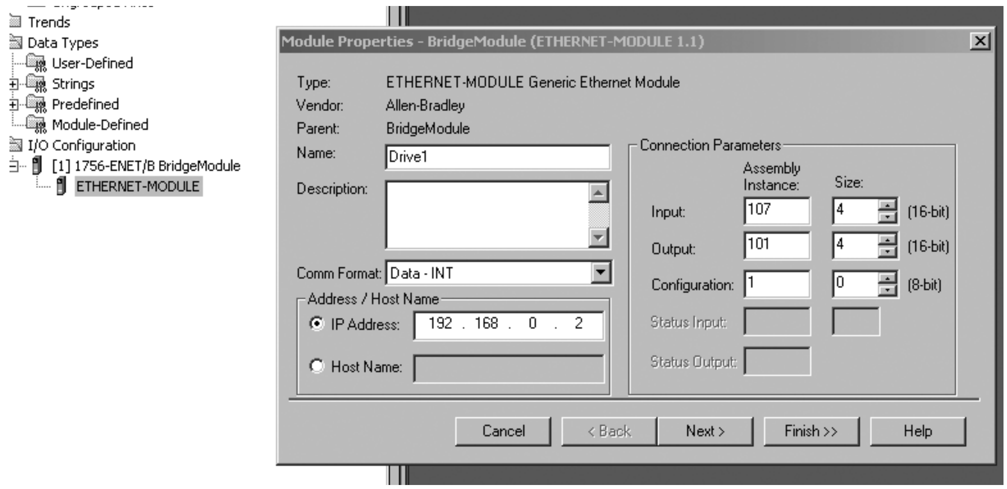


Figure C-4: Module Properties—BridgeModule

Tags from each drive may now be accessed using standard ladder instructions. For example, in **Figure C-6**, move instructions are used to move the speed and start commands for drive4. Notice that the use of INT data types in the scan list allow for simplified tag access. For example, the speed reference can be changed without having to use math operators to adjust the upper and lower bytes.

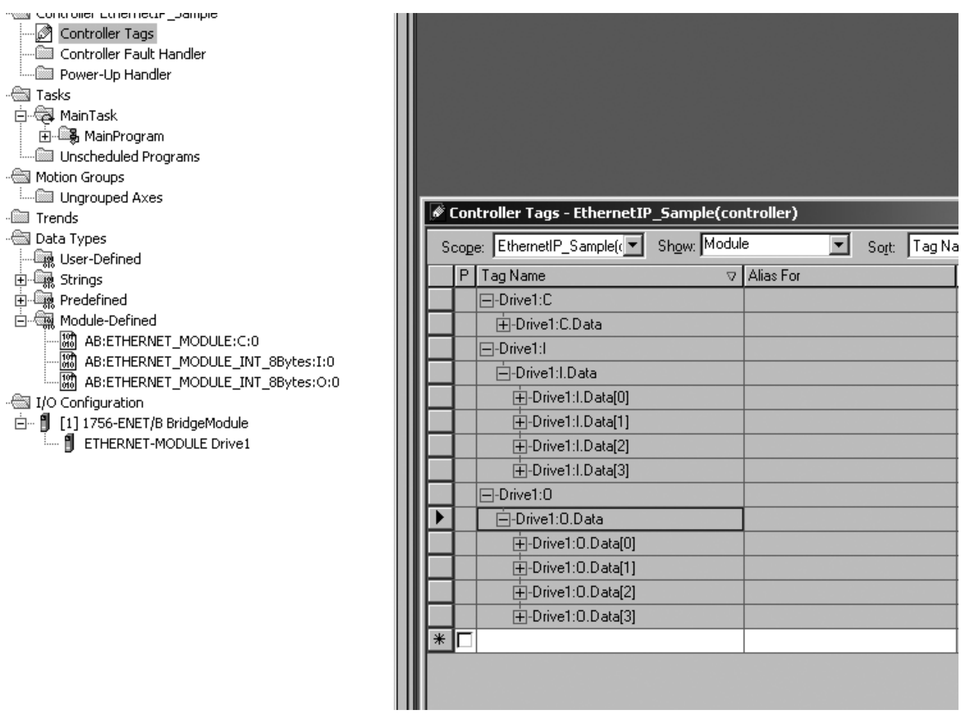


Figure C-5: Controller Tags—EthernetIP_Sample (Controller)

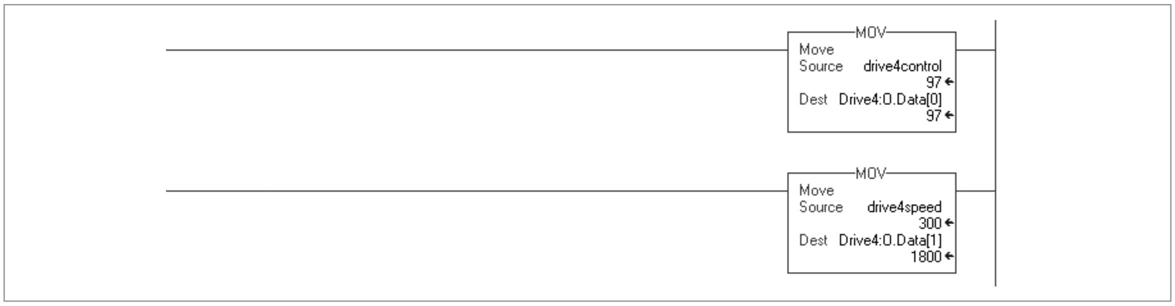


Figure C-6: Move Instructions

Explicit messages

The ladder logic in **Figure C-7** creates and sends an explicit message that changes the input and output assembly instance numbers used by the drive. It does this by using a message block, configured to send a Set Attribute Single CIP message. The configuration of the drive's input and output assemblies is done by changing attributes 3, and 4 of the selector class (0BE hex), instance 1. These items are used in the class, instance, and attribute argument fields of the configuration dialog in **Figure C-8**.

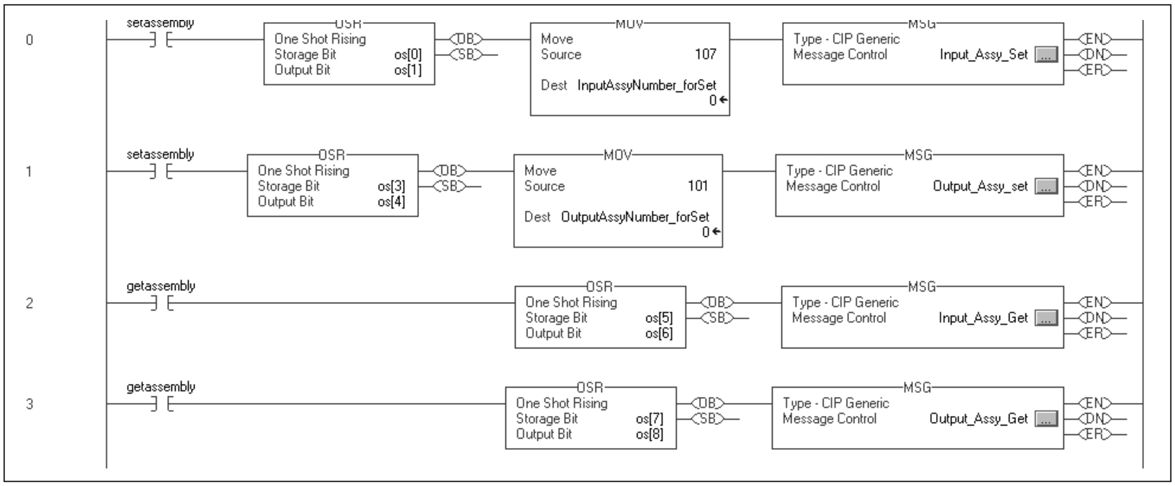


Figure C-7: Ladder Logic Message Blocks in RSLogix5000

Closing the SetAssembly contact fires a one shot, which in turn sets the variable InputAssyNumberForSet to a value of 107. This variable is used as the source element in the message configuration dialog (**Figure C-8**). You must also set the device path on the communication tab to the name of the drive you wish to send the message to, in this case Drive1. This device path determines which drive receives the explicit message.

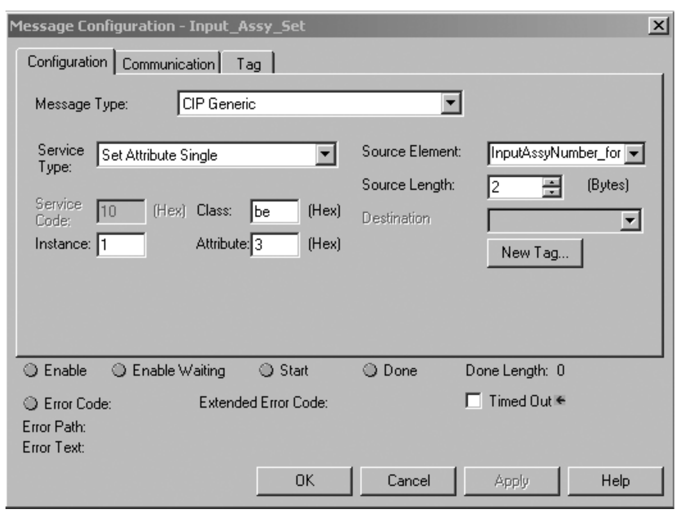


Figure C-8: Message Configuration for RSLogix5000

Forcing the **GetAssembly** contact fires a one-shot that triggers another message block that sends a **Get Attribute Single** message. The result of the get attribute single message is then placed in the destination element, **InputAssyNumberForGet**. This message response verifies that the drive has correctly received and responded to the previous setAttributeSingle message.

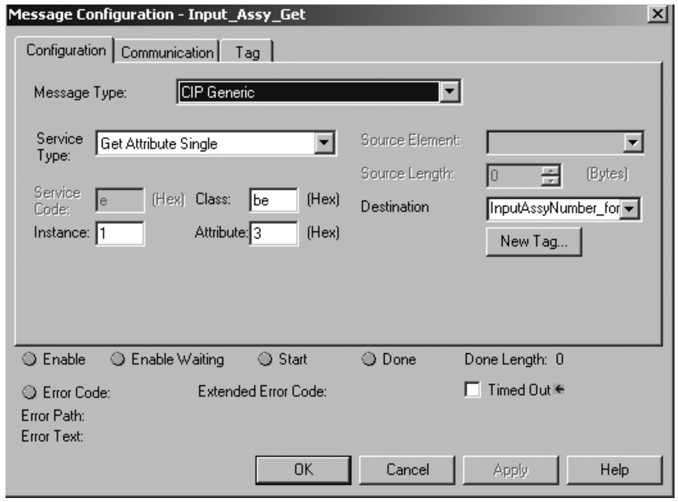


Figure C-9: Message Configuration

It's important to remember that explicit messages use PLC processor cycles that are best used to scan ladder logic. In the sample logic of figure A, the explicit message that sets the I/O assembly numbers is required to run only one time. Once the drive is configured to use a specific I/O assembly, it retains that information and the logic no longer has to run. This is the reason that a one-shot function block is used; it ensures that only one message is sent to the drive, and then will not execute again until the setAssembly contact opens a closes again.

Using explicit messages with I/O assemblies

Some PLCs, such as the Rockwell SLC500, do not allow for polled messaging over an EtherNet/IP. It is possible to transfer data using an I/O assembly as a template, but an explicit message must be used in place of the usual polled (implicit) message. The CIP specification provides for explicit access to the I/O assemblies via the "assembly object" class. The use of a "get attribute single" or "set attribute single" service to class 4, instance N, where N is the assembly number, attribute 3 (assembly data) is used. The same ladder logic structure used in figure A may be used, but a mechanism must be employed to periodically trigger the explicit messages. A timer may be used for this purpose. The timer should be set to a reasonable interval for reading information (~100 ms). The set service need only be called when control, speed change, or some other parameter write to the drive is required. A timer is still recommended to throttle messages, as event driven changes (such as a very slight speed change) may result in calling the message block logic too frequently. Excessive calls to message blocks can result in poor ladder logic performance.

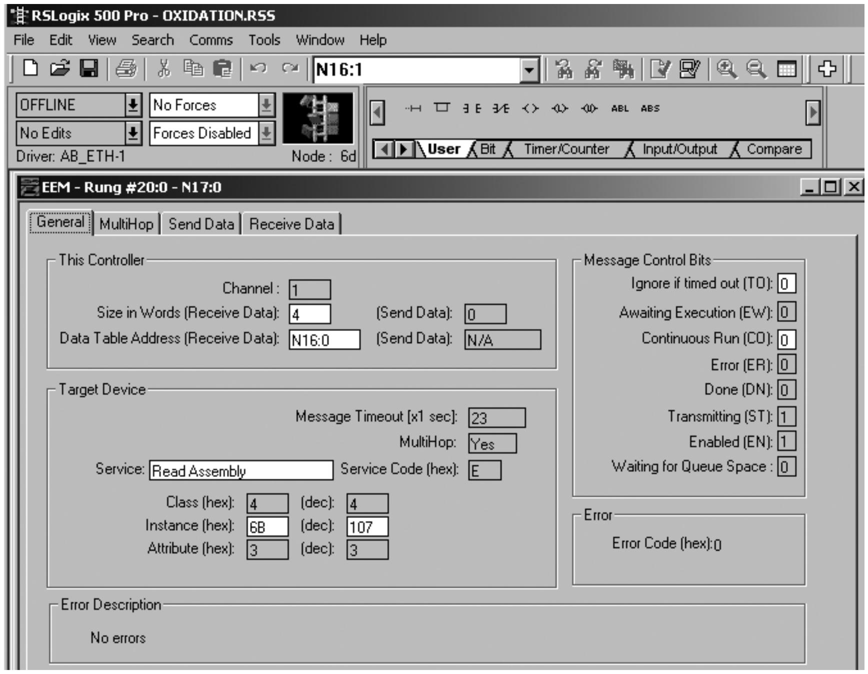


Figure C-10: RSLogix500 Configuration of Get Attribute Single

Example configuration dialogs for getting and setting RSLogix 500 message blocks are shown in **Figure C-10** and **Figure C-11. Figure C-10** shows configuration of the read assembly message block, which is used to get input information from the drives assembly number 107. **Figure C-11** shows the equivalent write assembly message block.

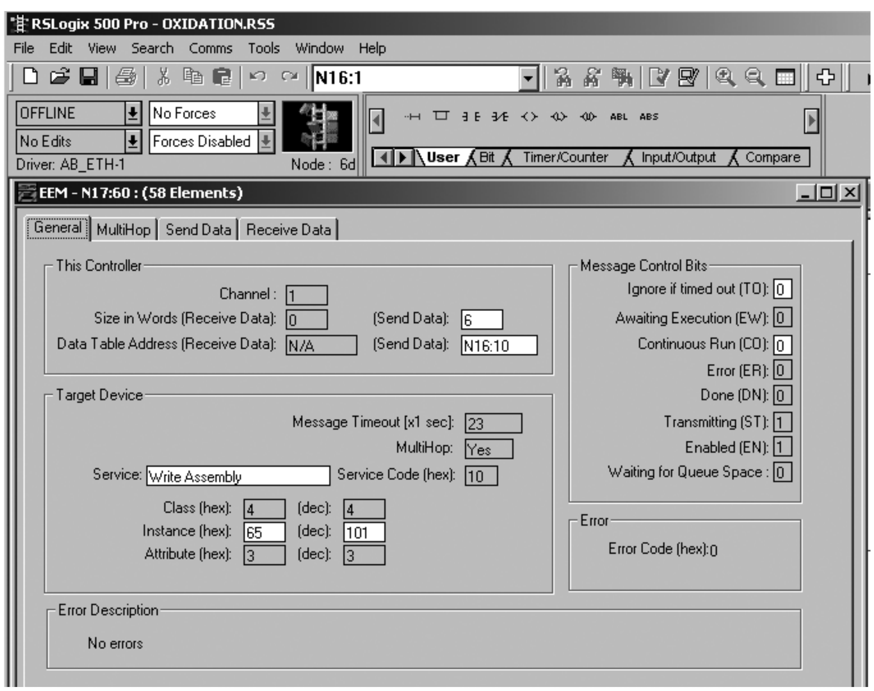


Figure C-11: RSLogix500 Configuration of Set Attribute Single

Eaton's Electrical Sector is a global leader in power distribution, power quality, control and automation, and monitoring products. When combined with Eaton's full-scale engineering services, these products provide customer-driven PowerChain™ solutions to serve the power system needs of the data center, industrial, institutional, public sector, utility, commercial, residential, IT, mission critical, alternative energy and OEM markets worldwide.

PowerChain solutions help enterprises achieve sustainable and competitive advantages through proactive management of the power system as a strategic, integrated asset throughout its life cycle, resulting in enhanced safety, greater reliability and energy efficiency. For more information, visit www.eaton.com/electrical.

Eaton Corporation
Electrical Sector
1111 Superior Ave.
Cleveland, OH 44114
United States
877-ETN-CARE (877-386-2273)
Eaton.com

Printed in USA
Publication No. MN04012010E / Z10054
September 2010

Powering Business Worldwide

PowerChain Management®