AKD®2G

EtherNet/IP Communications Manual

EtherNet/IP™

Edition: Revision A, September 2021

Part Number 907-200008-00

For safe and proper use, follow these instructions. Keep for future use.

Record of Document Revisions:

Revision	Remarks
A, 09/2021	Launch version

- AKD is a registered trademark of Kollmorgen Corporation
- ControlNet is a registered trademark or ODVA, Inc.
- DeviceNet is a registered trademarks of ODVA, Inc.
- EtherNet/IP is a registered trademark of ODVA, Inc.
- Studio 5000 is a registered trademark of Rockwell Automation

Technical changes which improve the performance of the device may be made without prior notice.

Table of Contents

1 General

1.1 About this Manual

This manual describes the installation, setup, range of functions, and software protocol for the AKD®2G EtherNet/IP™ product series. All AKD2G EtherNet/IP drives have built-in EtherNet/IP functionality - an additional option card is not required.

A digital version of this manual (PDF format) is available on the DVD included with your drive. Manual updates can be downloaded from the Kollmorgen® website.

Related documents for the AKD2G series include:

- Using AKD2G EtherNet/IP with Studio 5000™: This manual provides an easy start guide for Studio 5000 programs, as well as a reference to the sample add-on instructions.
- AKD2G Installation Manual: This manual provides instructions for installation and drive setup.

Additional documentation:

- The CIP Networks Library Volume 1: Common Industrial Protocol. ODVA, Inc.
- The CIP Networks Library Volume 2: EtherNet/IP Adaptation of CIP. ODVA. Inc.

1.2 Overview

EtherNet/IP is an industrial communication protocol based on TCP/IP and UDP/IP. It is used as a high level network for industrial automation applications. EtherNet/IP shares a common data structure with DeviceNet® and ControlNet®, but built on Ethernet as a physical medium. The protocol uses two communication channels:

Explicit Messages:

The AKD2G supports parameter access using Explicit Messaging.

Explicit Messages are used for reading or writing values on-demand, for drive configuration, and occasional reads or writes of parameter values. Communication rates depend on the particular parameter or command, and can range from approximately 5ms to 5s.

Explicit Messages allow you to access a single parameter value at a time independent of cyclic messaging and the I/O Assemblies. The desired parameter is selected by specifying the class object number, instance number, and attribute number in an explicit message.

I/O Assembly Messages:

I/O Assembly Messages are data structures usually sent on a timed cyclic basis for drive control and status. The data structure is predetermined and only certain values can be read and written.

I/O Assembly Messaging is used for most motion control. Control bits in a Command Message are used to enable the amplifier, do a controlled stop of the motor, initiate motion, and initiate stored motion block programs. Command Messages can also set the Target Position, Target Velocity, Acceleration, Deceleration, and Torque Setpoints. Status bits in a Response Message display error states and the general state of the amplifier. Response Messages can also display the actual position, commanded position, actual velocity and torque.

Typically, Explicit Messaging is used to configure the amplifier and I/O Assemblies are used to control movement. Most programmable logic controllers (PLCs) will support both types of messaging simultaneously. Check the documentation for your controller to determine which messaging types are supported.

I/O Assembly Messages combine many control and status bits into command and Response Messages. They are less versatile than Explicit Messages (only certain parameters are accessible), but several control values may be changed within one message. For this reason, Explicit Messaging is better for configuration and I/O Assembly Messaging is better for motion control.

- The Position Controller Object and Position Controller Supervisor Objects are used to set the operational mode (torque, velocity, or position), home, and configure motion.
- Additional configurations over EtherNet/IP can be made using the Parameter Object which provides access to drive parameters, including parameters accessible through WorkBench.
- Motion sequences may be pre-programmed into the amplifier using the AKD2G motion tasking feature. Once the Motion Task sequence has been configured, it may be executed with the Command Assembly Message Block Number field and Start Block bit.

I/O Assembly Messages and Explicit Messages may be used simultaneously.

1.2.1 Terminology

Several terms in this document have synonyms. For clarity, this document prefers the first (bolded) term.

Response Assembly:	Also known as "Input Assembly" or "Target to Originator Connection".
Command Assembly:	Also known as "Output Assembly" or "Originator to Target Connection".
Expected Packet Rate:	Also known as "Requested Packet Interval".

AKD2G defines one Command Assembly (sent from the controller to the drive) and one Response Assembly (sent from the drive to the controller).

1.3 AKD2G EtherNet/IP Features

1.3.1 Supported Features

AKD2G follows the ODVA standard for EtherNet/IP. It provides necessary standard objects, as well as certain vendor-specific objects. CIP-Motion (for real-time multi-axis synchronized motion control) is not supported.

The following general drive features are supported through EtherNet/IP:

NOTE

Kollmorgen recommends maintaining the drive, its setup, and its configuration via WorkBench, not via the PLC and over EtherNet/IP.

- Drive setup and configuration
  - Access a full range of drive parameters
  - Configure parameters through user programs
  - Setup Motion Tasks
- Position Control
  - Setup and trigger Homing
  - Point-to-point moves
  - Absolute and Relative Motion
  - Configure and execute Motion Task sequences
- Velocity Control
  - Initiate Jog Moves
  - Initiate Velocity Setpoint Mode
- Torque Control
  - Write torque commands
  - Read actual torque
- Status and actual values
  - Monitor drive status (enabled, faulted, homed, in position, in motion, etc) on every cycle
  - Monitor actual position and Velocity on every cycle
  - Monitor any drive value using Explicit Messaging on-demand

1.3.2 Expected Packet Rate

The Expected Packet Rate (EPR) is also called the Requested Packet Interval (RPI). EtherNet/IP's fastest supported cyclic rate on the AKD2G is 1 millisecond.

1.3.3 Connection Port

The AKD2G uses the fieldbus ports X11 and X12 on top of the AKD2G. These ports act as a network switch and can be daisy chained to connect to other AKD2Gs or other EtherNet/IP devices.

1.3.4 Network Topology

The AKD2G can be connected to an EtherNet/IP network in two manners:

- as another node on the network in a line topology
- as another node on the network in star topology (using a switch)

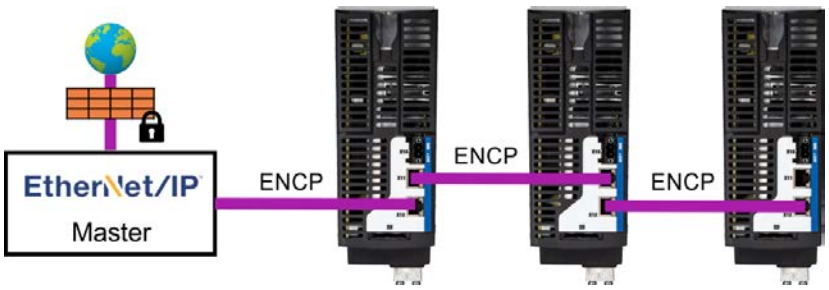
1.3.5 EtherNet/IP

AKD2G with connectivity option **I** can be connected as slaves to a EtherNet/IP network by using RJ45 connectors X11 (Port 2) and X12 (Port 1). The communication status is indicated by the built-in connector LEDs.

	Conn.	Name	Function
	X11 Port 2	NS	Green - On = Connected Green - Flashing = No connections Red - On = Duplicate IP Red Flashing = Connection Timeout Red/Green Flashing = Self-test Off = No power, No IP Address
	X11 Port 2	Link/ Activity	On/Blinking: Physical link/Data Traffic on. Static off: No link.
	X12 Port 1	MS	Green - On = Device Operational Green - Flashing = Standby Red - On = Major Fault Red Flashing = Minor Fault Red/Green Flashing = Self-test Off = No power
	X12 Port 1	Link/ Activity	On/Blinking: Physical link/Data Traffic on. Static off: No link.

Bus topology example (EtherNet/IP)

We suggest to use Kollmorgen ENCP cables. For more possible system solutions refer to the WorkBench Online Help.



--- / ---

2 Setup

2.1 Setting an IP Address in the Drive

The IP settings for the EtherNet/IP fieldbus ports are setup separately from the WorkBench service port and are managed with the commands EIP.IPADDRESS, EIP.IPSUBNET, EIP.IPGATEWAY, and EIP.IPMODE. Once the commands are set, they must be applied with EIP.IPRESET or DRV.NVSAVE. The drive should now be able to communicate on the EtherNet/IP network.

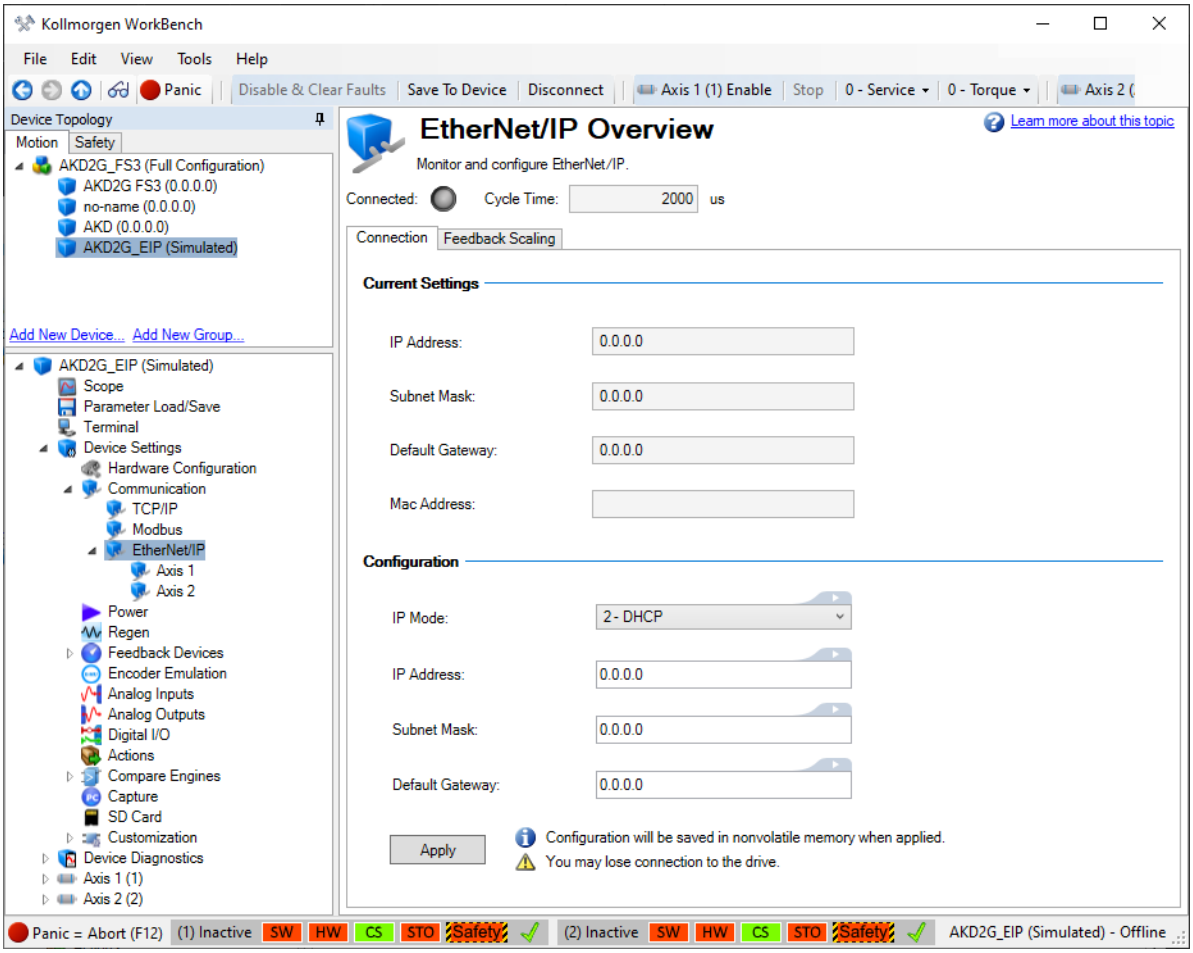
DRV.INFO and EIP.INFO can be used to view the currently active settings. The active settings may differ from the commands if the settings have not been applied with EIP.IPRESET or DRV.NVSAVE.

Setting an IP Address using WorkBench

The fields under Current Settings are automatically populated with the drive's current IP settings.

The Configuration section allows you to set the EtherNet/IP address on the drive.

1. Set IP Mode to Static, BOOTP, or DHCP.
   If using a Static address, you will need to set the IP Address, Subnet Mask, and Default Gateway.
2. Click Apply.



2.2 Controller Setup

Some controllers request an Electronic Data Sheet (EDS file) for configuring an EtherNet/IP node. The AKD2G EtherNet/IP EDS file can be found on the Kollmorgen website and on the product DVD.

The IP address of the controller must be set to the same subnet as the AKD2G.

The controller will typically need to be setup to know the IP address of the AKD2G. The process required will vary by controller.

2.3 Setting Expected Packet Rate in the Controller

The controller is responsible for setting the Expected Packet Rate (EPR) at which the AKD2G and controller will send cyclic messages.

The fastest supported cyclic rate for EtherNet/IP on AKD2G is 1 millisecond.

If the rate is set to too short of a time, communication may timeout between the drive and controller, resulting in fault F7000 Fieldbus Communication Lost. In this case, the EPR should be set to a larger value.

3 Communication Profile

3.1 Explicit Messaging

On-Demand Messaging

Explicit Messages allow you to access a single parameter value at a time independent of cyclic messaging and the I/O Assemblies. The desired parameter is selected by specifying the class object number, instance number, and attribute number in an explicit message.

TIP

See Overview for an overview of Explicit versus I/O Assembly Messaging.

3.1.1 Supported Services

- 0x10 – Write Value
- 0x0E – Read Value

3.1.2 Supported Objects

AKD2G supports a number of standard and vendor-specific objects for motion control. See Drive Objects for more information.

Parameter Object

Class Code: 0x64	
Instance	For axis-specific parameters, the instance number references the axis ID. For array parameters, the array index can be referenced by multipling the array index by 100 and adding it to the instance. For any other parameters, the instance is ignored and may be set to 1. For example, to reference Motion Task 3 for Axis 2, the instance would be 302. To reference Motion Task 0 for Axis 1, the instance would be 1.
Description	The parameter object gives direct access to amplifier configuration parameters.
Attribute	The attribute number references the desired parameter. See Appendix B: EtherNet/IP Objects List for a list of available parameters. See Drive Objects for more information.

Position Controller Object

Position Controller Object Attributes can be accessed (GET/SET) by sending a message using Class Code 0x66, instance number, and attribute ID or alternatively using the Get Attribute or Command Type 0x1B - Set Attribute of Position Controller Object methods in this manual.

Class Code: 0x66	
Instance	The instance is the Axis Number (1 or 2) of the attribute to GET/SET.
Description	The Position Controller Object is used to set the operating mode (Torque, Velocity, Position), configure motion profiles, and initiate movement.
Attribute	See Drive Objects for more information. A list of Position Controller Object attributes can be found in Appendix A: EtherNet/IP Objects and Attributes

AKD2G also supports the required standard objects for EtherNet/IP communication. Typically the controller will automatically configure these objects, and the user program will not need to directly use them:

- Identity
- Message Router
- Assembly
- Connection Manager
- TCP/IP
- Ethernet Link

3.1.3 Data Types

The table below describes the data type, number of bytes, minimum range, and maximum range.

Data Type	Number of Bytes	Minimum Value	Maximum Value	Abbreviation
Boolean	1	0(false)	1(true)	Bool
Short Integer	1	-16	15.875	S8
Unsigned Short Integer	1	0	31.875	U8
Integer	2	-4096	4095.875	S16
Unsigned Integer	2	0	8191.875	U16
Double Integer	4	-2^{28}	$2^{28}-1$	S32
Unsigned Double Integer	4	0	$2^{28}-1$	U32

3.1.4 Error Codes

The following error codes may be returned in response to an Explicit Message.

Error	Error Code
Not Settable	0x0E
Attribute Not Supported	0x14
Service Not Supported	0x08
Class Not Supported	0x16
Value is Out of Range	0x09

3.2 I/O Assembly Messaging

Cyclic Messaging

Cyclic data exchange includes the transmission and reception of data values between the master and the drive. These data values include setpoint values (Position Setpoint, Velocity Setpoint or Control Word) and actual values (Actual Position Value, Actual Velocity or Status Word).

The data commands and responses contain multiple values in pre-defined data structures, called assemblies. Assemblies are transmitted on a timer according to the Expected Packet Rate (EPR), which is also referred to as the Requested Packet Interval (RPI).

AKD2G defines one Command Assembly (sent from the controller to the drive) and one Response Assembly (sent from the drive to the controller).

TIP

See Overview for an overview of Explicit versus I/O Assembly Messaging.

3.2.1 Controller Configuration

Controllers must be configured with the correct assembly information in order to open an I/O connection to the AKD2G. This setup will differ depending on the controller type.

In addition to configuring the IP address of the AKD2G in the controller setup, the following values must be configured:

Connection Parameter	Assembly Instance	Size	Run/Idle Header?
Input	102	128 bytes	No
Output	101	128 bytes	Yes
Configuration	100	0 bytes	N/A
Requested Packet Interval (RPI)	1ms or greater if simultaneous WorkBench use is required		
I/O Connection Type	Multicast, Class 1 Type		

3.2.2 Drive Assembly

Drive Assemblies map assemblies to axes. Each Response Assembly is concatenated, one per axis to make up a Drive Response Assembly. Similarly, Command Assemblies are concatenated to make up a Drive Command Assembly.

Assembly	Offset at which bytes start	
	Axis 1	Axis 2
Drive Response Assembly	offset 0	offset 64
Drive Command Assembly	offset 0	offset 64

3.2.3 Command Assembly

Command Assemblies contain a Control Word and several fields used for setting values, requesting response data, and commanding moves. A Command Assembly may be used to send one data command at a time (Target Position, Target Velocity, Acceleration, Deceleration, or Torque). The command type is specified in the Command Type field. A Command Assembly also specifies a Response Type which requests a particular kind of data in the Response Assembly.

- A Command Assembly may contain both a Command Type and a Response Type to transmit a command and to request a particular response in the same assembly.
- A valid Command Type is required to be set in each Command Assembly. Data outside the allowed range will result in an Error Response Assembly.
- The amplifier must be Homed before motion is begun in Position Mode. Failure to home the amplifier will result in a warning (W6002 Homing is needed) that must be cleared before amplifier operation can continue.
- Each Command Assembly byte location is offset by the Axis Offset, defined above in the Drive Assembly description.

3.2.3.1 Command Assembly Data Structure

Axis 1 Byte	Axis 2 Byte	Data	Comment	Command
0	64	Control Word	The Control Word contains bits for enabling, moving, and handshaking with the drive.	AXIS#.EIP.CMD.CONTROL1
1	65	Block #	The Block Number is used to start a particular Motion Task, in combination with the Start Block bit in the Control Word.	AXIS#.EIP.CMD.BLOCKNUM
2	66	Command Type	Specifies the desired command to execute, such as Set Position or Set Parameter.	AXIS#.EIP.CMD.CMDTYPE
3	67	Response Type	Specifies the desired response data to return in the Response Assembly.	AXIS#.EIP.CMD.RSPTYPE
4-7	68-71	Data	The command data for most Command Types*	AXIS#.EIP.CMD.DATA
8-11	72-75	Position	Position data for Command Type 6 (Position Move)*	AXIS#.EIP.CMD.P
12-15	76-79	Velocity	Velocity data for Command Type 6 (Position Move) and 7 (Jog)*	AXIS#.EIP.CMD.V
16-19	80-83	Acceleration	Acceleration data for Command Type 6 (Position Move) and 7 (Jog)*	AXIS#.EIP.CMD.ACC
20-23	84-87	Deceleration	Deceleration data for Command Type 6 (Position Move) and 7 (Jog)*	AXIS#.EIP.CMD.DEC
24-27	88-91	Parameter/ Attribute Data	Command Data for Command Type 0x1B (Set Position Controller Attribute) and 0x1F (Set Parameter)*	AXIS#.EIP.CMD.DATA2
28	92	Attribute to Get	Index of desired Position Controller Attribute value to return in the Response Assembly bytes 24-27 for Axis 1; bytes 88-91 for Axis 2)	AXIS#.EIP.CMD.ATTRIBUTE
29-31	93-95	Reserved		AXIS#.EIP.PADBYTE
32-63	96-127	Command Dynamic Map	Data for user mappable parameters. See AXIS#.EIP.DYNAMICCMDMAP.	AXIS#.EIP.DYNAMICCMDDATA

*Least significant byte first for all data fields

TIP

See Using WorkBench to Dynamically Map EtherNet/IP Parameters in the online help for instructions on how to easily map parameters.

3.2.3.2 Control Word

Byte 0 for Axis 1; Byte 64 for Axis 2 of the Command Assembly.

Bit	Definition	Description
0	Load/Start	This bit is used for data handshaking between the controller and amplifier.
1	Start Block	Executes a Motion Task sequence previously generated and stored in the drive. Enter the starting block number into the Block Number field (byte 1) and transition this bit to High (1). The Load/Start flag must be Low (0) while transitioning Start Block.
2	Relative	This bit is used in only in Position Mode. This bit indicates whether a move executed with Command Type 1 (Target Position) or 6 (Position Move) should be absolute (0) or incremental (1).
3	Direction	Used with Command Type 0x07 - Jog Move in both Velocity Operation Mode or Position Operation Mode and Command Type 0x08 - Velocity Setpoint in Velocity Mode. Positive direction = 1 and Negative direction = 0.
4	Smooth Stop	Setting this bit causes the amplifier to decelerate to a stop. The smooth stop is the same as AXIS#.STOP . It will decelerate based on the DEC settings at the time of motion initiation.
5	Hard Stop	Setting this bit causes the amplifier to execute a Controlled Stop. The Enable bit must be cleared and reset in order to enable motion again. Controlled Stop configuration and settings can be found in WorkBench's Enable/Disable screen under Settings for the respective axis.
6	Reserved	
7	Enable	Setting this bit enables the amplifier.

To transmit a command to the amplifier:

1. Set the Command Type and load data into the data fields.
2. Toggle Load/Start to High (1).

   The amplifier will accept data only when Load/Start transitions from 0 to 1.

If the Command Type matches the operating mode (see list below for matches) the amplifier will start motion when the data is loaded. When the data has been loaded successfully, the amplifier will set the Load Complete response flag to High (1).

- Target Position or Position Move in Position Mode
- Target Velocity or Jog in Velocity Mode
- Torque in Torque Mode

3.2.3.3 Control Word 2

This is an additional user mappable control word that can be mapped using AXIS#.EIP.DYNAMICCMDMAP.

Bit	Definition	Description
0	Clear user map error	This bit clears an error bit that's set in Status Word 3 when one of the dynamic objects fails to set the associated object value.
1	Reserved	
2	Reserved	
3	Fault acknowledge	This bit will clear faults on the associated axis on a rising edge.
4-7	Reserved	

3.2.3.4 Command Type 0x05 - Torque

This command type is used to change the target torque. This can only be used in Torque Mode. Motion will begin as soon as the value is loaded.

1. Write 0x05 to the Command Type field (byte 2 for Axis 1; byte 66 for Axis 2) of the Command Assembly.
2. Put drive in Torque Mode by sending a message to Position Controller class 0x66, Instance 1 or 2 where Instance=Axis Number, Attribute 3 Operation Mode.
3. The Command Source must be Fieldbus.
4. Load the desired torque value in bytes 4-7 for Axis 1; bytes 68-71 for Axis 2.
5. Set the Load/Start bit to begin the move.

Torque values are in milliamps [mA].

3.2.3.5 Command Type 0x06 - Position Move

This command type is used to start a trajectory (Position Mode only) of the specified distance, velocity, acceleration and deceleration. Since all command values are sent to the drive in a single assembly, this is the preferred way motion should be commanded.

The trajectory can be Absolute or Relative, depending on the value of the Relative bit. The move will begin as soon as the command is loaded.

The Position Move is loaded into an internal move and can be viewed within WorkBenchusing terminal and AXIS#.FBUS Parameters.

1. Write 0x06 to the Command Type field (byte 2 for Axis 1; byte 66 for Axis 2) of the Command Assembly.
2. Put drive in Position Mode by sending a message to Position Controller class 0x66, Instance 1 or 2 where Instance=Axis Number, Attribute 3 Operation Mode.
3. The Command Source must be Fieldbus.
4. Load Target Position, Velocity, Acceleration and Deceleration into bytes 8-23 for Axis 1; bytes 72-87 for Axis 2 (See Command Assembly Data Structure).
5. Set the Load/Start bit to begin the move.

Position values are scaled according to AXIS#.EIP.POSUNIT. Velocity and Acceleration values are scaled according to AXIS#.EIP.PROFUNIT.

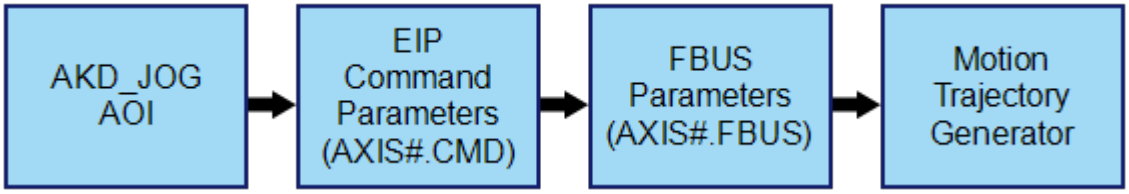
3.2.3.6 Command Type 0x07 - Jog Move

This command type is used to change the Target Velocity, Acceleration and Deceleration in Velocity and Position Mode. The Direction bit sets the desired direction. The move will begin as soon as the Target Velocity is loaded and the Load/Start bit turns from OFF (0) to ON (1); rising edge.

1. Write 0x07 to the Command Type field (byte 2 for Axis 1; byte 66 for Axis 2) of the Command Assembly.
2. Put drive in Velocity Mode or Position Mode by sending a message to Position Controller class 0x66, Instance 1 or 2 where Instance=Axis 1 or 2, Attribute 3 Operation Mode.
3. Load Target Velocity, Acceleration and Deceleration into bytes 12-23 for Axis 1; bytes 76-87 for Axis 2 (See Command Assembly Data Structure).
4. Set the Load/Start bit to begin the move.

- Velocity and Acceleration values are scaled according to AXIS#.EIP.PROFUNIT.
- The Command Source must be Fieldbus.
- The Operation Mode can be set to either Velocity or Position.

Jog Mode (Velocity Mode Operation)



The AXIS#.EIP parameter values are passed to the drive as follows:

AXIS#.EIP.CMD.ACC → AXIS#.FBUS.ACC

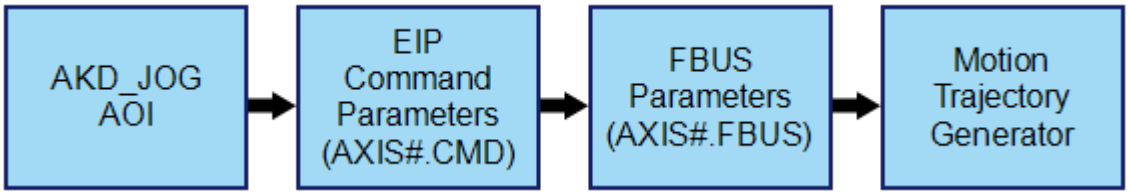
AXIS#.EIP.CMD.DEC → AXIS#.FBUS.DEC

AXIS#.EIP.CMD.V → AXIS#.VL.CMD

AXIS#.EIP.CMD.P → 0.

Note that neither AXIS#.FBUS.V or AXIS#.VL.CMDU are utilized.
The Direction bit 3 in the Control Word (AXIS#.EIP.CMD.CONTROL1) sets the Direction 0 = Reverse; 1 = Forward and sets the sign of the AXIS#.VL.CMD.

Jog Mode (Position Mode Operation)



In Position operation mode the AXIS#.EIP parameter values are passed to the corresponding AXIS#.FBUS parameters.

AXIS#.EIP.CMD.ACC → AXIS#.FBUS.ACC

AXIS#.EIP.CMD.DEC → AXIS#.FBUS.DEC

AXIS.EIP.CMD.V → AXIS#.FBUS.V

AXIS#.EIP.CMD.P → 0.

TIP

- No “P”, position value passed from EIP (AXIS#.EIP.CMD.P=0)
- AXIS#.FBUS.P's magnitude is set to the equivalent scaling of 32 revolutions.
- The signed value (positive or negative) is set by Direction bit 3 in the Control Word (AXIS#.EIP.CMD.CONTROL1)
- The Control Word sets the Direction (0 = Reverse; 1 = Forward) and the sign of the AXIS#.VL.CMD.

This is used internally as follows:

In Position Mode, on execution AXIS#.FBUS.P is set by the Jog Mode Command Type 0x07 to a value to -2097152.000 Counts when the Direction bit is 0 and +2097152.000 Counts when the Direction bit is 1. This value assumes WorkBench is set to the default EIP position units (65536 EIP counts/revolution). The result is a continuous move at the target velocity set by AXIS#.FBUS.V which is sourced from AXIS#.EIP.CMD.V as shown above.

3.2.3.7 Command Type 0x08 - Velocity Setpoint

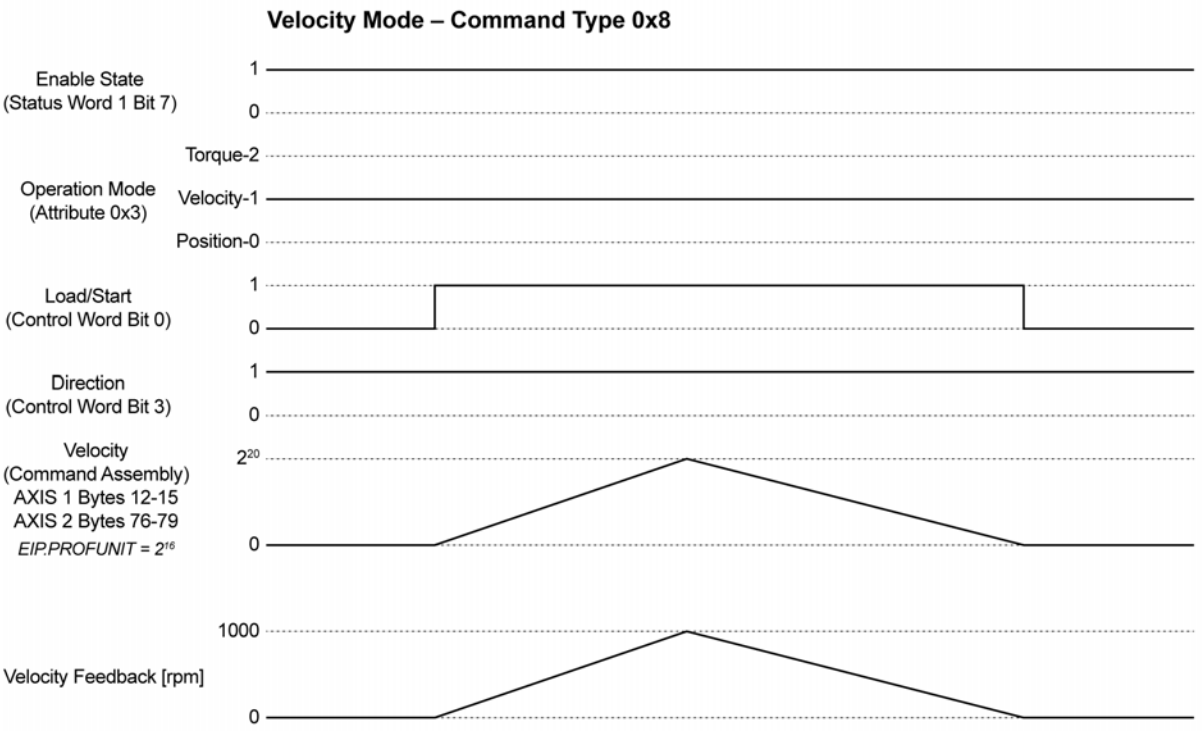
This command type is used to change the Target Velocity every cyclic cycle. The operation mode must be set to Velocity and the drive enabled, otherwise this command will return an error using Response Assembly Response Type (byte 3 for Axis 1; byte 67 for Axis 2) = 0x14 and Data (bytes 4-7 for Axis 1; bytes 68-71 for Axis 2). The command source must be Fieldbus. The drive will follow commanded velocity while the Load/Start bit is set (level triggered).

NOTE

For this command type, the behavior is different than the other command types. There is no data handshake using the Load/Start bit in the Control Word (byte 0 for Axis 1; byte 64 for Axis 2) in the Command Assembly.

In this mode, when the Load/Start bit is set to High (1), the drive will follow the commanded velocity from Velocity field (bytes 12-15 for Axis 1; bytes 76-79 for Axis 2). When the Load/Start bit is Low (0), the drive will continue to follow the last commanded velocity (ignores changes to the Velocity field) until you change one of the following:

- Hard Stop bit is set to High (1) in the Control Word (byte 0 for Axis 1; byte 64 for Axis 2).
- Smooth Stop bit is set to High (1) in the Control Word (byte 0 for Axis 1; byte 64 for Axis 2).
- Enable bit is cleared (0) in the Control Word (byte 0 for Axis 1; byte 64 for Axis 2).
- Fault occurs.
- Or some other condition that would stop motion occurs (STO, etc.)



Setup motion:

1. Write 0x8 to the Command Type field (byte 2 Axis 1; byte 66 Axis 2) in the Command Assembly.

   NOTE

   In this mode, the drive ignores Acceleration (bytes 16-19 Axis 1; bytes 80-83 Axis 2) and Deceleration (bytes 20-23 Axis 1; bytes 84-87 Axis 2).
2. Setup scaling for Velocity, Acceleration, and Deceleration. Send a message to Position Controller class 0x66, Instance 1 or 2 where Instance = Axis Number, Attribute 0x05: Profile Units.
3. Setup Acceleration one time after power cycle. Send a message to Position Controller class 0x66, Instance 1 or 2 where Instance = Axis Number, Attribute 0x08: Acceleration. Or use keyword AXIS#.FBUS.ACC. This parameter can be saved to non-volatile memory so that it is restored after a power cycle.
4. Setup Deceleration one time after power cycle. Send a message to Position Controller class 0x66, Instance 1 or 2 where Instance = Axis Number, Attribute 0x09: Deceleration. Or use keyword AXIS#.FBUS.DEC. This parameter can be saved to non-volatile memory so that it is restored after a power cycle.
5. Put drive in Velocity Mode. Send a message to Position Controller class 0x66, Instance 1 or 2 where Instance = Axis Number, Attribute 0x03: Operation Mode with a value 1: Velocity Mode.
6. The Command Source must be Fieldbus.
7. Enable the drive. This is accomplished by setting the Enable bit, which is bit 7 of the Control Word in the Command Assembly.
8. To enable errors: Write 0x14 to Response Type field (byte 3 for Axis 1; byte 67 for Axis 2) in the Command Assembly. Then read the Data field (bytes 4-7 for Axis 1; bytes 68-71 for Axis 2) in the Response Assembly.
9. Set your initial Target Velocity using the Velocity field of the Command Assembly (bytes 12-15 for Axis 1; bytes 76-79 for Axis 2).
10. Enable following of Velocity Setpoint. This is accomplished by setting the Load/Start bit which is bit 0 of the Control Word in the Command Assembly.

3.2.3.8 Command Type 0x09 – Motion Task

This Command Type is used to configure a Motion Task (Position mode only) using the specified Block/Motion Task number, Position, Velocity, Acceleration, and Deceleration.

The Position Move is loaded into the Motion Task specified by the block number in the Command Assembly and can be viewed within WorkBench.

1. Write 0x09 to the Command Type field (byte 2 for Axis ; byte 66 for Axis 2) of the Command Assembly.
2. Put drive in Position Mode by sending a message to Position Controller class 0x66, Instance 1 or 2 where Instance = Axis Number, Attribute 3 Operation Mode.
3. The Command Source must be Fieldbus.
4. Load Block Number, Target Position, Velocity, Acceleration and Deceleration into bytes 8-23 for Axis 1 or bytes 72-87 for Axis 2 (See Command Assembly Data Structure).
5. Set the Load/Start bit to load the data.
6. Optionally set the Start Block bit to start the move immediately after loading the data.

Position values are scaled according to AXIS#.EIP.POSUNIT. Velocity and acceleration values are scaled according to AXIS#.EIP.PROFUNIT.

NOTE

This command type will not override the specific Motion Task's movement type of Absolute or Relative. AXIS#.MT.CNTL must be set ahead of time to the desired movement type.

3.2.3.9 Command Type 0x1B - Set Attribute of Position Controller Object

This command type is used to set a value in the Position Controller Object, such as for configuring and triggering a home move. See the chapter on Drive Objects and Appendix A: EtherNet/IP Objects and Attributes for a listing of available attributes in this object.

1. Write 0x1B to the Command Typefield (byte 2 for Axis 1; byte 66 for Axis 2) of the Command Assembly.
2. Load the desired Attribute number (least significant byte first) into bytes 4-5 for Axis 1 or bytes 68-69 for Axis 2.
3. Load the desired Parameter/Attribute Data value into bytes 24-27 for Axis 1 or bytes 88-91 for Axis 2 (See the Data Structure section).
4. Set the Load/Start bit to set the value in the drive.

3.2.3.10 Command Type 0x1F - Read or Write Parameter Value

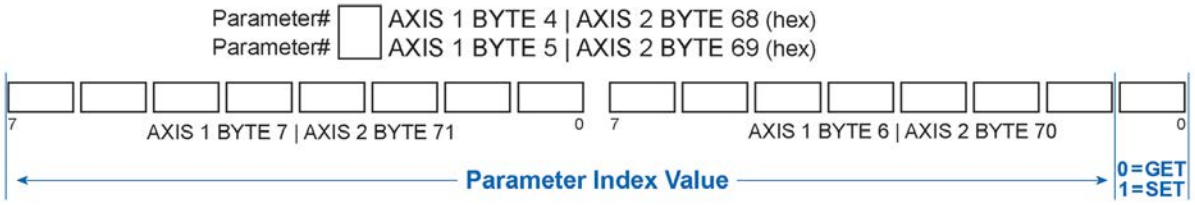
This command type is used to configure or read any parameter in the drive. See Appendix B: EtherNet/IP Objects List for a listing of parameter indexes, data types, and scaling.

Use this command to either read or write the desired parameter. To determine whether this is a read or write command, use byte 6 Bit 0 for Axis 1 or byte 70 Bit 0 for Axis 2.

Some parameters can take a long time to execute. When the command has completed, the Load Complete status bit will be set in the response, or else an Error Response Assembly will be returned.

1. Write 0x1F to the Command Type field (byte 2 for Axis 1; byte 66 for Axis 2) of the Command Assembly.
2. Load the desired parameter Index into (first half of the Data field, least significant byte first) into bytes 4-5 for Axis 1 or bytes 68-69 Axis 2.
3. Set byte 6 bit 0 for Axis 1 or byte 70 bit 0 for Axis 2 according to whether you wish to read or write the parameter. 0 = read, 1 = write.
4. For Axis 1 set byte 6 bits 1-7 and byte 7 bits 0-7 to 0 for non-array type parameters or to the binary equivalent for the parameter array index for array-type parameters.
   For Axis 2 set byte 70 bits 1-7 and byte 71 bits 0-7 to 0 for non-array type parameters or to the binary equivalent for the parameter array index for array-type parameters.
5. If writing a parameter, load the Parameter/Attribute Data value into bytes 24-27 for Axis 1 or bytes 88-91for Axis 2.
6. Set the Load/Start bit to execute the command.
7. If reading a parameter, the value will be returned in bytes 24-27 for Axis 1 or bytes 88-91 for Axis 2 of the response.

Definition:



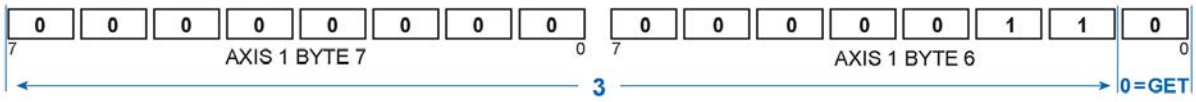
Example:

AKD2G AXIS1 Motion Task: 6307, AXIS#.MT.P, Position, 4 Byte Signed, ReadWrite, 2, 0 to 31
[Parameter 6307 (decimal) = 0x18A3 (hex)]

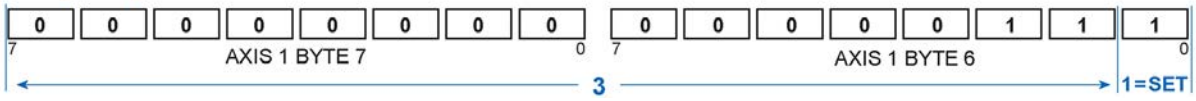
Motion Task 3 (Array Index = 3)

Parameter# A3 BYTE 4
Parameter# 18 BYTE 5

Get (Read) Parameter:



Set (Write) Parameter:



3.2.3.11 Get Attribute

The Get Attribute field operates differently from the Command Types listed above, as it does not make use of the Command Type field or require Load/Start to be set.

To read an attribute of the Position Controller in each cycle, set byte 28 (Attribute to Get) for Axis 1 or byte 92 for Axis 2 (Attribute to Get) to the desired attribute number. The data will be returned in each Response Assembly in bytes 24-27 Axis 1; bytes 88-91 Axis 2. On execution of the command, the Attribute to Get number will be mirrored in the Response Assembly in byte 28 (Attribute to Get) for Axis 1 or byte 92 for Axis 2.

1. Load the desired attribute number of the Position Controller Object into byte 28 for Axis 1; byte 92 for Axis 2, Attribute to Get.
2. The value will be updated each communication cycle in bytes 24-27 for Axis 1 or bytes 88-91 for Axis 2 of the Response Assembly.

NOTE

Attribute to Get and Command Type 0x1F Read Parameter Value both use bytes 24-27 for Axis 1; bytes 88-91 for Axis 2 of the Response Assembly. When using command 0x1F to read a parameter, set the Attribute to Get field to 0.

3.2.4 Response Assembly

In I/O Assembly Messaging, the amplifier transmits a Response Assembly back to the controller. The Response Assembly has a number of predefined status words and data values. In addition, it can contain one data value which is selected by the Response Type field of the Command Assembly.

3.2.4.1 Response Assembly Data Structure

Each Response Assembly byte location is offset by the Axis Offset defined above in the Drive Assembly description.

Axis 1 Byte	Axis 2 Byte	Data	Comment	Command
0	64	Status Word 1	Various status bits	AXIS#.EIP.RSP.STATUS1
1	65	Executing Block #	The index of the Motion Task which is currently being executed	AXIS#.EIP.RSP.BLOCKNUM
2	66	Status Word 2	Various status bits	AXIS#.EIP.RSP.STATUS2
3	67	Response Type	Specifies the Response Type of this assembly, echoing the Response Type set in the Command Assembly.	AXIS#.EIP.RSP.RSPTYPE
4-7	68-71	Data	The response data for most Response Types*	AXIS#.EIP.RSP.DATA
8-11	72-75	Position	Actual Position*	AXIS#.EIP.RSP.P
12-15	76-79	Velocity	Actual Velocity*	AXIS#.EIP.RSP.V
16-19	80-83	Motion Status	Status bits. This provides the status word AXIS#.MOTIONSTAT.	AXIS#.EIP.RSP.MOTIONSTAT
20-23	84-87	Reserved		AXIS#.EIP.PADBYTE
24-27	88-91	Parameter/ Attribute Data	Response Data for Command Type 0x1F (Set Parameter) and the Attribute to Get*	AXIS#.EIP.RSP.DATA2
28	92	Attribute to Get	Mirrors the Attribute to Get from the Command Assembly. If non-zero, the data will be in the Parameter Data field.	AXIS#.EIP.RSP.ATTRIBUTE
29-31	93-95	Reserved		AXIS#.EIP.PADBYTE
32-63	96-127	Response Dynamic Map	Data for user mappable parameters. See AXIS#.EIP.DYNAMICRSPMAP.	AXIS#.EIP.DYNAMICRSPDATA

** Least significant byte first for all data fields*

Status 1, Status 2, Actual Position, Actual Velocity, and Motion Status data are updated in every Response Assembly.

Data in bytes 4-7 for Axis 1; bytes 68-71 for Axis 2 will be updated depending on the value of the Response Type.

Parameter/Attribute Data in bytes 24-27 for Axis 1; bytes 88-91 for Axis 2 will be updated when Attribute to Get is non-zero or when a Get Parameter command is completed.

TIP

See Using WorkBench to Dynamically Map EtherNet/IP Parameters in the online help for instructions on how to easily map parameters.

3.2.4.2 Status Word 1

Byte 0 for Axis 1; Byte 64 for Axis 2 of the Response Assembly.

Bit	Definition	Description
0	In Motion	This bit indicates whether a trajectory is in progress (1) or has completed (0). This bit is set immediately when motion begins and remains set for the entire motion.
1	Block in Execution	When set, this bit indicates the axis is running a Motion Task. The Index, also called the Block Number, of the Motion Task can be monitored in the Response Assembly's Executing Block # (byte1 for Axis 1; byte 65 for Axis 2) in Response Assembly.
2	In Position	This bit indicates whether or not the motor is on the last targeted position. (1 = On Target)
3	General Fault	This bit indicates whether or not a fault has occurred.
4	Current Direction	This bit reflects the actual direction of motion.
5	Homed	This bit is set when the drive has been successfully Homed.
6	Reserved	
7	Enable State	This bit reflects the Enable State of the amplifier.

3.2.4.3 Status Word 2

Byte 2 for Axis 1; Byte 66 for Axis 2 of the Response Assembly.

Bit	Definition	Description
0	Reserved	
1	Pos HW Limit	This bit indicates the state of the Positive Hardware Limit Input.
2	Neg HW Limit	This bit indicates the state of the Negative Hardware Limit Input.
3	Pos SW Limit	This bit indicates when the position is greater than or equal to the Positive Software Limit Position.
4	Neg SW Limit	This bit indicates when the position is less than or equal to the Negative Software Limit Position.
5-6	Reserved	
7	Load Complete	This bit indicates that the command data contained in the Command Message has been successfully loaded into the device. Used for handshaking between the controller and amplifier – see Data Handshake.

3.2.4.4 Status Word 3

This is an additional user mappable status word that can be mapped using AXIS#.EIP.DYNAMICRSPMAP.

Bit	Definition	Description
0	Hw Enable	This bit is set if the axis' Hw Enable has voltage.
1	STO active	This bit is set if voltage is applied to both STO inputs.
2	User map error	This bit is set if a dynamic object fails to set a value (i.e.: out of range, invalid value, etc).
3	Warning present	This bit is set if a warning is active.
4-7	Reserved	

3.2.4.5 Response Type 0x05 - Actual Torque

This I/O Response Assembly is used to return the Actual Torque (current) of the motor in milliarms. Data will be received in the Data field, bytes 4-7 for Axis 1; bytes 68-71 for Axis 2. Set Response Type = 0x05 (byte3 for Axis 1; byte 67 for Axis 2) in the Command Assembly to read this value.

3.2.4.6 Response Type 0x14 - Command/Response Error

This I/O Response identifies an error that has occurred and will always be returned as the result of an invalid Command Assembly. The Response Type field of the Response Assembly echoes the matching field from the previous Command Assembly with one exception. In the case of an invalid Command Assembly, the Response Assembly Type field of the Response Assembly will be set to 0x14 and error codes will be returned in the Data field.

Error Code (hex)	Additional Code (hex)	EtherNet/IP Error
Axis 1 = Byte 4 Axis 2 = Byte 68	Axis 1 = Byte 5 Axis 2 = Byte 69	
0	FF	NO ERROR
2	FF	RESOURCE_UNAVAILABLE
5	FF	PATH_UNKNOWN
5	1	COMMAND_AXIS_INVALID
5	2	RESPONSE_AXIS_INVALID
8	FF	SERVICE_NOT_SUPP
8	1	COMMAND_NOT_SUPPORTED
8	2	RESPONSE_NOT_SUPPORTED
9	FF	INVALID_ATTRIBUTE_VALUE
B	FF	ALREADY_IN_STATE
C	FF	OBJ_STATE_CONFLICT
D	FF	OBJECT_ALREADY_EXISTS
E	FF	ATTRIBUTE_NOT_SETTABLE
F	FF	ACCESS_DENIED
10	FF	DEVICE_STATE_CONFLICT
11	FF	REPLY_DATA_TOO_LARGE
13	FF	NOT_ENOUGH_DATA
14	FF	ATTRIBUTE_NOT_SUPP
15	FF	TOO_MUCH_DATA
16	FF	OBJECT_DOES_NOT_EXIST
17	FF	FRAGMENTATION_SEQ_ERR
20	FF	INVALID_PARAMETER

3.2.4.6.1 Data Field - On Command Error

The Data Field (bytes 4-7 Axis 1; bytes 68-71 for Axis 2 of the Response Assembly) has the following contents when the Response Type field is set to 0x14.

Axis 1 Byte	Axis 2 Byte	Data	Comment
4	68	Error Code	Refer to table in Response Type 0x14 - Command/Response Error.
5	69	Additional Code	Refer to table in Response Type 0x14 - Command/Response Error.
6	70	Command Type	Echoes byte 2 (Axis 1) or byte 66 (Axis 2) from the Command Assembly Data Structure.
7	71	Response Type	Echoes byte 3 (Axis 1) or bye 67 (Axis 2) from the Command Assembly Data Structure.

3.2.5 Data Handshake

Data handshaking is used to transmit data commands with I/O Assembly Messaging. To transmit a command to the amplifier, set the Command Type and load data into the data fields, then toggle the Load/Start bit to High (1). The amplifier will accept data only when Load/Start transitions from 0 to 1. If the data is loaded successfully, the amplifier will set the Load Complete response flag to High (1). Load Complete will be cleared by the amplifier after Load/Start is cleared by the controller. If the data does not load successfully due to an error in the Command Assembly, the amplifier will load an error response into the Response Assembly (Axis 1: byte 4 Error Code, byte 5 Additional Code, Bytes 6-7 echo Command Assembly bytes 2-3. Axis 2: byte 68 Error Code, byte 69 Additional Code, Bytes 70-71 echo Command Assembly bytes 66-67.) See Response Type 0x14 - Command/Response Error for more information.

I/O Assembly Messaging Handshaking Sequence	Example
1. Controller loads a valid Command Type and data into the Command Assembly with Load/Start set to Low (0).	Load a Target Position command of 1000. C: 0x80 0x00 0x21 0x20 0xE8 0x03 0x00 0x00 Enable = 1, Load/Start = 0, Command Axis = 1, Command Type = 1, Response Axis = 1, Response Type = 0 (none), Data = 1000
2. Amplifier clears the Load Complete flag in the Response Assembly when Load/Start is Low (0) in the Command Assembly.	Respond with status flags. No command yet. R: 0x84 0x00 0x00 0x20 0x00 0x00 0x00 0x00 Enabled = 1, In Position = 1, Load Complete = 0, Response Axis = 1, Response Type = 0 (none), Data = 0
3. Controller checks that the Load Complete flag in the Response Assembly is Low (0) to ensure that the amplifier is ready to receive data. Controller sets the Load Data flag in the Command Assembly.	Set the Load Data flag. C: 0x81 0x00 0x21 0x20 0xE8 0x03 0x00 0x00 Enable = 1, Load/Start= 1, Command Axis = 1, Command Type = 1, Response Axis = 1, Data = 1000
4. Amplifier sees the Load/Start flag transition to High (1) and attempts to execute the command specified in the Command Type field on the data in the Data bytes. If successful, the amplifier sets the Load Complete flag. If the command fails or the Command Assembly is invalid, the amplifier will set Response Type to Error and load error information in the Response Assembly Data fields. If the command matches the operating mode (e.g. Target Position in positioning mode), the amplifier will start motion.	If no error, execute the requested command R: 0x81 0x00 0x80 0x20 0x00 0x00 0x00 0x00 Enabled = 1, In Motion = 1, Load Complete = 1, Response Axis = 1, Response Type = 0 (none), Data = 0 If there was an error (e.g. data out of range): R: 0x80 0x00 0x00 0x34 0x09 0xFF 0x21 0x20 Enabled = 1, Load Complete = 0, Response Axis = 1, Response Type = 0x14 (Error), Error codes = 0x09FF (Invalid Attribute), bytes 6-7 for Axis 1. Bytes 70-71 echo Command Assembly bytes 66-67 for Axis 2.
5. Controller waits for either the Load Complete flag to transition to High (1) or for an Error Response Type in the Response Assembly, then clears Load/Start. Ready for next command	Clear Load/Start C: 0x80 0x00 0x21 0x20 0xE8 0x03 0x00 0x00 Enable = 1, Load/Start = 0, Command Axis = 1, Command Type = 1, Response Axis = 1, Data = 1000

4 Operation Modes

4.1 Velocity Mode

In this mode, the Axis is controlled via a speed setpoint sent from the controller to the drive using I/O Assembly Messaging (the Jog command). When changing velocity, the commanded acceleration and deceleration rates will be used.

4.1.1 Setup Velocity Mode

Before Jog commands may be issued, the following conditions must be met:

- Faults are cleared. (If necessary, query the General Fault bit in Status Word 1 and issue an Explicit Message to clear faults.)
- Axis is enabled (Set Enable bit in the Control Word)
- Axis is in Velocity Mode (Set Attribute 3 Operational Mode of the Position Controller Object)
- The Command Source must be Fieldbus.
- Smooth Stop and Hard Stop bits are cleared in Status Word 1.
- Position Limits are cleared (Check bits in Status Word 2)

4.1.2 Velocity Moves

Once the Axis is ready to go, issue Jog commands (Command Type 0x07) to set a speed setpoint in the drive. Target Velocity, Acceleration, Deceleration, and Direction should all be loaded before setting the Load/Start bit to initiate the move.

NOTE

The actual direction is determined by both the Target Velocity Sign and the Direction bit. When a positive velocity with negative Direction bit or negative velocity with positive Direction bit are used, the Velocity will be negative. When a positive Velocity with positive Direction bit or negative Velocity with negative Direction bit are used, the Velocity will be positive.

While in motion, you may issue another Jog command to immediately change Velocity and Direction at the desired Acceleration and Deceleration rates.

While a Jog is operating, the In Motion bit in Status Word 1 will be set and In Position will be cleared. The Direction status bit will reflect the actual direction of motion.

Set the Smooth Stop bit to stop the motor at the previously set deceleration rate and remain enabled after decelerating to a stop.

Set the Hard Stop bit to immediately stop at the Controlled Stop rate and disable the motor. To clear this Controlled Stop condition, you must clear the Hard Stop and Enable bits, then set the Enable bit.

Velocity Move values can be verified in WorkBench. From the terminal, the affected values are AXIS#.VL.CMD.

4.2 Position Mode

In this mode, the Axis runs an internal trajectory generator for moving between commanded positions. These positions can be sent directly from the controller (point-to-point moves), or pre-programmed in Motion Task sequences.

4.2.1 Setup Position Mode

Before Position Move commands may be issued, the following conditions must be met:

- Faults are cleared (If necessary, query the General Fault bit in Status Word 1 and issue an Explicit Message to clear faults.)
- Axis is enabled (Set Enable bit in the Control Word)
- Axis is in Position Mode (Set Attribute 3 Operational Mode of the Position Controller Object)
- The Command Source must be Fieldbus.
- Smooth Stop and Hard Stop bits are cleared in Status Word 1.
- Position Limits are cleared (Check bits in Status Word 2)
- Axis is Homed (Check Homed bit in Status Word 1)

4.2.2 Homing

Once all conditions listed under Setup Position Mode have been met (with the exception of homing), the Axis may be Homed.

The Homing Mode may be selected using attribute 0x64, Home Mode of the Position Controller Object, or by setting the Homing Mode directly in WorkBench. Please see the AKD2G Online Help for a description of Homing modes.

To execute homing, write a value of 1 to attribute 0x65 Start Home Move.

When homing is complete, the Homed flag in Status Word 1 of the Response Assembly will be set.

4.2.3 Position Moves (point-to-point)

Once all conditions listed under Setup Position Mode have been met, and the Axis has been Homed, issue Position Move commands (Command Type 0x06) to move to a desired position. Target Position, Velocity, Acceleration, Deceleration, and Incremental (bit in Command Word) should all be loaded before setting the Load/Start bit to initiate the move.

While in motion, you may issue another Position Move command to interrupt the move with a new target position. In Position Mode, Jog Moves (Command Type 0x07) work in a similar way, and can be blended with Position Moves.

While a Position Move is operating, the In Motion bit in Status Word 1 will be set and In Position will be cleared. The Direction status bit will reflect the actual direction of motion. In Position will be set when the target position is reached.

Set the Smooth Stop bit to stop the motor at the previously set deceleration rate and remain enabled.

Set the Hard Stop bit to immediately stop at the Controlled Stop rate and disable. To clear this Controlled Stop condition, you must clear the Hard Stop and Enable bits, then set the Enable bit.

The Position Move is loaded into an internal move and can be viewed within WorkBench using terminal and AXIS#.FBUS parameters.

4.2.4 Running a Stored Motion Task Sequence

As an alternative to issuing a single point-to-point position command, EtherNet/IP can be used to start a predefined Motion Task or sequence of motion tasks.

A motion tasking sequence may be setup in WorkBench and then executed later through EtherNet/IP. Motion Tasks may also be setup directly through EtherNet/IP as demonstrated in the sample programs.

To execute a Motion Task sequence, set Block Number equal to the index of the Motion Task to begin executing and transition the Start Block bit high. The drive must be enabled and the stop and Load/Start bits must be low.

When a stored Motion Task is running, the Response Assembly will report this with the Block in Execution status bit, and the executing task will be given in the Block # response byte.

To stop an executing sequence, set the Smooth Stop or Hard Stop bit.

4.3 Torque Mode

In this mode, the axis runs at constant torque using the latest command value received from the controller.

4.3.1 Setup Torque Mode

Before Torque Move commands may be issued, the following conditions must be met:

- Faults are cleared (If necessary, query the General Fault bit in Status Word 1 and issue an Explicit Message to clear faults.)
- Axis is enabled (Set Enable bit in the Control Word)
- Axis is in Torque Mode (Set Attribute 3 Operational Mode of the Position Controller Object)
- The Command Source must be Fieldbus.
- Smooth Stop and Hard Stop bits are cleared in Status Word 1.
- Position Limits are cleared (Check bits in Status Word 2)

4.3.2 Torque Moves

Once the axis is setup for Torque Mode, issue Torque commands (Command Type 0x05) to set a torque setpoint in the axis. Torque commands and values are scaled in milliarms.

- While in motion, issue another Torque command to immediately change the target torque.
- While a torque command is active, the In Motion bit in Status Word 1 will be set and In Position will be cleared. The Direction status bit will reflect the actual direction of motion.
- Set the Smooth Stop bit to stop the motor at the previously set deceleration rate and remain enabled.
- Set the Hard Stop bit to immediately stop at the Controlled Stop rate and disable. To clear this Controlled Stop condition, you must clear the Hard Stop and Enable bits, then set the Enable bit.
- Torque Move values can be verified in WorkBench. From the terminal, the effective value is AXIS#.IL.CMD.

5 Drive Objects

5.1 Position Controller Class 0x66

The following attributes are supported in the Position Controller class. The instance number represents the axis number.

Attribute 0x01: Number of Attributes	
Description	The total number of attributes supported by the unit in the Position Controller class.
Access Rule	Get
Data Type	Unsigned Short Integer
Range	N/A
Default	N/A
Non-Volatile	N/A
See Also	N/A

Attribute 0x02: Attribute List	
Description	Returns an array with a list of the attributes supported by this unit in the Position Controller class. The length of this list is specified in Number of Attributes.
Access Rule	Get
Data Type	Array of Unsigned Short Integer
Range	Array size is defined by Attribute 1
Default	N/A
Non-Volatile	N/A
See Also	N/A

Attribute 0x03: Operation Mode	
Description	This attribute is used to Get or Set the operating mode. 0 = Position AXIS#.OPMODE 2). 1 = velocity (AXIS#.OPMODE 1). 2 = Torque (AXIS#.OPMODE 0). This attribute must be set before any move is attempted.
Access Rule	Get / Set
Data Type	Unsigned Short Integer
Range	0 = Position Mode 1 = Velocity Mode 2 = Torque Mode 3 = Other (read only)
Default	0
Non-Volatile	No
See Also	N/A

Attribute 0x04: Position Units	
Description	This ratio value is the number of 32-bit actual position feedback counts equal to one position unit.
Access Rule	Get / Set
Data Type	U32
Range	0 to 2^{31}
Default	65536 (16 bits/revolution)
Non-Volatile	Yes
See Also	See AXIS#.EIP.POSUNIT.

Attribute 0x05: Profile Units	
Description	This ratio value is the number of 32-bit actual position feedback counts per second (velocity) or second squared (acceleration) equal to one Velocity or Acceleration unit.
Access Rule	Get / Set
Data Type	U32
Range	0 to 2^{31}
Default	65536 (16 bits/revolution)
Non-Volatile	Yes
See Also	See AXIS#.EIP.PROFUNIT.

Attribute 0x06: Target Position	
Description	This attribute specifies the Target Position in counts. Set Start Trajectory = 1 (Attribute 11) or the Polled I/O Start Trajectory/Load Data bit to initiate the positioning move.
Access Rule	Get / Set
Data Type	S32
Range	-2^{31} to 2^{31}
Default	0
Non-Volatile	No
See Also	AXIS#.FBUS.P when using Command Type 6 (Position Move).

Attribute 0x07: Target Velocity	
Description	This attribute specifies the Target Velocity in counts per second. Target Velocity is used with Command Type 6 (Position Move) and Command Type 7 (Jog Move). Units are determined by AXIS#.EIP.PROFUNIT, Position Controller attribute 5.
Access Rule	Get / Set
Data Type	S32
Range	-2^{31} to 2^{31}
Default	0
Non-Volatile	Yes
See Also	AXIS#.FBUS.V when using Command Type 6 (Position Move) or 7 (Jog Move) while in Position Mode. Otherwise goes directly to AXIS#.VL.CMD for other types.

Attribute 0x08: Acceleration	
Description	This attribute specifies the Acceleration for Command Type 6 (Position Move), Command Type 7 (Jog Move), and Command Type 8 (Velocity Setpoint). Units are determined by Position Controller Attribute 5 (AXIS#.EIP.PROFUNIT). All Position Moves initiated through a Command Assembly or Command Block Object use this acceleration rate. To set different acceleration rates for multiple motion blocks (tasks) the Motion Tasks must be setup separately, either via WorkBench or EtherNet/IP commands.
Access Rule	Get / Set
Data Type	U32
Range	Set to a positive number
Default	According to setup
Non-Volatile	Yes
See Also	AXIS#.FBUS.ACC

Attribute 0x09: Deceleration	
Description	This attribute specifies the Deceleration for Command Type 6 (Position Move), Command Type 7 (Jog Move), and Command Type 8 (Velocity Setpoint). Units are determined by Position Controller Attribute 5 (AXIS#.EIP.PROFUNIT). All Position Moves initiated through a Command Assembly or Command Block Object use this acceleration rate. To set different acceleration rates for multiple motion blocks (tasks) the Motion Tasks must be setup separately, either via WorkBench or EtherNet/IP commands.
Access Rule	Get / Set
Data Type	U32
Range	Set to a positive number
Default	According to setup
Non-Volatile	Yes
See Also	AXIS#.FBUS.DEC

Attribute 0x0A: Move Type	
Description	This bit is used to define the position value as either Absolute or Incremental in Position Mode.
Access Rule	Get / Set
Data Type	Boolean
Range	0 = Absolute Position 1 = Incremental Position
Default	1
Non-Volatile	No
See Also	N/A

Attribute 0x0B: Trajectory Start/Complete	
Description	Set High (1) to start a Trajectory Move. Reads High (1) while In Motion and Low (0) when motion is complete
Access Rule	Get / Set
Data Type	Boolean
Range	0 = Move Complete 1 = Start Trajectory (In Motion)
Default	0
Non-Volatile	No
See Also	N/A

Attribute 0x3A: Load Data Complete	
Description	Indicated the drive has successfully loaded the previous command value. It is used in combination with attribute 0x0B Trajectory Start/Complete to handshake motion starts between the AKD2G and controller.
Access Rule	Get
Data Type	Boolean
Range	0 = Load not complete 1 = Load complete
Default	0
Non-Volatile	No
See Also	N/A

Attribute 0x11: Enable	
Description	This flag is used to control the enable output. Clearing this bit sets the enable output inactive and the currently executing motion profile is aborted.
Access Rule	Get / Set
Data Type	Boolean
Range	0 = Disable 1 = Enable
Default	0
Non-Volatile	N/A
See Also	AXIS#.EN and AXIS#.DIS

Attribute 0x19: Torque	
Description	Set a new torque command (AXIS#.FBUS.IL.CMD) in Torque Mode or read the current torque command. The Trajectory Start attribute is used to begin motion.
Access Rule	Get / Set
Data Type	S32
Range	-3280 to 3280 (3280 = peak torque)
Default	0
Non-Volatile	No
See Also	AXIS#.FBUS.IL.CMD

Attribute 0x64: Home Mode	
Description	Set the desired Homing Mode.
Access Rule	Get / Set
Data Type	U16
Range	N/A
Default	0
Non-Volatile	Yes
See Also	AXIS#.HOME.MODE

Attribute 0x65: Start Home Move	
Description	Start homing.
Access Rule	Get / Set
Data Type	Boolean
Range	0 = Do not Move Home 1 = Initiate a Home Move
Default	0
Non-Volatile	No
See Also	AXIS#.HOME.MOVE

5.2 Parameter Object Class 0x64

Most drive parameters can be read and or written through the Parameter Object. This includes many drive parameters also available through the Position Controller and Position Controller Supervisor classes.

For an Explicit Message to the Parameter Object, the attribute number of the desired parameter can be found in Appendix B: EtherNet/IP Objects List. See the Appendix for attribute numbers, data types, and scaling. Note that Float types are scaled by 1000 to get an integer value.

Amplifier commands such as AXIS#.HOME.MOVE and DRV.NVSAVE are executed by sending a Set Value command with a data length of 1 and any value 0 to 255. Reading the value will not execute the process.

For example, send the following Explicit Message to initiate homing (AXIS2.HOME.MOVE, attribute = 6108):

[class = 0x64, instance = 2 (axis ID), attribute = 6108, data length = 1, data value = 0x01].

5.2.1 Read a Parameter Value

To read a parameter value through Explicit Messaging, use Service 0x0E (Read Value), Class 0x64 (Parameter class).

The attribute number corresponds to the index (ID) of the desired parameter. This number may be found in Appendix B: EtherNet/IP Objects List. The instance number specifies the axis ID for axis-based parameters (AXIS#.*) and/or the array index for array-based parameters. The array index is specified by multiplying the value of the array index by 100 (array index * 100).

For example, to reference Motion Task 3 for Axis 2, the instance would be 302. To reference Motion Task 0 for Axis 1, the instance would be 1.

For drive level non-array parameters (i.e.: DRV.NVSAVE), the instance is ignored.
See examples in Appendix C: Examples of Explicit Messaging.

5.2.2 Write a Parameter Value

To set a parameter value through Explicit Messaging, use Service 0x10 (Write Value), Class 0x64 (Parameter class).

The attribute number corresponds to the index (ID) of the desired parameter. This number may be found in Appendix B: EtherNet/IP Objects List.

The length of the data written must match the length of the parameter. The instance number specifies the axis ID for axis-based parameters (AXIS#.*) and/or the array index for array-based parameters. The array index is specified by multiplying the value of the array index by 100 (array index * 100).

For example, to reference Motion Task 3 for Axis 2, the instance would be 302. To reference Motion Task 0 for Axis 1, the instance would be 1.

For drive level non-array parameters (i.e.: DRV.NVSAVE), the instance is ignored.
See examples in Appendix C: Examples of Explicit Messaging.

5.2.3 Execute a Command Parameter

Some parameters are actually commands which do not take a value, but execute a drive function such as AXIS#.HOME.MOVE or DRV.CLRFAULTS . To execute a command, write a value of 1 to the parameter.

The index (ID) number of the desired parameter can be found in Appendix B: EtherNet/IP Objects List.

To execute a command parameter through Explicit Messaging, use Service 0x10 (Write Value), Class 0x64 (Parameter class), Data = 0x01.

See examples in Appendix C: Examples of Explicit Messaging.

6 Handling Faults

Drive fault conditions are reported with the General Fault bit in Status Word 1 of the Response Assembly.

Specific fault numbers can be read through fault registers using the Parameter Class. The fault registers AXIS#.FAULT# at indexes 6800-6809. FAULT1 will always list the highest-priority fault.

Faults may be cleared by sending a message to the DRV.CLRFAULTS index 2002 of the Parameter Class. Write a 1-byte value (any value) to the parameter to execute the command.

Transmit the following Explicit Message:

Service	0x10 (Write)
Class	0x64 (Parameter)
Instance	1
Attribute	2002 (0x7D2) DRV.CLRFAULTS
Data Length	1 byte
Data Value	1

Individual Axis faults may be cleared by using AXIS#.CLRFAULTS.

5022 (0x139E)

Transmit the following Explicit Message:

Service	0x10 (Write)
Class	0x64 (Parameter)
Instance	1 or 2 (Axis #)
Attribute	5022 (0x139E) AXIS#.CLRFAULTS
Data Length	1 byte
Data Value	1

7 Saving to Non-Volatile Memory

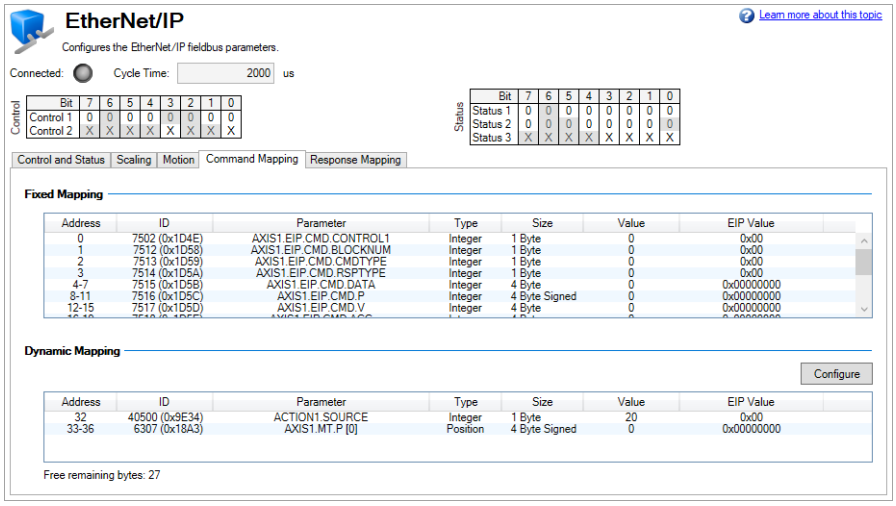
Drive parameters are typically stored in RAM and only stored to non-volatile memory when a Save is commanded through an Explicit Message to the Parameter Object.

Transmit the following Explicit Message:

Service	0x10 (Write)
Class	0x64 (Parameter)
Instance	1
Attribute	2005 (7D5) DRV.NVSAVE
Data Length	1 byte
Data Value	1

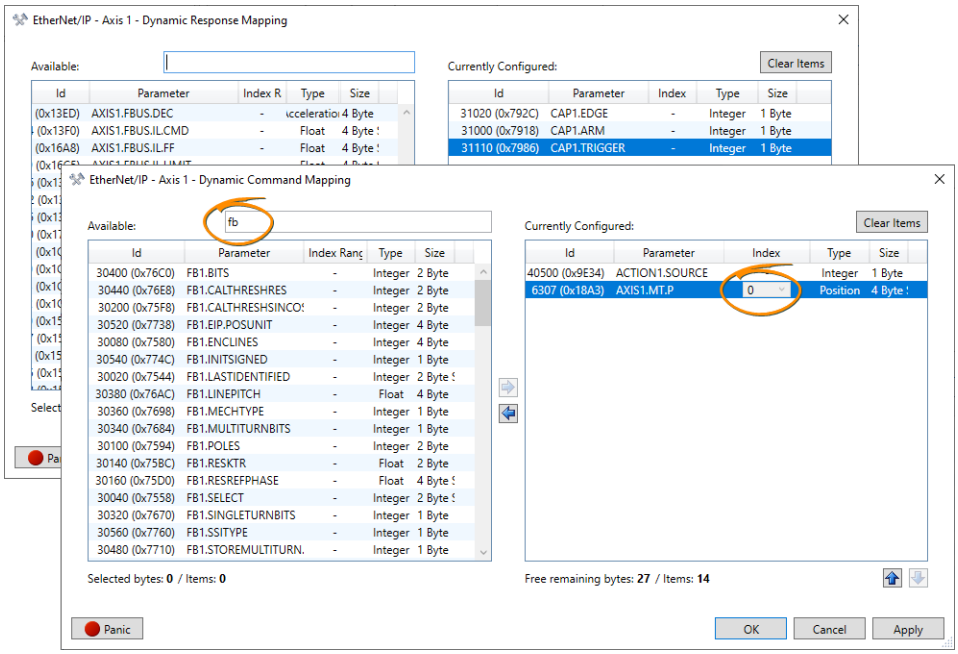
8 Using WorkBench to Dynamically Map EtherNet/IP Parameters

The EtherNet/IP View has tabs for Command Mapping and Response Mapping of commands for each axis. The tabs look much the same with sections for Fixed Mapping and Dynamic Mapping. The commands seen in the Fixed Mapping section are set for Kollmorgen drives, while the commands seen in the Dynamic Mapping section are user-configurable.



Following are the steps for setting up Dynamic Mapping of parameters.

1. Click the **Configure** button to open a dialog which lists all available commands.
2. Find the desired command. Using the search field to narrow your list of choices is recommended.

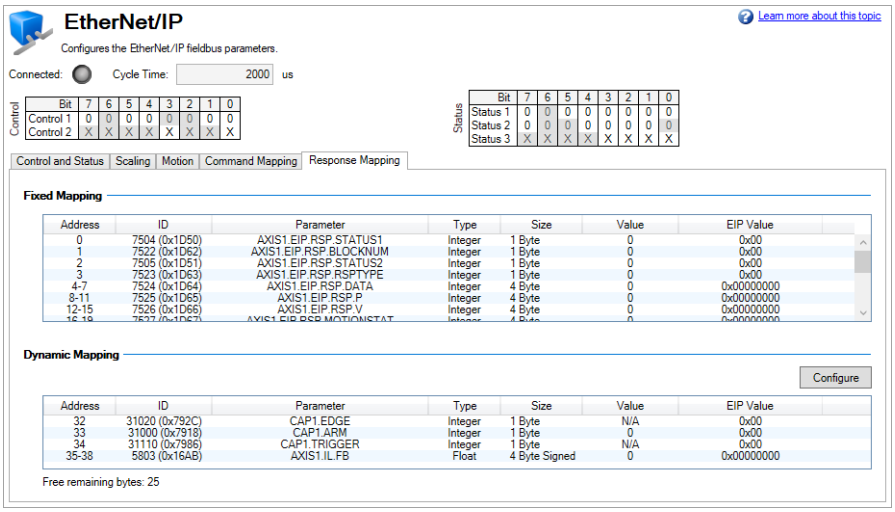


3. Select the desired parameter and then press the right arrow key in the center of the screen to move the parameter into the "Currently Configured" list.

   Array-type parameters added to the Currently Configured List include an available field under the Index column for selecting and assigning which index of the parameter to dynamically map, e.g. AXIS1.MT.P for motion task 3 the index=3.

4. Click **Apply** and then **OK**.

In the following image we see that Axis 1 Current Loop Feedback has been dynamically mapped. Under the Dynamic Mapping list AXIS1.IL.FB (among others)is mapped to the response assembly's bytes 35-38 (4 bytes) and the Free Remaining Bytes are now reduced by four bytes from 32 to 25 with the total of 7 bytes having been mapped.



NOTE

It is up to the programmer to copy the corresponding bytes of data for each dynamically mapped parameter from the IO Cyclic Data (Command and Response Assemblies) into their EIP Master program to be used.

TIP

While using WorkBench is the preferred method, the following commands can be used in the Terminal to dynamically map parameters.

- AXIS#.EIP.DYNAMICCMDMAP
- AXIS#.EIP.DYNAMICCMDINDEX
- AXIS#.EIP.DYNAMICRSPDATA
- AXIS#.EIP.DYNAMICRSPINDEX

9 Units

Position, Velocity and Acceleration are scaled differently for EtherNet/IP than for WorkBench. In WorkBench, these values are displayed as floating point numbers and can been configured in many ways. In EtherNet/IP, these values are integers and are scaled as a ratio of position units to actual position counts.

9.1 Position Units

Position values are scaled according to the EtherNet/IP Position Controller Device standard. One “Position Units” scaling value is defined as the number of actual position feedback counts (at 32 bits per revolution) equal to one position unit.

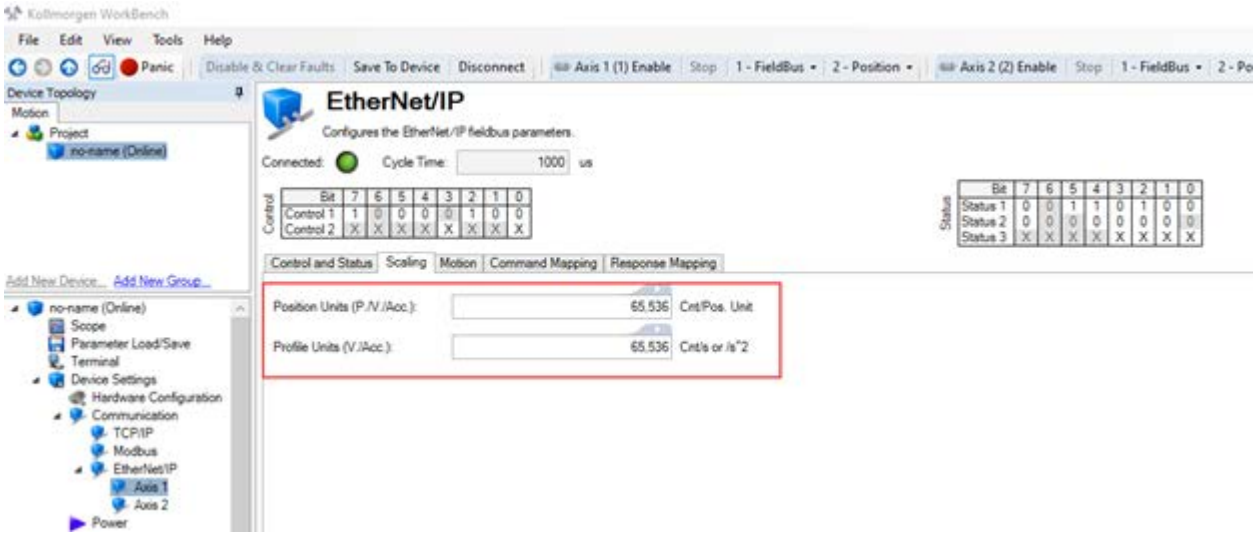
- From WorkBench, this scaling parameter is visible on the EtherNet/IP screen or as AXIS#.EIP.POSUNIT in the terminal.
- From EtherNet/IP, this value can be accessed at attribute 0x04 Position Units of the Position Controller Object.

The default value is 2^{16} = 65536, which provides 2^{32} / 2^{16} = 2^{16} counts per revolution. A value of 1 would provide 2^{32} / 1 = 2^{32} counts per revolution.

NOTE

When dynamically mapping Response or Command Assembly, Position Units are scaled from 8 bytes to 4 bytes. This results in a loss of precision for the parameters AXIS#.PL.CMD , AXIS#.PL.ERR , and AXIS#.SWLS.LIMIT#.

Example: EtherNet/IP units for Axis 1



9.2 Velocity and Acceleration Units

Velocity and Acceleration values are scaled according to the EtherNet/IP Position Controller Device standard. One “Profile Units” scaling value is defined, which affects both Velocity and Acceleration.

- For Velocity values, Profile Units gives the number of actual position feedback counts (at 32 bits per revolution) per second equal to one velocity unit.
- For Acceleration values, Profile Units gives the number of actual position feedback counts (at 32 bits per revolution) per second^2 equal to one acceleration unit.
- From WorkBench, this scaling parameter is visible in the EtherNet/IP screen or as AXIS#.EIP.PROFUNIT in the terminal.
- From EtherNet/IP, this value can be accessed at Attribute 0x05 Profile Units of the Position Controller Object.

The default value is 2^{16} = 65536, which provides 2^{32} / 2^{16} = 2^{16} counts per second per revolution. A value of 1 would provide 2^{32} / 1 = 2^{32} counts per second per revolution.

9.3 Torque Units

Torque commands and values are scaled in milliamps [mA].

9.4 Other Floating Point Values

Other parameters which are displayed as floating point values in WorkBench are provided with three-digit accuracy over EtherNet/IP. For example, a velocity loop gain AXIS#.VL.KP of 1.200 would be read over EtherNet/IP as 1200.

9.5 Scaling

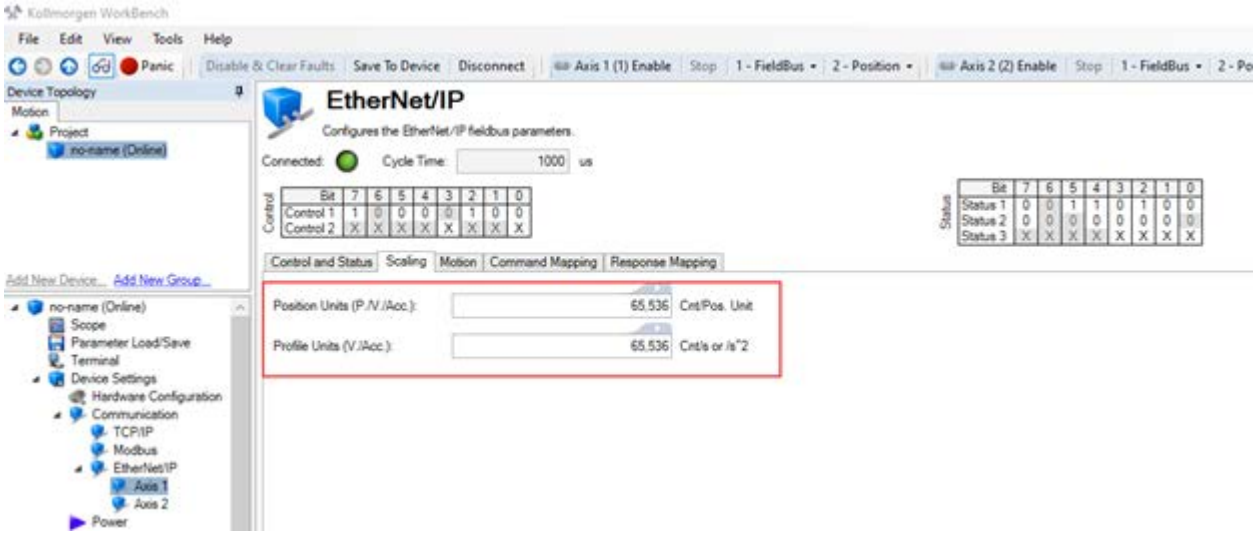
9.5.1 Scaling Best Practices

It is important to note that while WorkBench provides a way to scale the Axis units on the Units screen in the WorkBench project tree, those units only pertain to how the values and units are shown within WorkBench. These units do not affect how the Position, Velocity, and Acceleration are read or written to or from the controller (i.e. PLC, HMI, etc.) using EtherNet/IP.

The most intuitive approach is to scale the WorkBench units in the same way as EtherNet/IP scaling so the counts in the PLC equal the counts set or read in the drive while monitoring with WorkBench.

9.5.2 EtherNet/IP Scaling in WorkBench

The figure below shows the default scaling for Axis 1.



9.5.3 EtherNet/IP Position Unit Scaling

The default scaling for AXIS#.EIP.POSUNIT is 65536, as shown above.

65536 = 2^{16}, so the actual counts per revolution of the motor is found using the formula:

Definition:

$$EIP\ Counts\ per\ motor\ rev = \frac{AKD\ internal\ feedback\ counts}{AXIS\#.EIP.POSUNIT} = \frac{2^{32}}{AXIS\#.EIP.POSUNIT}$$

Default Scaling:

$$AXIS\#.EIP.POSUNIT = 2^{16}\ or\ 65536$$

$$EIP\ Counts\ per\ motor\ rev = \frac{AKD\ internal\ feedback\ counts}{AXIS\#.EIP.POSUNIT}$$

$$= \frac{2^{32}}{2^{16}} = 2^{16}\ or\ 65536\ EIP\ counts\ per\ motor\ rev$$

This means if a Position Move is commanded over EtherNet/IP (e.g., a Relative Move) and is programmed with a position attribute value of 65536, when the move is triggered the motor will make 1 revolution. This is true regardless of the WorkBench units.

The implication is the PLC/HMI must do the conversions from real-world units (i.e. inches, inches/sec, etc.) to revolutions and revolutions/second, etc. then convert these units to counts and counts/revolution, etc. based on the EtherNet/IP scaling. Also keep in mind the values entered are integer based and not floating point. This means the smallest positional value/increment that can be commanded is 1 count (not fractions of counts).

Example:

Horizontal axis, 0.2 inch/revolution ballscrew, 5:1 gearbox. Desired units are inches, inches/sec, inches/sec^2.

1 inch	1 revolution of ballscrew	5 motor revolution	65536 counts	=	1638400 counts
	0.2 inch	1 revolution of ballscrew	1 motor revolution		

9.5.4 EtherNet/IP Velocity and Acceleration Unit Scaling

Velocity and Acceleration units are also determined by the EtherNet/IP scaling.

Below, the example shows if the default value of 65536 is used for AXIS#.EIP.PROFUNIT then 65536 counts/sec is 1 revolution/sec or 60 RPM. For 10 rps, 655360 is used to set the Position Move's Speed value.

Definition:

$$EIP\ Counts\ per\ motor\ rev/s\ =\ \frac{AKD\ internal\ feedback\ counts/s}{AXIS\#.EIP.PROFUNIT} = \frac{2^{32}}{AXIS\#.EIP.PROFUNIT}$$

Default scaling:

$$AXIS\#.EIP.PROFUNIT\ =\ 2^{16}\ or\ 65536$$

$$EIP\ Counts/s\ per\ motor\ rev/s\ =\ \frac{AKD\ internal\ feedback\ counts/s}{AXIS\#.EIP.PROFUNIT}$$

$$=\ \frac{2^{32}}{2^{16}}\ =\ 2^{16}\ or\ 65536\ EIP\ Counts/s\ per\ motor\ rev/s$$

NOTE

- Note: 65536 counts/s in PLC = 1 revolution/s of actual motor speed = 60 RPM
- So PLC value = motor speed in revolution/s * 65536
- Or PLC value = motor speed in RPM * 65536 / 60

Using the same example as above, lets suppose the PLC/HMI wants to set the Target Velocity during the move to be 5 inches/sec.

Horizontal axis, 0.2 inch/revolution ballscrew, 5:1 gearbox. Desired units are inches, inches/sec, inches/sec^2.

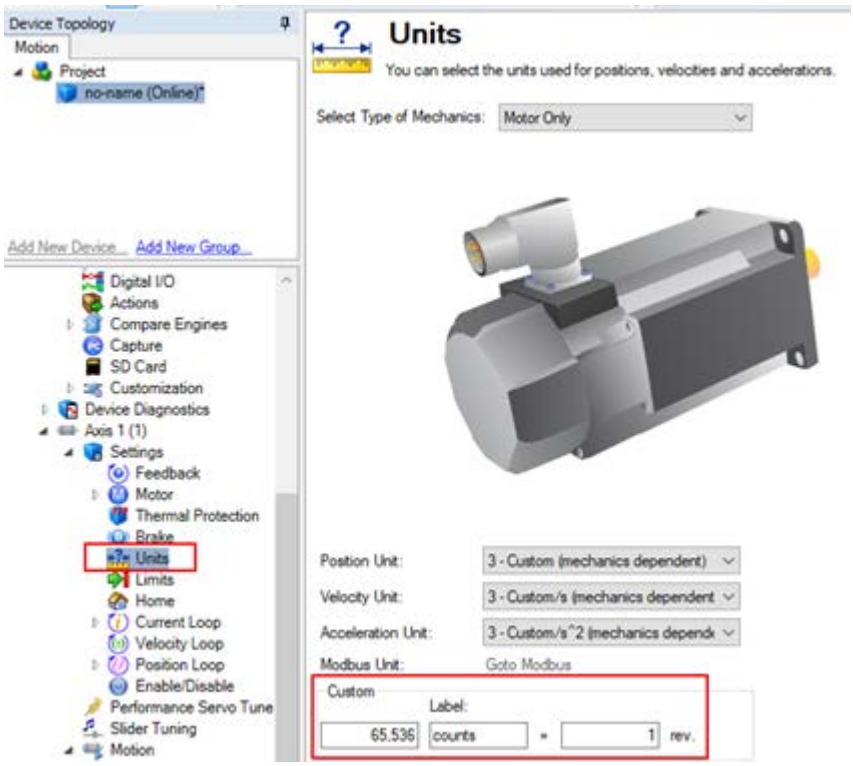
5 inches	1 revolution of ballscrew	5 motor revolution	65536 counts	=	8192000 counts
1 sec	0.2 inch	1 revolution of ballscrew	1 motor revolution		sec

For acceleration and deceleration units, it follows the same convention with counts/sec^2.

As mentioned previously, WorkBench Units can be set to match EtherNet/IPUnits.

On the Units screen:

- Set Select Type of Mechanics to **Motor Only**.
- Set the Position Unit to **3 - Custom (mechanics dependent)**
- Set the Velocity Unit to **3 - Custom/s (mechanics dependent}**
- Set the Acceleration Unit to **3 - Custom/s^2 mechanics dependent**.
- The Custom dialog loads displaying 65536 counts = 1 revolution.

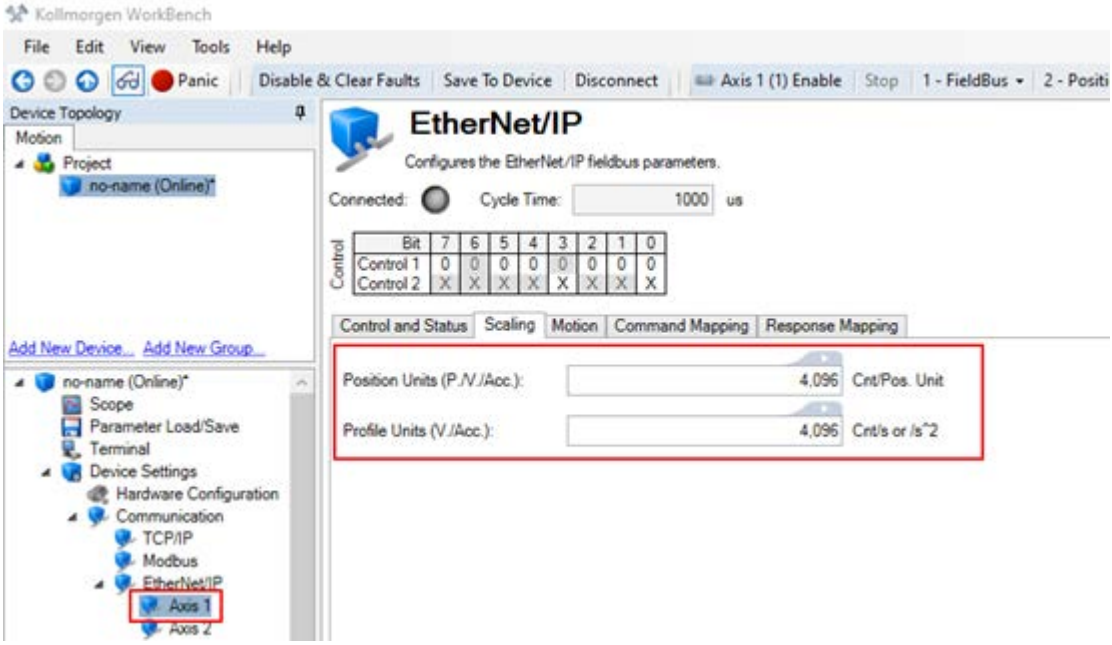


9.5.5 Higher Resolution Scaling

In many cases, the default scaling and resolution is adequate. However, some applications require a higher resolution than 65536 counts per motor revolution.

In the following example, the resolution is increased from 65536 (2^{16}) counts to 1048576 (2^{20}) counts per revolution.This is done by setting the AXIS#.EIP.POSUNIT and AXIS#.EIP.PROFUNIT to 4096 (2^{12}). The derivation of scaling formulas and PLC math examples, and the conversion from real-world units to EtherNet/IP counts are shown for the higher resolution scaling below.

Changing the Ethernet/IP Scaling



9.5.6 Higher Resolution Scaling Example

EtherNet/IP Position Unit

$$AXIS\#.EIP.POSUNIT = 2^{12} \text{ or } 4096$$

$$EIP \text{ counts per motor rev} = \frac{AKD \text{ internal feedback counts}}{AXIS\#.EIP.POSUNIT}$$

$$= \frac{2^{32}}{2^{12}} = 2^{20} \text{ or } 1048576 \text{ } EIP \text{ counts per motor rev}$$

EtherNet/IP Velocity and Acceleration

$$AXIS\#.EIP.PROFUNIT = 2^{12} \text{ or } 4096$$

$$EIP \text{ counts per motor rev / s} = \frac{AKD \text{ internal feedback counts}}{AXIS\#.EIP.PROFUNIT}$$

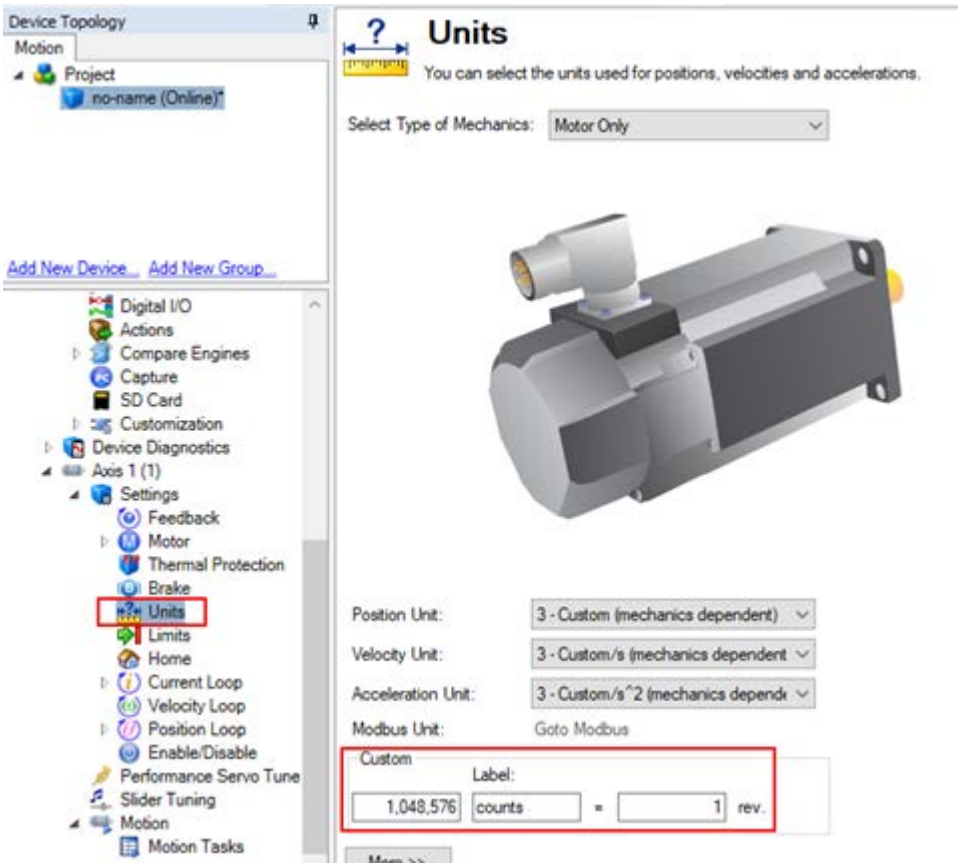
$$= \frac{2^{32}}{2^{12}} = 2^{20} \text{ or } 1048576 \text{ } EIP \text{ counts per motor rev / s}$$

NOTE

- Note: 1048576 counts/s in PLC = 1 rev/s of actual motor speed = 1 rps = 60 RPM
- So PLC value = motor speed in rev/s * 1048576
- Or PLC value = motor speed in RPM *1048576 / 60

Setting WorkBench Units to Match EtherNet/IP

To set the WorkBench Units to match EtherNet/IP scaling enter 1048576 into the Custom field.


Device Topology
Motion
Project
no-name (Online)*
Add New Device...
Add New Group...
Digital I/O
Actions
Compare Engines
Capture
SD Card
Customization
Device Diagnostics
Axis 1 (1)
Settings
Feedback
Motor
Thermal Protection
Brake
Units
Limits
Home
Current Loop
Velocity Loop
Position Loop
Enable/Disable
Performance Servo Tune
Slider Tuning
Motion
Motion Tasks
Units
You can select the units used for positions, velocities and accelerations.
Select Type of Mechanics:
Motor Only
Position Unit:
3 - Custom (mechanics dependent)
Velocity Unit:
3 - Custom/s (mechanics dependent
Acceleration Unit:
3 - Custom/s^2 (mechanics dependc
Modbus Unit:
Goto Modbus
Custom
Label:
1,048,576
counts
=
1
rev.

10 EtherNet/IP Commands

Following is a list of commands used by EtherNet/IP. Please see the AKD2G Online Help for full documentation on the commands.

Command	Description
AXIS#.EIP.CMD.ACC	This parameter allows reading the associated axis' EtherNet/IP 32-bit acceleration field from the fixed part of the command mapping.
AXIS#.EIP.CMD.ATTRIBUTE	This parameter allows reading the associated axis' EtherNet/IP attribute byte from the fixed part of the command mapping.
AXIS#.EIP.CMD.BLOCKNUM	This parameter allows reading the associated axis' EtherNet/IP block number byte from the fixed part of the command mapping.
AXIS#.EIP.CMD.CMDTYPE	This parameter allows reading the associated axis' EtherNet/IP command type byte from the fixed part of the command mapping.
AXIS#.EIP.CMD.CONTROL1	This parameter allows recording, triggering, and viewing the associated axis' EtherNet/IP control word signal from the fixed part of the command mapping.
AXIS#.EIP.CMD.CONTROL2	A second control byte for EtherNet/IP that can be mapped in the dynamic portion of the command map using AXIS#.EIP.DYNAMICCMDMAP.
AXIS#.EIP.CMD.DATA	This parameter allows reading the associated axis' EtherNet/IP 32-bit data field from the fixed part of the command mapping. The meaning of the data contained in the data field changes based on the command type specified.
AXIS#.EIP.CMD.DATA2	This parameter allows reading the associated axis' EtherNet/IP 32-bit parameter/attribute data field from the fixed part of the command mapping. The meaning of the data contained in the data field changes based on the command type specified and the parameter/attribute specified.
AXIS#.EIP.CMD.DEC	This parameter allows reading the associated axis' EtherNet/IP 32-bit deceleration field from the fixed part of the command mapping.
AXIS#.EIP.CMD.P	This parameter allows reading the associated axis' EtherNet/IP 32-bit position field from the fixed part of the command mapping.
AXIS#.EIP.CMD.RSPTYPE	This parameter allows reading the associated axis' EtherNet/IP response type byte from the fixed part of the command mapping.
AXIS#.EIP.CMD.V	This parameter allows reading the associated axis' EtherNet/IP 32-bit velocity field from the fixed part of the command mapping.
AXIS#.EIP.DYNAMICCMDDATA	This parameter shows the data received by the AKD2G over the EtherNet/IP network. Each entry corresponds to a mapping entry in AXIS#.EIP.DYNAMICCMDMAP.
AXIS#.EIP.DYNAMICCMDINDEX	This parameter is used to set the array index of the corresponding parameter in AXIS#.EIP.DYNAMICCMDMAP. If the mapped parameter is not an array parameter, the index should be left as 0. If mapped parameter is an array, the index will be used when setting/getting the corresponding value.

Command	Description
AXIS#.EIP.DYNAMICCMDMAP	This parameter is used to set parameters in the dynamic portion of the command assembly using the syntax AXIS#.EIP.DYNAMICCMDMAP <index> <parameter id>. Reading the keyword with no index specified will show all index values. A 0 represents an unassigned value.
AXIS#.EIP.DYNAMICRSPDATA	This parameter shows the data transmitted by the AKD2G over EtherNet/IP network. Each entry corresponds to a mapping entry in AXIS#.EIP.DYNAMICRSPMAP.
AXIS#.EIP.DYNAMICRSPINDEX	This parameter is used to set the array index of the corresponding parameter in AXIS#.EIP.DYNAMICRSPMAP. If the mapped parameter is not an array parameter, the index should be left as 0. If mapped parameter is an array, the index will be used when setting/getting the corresponding value. **NOTE** If the array index specified is outside the range an error will occur internally and the error bit of AXIS#.EIP.RSP.STATUS3 will be set.
AXIS#.EIP.DYNAMICRSPMAP	This parameter is used to set parameters in the dynamic portion of the response assembly using the syntax AXIS#.EIP.DYNAMICRSPMAP <index> <parameter id>. Reading the keyword with no index specified will show all index values. A 0 represents an unassigned value.
AXIS#.EIP.FIXEDCMDDATA	This parameter shows the data received by the AKD2G over the EtherNet/IP network. Each entry corresponds to a mapping entry in AXIS#.EIP.FIXEDCMDMAP.
AXIS#.EIP.FIXEDCMDMAP	This parameter is used to read the signal IDs assigned to the fixed portion of the command assembly. By default it will match the command assembly structure defined for assembly 101.
AXIS#.EIP.FIXEDRSPDATA	This parameter shows the data received by the AKD2G over EtherNet/IP network. Each entry corresponds to a mapping entry in AXIS#.EIP.FIXEDRSPMAP.
AXIS#.EIP.FIXEDRSPMAP	This parameter is used to read the signal ids assigned to the fixed portion of the response assembly. By default it will match the response assembly structure defined for assembly 102.
AXIS#.EIP.OBJECTLIST	This command lists the objects available to map in the dynamic portion of the command and response maps using AXIS#.EIP.DYNAMICCMDMAP and AXIS#.EIP.RSPMAP. The information displayed is: • Object number • Object name • Data type (Integer, Float, Position, Velocity, Acceleration, Varies, None) • Data size (1 Byte Command, 1 Byte, 2 Byte, 4 Byte, 8 Byte) • Data access (ReadOnly, WriteOnly, ReadWrite)

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>AXIS#.EIP.OPMODE</td><td>This parameter allows reading or writing the EtherNet/IP position controller class' OpMode attribute. It will change the AXIS#.OPMODE, but note that the values have different value meanings to match the EtherNet/IP position controller definition.

<table><tr><th>Mode</th><th>Description</th></tr><tr><td>0</td><td>Position Operation Mode</td></tr><tr><td>1</td><td>Velocity Operation Mode</td></tr><tr><td>2</td><td>Torque Operation Mode</td></tr></table></td></tr>
<tr><td>AXIS#.EIP.PADBYTE</td><td>This parameter is used to represent padding bytes in both the Command Assembly and the Response Assembly when using the mapping parameters. Writing a value has no affect and the reading will always be 0.</td></tr>
<tr><td>AXIS#.EIP.POSUNIT</td><td>Position values are scaled according to theEtherNet/IP Position Controller Device standard. One “Position Units” scaling value is defined per axis, which gives the number of actual position feedback counts (at 32 bits per revolution) equal to one position unit.
From WorkBench, this scaling parameter is visible on the EtherNet/IP screen or as AXIS#.EIP.POSUNIT in the terminal.
From EtherNet/IP, this value can be accessed using explicit messages by using either the vendor parameter class 0x64 with attribute 7500 from the ID list or via attribute 4 of the Position Controller class 0x66. In both cases, the instance ID specifies the axis ID.
The default value is 2^{16} = 65536, which provides 2^{32} / 2^{16} = 2^{16} counts per revolution. A value of 1 would provide 2^{32} / 1 = 2^{32} counts per revolution.
See Appendix B: EtherNet/IP Objects List for a list of the Parameter IDs.</td></tr>
</table>

Command	Description
AXIS#.EIP.PROFUNIT	Velocity and acceleration values are scaled according to the EtherNet/IP Position Controller Device standard. One “Profile Units” scaling value is defined per axis, which affects all velocity and acceleration values. For velocity values, Profile Units gives the number of actual position feedback counts (at 32 bits per revolution) per second equal to one velocity unit. For acceleration values, Profile Units gives the number of actual position feedback counts (at 32 bits per revolution) per second^2 equal to one acceleration unit. From WorkBench, this scaling parameter is visible on the EtherNet/IP screen or as AXIS#.EIP.PROFUNIT in the terminal. From EtherNet/IP, this value can be accessed using explicit messages by using either the vendor parameter class 0x64 with attribute 7501 from the ID list or by using attribute 5 of the Position Controller class 0x66. In both cases, the instance ID specifies the axis ID. The default value is 2^{16} = 65536, which provides 2^{32} / 2^{16} = 2^{16} counts per second per revolution. A value of 1 would provide 2^{32} / 1 = 2^{32} counts per second per revolution. See Appendix B: EtherNet/IP Objects List for a list of the Parameter IDs.
AXIS#.EIP.RSP.ATTRIBUTE	This parameter allows reading the associated axis’ EtherNet/IP attribute byte from the fixed part of the response mapping.
AXIS#.EIP.RSP.BLOCKNUM	This parameter allows reading the associated axis’ EtherNet/IP block number byte from the fixed part of the response mapping.
AXIS#.EIP.RSP.DATA	This parameter allows reading the associated axis’ EtherNet/IP 32-bit data field from the fixed part of the response mapping. The data meaning changes based on the response type specified.
AXIS#.EIP.RSP.DATA2	This parameter allows reading the associated axis’ EtherNet/IP 32-bit parameter/attribute data field from the fixed part of the response mapping. The meaning of the data changes based on the parameter/attribute specified.
AXIS#.EIP.RSP.MOTIONSTAT	This parameter allows reading the associated axis’ EtherNet/IPP 32-bit motion status field from the fixed part of the response mapping. The value should match AXIS#.MOTIONSTAT.
AXIS#.EIP.RSP.P	This parameter allows reading the associated axis’ EtherNet/IP 32-bit actual position from the fixed part of the response mapping.
AXIS#.EIP.RSP.RSPTYPE	This parameter allows reading the associated axis’ EtherNet/IP response type byte from the fixed part of the response mapping.
AXIS#.EIP.RSP.STATUS1	This parameter allows reading the first status byte from the EtherNet/IP cyclic response.
AXIS#.EIP.RSP.STATUS2	This parameter allows reading the second status byte from the EtherNet/IP cyclic response.
AXIS#.EIP.RSP.STATUS3	This parameter allows reading the optional third status byte from the EtherNet/IP cyclic response.

Command	Description
AXIS#.EIP.RSP.V	This parameter allows reading the associated axis' EtherNet/IP 32-bit actual velocity from the fixed part of the response mapping.
AXIS#.EIP.RSPMAP	This parameter is used to set parameters in the dynamic portion of the response assembly using the syntax AXIS#.EIP.RSPMAP <index> <parameter id>. Reading the keyword with no index specified will show all index values. A 0 represents an unassigned value.
AXIS#.EIP.STATUS1	This parameter allows reading the first status byte from the EtherNet/IP cyclic response.
AXIS#.EIP.STATUS2	This parameter allows reading the second status byte from the EtherNet/IP cyclic response.
AXIS#.EIP.STATUS3	This parameter allows reading the optional third status byte from the EtherNet/IP cyclic response.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>EIP.INFO</td><td>This keyword provides information about the EtherNet/IP operation of the drive.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Bus State</td><td>Indicates whether the EtherNet/IP ports are enabled or not.</td></tr>
<tr><td>Connected</td><td>Indicates whether the EtherNet/IP master has an active connection to the drive or not.</td></tr>
<tr><td>Cycle Time</td><td>The current cycle period (RPI) configured by the connection</td></tr>
<tr><td>Originator Serial</td><td>Serial number of the device connected to the drive</td></tr>
<tr><td>Originator Vendor ID</td><td>Vendor ID of the device connected to the drive</td></tr>
<tr><td>Name</td><td>Fieldbus stack name.</td></tr>
<tr><td>Version</td><td>Fieldbus stack version.</td></tr>
<tr><td>Firmware Date</td><td>Fieldbus stack date.</td></tr>
<tr><td>EIP MAC #</td><td>The three MAC addresses required by the EtherNet/IP stack.</td></tr>
<tr><td>Port 1 Link</td><td>0 – No active Ethernet link on Port 1
1 – Active Ethernet link on Port 1</td></tr>
<tr><td>Port 2 Link</td><td>0 – No active Ethernet link on Port 2
1 – Active Ethernet link on Port 2</td></tr>
<tr><td>IP Address</td><td>The current EtherNet/IP IP address. See EIP.IPADDRESS.</td></tr>
<tr><td>Subnet Mask</td><td>The current EtherNet/IP IP subnet mask. See EIP.IPSUBNET.</td></tr>
<tr><td>Gateway</td><td>The current EtherNet/IP IP gateway. See EIP.IPGATEWAY.</td></tr>
<tr><td>Name server1</td><td>The current EtherNet/IP DNS server 1, this can be set using the TCP/IP class from the EtherNet/IP network</td></tr>
<tr><td>Name server2</td><td>The current EtherNet/IP DNS server 2, this can be set using the TCP/IP class from the EtherNet/IP network</td></tr>
<tr><td>Domain Name</td><td>The current EtherNet/IP domain name, this can be set using the TCP/IP class from the EtherNet/IP network</td></tr>
<tr><td>IP Mode</td><td>The current EtherNet/IP IP mode. See EIP.IPMODE.</td></tr>
</table>
</td></tr>
<tr><td>EIP.IPADDRESS</td><td>This parameter is used to set the IP address of the drive for the EtherNet/IP fieldbus when EIP.IPMODE is set to 0 for static IP.</td></tr>
</table>

Command	Description
EIP.IPGATEWAY	This parameter is used to set the gateway IP of the drive for the EtherNet/IP fieldbus when EIP.IPMODE is set to 0 for static IP. It determines the IP addresses the drive can communicate with outside of its current subnet.
EIP.IPMODE	This parameter is used to determine how the drive acquires an IP address for EtherNet/IP communication.
EIP.IPRESET	This command is used to acquire a new IP address for EtherNet/IP communication based on which EIP.IPMODE is specified.
EIP.IPSUBNET	This parameter is used to set the IP subnet mask of the drive on the EtherNet/IP fieldbus when EIP.IPMODE is set to 0 for static IP. It determines the IP addresses the drive can communicate with.

11 Appendix A: EtherNet/IP Objects and Attributes

The following table provides supported attributes.

11.1 Position Controller Object 0x66

Attribute ID (Decimal Value)	Name	Access Rule	Type	Description
1	Number of Attributes	Get	USINT	Returns the total number of attributes supported by this object in this device.
2	Attribute List	Get	Array of USINT	Returns an array with a list of the attributes supported by this object in this device.
3	Mode	Get/Set	USINT	Operating mode. 0 = Position Mode (default), 1 = Velocity Mode, 2 = Torque Mode. Values get converted to AXIS#.OPMODE.
4	Position Units	Get/Set	DINT	Position Units ratio value is the number of actual position feedback counts equal to one position unit (default 1). See AXIS#.EIP.POSUNIT.
5	Profile Units	Get/Set	DINT	Profile Units ratio value is the number of actual position feedback counts per second or second^2 equal to one velocity, acceleration or deceleration unit (default 1). See AXIS#.EIP.PROFUNIT.
6	Target Position	Get/Set	DINT	Specifies the target position in counts. See AXIS#.FBUS.P.
7	Target Velocity	Get/Set	DINT	Sets AXIS#.FBUS.V when using Command Type 6 (Position Move) or 7 (Jog Move) while in Position Mode. Otherwise goes directly to AXIS#.VL.CMD.
8	Acceleration	Get/Set	DINT	Sets the accelerationrate to use during position and velocity moves. Sets AXIS#.FBUS.ACC.
9	Deceleration	Get/Set	DINT	Sets the deceleration rate to use during position and velocity moves. Sets AXIS#.FBUS.DEC.
10	Incremental Position Flag	Get/Set	BOOL	Incremental Position Flag 0: = absolute, 1: = incremental.
11	Load Data/Profile Handshake	Get/Set	BOOL	Used to Load Command Data, Start a Profile Move, and indicate that a Profile Move is in progress.
17	Enable	Get/Set	BOOL	Enable Output (same as AXIS#.EN and AXIS#.DIS).
25	Torque	Get/Set	DINT	Output torque in milliamps [mA]. Sets AXIS#.FBUS.IL.CMD. This is only used when AXIS#.OPMODE is in Torque Mode and AXIS#.CMDSOURCE is fieldbus (i.e.: Command Type 5 - Torque Move).

Attribute ID (Decimal Value)	Name	Access Rule	Type	Description
58	Load Data Complete	Get/Set	BOOL	Indicates that valid data for a valid I/O Command Message type has been loaded into the position controller device.
100	Home Mode	Get/Set	INT	See AXIS#.HOME.MODE in WorkBench Online Help.
101	Home Move	Set	BOOL	Initiate a Home Move. See AXIS#.HOME.MOVE.

12 Appendix B: EtherNet/IP Objects List

The parameters in this list correspond to drive parameters available in WorkBench and are described in the WorkBench help documentation and the WorkBench Online Help..

- Position values are scaled according to AXIS#.EIP.POSUNIT.
- Velocity and Acceleration values are scaled according to AXIS#.EIP.PROFUNIT.
- Other floating point values are multiplied by 1000, such that a value displayed in WorkBench as 1.001 will be transmitted through EtherNet/IP as 1001.

Name	ID	Hex	Data Type	Access
DRV.DIS	2000	7D0	Unsigned8	Read/Write
DRV.BLINKDISPLAY	2001	7D1	Unsigned8	Read/Write
DRV.CLRFAULTS	2002	7D2	Unsigned8	Read/Write
DRV.TYPE	2003	7D3	Unsigned8	Read Only
DRV.RSTVAR	2004	7D4	Unsigned8	Read/Write
DRV.NVSAVE	2005	7D5	Unsigned8	Read/Write
DRV.STOP	2006	7D6	Unsigned8	Read/Write
DRV.NVLOAD	2007	7D7	Unsigned8	Read/Write
DRV.RUNTIME	2008	7D8	Unsigned32	Read Only
DRV.NVCHECK	2009	7D9	Unsigned32	Read Only
DRV.CUSTOMIDENTIFIER	2010	7DA	Unknown	Read/Write
DRV.NAME	2011	7DB	Unknown	Read/Write
DRV.NVVER	2012	7DC	Unknown	Read/Write
DRV.REBOOT	2013	7DD	Unsigned32	Read/Write
DRV.DOWNLOADALLOWED	2014	7DE	Unsigned8	Read Only
DRV.ADDUSER	2015	7DF	Unknown	Read/Write
DRV.DELUSER	2016	7E0	Unknown	Read/Write
DRV.LOGOUT	2017	7E1	Unsigned8	Read/Write
DRV.SETUSERPWD	2018	7E2	Unknown	Read/Write
DRV.TEMPFTHRESH	2019	7E3	Unsigned8	Read Only
DRV.TEMPWTHRESH	2020	7E4	Unsigned8	Read Only
IP.DEFAULTINTERFACE	2200	898	Unsigned8	Read/Write
IP.MODE	2201	899	Unsigned16	Read/Write
IP.ADDRESS	2202	89A	Unsigned32	Read/Write
IP.GATEWAY	2203	89B	Unsigned32	Read/Write
IP.PROTOCOL	2204	89C	Unsigned8	Read/Write
IP.RESET	2205	89D	Unsigned8	Read/Write
IP.SUBNET	2206	89E	Unsigned32	Read/Write
REC.ACTIVE	2300	8FC	Unsigned8	Read Only
REC.DONE	2301	8FD	Unsigned8	Read Only
REC.GAP	2302	8FE	Unsigned16	Read/Write
REC.NUMPOINTS	2303	8FF	Unsigned16	Read/Write
REC.OFF	2304	900	Unsigned8	Read/Write
REC.STOPTYPE	2305	901	Unsigned8	Read/Write
REC.TRIG	2306	902	Unsigned8	Read/Write
REC.TRIGPOS	2307	903	Unsigned8	Read/Write
REC.RETRIEVESIZE	2308	904	Unsigned16	Read/Write
REC.TRIGSLOPE	2309	905	Unsigned8	Read/Write
REC.TRIGTYPE	2310	906	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
REC.CH1	2312	908	Unknown	Read/Write
REC.CH2	2313	909	Unknown	Read/Write
REC.CH3	2314	90A	Unknown	Read/Write
REC.CH4	2315	90B	Unknown	Read/Write
REC.CH5	2316	90C	Unknown	Read/Write
REC.CH6	2317	90D	Unknown	Read/Write
REC.RETRIEVEFRMT	2318	90E	Unsigned8	Read/Write
REC.TRIGMASK	2319	90F	Unsigned32	Read/Write
REC.TRIGPARAM	2320	910	Unknown	Read/Write
REGEN.POWER	2400	960	Unsigned32	Read Only
REGEN.REXT	2401	961	Unsigned16	Read/Write
REGEN.TEXT	2402	962	Unsigned32	Read/Write
REGEN.TYPE	2403	963	Signed8	Read/Write
REGEN.WATTEXT	2404	964	Unsigned16	Read/Write
REGEN.POWERFILTERED	2405	965	Unsigned32	Read Only
VBUS.VALUE	2500	9C4	Signed32	Read Only
VBUS.CAP	2501	9C5	Unsigned32	Read Only
VBUS.DCOPERATION	2502	9C6	Unsigned8	Read/Write
VBUS.ICAP	2503	9C7	Signed32	Read Only
VBUS.ICAPLIMIT	2504	9C8	Signed32	Read Only
VBUS.INRUSHOFF	2505	9C9	Unsigned16	Read Only
VBUS.INRUSHON	2506	9CA	Unsigned16	Read Only
VBUS.OVFTHRESH	2507	9CB	Unsigned16	Read Only
VBUS.OVWTHRESH	2508	9CC	Unsigned16	Read/Write
VBUS.THREEPHASE	2509	9CD	Unsigned8	Read/Write
VBUS.UVFTHRESH	2510	9CE	Unsigned16	Read/Write
VBUS.UVWTHRESH	2511	9CF	Unsigned16	Read/Write
VBUS.UVMODE	2512	9D0	Unsigned8	Read/Write
VBUS.ACNOMINAL	2513	9D1	Unsigned16	Read/Write
VBUS.DCNOMINAL	2514	9D2	Unsigned16	Read/Write
FBUS.TYPE	2600	A28	Unsigned8	Read Only
DIN.STATES	2700	A8C	Unsigned32	Read Only
DIO.STATES	2701	A8D	Unsigned32	Read Only
DOUT.STATES	2702	A8E	Unsigned32	Read Only
FBUS.DOUT.STATES	2703	A8F	Unsigned32	Read/Write
MW.USERBUFFER	2800	AF0	Signed32[32]	Read/Write
MW.MODEL1.STATE	2801	AF1	Unsigned8	Read/Write
MW.MODEL2.STATE	2802	AF2	Unsigned8	Read/Write
HW.FANSPEED1	2900	B54	Signed32	Read Only
GANTRY.PL.ERR	3000	BB8	Signed32	Read Only
GANTRY.PL.ERRFTHRESH	3001	BB9	Signed32	Read/Write
GANTRY.PL.ERRWTHRESH	3002	BBA	Signed32	Read/Write
GANTRY.STATE	3003	BBB	Unsigned8	Read Only
GANTRY.HOME.REQUIRED	3004	BBC	Unsigned8	Read/Write
GANTRY.PL.ERRTTHRESH	3005	BBD	Unsigned16	Read/Write
BRAKE1.AXIS	3100	C1C	Unsigned8	Read/Write
BRAKE2.AXIS	3101	C1D	Unsigned8	Read/Write
BRAKE1.STATE	3105	C21	Unsigned8	Read Only
BRAKE2.STATE	3106	C22	Unsigned8	Read Only

Name	ID	Hex	Data Type	Access
USER.INT1	3200	C80	Signed16	Read/Write
USER.INT2	3201	C81	Signed16	Read/Write
USER.INT3	3202	C82	Signed16	Read/Write
USER.INT4	3203	C83	Signed16	Read/Write
USER.INT5	3204	C84	Signed16	Read/Write
USER.INT6	3205	C85	Signed16	Read/Write
USER.INT7	3206	C86	Signed16	Read/Write
USER.INT8	3207	C87	Signed16	Read/Write
USER.INT9	3208	C88	Signed16	Read/Write
USER.INT10	3209	C89	Signed16	Read/Write
EIP.SAMPLEPERIOD	3302	CE6	Unsigned32	Read/Write
SD.LOGEN	3400	D48	Unsigned8	Read/Write
SD.STATUS	3401	D49	Unsigned8	Read Only
LOG.SOURCE	3500	DAC	Unsigned32	Read/Write
X22.MODE	3600	E10	Unsigned8	Read/Write
X23.MODE	3601	E11	Unsigned8	Read/Write
MODBUS.CLRERRORS	3700	E74	Unsigned8	Read/Write
MODBUS.EN	3702	E76	Unsigned8	Read/Write
MODBUS.ENDIAN	3703	E77	Unsigned8	Read/Write
MODBUS.ERRORCOUNT	3704	E78	Unsigned16	Read Only
MODBUS.ERRORMODE	3705	E79	Unsigned8	Read/Write
MODBUS.KEEPALIVE	3706	E7A	Unsigned8	Read/Write
MODBUS.MSGLOG	3707	E7B	Unsigned8	Read/Write
MODBUS.RSTMAP	3708	E7C	Unsigned8	Read/Write
MODBUS.WATCHDOG	3709	E7D	Unsigned16	Read/Write
AXIS#.NAME	5000	1388	Unknown	Read/Write
AXIS#.ZEROT	5001	1389	Unsigned32	Read/Write
AXIS#.ZEROV	5002	138A	Signed32	Read/Write
AXIS#.ACTIVE	5003	138B	Unsigned8	Read Only
AXIS#.CMDSOURCE	5004	138C	Unsigned8	Read/Write
AXIS#.DBILIMIT	5005	138D	Unsigned32	Read/Write
AXIS#.DIR	5006	138E	Unsigned8	Read/Write
AXIS#.DIS	5007	138F	Unsigned8	Read/Write
AXIS#.DISMODE	5008	1390	Unsigned8	Read/Write
AXIS#.DISSOURCES	5009	1391	Unsigned16	Read Only
AXIS#.DISTO	5010	1392	Unsigned32	Read/Write
AXIS#.EN	5011	1393	Unsigned8	Read/Write
AXIS#.ENDEFAULT	5012	1394	Unsigned8	Read/Write
AXIS#.ICONT	5013	1395	Signed32	Read Only
AXIS#.IPEAK	5014	1396	Signed32	Read Only
AXIS#.OPMODE	5015	1397	Unsigned8	Read/Write
AXIS#.STOP	5016	1398	Unsigned8	Read/Write
AXIS#.SETUPREQBITS	5017	1399	Unsigned32	Read Only
AXIS#.WARNING1	5018	139A	Unsigned32	Read Only
AXIS#.WARNING2	5019	139B	Unsigned32	Read Only
AXIS#.WARNING3	5020	139C	Unsigned32	Read Only
AXIS#.MOTIONDISSOURCES	5021	139D	Unsigned16	Read Only
AXIS#.CLRFAULTS	5022	139E	Unsigned8	Read/Write
AXIS#.DISSOURCESMASK	5023	139F	Unsigned16	Read Only

Name	ID	Hex	Data Type	Access
AXIS#.FAULTED	5024	13A0	Unsigned8	Read Only
AXIS#.MOTIONSTAT	5025	13A1	Unsigned32	Read Only
AXIS#.TEMP	5026	13A2	Signed32	Read Only
AXIS#.TEMPFTHRESH	5027	13A3	Signed32	Read Only
AXIS#.TEMPWTHRESH	5028	13A4	Signed32	Read Only
AXIS#.UTFTHRESH	5029	13A5	Signed32	Read Only
AXIS#.UTWTHRESH	5030	13A6	Signed32	Read Only
AXIS#.ZEROACC	5031	13A7	Unsigned32	Read/Write
AXIS#.ZEROREACHED	5032	13A8	Unsigned8	Read Only
AXIS#.MOTIONCONTROL	5033	13A9	Unsigned16	Read/Write
AXIS#.CLRWARNINGS	5034	13AA	Unsigned8	Read/Write
AXIS#.FBUS.ACC	5100	13EC	Unsigned32	Read/Write
AXIS#.FBUS.DEC	5101	13ED	Unsigned32	Read/Write
AXIS#.FBUS.PROTECTION	5102	13EE	Unsigned8	Read/Write
AXIS#.FBUS.BLOCKING	5103	13EF	Unsigned8	Read Only
AXIS#.FBUS.IL.CMD	5104	13F0	Signed32	Read/Write
AXIS#.FBUS.V	5105	13F1	Signed32	Read/Write
AXIS#.FBUS.P	5106	13F2	Signed32	Read/Write
AXIS#.CS.DEC	5300	14B4	Unsigned32	Read/Write
AXIS#.CS.STATE	5301	14B5	Unsigned8	Read Only
AXIS#.GUI.PARAM01	5400	1518	Signed32	Read/Write
AXIS#.GUI.PARAM02	5401	1519	Signed32	Read/Write
AXIS#.GUI.PARAM03	5402	151A	Signed32	Read/Write
AXIS#.GUI.PARAM04	5403	151B	Signed32	Read/Write
AXIS#.GUI.PARAM05	5404	151C	Signed32	Read/Write
AXIS#.GUI.PARAM06	5405	151D	Signed32	Read/Write
AXIS#.GUI.PARAM07	5406	151E	Signed32	Read/Write
AXIS#.GUI.PARAM08	5407	151F	Signed32	Read/Write
AXIS#.GUI.PARAM09	5408	1520	Signed32	Read/Write
AXIS#.GUI.PARAM10	5409	1521	Signed32	Read/Write
AXIS#.GEAR.ACC	5500	157C	Unsigned32	Read/Write
AXIS#.GEAR.DEC	5501	157D	Unsigned32	Read/Write
AXIS#.GEAR.IN	5502	157E	Unsigned16	Read/Write
AXIS#.GEAR.MOVE	5503	157F	Unsigned8	Read/Write
AXIS#.GEAR.OUT	5504	1580	Signed16	Read/Write
AXIS#.GEAR.FBSOURCE	5505	1581	Unsigned8	Read/Write
AXIS#.GEAR.STATE	5506	1582	Signed16	Read Only
AXIS#.GEAR.AUTOSTART	5507	1583	Unsigned8	Read/Write
AXIS#.HWLS.NEGSTATE	5600	15E0	Unsigned8	Read Only
AXIS#.HWLS.POSSTATE	5601	15E1	Unsigned8	Read Only
AXIS#.HWLS.NEGSOURCE	5602	15E2	Unsigned8	Read/Write
AXIS#.HWLS.POSSOURCE	5603	15E3	Unsigned8	Read/Write
AXIS#.LOAD.INERTIA	5700	1644	Unsigned32	Read/Write
AXIS#.FBUS.IL.FF	5800	16A8	Signed32	Read/Write
AXIS#.IL.CMD	5801	16A9	Signed32	Read Only
AXIS#.IL.CMDU	5802	16AA	Signed16	Read/Write
AXIS#.IL.FB	5803	16AB	Signed32	Read Only
AXIS#.IL.FF	5804	16AC	Signed32	Read Only
AXIS#.IL.FOLDFTHRESH	5805	16AD	Signed32	Read Only

Name	ID	Hex	Data Type	Access
AXIS#.IL.FOLDFTHRESHU	5806	16AE	Signed32	Read/Write
AXIS#.IL.FOLDWTHRESH	5807	16AF	Signed32	Read/Write
AXIS#.IL.FRICTION	5808	16B0	Signed32	Read/Write
AXIS#.IL.IFOLD	5809	16B1	Signed32	Read Only
AXIS#.IL.KACCFF	5810	16B2	Signed32	Read/Write
AXIS#.IL.AINSOURCE	5811	16B3	Unsigned8	Read/Write
AXIS#.IL.KVFF	5812	16B4	Signed32	Read/Write
AXIS#.IL.LIMITN	5813	16B5	Signed32	Read/Write
AXIS#.IL.LIMITP	5814	16B6	Signed32	Read/Write
AXIS#.IL.MIFOLD	5815	16B7	Signed32	Read Only
AXIS#.IL.OFFSET	5816	16B8	Signed32	Read/Write
AXIS#.IL.VCMD	5817	16B9	Signed32	Read Only
AXIS#.IL.BW	5818	16BA	Unsigned16	Read/Write
AXIS#.IL.KP	5819	16BB	Unsigned32	Read Only
AXIS#.IL.DIFOLD	5820	16BC	Signed32	Read Only
AXIS#.IL.FBSOURCE	5821	16BD	Unsigned8	Read/Write
AXIS#.IL.CMDACC	5822	16BE	Signed32	Read Only
AXIS#.IL.PWMFREQ	5823	16BF	Unsigned32	Read Only
AXIS#.IL.KPLOOKUP	5824	16C0	Unsigned32[256]	Read Only
AXIS#.IL.DI2T	5825	16C1	Signed32	Read Only
AXIS#.IL.AINSCALE	5826	16C2	Signed32	Read/Write
AXIS#.IL.MI2T	5827	16C3	Signed32	Read Only
AXIS#.IL.PWMQUIET	5828	16C4	Unsigned8	Read/Write
AXIS#.FBUS.IL.LIMIT	5829	16C5	Signed32	Read/Write
AXIS#.PL.CMD	5900	170C	Signed32	Read Only
AXIS#.PL.ERR	5901	170D	Signed32	Read Only
AXIS#.PL.ERRFTHRESH	5902	170E	Signed32	Read/Write
AXIS#.PL.ERRWTHRESH	5903	170F	Signed32	Read/Write
AXIS#.PL.FB	5904	1710	Signed32	Read Only
AXIS#.PL.FBSOURCE	5905	1711	Unsigned8	Read/Write
AXIS#.PL.INTOUTMAX	5906	1712	Signed32	Read/Write
AXIS#.PL.KI	5907	1713	Signed32	Read/Write
AXIS#.PL.KP	5908	1714	Signed32	Read/Write
AXIS#.PL.MODP1	5909	1715	Signed32	Read/Write
AXIS#.PL.MODP2	5910	1716	Signed32	Read/Write
AXIS#.PL.MODPDIR	5911	1717	Unsigned8	Read/Write
AXIS#.PL.MODPEN	5912	1718	Unsigned8	Read/Write
AXIS#.PL.AINSCALE	5913	1719	Signed32	Read/Write
AXIS#.PL.AINSOURCE	5915	171B	Unsigned8	Read/Write
AXIS#.PL.KITHRESH	5916	171C	Signed32	Read/Write
AXIS#.PL.OFFSET	5917	171D	Signed32	Read/Write
AXIS#.VL.ARPF1	6000	1770	Signed32	Read/Write
AXIS#.VL.ARPF2	6001	1771	Signed32	Read/Write
AXIS#.VL.ARPF3	6002	1772	Signed32	Read/Write
AXIS#.VL.ARPF4	6003	1773	Signed32	Read/Write
AXIS#.VL.ARPQ1	6004	1774	Signed32	Read/Write
AXIS#.VL.ARPQ2	6005	1775	Signed32	Read/Write
AXIS#.VL.ARPQ3	6006	1776	Signed32	Read/Write
AXIS#.VL.ARPQ4	6007	1777	Signed32	Read/Write

Name	ID	Hex	Data Type	Access
AXIS#.VL.ARTYPE1	6008	1778	Signed8	Read/Write
AXIS#.VL.ARTYPE2	6009	1779	Signed8	Read/Write
AXIS#.VL.ARTYPE3	6010	177A	Signed8	Read/Write
AXIS#.VL.ARTYPE4	6011	177B	Signed8	Read/Write
AXIS#.VL.ARZF1	6012	177C	Signed32	Read/Write
AXIS#.VL.ARZF2	6013	177D	Signed32	Read/Write
AXIS#.VL.ARZF3	6014	177E	Signed32	Read/Write
AXIS#.VL.ARZF4	6015	177F	Signed32	Read/Write
AXIS#.VL.ARZQ1	6016	1780	Signed32	Read/Write
AXIS#.VL.ARZQ2	6017	1781	Signed32	Read/Write
AXIS#.VL.ARZQ3	6018	1782	Signed32	Read/Write
AXIS#.VL.ARZQ4	6019	1783	Signed32	Read/Write
AXIS#.FBUS.VL.FF	6020	1784	Signed32	Read/Write
AXIS#.VL.CMD	6021	1785	Signed32	Read Only
AXIS#.VL.CMDU	6022	1786	Signed32	Read/Write
AXIS#.VL.ERR	6023	1787	Signed32	Read Only
AXIS#.VL.FB	6024	1788	Signed32	Read Only
AXIS#.VL.FBFILTER	6025	1789	Signed32	Read Only
AXIS#.VL.FBSOURCE	6026	178A	Unsigned8	Read/Write
AXIS#.VL.FF	6027	178B	Signed32	Read Only
AXIS#.VL.KI	6028	178C	Signed32	Read/Write
AXIS#.VL.KP	6029	178D	Signed32	Read/Write
AXIS#.VL.KVFF	6030	178E	Signed32	Read/Write
AXIS#.VL.LIMITN	6031	178F	Signed32	Read/Write
AXIS#.VL.LIMITP	6032	1790	Signed32	Read/Write
AXIS#.VL.LMJR	6033	1791	Unsigned32	Read/Write
AXIS#.VL.THRESH	6034	1792	Signed32	Read/Write
AXIS#.VL.FBUNFILTERED	6035	1793	Signed32	Read Only
AXIS#.VL.VFTHRESH	6036	1794	Signed32	Read Only
AXIS#.VL.AINSCALE	6037	1795	Unsigned32	Read/Write
AXIS#.VL.AINACC	6038	1796	Signed32	Read/Write
AXIS#.VL.AINDEC	6039	1797	Signed32	Read/Write
AXIS#.VL.AINSOURCE	6040	1798	Unsigned8	Read/Write
AXIS#.VL.ACCFILTERED	6041	1799	Signed32	Read Only
AXIS#.VL.KIMODE	6042	179A	Unsigned8	Read/Write
AXIS#.HOME.ACC	6100	17D4	Unsigned32	Read/Write
AXIS#.HOME.AUTOMOVE	6101	17D5	Unsigned8	Read/Write
AXIS#.HOME.DEC	6102	17D6	Unsigned32	Read/Write
AXIS#.HOME.DIR	6103	17D7	Unsigned16	Read/Write
AXIS#.HOME.DIST	6104	17D8	Signed32	Read/Write
AXIS#.HOME.CREEPFACTOR	6105	17D9	Unsigned32	Read/Write
AXIS#.HOME.IPEAK	6106	17DA	Signed32	Read/Write
AXIS#.HOME.MODE	6107	17DB	Unsigned16	Read/Write
AXIS#.HOME.MOVE	6108	17DC	Unsigned8	Read/Write
AXIS#.HOME.P	6109	17DD	Signed32	Read/Write
AXIS#.HOME.PERRTHRESH	6110	17DE	Signed32	Read/Write
AXIS#.HOME.SET	6111	17DF	Unsigned8	Read/Write
AXIS#.HOME.V	6112	17E0	Signed32	Read/Write
AXIS#.HOME.MAXDIST	6113	17E1	Signed32	Read/Write

Name	ID	Hex	Data Type	Access
AXIS#.HOME.CLEAR	6114	17E2	Unsigned8	Read/Write
AXIS#.HOME.MULTITURNMODE	6115	17E3	Unsigned8	Read/Write
AXIS#.HOME.OFFSET	6116	17E4	Signed32	Read Only
AXIS#.HOME.OFFSETUSER	6117	17E5	Signed32	Read/Write
AXIS#.HOME.SWITCHSOURCE	6118	17E6	Unsigned8	Read/Write
AXIS#.HOME.SWITCHSTATE	6119	17E7	Unsigned8	Read Only
AXIS#.MOTOR.AUTOSET	6200	1838	Unsigned8	Read/Write
AXIS#.MOTOR.BRAKE	6201	1839	Unsigned8	Read Only
AXIS#.MOTOR.BRAKECONTROL	6202	183A	Unsigned8	Read/Write
AXIS#.MOTOR.CTF0	6203	183B	Unsigned32	Read Only
AXIS#.MOTOR.ICONT	6204	183C	Unsigned32	Read Only
AXIS#.MOTOR.IDDATAVALID	6205	183D	Unsigned8	Read Only
AXIS#.MOTOR.INERTIA	6206	183E	Unsigned32	Read Only
AXIS#.MOTOR.IPEAK	6207	183F	Unsigned32	Read Only
AXIS#.MOTOR.KT	6208	1840	Unsigned32	Read Only
AXIS#.MOTOR.LQLL	6209	1841	Unsigned32	Read Only
AXIS#.MOTOR.PHASE	6210	1842	Unsigned16	Read/Write
AXIS#.MOTOR.PITCH	6211	1843	Unsigned32	Read/Write
AXIS#.MOTOR.POLES	6212	1844	Unsigned16	Read Only
AXIS#.MOTOR.R	6213	1845	Unsigned32	Read Only
AXIS#.MOTOR.RTYPE	6214	1846	Unsigned8	Read/Write
AXIS#.MOTOR.TBRAKEAPP	6215	1847	Unsigned16	Read Only
AXIS#.MOTOR.TBRAKERLS	6216	1848	Unsigned16	Read Only
AXIS#.MOTOR.TEMP	6217	1849	Unsigned32	Read Only
AXIS#.MOTOR.TEMPFAULT	6218	184A	Unsigned32	Read Only
AXIS#.MOTOR.TEMPWARN	6219	184B	Unsigned32	Read/Write
AXIS#.MOTOR.TYPE	6220	184C	Unsigned8	Read Only
AXIS#.MOTOR.VMAX	6221	184D	Unsigned16	Read Only
AXIS#.MOTOR.VOLTMAX	6222	184E	Unsigned16	Read Only
AXIS#.MOTOR.TBRAKETO	6223	184F	Signed32	Read/Write
AXIS#.MOTOR.BRAKEIMM	6224	1850	Unsigned8	Read/Write
AXIS#.MOTOR.VOLTMIN	6225	1851	Unsigned16	Read/Write
AXIS#.MOTOR.VOLTRATED	6226	1852	Unsigned16	Read/Write
AXIS#.MOTOR.VRATED	6227	1853	Signed32	Read/Write
AXIS#.MOTOR.IMTR	6228	1854	Unsigned16	Read/Write
AXIS#.MOTOR.IMID	6229	1855	Unsigned16	Read/Write
AXIS#.MOTOR.LDLL	6230	1856	Unsigned32	Read Only
AXIS#.MOTOR.LISAT	6231	1857	Unsigned32	Read Only
AXIS#.MOTOR.IDMAX	6232	1858	Unsigned32	Read Only
AXIS#.MOTOR.PHSADVK1	6233	1859	Signed32	Read Only
AXIS#.MOTOR.PHSADVK2	6234	185A	Signed32	Read Only
AXIS#.MOTOR.TEMPC	6235	185B	Signed16	Read Only
AXIS#.MOTOR.FIELDWEAKENING	6236	185C	Unsigned8	Read/Write
AXIS#.MOTOR.NAME	6237	185D	Unknown	Read Only
AXIS#.MOTOR.BRAKEPOWERDELAY	6238	185E	Unsigned16	Read Only
AXIS#.MOTOR.BRAKEPOWERLOW	6239	185F	Unsigned16	Read Only
AXIS#.MOTOR.BRAKEPOWERSAVING	6240	1860	Unsigned8	Read Only
AXIS#.MOTOR.KE	6241	1861	Unsigned32	Read Only
AXIS#.MOTOR.SERIALNUM	6242	1862	Unknown	Read Only

Name	ID	Hex	Data Type	Access
AXIS#.MOTOR.RSOURCE	6243	1863	Unsigned8	Read/Write
AXIS#.MOTOR.TEMPSOURCE	6244	1864	Unsigned8	Read/Write
AXIS#.MT.ACC	6300	189C	Unsigned32[32]	Read/Write
AXIS#.MT.CLEAR	6301	189D	Unsigned8[32]	Read/Write
AXIS#.MT.CNTL	6302	189E	Unsigned32[32]	Read/Write
AXIS#.MT.CONTINUE	6303	189F	Unsigned8	Read/Write
AXIS#.MT.DEC	6304	18A0	Unsigned32[32]	Read/Write
AXIS#.MT.MOVE	6305	18A1	Unsigned8[32]	Read/Write
AXIS#.MT.MTNEXT	6306	18A2	Signed8[32]	Read/Write
AXIS#.MT.P	6307	18A3	Signed32[32]	Read/Write
AXIS#.MT.TNEXT	6308	18A4	Unsigned16[32]	Read/Write
AXIS#.MT.V	6309	18A5	Signed32[32]	Read/Write
AXIS#.MT.VCMD	6310	18A6	Signed32	Read Only
AXIS#.MT.FEEDRATE	6311	18A7	Unsigned32	Read/Write
AXIS#.MT.CAP	6312	18A8	Unsigned8[32]	Read/Write
AXIS#.MT.CLEARALL	6313	18A9	Unsigned8	Read/Write
AXIS#.MT.DISALLOWINTERRUPT	6314	18AA	Unsigned8[32]	Read/Write
AXIS#.MT.DISALLOWZEROSTARTVEL	6315	18AB	Unsigned8[32]	Read/Write
AXIS#.MT.RUNNINGTASK	6316	18AC	Signed8	Read Only
AXIS#.MT.TRANSITION	6317	18AD	Unsigned8[32]	Read/Write
AXIS#.MT.TYPE	6318	18AE	Unsigned8[32]	Read/Write
AXIS#.SM.I1	6400	1900	Signed32	Read/Write
AXIS#.SM.I2	6401	1901	Signed32	Read/Write
AXIS#.SM.MODE	6402	1902	Unsigned16	Read/Write
AXIS#.SM.MOVE	6403	1903	Unsigned8	Read/Write
AXIS#.SM.T1	6404	1904	Unsigned16	Read/Write
AXIS#.SM.T2	6405	1905	Unsigned16	Read/Write
AXIS#.SM.V1	6406	1906	Signed32	Read/Write
AXIS#.SM.V2	6407	1907	Signed32	Read/Write
AXIS#.SM.ACC	6408	1908	Unsigned32	Read/Write
AXIS#.SM.DEC	6409	1909	Unsigned32	Read/Write
AXIS#.SWLS.EN	6500	1964	Unsigned8	Read/Write
AXIS#.SWLS.LIMIT0	6501	1965	Signed32	Read/Write
AXIS#.SWLS.LIMIT1	6502	1966	Signed32	Read/Write
AXIS#.SWLS.STATE	6503	1967	Unsigned8	Read Only
AXIS#.UNIT.ACCLINEAR	6600	19C8	Unsigned8	Read/Write
AXIS#.UNIT.ACCROTARY	6601	19C9	Unsigned8	Read/Write
AXIS#.UNIT.PIN	6602	19CA	Unsigned32	Read/Write
AXIS#.UNIT.PLINEAR	6603	19CB	Unsigned8	Read/Write
AXIS#.UNIT.POUT	6604	19CC	Unsigned32	Read/Write
AXIS#.UNIT.PROTARY	6605	19CD	Unsigned8	Read/Write
AXIS#.UNIT.VLINEAR	6606	19CE	Unsigned8	Read/Write
AXIS#.UNIT.VROTARY	6607	19CF	Unsigned8	Read/Write
AXIS#.UNIT.LABEL	6608	19D0	Unknown	Read/Write
AXIS#.WS.ARM	6700	1A2C	Unsigned8	Read/Write
AXIS#.WS.DISTMAX	6701	1A2D	Signed32	Read/Write
AXIS#.WS.DISTMIN	6702	1A2E	Signed32	Read/Write
AXIS#.WS.IMAX	6703	1A2F	Signed32	Read/Write
AXIS#.WS.MODE	6704	1A30	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
AXIS#.WS.NUMLOOPS	6705	1A31	Unsigned8	Read/Write
AXIS#.WS.STATE	6706	1A32	Unsigned8	Read Only
AXIS#.WS.T	6707	1A33	Unsigned8	Read/Write
AXIS#.WS.TDELAY1	6708	1A34	Unsigned8	Read/Write
AXIS#.WS.TDELAY2	6709	1A35	Unsigned8	Read/Write
AXIS#.WS.TDELAY3	6710	1A36	Unsigned16	Read/Write
AXIS#.WS.VTHRESH	6711	1A37	Signed32	Read/Write
AXIS#.WS.DISARM	6712	1A38	Unsigned8	Read/Write
AXIS#.WS.FREQ	6713	1A39	Unsigned32	Read/Write
AXIS#.WS.TDELAY4	6714	1A3A	Unsigned16	Read/Write
AXIS#.WS.CHECKT	6715	1A3B	Unsigned16	Read/Write
AXIS#.WS.CHECKV	6716	1A3C	Signed32	Read/Write
AXIS#.WS.TSTANDSTILL	6717	1A3D	Unsigned16	Read/Write
AXIS#.WS.TIRAMP	6718	1A3E	Unsigned16	Read/Write
AXIS#.WS.CHECKMODE	6719	1A3F	Unsigned8	Read/Write
AXIS#.FAULT1	6800	1A90	Unsigned16	Read Only
AXIS#.FAULT2	6801	1A91	Unsigned16	Read Only
AXIS#.FAULT3	6802	1A92	Unsigned16	Read Only
AXIS#.FAULT4	6803	1A93	Unsigned16	Read Only
AXIS#.FAULT5	6804	1A94	Unsigned16	Read Only
AXIS#.FAULT6	6805	1A95	Unsigned16	Read Only
AXIS#.FAULT7	6806	1A96	Unsigned16	Read Only
AXIS#.FAULT8	6807	1A97	Unsigned16	Read Only
AXIS#.FAULT9	6808	1A98	Unsigned16	Read Only
AXIS#.FAULT10	6809	1A99	Unsigned16	Read Only
AXIS#.JOG.ACC	6900	1AF4	Unsigned32	Read/Write
AXIS#.JOG.DEC	6901	1AF5	Unsigned32	Read/Write
AXIS#.JOG.MOVEN	6902	1AF6	Unsigned8	Read/Write
AXIS#.JOG.MOVEP	6903	1AF7	Unsigned8	Read/Write
AXIS#.JOG.V	6904	1AF8	Signed32	Read/Write
AXIS#.HWEN.MODE	7000	1B58	Unsigned8	Read/Write
AXIS#.HWEN.SOURCE	7001	1B59	Unsigned8	Read/Write
AXIS#.HWEN.STATE	7002	1B5A	Unsigned8	Read Only
AXIS#.SETTLE.P	7100	1BBC	Signed32	Read/Write
AXIS#.SETTLE.V	7101	1BBD	Signed32	Read/Write
AXIS#.FAULT6004.ACTION	7200	1C20	Unsigned8	Read/Write
AXIS#.FIELDWEAKENING.LOOPBW	7300	1C84	Signed32	Read/Write
AXIS#.FIELDWEAKENING.CURRFILTERBW	7301	1C85	Signed32	Read/Write
AXIS#.FIELDWEAKENING.VOLTFILTERBW	7302	1C86	Signed32	Read/Write
AXIS#.FIELDWEAKENING.VBUSMARGIN	7303	1C87	Signed32	Read/Write
AXIS#.SENSORLESS.BWU	7400	1CE8	Signed32	Read/Write
AXIS#.SENSORLESS.BW	7401	1CE9	Signed32	Read Only
AXIS#.SENSORLESS.FAULTANGLE	7402	1CEA	Signed32	Read/Write
AXIS#.SENSORLESS.FAULTTIME	7403	1CEB	Signed32	Read/Write
AXIS#.SENSORLESS.RPMSTART	7404	1CEC	Signed32	Read/Write
AXIS#.SENSORLESS.ISTART	7405	1CED	Signed32	Read/Write
AXIS#.SENSORLESS.ENPHASELEAD	7406	1CEE	Unsigned8	Read/Write
AXIS#.EIP.POSUNIT	7500	1D4C	Unsigned32	Read/Write
AXIS#.EIP.PROFUNIT	7501	1D4D	Unsigned32	Read/Write

Name	ID	Hex	Data Type	Access
AXIS#.EIP.CMD.CONTROL1	7502	1D4E	Unsigned8	Read/Write
AXIS#.EIP.CMD.CONTROL2	7503	1D4F	Unsigned8	Read/Write
AXIS#.EIP.RSP.STATUS1	7504	1D50	Unsigned8	Read Only
AXIS#.EIP.RSP.STATUS2	7505	1D51	Unsigned8	Read Only
AXIS#.EIP.RSP.STATUS3	7506	1D52	Unsigned8	Read Only
AXIS#.EIP.OPMODE	7507	1D53	Unsigned8	Read/Write
AXIS#.EIP.DYNAMICCMDMAP	7508	1D54	Unsigned16[16]	Read/Write
AXIS#.EIP.DYNAMICRSPMAP	7509	1D55	Unsigned16[16]	Read/Write
AXIS#.EIP.DYNAMICCMDDATA	7510	1D56	Unsigned32[16]	Read Only
AXIS#.EIP.DYNAMICRSPDATA	7511	1D57	Unsigned32[16]	Read Only
AXIS#.EIP.CMD.BLOCKNUM	7512	1D58	Unsigned8	Read/Write
AXIS#.EIP.CMD.CMDTYPE	7513	1D59	Unsigned8	Read/Write
AXIS#.EIP.CMD.RSPTYPE	7514	1D5A	Unsigned8	Read/Write
AXIS#.EIP.CMD.DATA	7515	1D5B	Unsigned32	Read/Write
AXIS#.EIP.CMD.P	7516	1D5C	Signed32	Read/Write
AXIS#.EIP.CMD.V	7517	1D5D	Unsigned32	Read/Write
AXIS#.EIP.CMD.ACC	7518	1D5E	Unsigned32	Read/Write
AXIS#.EIP.CMD.DEC	7519	1D5F	Unsigned32	Read/Write
AXIS#.EIP.CMD.DATA2	7520	1D60	Unsigned32	Read/Write
AXIS#.EIP.CMD.ATTRIBUTE	7521	1D61	Unsigned8	Read/Write
AXIS#.EIP.RSP.BLOCKNUM	7522	1D62	Unsigned8	Read Only
AXIS#.EIP.RSP.RSPTYPE	7523	1D63	Unsigned8	Read Only
AXIS#.EIP.RSP.DATA	7524	1D64	Unsigned32	Read Only
AXIS#.EIP.RSP.P	7525	1D65	Unsigned32	Read Only
AXIS#.EIP.RSP.V	7526	1D66	Unsigned32	Read Only
AXIS#.EIP.RSP.MOTIONSTAT	7527	1D67	Unsigned32	Read Only
AXIS#.EIP.RSP.DATA2	7528	1D68	Unsigned32	Read Only
AXIS#.EIP.RSP.ATTRIBUTE	7529	1D69	Unsigned8	Read Only
AXIS#.EIP.PADBYTE	7530	1D6A	Unsigned8	Read/Write
AXIS#.EIP.FIXEDCMDMAP	7531	1D6B	Unsigned16[32]	Read Only
AXIS#.EIP.FIXEDRSPMAP	7532	1D6C	Unsigned16[32]	Read Only
AXIS#.EIP.FIXEDCMDDATA	7533	1D6D	Unsigned32[32]	Read Only
AXIS#.EIP.FIXEDRSPDATA	7534	1D6E	Unsigned32[32]	Read Only
AXIS#.SAFE.STO.A	10000	2710	Unsigned8	Read Only
AXIS#.SAFE.STO.B	10001	2711	Unsigned8	Read Only
AXIS#.SAFE.STO.ACTIVE	10002	2712	Unsigned8	Read Only
AXIS#.SAFE.STO.REPORTFAULT	10003	2713	Unsigned8	Read/Write
DIN1.STATE	20000	4E20	Unsigned8	Read Only
DIN2.STATE	20001	4E21	Unsigned8	Read Only
DIN3.STATE	20002	4E22	Unsigned8	Read Only
DIN4.STATE	20003	4E23	Unsigned8	Read Only
DIN5.STATE	20004	4E24	Unsigned8	Read Only
DIN6.STATE	20005	4E25	Unsigned8	Read Only
DIN7.STATE	20006	4E26	Unsigned8	Read Only
DIN8.STATE	20007	4E27	Unsigned8	Read Only
DIN9.STATE	20008	4E28	Unsigned8	Read Only
DIN10.STATE	20009	4E29	Unsigned8	Read Only
DIN11.STATE	20010	4E2A	Unsigned8	Read Only
DIN12.STATE	20011	4E2B	Unsigned8	Read Only

Name	ID	Hex	Data Type	Access
DIN1.INV	20050	4E52	Unsigned8	Read/Write
DIN2.INV	20051	4E53	Unsigned8	Read/Write
DIN3.INV	20052	4E54	Unsigned8	Read/Write
DIN4.INV	20053	4E55	Unsigned8	Read/Write
DIN5.INV	20054	4E56	Unsigned8	Read/Write
DIN6.INV	20055	4E57	Unsigned8	Read/Write
DIN7.INV	20056	4E58	Unsigned8	Read/Write
DIN8.INV	20057	4E59	Unsigned8	Read/Write
DIN9.INV	20058	4E5A	Unsigned8	Read/Write
DIN10.INV	20059	4E5B	Unsigned8	Read/Write
DIN11.INV	20060	4E5C	Unsigned8	Read/Write
DIN12.INV	20061	4E5D	Unsigned8	Read/Write
DIN1.FILTER	20100	4E84	Unsigned8	Read/Write
DIN2.FILTER	20101	4E85	Unsigned8	Read/Write
DIN3.FILTER	20102	4E86	Unsigned8	Read/Write
DIN4.FILTER	20103	4E87	Unsigned8	Read/Write
DIN5.FILTER	20104	4E88	Unsigned8	Read/Write
DIN6.FILTER	20105	4E89	Unsigned8	Read/Write
DIN7.FILTER	20106	4E8A	Unsigned8	Read/Write
DIN8.FILTER	20107	4E8B	Unsigned8	Read/Write
DIN9.FILTER	20108	4E8C	Unsigned8	Read/Write
DIN10.FILTER	20109	4E8D	Unsigned8	Read/Write
DIN11.FILTER	20110	4E8E	Unsigned8	Read/Write
DIN12.FILTER	20111	4E8F	Unsigned8	Read/Write
DOUT1.STATE	21000	5208	Unsigned8	Read Only
DOUT2.STATE	21001	5209	Unsigned8	Read Only
DOUT3.STATE	21002	520A	Unsigned8	Read Only
DOUT4.STATE	21003	520B	Unsigned8	Read Only
DOUT5.STATE	21004	520C	Unsigned8	Read Only
DOUT6.STATE	21005	520D	Unsigned8	Read Only
DOUT7.STATE	21006	520E	Unsigned8	Read Only
DOUT8.STATE	21007	520F	Unsigned8	Read Only
DOUT9.STATE	21008	5210	Unsigned8	Read Only
DOUT1.STATEU	21050	523A	Unsigned8	Read/Write
DOUT2.STATEU	21051	523B	Unsigned8	Read/Write
DOUT3.STATEU	21052	523C	Unsigned8	Read/Write
DOUT4.STATEU	21053	523D	Unsigned8	Read/Write
DOUT5.STATEU	21054	523E	Unsigned8	Read/Write
DOUT6.STATEU	21055	523F	Unsigned8	Read/Write
DOUT7.STATEU	21056	5240	Unsigned8	Read/Write
DOUT8.STATEU	21057	5241	Unsigned8	Read/Write
DOUT9.STATEU	21058	5242	Unsigned8	Read/Write
DOUT1.SOURCE	21150	529E	Unsigned8	Read/Write
DOUT2.SOURCE	21151	529F	Unsigned8	Read/Write
DOUT3.SOURCE	21152	52A0	Unsigned8	Read/Write
DOUT4.SOURCE	21153	52A1	Unsigned8	Read/Write
DOUT5.SOURCE	21154	52A2	Unsigned8	Read/Write
DOUT6.SOURCE	21155	52A3	Unsigned8	Read/Write
DOUT7.SOURCE	21156	52A4	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
DOUT8.SOURCE	21157	52A5	Unsigned8	Read/Write
DOUT9.SOURCE	21158	52A6	Unsigned8	Read/Write
DOUT1.SOURCEID	21200	52D0	Unsigned8	Read/Write
DOUT2.SOURCEID	21201	52D1	Unsigned8	Read/Write
DOUT3.SOURCEID	21202	52D2	Unsigned8	Read/Write
DOUT4.SOURCEID	21203	52D3	Unsigned8	Read/Write
DOUT5.SOURCEID	21204	52D4	Unsigned8	Read/Write
DOUT6.SOURCEID	21205	52D5	Unsigned8	Read/Write
DOUT7.SOURCEID	21206	52D6	Unsigned8	Read/Write
DOUT8.SOURCEID	21207	52D7	Unsigned8	Read/Write
DOUT9.SOURCEID	21208	52D8	Unsigned8	Read/Write
DIO1.STATE	22000	55F0	Unsigned8	Read Only
DIO2.STATE	22001	55F1	Unsigned8	Read Only
DIO3.STATE	22002	55F2	Unsigned8	Read Only
DIO4.STATE	22003	55F3	Unsigned8	Read Only
DIO5.STATE	22004	55F4	Unsigned8	Read Only
DIO6.STATE	22005	55F5	Unsigned8	Read Only
DIO1.DIR	22050	5622	Unsigned8	Read/Write
DIO2.DIR	22051	5623	Unsigned8	Read/Write
DIO3.DIR	22052	5624	Unsigned8	Read/Write
DIO4.DIR	22053	5625	Unsigned8	Read/Write
DIO5.DIR	22054	5626	Unsigned8	Read/Write
DIO6.DIR	22055	5627	Unsigned8	Read/Write
DIO1.INV	22100	5654	Unsigned8	Read/Write
DIO2.INV	22101	5655	Unsigned8	Read/Write
DIO3.INV	22102	5656	Unsigned8	Read/Write
DIO4.INV	22103	5657	Unsigned8	Read/Write
DIO5.INV	22104	5658	Unsigned8	Read/Write
DIO6.INV	22105	5659	Unsigned8	Read/Write
DIO1.STATEU	22150	5686	Unsigned8	Read/Write
DIO2.STATEU	22151	5687	Unsigned8	Read/Write
DIO3.STATEU	22152	5688	Unsigned8	Read/Write
DIO4.STATEU	22153	5689	Unsigned8	Read/Write
DIO5.STATEU	22154	568A	Unsigned8	Read/Write
DIO6.STATEU	22155	568B	Unsigned8	Read/Write
DIO1.SOURCE	22200	56B8	Unsigned8	Read/Write
DIO2.SOURCE	22201	56B9	Unsigned8	Read/Write
DIO3.SOURCE	22202	56BA	Unsigned8	Read/Write
DIO4.SOURCE	22203	56BB	Unsigned8	Read/Write
DIO5.SOURCE	22204	56BC	Unsigned8	Read/Write
DIO6.SOURCE	22205	56BD	Unsigned8	Read/Write
DIO1.FILTER	22250	56EA	Unsigned8	Read/Write
DIO2.FILTER	22251	56EB	Unsigned8	Read/Write
DIO3.FILTER	22252	56EC	Unsigned8	Read/Write
DIO4.FILTER	22253	56ED	Unsigned8	Read/Write
DIO5.FILTER	22254	56EE	Unsigned8	Read/Write
DIO6.FILTER	22255	56EF	Unsigned8	Read/Write
DIO1.SOURCEID	22300	571C	Unsigned8	Read/Write
DIO2.SOURCEID	22301	571D	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
DIO3.SOURCEID	22302	571E	Unsigned8	Read/Write
DIO4.SOURCEID	22303	571F	Unsigned8	Read/Write
DIO5.SOURCEID	22304	5720	Unsigned8	Read/Write
DIO6.SOURCEID	22305	5721	Unsigned8	Read/Write
DIO1.TERM	22350	574E	Unsigned8	Read/Write
DIO2.TERM	22351	574F	Unsigned8	Read/Write
DIO3.TERM	22352	5750	Unsigned8	Read/Write
DIO4.TERM	22353	5751	Unsigned8	Read/Write
DIO5.TERM	22354	5752	Unsigned8	Read/Write
DIO6.TERM	22355	5753	Unsigned8	Read/Write
AIN1.CUTOFF	23000	59D8	Unsigned32	Read/Write
AIN2.CUTOFF	23001	59D9	Unsigned32	Read/Write
AIN1.OFFSET	23050	5A0A	Signed16	Read/Write
AIN2.OFFSET	23051	5A0B	Signed16	Read/Write
AIN1.VALUE	23100	5A3C	Signed16	Read Only
AIN2.VALUE	23101	5A3D	Signed16	Read Only
AIN1.DEADBAND	23150	5A6E	Signed16	Read/Write
AIN2.DEADBAND	23151	5A6F	Signed16	Read/Write
AIN1.DEADBANDMODE	23200	5AA0	Unsigned16	Read/Write
AIN2.DEADBANDMODE	23201	5AA1	Unsigned16	Read/Write
AIN1.ZERO	23250	5AD2	Unsigned8	Read/Write
AIN2.ZERO	23251	5AD3	Unsigned8	Read/Write
AOUT1.VALUE	24000	5DC0	Signed16	Read Only
AOUT2.VALUE	24001	5DC1	Signed16	Read Only
AOUT1.SOURCE	24050	5DF2	Unsigned8	Read/Write
AOUT2.SOURCE	24051	5DF3	Unsigned8	Read/Write
AOUT1.CUTOFF	24100	5E24	Unsigned32	Read/Write
AOUT2.CUTOFF	24101	5E25	Unsigned32	Read/Write
FBUS.AOUT1.VALUE	24150	5E56	Signed16	Read/Write
FBUS.AOUT2.VALUE	24151	5E57	Signed16	Read/Write
AOUT1.OFFSET	24200	5E88	Signed16	Read/Write
AOUT2.OFFSET	24201	5E89	Signed16	Read/Write
FB1.IDENTIFIED	30000	7530	Signed16	Read Only
FB2.IDENTIFIED	30001	7531	Signed16	Read Only
FB3.IDENTIFIED	30002	7532	Signed16	Read Only
FB4.IDENTIFIED	30003	7533	Signed16	Read Only
FB5.IDENTIFIED	30004	7534	Signed16	Read Only
FB1.LASTIDENTIFIED	30020	7544	Signed16	Read/Write
FB2.LASTIDENTIFIED	30021	7545	Signed16	Read/Write
FB3.LASTIDENTIFIED	30022	7546	Signed16	Read/Write
FB4.LASTIDENTIFIED	30023	7547	Signed16	Read/Write
FB5.LASTIDENTIFIED	30024	7548	Signed16	Read/Write
FB1.SELECT	30040	7558	Signed16	Read/Write
FB2.SELECT	30041	7559	Signed16	Read/Write
FB3.SELECT	30042	755A	Signed16	Read/Write
FB4.SELECT	30043	755B	Signed16	Read/Write
FB5.SELECT	30044	755C	Signed16	Read/Write
FB1.HALLSTATE	30060	756C	Unsigned8	Read Only
FB2.HALLSTATE	30061	756D	Unsigned8	Read Only

Name	ID	Hex	Data Type	Access
FB3.HALLSTATE	30062	756E	Unsigned8	Read Only
FB4.HALLSTATE	30063	756F	Unsigned8	Read Only
FB5.HALLSTATE	30064	7570	Unsigned8	Read Only
FB1.ENCLINES	30080	7580	Unsigned32	Read/Write
FB2.ENCLINES	30081	7581	Unsigned32	Read/Write
FB3.ENCLINES	30082	7582	Unsigned32	Read/Write
FB4.ENCLINES	30083	7583	Unsigned32	Read/Write
FB5.ENCLINES	30084	7584	Unsigned32	Read/Write
FB1.POLES	30100	7594	Unsigned16	Read/Write
FB2.POLES	30101	7595	Unsigned16	Read/Write
FB3.POLES	30102	7596	Unsigned16	Read/Write
FB4.POLES	30103	7597	Unsigned16	Read/Write
FB5.POLES	30104	7598	Unsigned16	Read/Write
FB1.RESKTR	30140	75BC	Unsigned16	Read/Write
FB2.RESKTR	30141	75BD	Unsigned16	Read/Write
FB3.RESKTR	30142	75BE	Unsigned16	Read/Write
FB4.RESKTR	30143	75BF	Unsigned16	Read/Write
FB5.RESKTR	30144	75C0	Unsigned16	Read/Write
FB1.RESREFPHASE	30160	75D0	Signed32	Read/Write
FB2.RESREFPHASE	30161	75D1	Signed32	Read/Write
FB3.RESREFPHASE	30162	75D2	Signed32	Read/Write
FB4.RESREFPHASE	30163	75D3	Signed32	Read/Write
FB5.RESREFPHASE	30164	75D4	Signed32	Read/Write
FB1.TRACKINGCAL	30180	75E4	Unsigned8	Read/Write
FB2.TRACKINGCAL	30181	75E5	Unsigned8	Read/Write
FB3.TRACKINGCAL	30182	75E6	Unsigned8	Read/Write
FB4.TRACKINGCAL	30183	75E7	Unsigned8	Read/Write
FB5.TRACKINGCAL	30184	75E8	Unsigned8	Read/Write
FB1.CALTHRESHSINCOS	30200	75F8	Unsigned16	Read/Write
FB2.CALTHRESHSINCOS	30201	75F9	Unsigned16	Read/Write
FB3.CALTHRESHSINCOS	30202	75FA	Unsigned16	Read/Write
FB4.CALTHRESHSINCOS	30203	75FB	Unsigned16	Read/Write
FB5.CALTHRESHSINCOS	30204	75FC	Unsigned16	Read/Write
FB1.MECHPOS	30220	760C	Unsigned32	Read Only
FB2.MECHPOS	30221	760D	Unsigned32	Read Only
FB3.MECHPOS	30222	760E	Unsigned32	Read Only
FB4.MECHPOS	30223	760F	Unsigned32	Read Only
FB5.MECHPOS	30224	7610	Unsigned32	Read Only
FB1.P	30240	7620	Signed32	Read Only
FB2.P	30241	7621	Signed32	Read Only
FB3.P	30242	7622	Signed32	Read Only
FB4.P	30243	7623	Signed32	Read Only
FB5.P	30244	7624	Signed32	Read Only
FB1.SIGNALAMPLITUDE	30260	7634	Signed32	Read Only
FB2.SIGNALAMPLITUDE	30261	7635	Signed32	Read Only
FB3.SIGNALAMPLITUDE	30262	7636	Signed32	Read Only
FB4.SIGNALAMPLITUDE	30263	7637	Signed32	Read Only
FB5.SIGNALAMPLITUDE	30264	7638	Signed32	Read Only
FB1.SIGNALCOS	30280	7648	Signed32	Read Only

Name	ID	Hex	Data Type	Access
FB2.SIGNALCOS	30281	7649	Signed32	Read Only
FB3.SIGNALCOS	30282	764A	Signed32	Read Only
FB4.SIGNALCOS	30283	764B	Signed32	Read Only
FB5.SIGNALCOS	30284	764C	Signed32	Read Only
FB1.SIGNALSIN	30300	765C	Signed32	Read Only
FB2.SIGNALSIN	30301	765D	Signed32	Read Only
FB3.SIGNALSIN	30302	765E	Signed32	Read Only
FB4.SIGNALSIN	30303	765F	Signed32	Read Only
FB5.SIGNALSIN	30304	7660	Signed32	Read Only
FB1.SINGLETURNBITS	30320	7670	Unsigned8	Read/Write
FB2.SINGLETURNBITS	30321	7671	Unsigned8	Read/Write
FB3.SINGLETURNBITS	30322	7672	Unsigned8	Read/Write
FB4.SINGLETURNBITS	30323	7673	Unsigned8	Read/Write
FB5.SINGLETURNBITS	30324	7674	Unsigned8	Read/Write
FB1.MULTITURNBITS	30340	7684	Unsigned8	Read/Write
FB2.MULTITURNBITS	30341	7685	Unsigned8	Read/Write
FB3.MULTITURNBITS	30342	7686	Unsigned8	Read/Write
FB4.MULTITURNBITS	30343	7687	Unsigned8	Read/Write
FB5.MULTITURNBITS	30344	7688	Unsigned8	Read/Write
FB1.MECHTYPE	30360	7698	Unsigned8	Read/Write
FB2.MECHTYPE	30361	7699	Unsigned8	Read/Write
FB3.MECHTYPE	30362	769A	Unsigned8	Read/Write
FB4.MECHTYPE	30363	769B	Unsigned8	Read/Write
FB5.MECHTYPE	30364	769C	Unsigned8	Read/Write
FB1.LINEPITCH	30380	76AC	Unsigned32	Read/Write
FB2.LINEPITCH	30381	76AD	Unsigned32	Read/Write
FB3.LINEPITCH	30382	76AE	Unsigned32	Read/Write
FB4.LINEPITCH	30383	76AF	Unsigned32	Read/Write
FB5.LINEPITCH	30384	76B0	Unsigned32	Read/Write
FB1.BITS	30400	76C0	Unsigned16	Read/Write
FB2.BITS	30401	76C1	Unsigned16	Read/Write
FB3.BITS	30402	76C2	Unsigned16	Read/Write
FB4.BITS	30403	76C3	Unsigned16	Read/Write
FB5.BITS	30404	76C4	Unsigned16	Read/Write
FB1.FAULTS	30420	76D4	Unsigned32[5]	Read Only
FB2.FAULTS	30421	76D5	Unsigned32[5]	Read Only
FB3.FAULTS	30422	76D6	Unsigned32[5]	Read Only
FB4.FAULTS	30423	76D7	Unsigned32[5]	Read Only
FB5.FAULTS	30424	76D8	Unsigned32[5]	Read Only
FB1.CALTHRESHRES	30440	76E8	Unsigned16	Read/Write
FB2.CALTHRESHRES	30441	76E9	Unsigned16	Read/Write
FB3.CALTHRESHRES	30442	76EA	Unsigned16	Read/Write
FB4.CALTHRESHRES	30443	76EB	Unsigned16	Read/Write
FB5.CALTHRESHRES	30444	76EC	Unsigned16	Read/Write
FB1.STOREMULTITURN.ENABLE	30460	76FC	Unsigned8	Read/Write
FB2.STOREMULTITURN.ENABLE	30461	76FD	Unsigned8	Read/Write
FB3.STOREMULTITURN.ENABLE	30462	76FE	Unsigned8	Read/Write
FB4.STOREMULTITURN.ENABLE	30463	76FF	Unsigned8	Read/Write
FB5.STOREMULTITURN.ENABLE	30464	7700	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
FB1.STOREMULTITURN.BITS	30480	7710	Unsigned8	Read/Write
FB2.STOREMULTITURN.BITS	30481	7711	Unsigned8	Read/Write
FB3.STOREMULTITURN.BITS	30482	7712	Unsigned8	Read/Write
FB4.STOREMULTITURN.BITS	30483	7713	Unsigned8	Read/Write
FB5.STOREMULTITURN.BITS	30484	7714	Unsigned8	Read/Write
FB1.EIP.POSUNIT	30520	7738	Unsigned32	Read/Write
FB2.EIP.POSUNIT	30521	7739	Unsigned32	Read/Write
FB3.EIP.POSUNIT	30522	773A	Unsigned32	Read/Write
FB4.EIP.POSUNIT	30523	773B	Unsigned32	Read/Write
FB5.EIP.POSUNIT	30524	773C	Unsigned32	Read/Write
FB1.INITSIGNED	30540	774C	Unsigned8	Read/Write
FB2.INITSIGNED	30541	774D	Unsigned8	Read/Write
FB3.INITSIGNED	30542	774E	Unsigned8	Read/Write
FB4.INITSIGNED	30543	774F	Unsigned8	Read/Write
FB5.INITSIGNED	30544	7750	Unsigned8	Read/Write
FB1.SSITYPE	30560	7760	Unsigned8	Read/Write
FB2.SSITYPE	30561	7761	Unsigned8	Read/Write
FB3.SSITYPE	30562	7762	Unsigned8	Read/Write
FB4.SSITYPE	30563	7763	Unsigned8	Read/Write
FB5.SSITYPE	30564	7764	Unsigned8	Read/Write
CAP1.ARM	31000	7918	Unsigned8	Read/Write
CAP2.ARM	31001	7919	Unsigned8	Read/Write
CAP1.COUNT	31010	7922	Unsigned16	Read Only
CAP2.COUNT	31011	7923	Unsigned16	Read Only
CAP1.EDGE	31020	792C	Unsigned8	Read/Write
CAP2.EDGE	31021	792D	Unsigned8	Read/Write
CAP1.P	31030	7936	Signed32	Read Only
CAP2.P	31031	7937	Signed32	Read Only
CAP1.PREEDGE	31040	7940	Unsigned8	Read/Write
CAP2.PREEDGE	31041	7941	Unsigned8	Read/Write
CAP1.PREMODE	31050	794A	Unsigned8	Read/Write
CAP2.PREMODE	31051	794B	Unsigned8	Read/Write
CAP1.PRESELECT	31060	7954	Unsigned8	Read/Write
CAP2.PRESELECT	31061	7955	Unsigned8	Read/Write
CAP1.REARM	31070	795E	Unsigned8	Read/Write
CAP2.REARM	31071	795F	Unsigned8	Read/Write
CAP1.SOURCE	31080	7968	Unsigned8	Read/Write
CAP2.SOURCE	31081	7969	Unsigned8	Read/Write
CAP1.STATE	31090	7972	Unsigned8	Read Only
CAP2.STATE	31091	7973	Unsigned8	Read Only
CAP1.TRIGGER	31110	7986	Unsigned8	Read/Write
CAP2.TRIGGER	31111	7987	Unsigned8	Read/Write
CMP1.ARM	31300	7A44	Unsigned8[8]	Read/Write
CMP2.ARM	31301	7A45	Unsigned8[8]	Read/Write
CMP1.DIR	31320	7A58	Unsigned8[8]	Read/Write
CMP2.DIR	31321	7A59	Unsigned8[8]	Read/Write
CMP1.REARM	31340	7A6C	Unsigned8[8]	Read/Write
CMP2.REARM	31341	7A6D	Unsigned8[8]	Read/Write
CMP1.SOURCE	31360	7A80	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
CMP2.SOURCE	31361	7A81	Unsigned8	Read/Write
CMP1.STARTVAL	31380	7A94	Signed32[8]	Read/Write
CMP2.STARTVAL	31381	7A95	Signed32[8]	Read/Write
CMP1.STATE	31400	7AA8	Unsigned8[8]	Read Only
CMP2.STATE	31401	7AA9	Unsigned8[8]	Read Only
CMP1.STATES	31420	7ABC	Unsigned16	Read Only
CMP2.STATES	31421	7ABD	Unsigned16	Read Only
CMP1.VAL	31440	7AD0	Signed32	Read Only
CMP2.VAL	31441	7AD1	Signed32	Read Only
CMP1.WIDTHT	31460	7AE4	Unsigned32[8]	Read/Write
CMP2.WIDTHT	31461	7AE5	Unsigned32[8]	Read/Write
CMP1.WIDTHTYPE	31480	7AF8	Unsigned8[8]	Read/Write
CMP2.WIDTHTYPE	31481	7AF9	Unsigned8[8]	Read/Write
CMP1.WIDTHVAL	31500	7B0C	Signed32[8]	Read/Write
CMP2.WIDTHVAL	31501	7B0D	Signed32[8]	Read/Write
CMP1.MODEN	31520	7B20	Unsigned8	Read/Write
CMP2.MODEN	31521	7B21	Unsigned8	Read/Write
CMP1.MODVAL1	31540	7B34	Signed32	Read/Write
CMP2.MODVAL1	31541	7B35	Signed32	Read/Write
CMP1.MODVAL2	31560	7B48	Signed32	Read/Write
CMP2.MODVAL2	31561	7B49	Signed32	Read/Write
CMP1.ADVANCET	31580	7B5C	Unsigned32	Read/Write
CMP2.ADVANCET	31581	7B5D	Unsigned32	Read/Write
CMP1.SOURCEVAL	31600	7B70	Signed32	Read Only
CMP2.SOURCEVAL	31601	7B71	Signed32	Read Only
EEO1.DIR	32000	7D00	Unsigned8	Read/Write
EEO2.DIR	32001	7D01	Unsigned8	Read/Write
EEO1.LINES	32005	7D05	Unsigned32	Read/Write
EEO2.LINES	32006	7D06	Unsigned32	Read/Write
EEO1.MODE	32010	7D0A	Unsigned8	Read/Write
EEO2.MODE	32011	7D0B	Unsigned8	Read/Write
EEO1.PULSEWIDTH	32015	7D0F	Signed32	Read/Write
EEO2.PULSEWIDTH	32016	7D10	Signed32	Read/Write
EEO1.SOURCE	32020	7D14	Unsigned8	Read/Write
EEO2.SOURCE	32021	7D15	Unsigned8	Read/Write
EEO1.ZMODE	32025	7D19	Unsigned8	Read/Write
EEO1.ZOFFSET	32030	7D1E	Unsigned32	Read/Write
ACTION.RUNNING	40000	9C40	Unsigned32	Read Only
ACTION1.ACTIVE	40100	9CA4	Unsigned8	Read/Write
ACTION2.ACTIVE	40101	9CA5	Unsigned8	Read/Write
ACTION3.ACTIVE	40102	9CA6	Unsigned8	Read/Write
ACTION4.ACTIVE	40103	9CA7	Unsigned8	Read/Write
ACTION5.ACTIVE	40104	9CA8	Unsigned8	Read/Write
ACTION6.ACTIVE	40105	9CA9	Unsigned8	Read/Write
ACTION7.ACTIVE	40106	9CAA	Unsigned8	Read/Write
ACTION8.ACTIVE	40107	9CAB	Unsigned8	Read/Write
ACTION9.ACTIVE	40108	9CAC	Unsigned8	Read/Write
ACTION10.ACTIVE	40109	9CAD	Unsigned8	Read/Write
ACTION11.ACTIVE	40110	9CAE	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
ACTION12.ACTIVE	40111	9CAF	Unsigned8	Read/Write
ACTION13.ACTIVE	40112	9CB0	Unsigned8	Read/Write
ACTION14.ACTIVE	40113	9CB1	Unsigned8	Read/Write
ACTION15.ACTIVE	40114	9CB2	Unsigned8	Read/Write
ACTION16.ACTIVE	40115	9CB3	Unsigned8	Read/Write
ACTION17.ACTIVE	40116	9CB4	Unsigned8	Read/Write
ACTION18.ACTIVE	40117	9CB5	Unsigned8	Read/Write
ACTION19.ACTIVE	40118	9CB6	Unsigned8	Read/Write
ACTION20.ACTIVE	40119	9CB7	Unsigned8	Read/Write
ACTION21.ACTIVE	40120	9CB8	Unsigned8	Read/Write
ACTION22.ACTIVE	40121	9CB9	Unsigned8	Read/Write
ACTION23.ACTIVE	40122	9CBA	Unsigned8	Read/Write
ACTION24.ACTIVE	40123	9CBB	Unsigned8	Read/Write
ACTION25.ACTIVE	40124	9CBC	Unsigned8	Read/Write
ACTION26.ACTIVE	40125	9CBD	Unsigned8	Read/Write
ACTION27.ACTIVE	40126	9CBE	Unsigned8	Read/Write
ACTION28.ACTIVE	40127	9CBF	Unsigned8	Read/Write
ACTION29.ACTIVE	40128	9CC0	Unsigned8	Read/Write
ACTION30.ACTIVE	40129	9CC1	Unsigned8	Read/Write
ACTION31.ACTIVE	40130	9CC2	Unsigned8	Read/Write
ACTION32.ACTIVE	40131	9CC3	Unsigned8	Read/Write
ACTION1.CONDITION	40200	9D08	Unsigned8	Read/Write
ACTION2.CONDITION	40201	9D09	Unsigned8	Read/Write
ACTION3.CONDITION	40202	9D0A	Unsigned8	Read/Write
ACTION4.CONDITION	40203	9D0B	Unsigned8	Read/Write
ACTION5.CONDITION	40204	9D0C	Unsigned8	Read/Write
ACTION6.CONDITION	40205	9D0D	Unsigned8	Read/Write
ACTION7.CONDITION	40206	9D0E	Unsigned8	Read/Write
ACTION8.CONDITION	40207	9D0F	Unsigned8	Read/Write
ACTION9.CONDITION	40208	9D10	Unsigned8	Read/Write
ACTION10.CONDITION	40209	9D11	Unsigned8	Read/Write
ACTION11.CONDITION	40210	9D12	Unsigned8	Read/Write
ACTION12.CONDITION	40211	9D13	Unsigned8	Read/Write
ACTION13.CONDITION	40212	9D14	Unsigned8	Read/Write
ACTION14.CONDITION	40213	9D15	Unsigned8	Read/Write
ACTION15.CONDITION	40214	9D16	Unsigned8	Read/Write
ACTION16.CONDITION	40215	9D17	Unsigned8	Read/Write
ACTION17.CONDITION	40216	9D18	Unsigned8	Read/Write
ACTION18.CONDITION	40217	9D19	Unsigned8	Read/Write
ACTION19.CONDITION	40218	9D1A	Unsigned8	Read/Write
ACTION20.CONDITION	40219	9D1B	Unsigned8	Read/Write
ACTION21.CONDITION	40220	9D1C	Unsigned8	Read/Write
ACTION22.CONDITION	40221	9D1D	Unsigned8	Read/Write
ACTION23.CONDITION	40222	9D1E	Unsigned8	Read/Write
ACTION24.CONDITION	40223	9D1F	Unsigned8	Read/Write
ACTION25.CONDITION	40224	9D20	Unsigned8	Read/Write
ACTION26.CONDITION	40225	9D21	Unsigned8	Read/Write
ACTION27.CONDITION	40226	9D22	Unsigned8	Read/Write
ACTION28.CONDITION	40227	9D23	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
ACTION29.CONDITION	40228	9D24	Unsigned8	Read/Write
ACTION30.CONDITION	40229	9D25	Unsigned8	Read/Write
ACTION31.CONDITION	40230	9D26	Unsigned8	Read/Write
ACTION32.CONDITION	40231	9D27	Unsigned8	Read/Write
ACTION1.CONDITIONVALUE	40300	9D6C	Signed32	Read/Write
ACTION2.CONDITIONVALUE	40301	9D6D	Signed32	Read/Write
ACTION3.CONDITIONVALUE	40302	9D6E	Signed32	Read/Write
ACTION4.CONDITIONVALUE	40303	9D6F	Signed32	Read/Write
ACTION5.CONDITIONVALUE	40304	9D70	Signed32	Read/Write
ACTION6.CONDITIONVALUE	40305	9D71	Signed32	Read/Write
ACTION7.CONDITIONVALUE	40306	9D72	Signed32	Read/Write
ACTION8.CONDITIONVALUE	40307	9D73	Signed32	Read/Write
ACTION9.CONDITIONVALUE	40308	9D74	Signed32	Read/Write
ACTION10.CONDITIONVALUE	40309	9D75	Signed32	Read/Write
ACTION11.CONDITIONVALUE	40310	9D76	Signed32	Read/Write
ACTION12.CONDITIONVALUE	40311	9D77	Signed32	Read/Write
ACTION13.CONDITIONVALUE	40312	9D78	Signed32	Read/Write
ACTION14.CONDITIONVALUE	40313	9D79	Signed32	Read/Write
ACTION15.CONDITIONVALUE	40314	9D7A	Signed32	Read/Write
ACTION16.CONDITIONVALUE	40315	9D7B	Signed32	Read/Write
ACTION17.CONDITIONVALUE	40316	9D7C	Signed32	Read/Write
ACTION18.CONDITIONVALUE	40317	9D7D	Signed32	Read/Write
ACTION19.CONDITIONVALUE	40318	9D7E	Signed32	Read/Write
ACTION20.CONDITIONVALUE	40319	9D7F	Signed32	Read/Write
ACTION21.CONDITIONVALUE	40320	9D80	Signed32	Read/Write
ACTION22.CONDITIONVALUE	40321	9D81	Signed32	Read/Write
ACTION23.CONDITIONVALUE	40322	9D82	Signed32	Read/Write
ACTION24.CONDITIONVALUE	40323	9D83	Signed32	Read/Write
ACTION25.CONDITIONVALUE	40324	9D84	Signed32	Read/Write
ACTION26.CONDITIONVALUE	40325	9D85	Signed32	Read/Write
ACTION27.CONDITIONVALUE	40326	9D86	Signed32	Read/Write
ACTION28.CONDITIONVALUE	40327	9D87	Signed32	Read/Write
ACTION29.CONDITIONVALUE	40328	9D88	Signed32	Read/Write
ACTION30.CONDITIONVALUE	40329	9D89	Signed32	Read/Write
ACTION31.CONDITIONVALUE	40330	9D8A	Signed32	Read/Write
ACTION32.CONDITIONVALUE	40331	9D8B	Signed32	Read/Write
ACTION1.RUNCOUNT	40400	9DD0	Unsigned32	Read Only
ACTION2.RUNCOUNT	40401	9DD1	Unsigned32	Read Only
ACTION3.RUNCOUNT	40402	9DD2	Unsigned32	Read Only
ACTION4.RUNCOUNT	40403	9DD3	Unsigned32	Read Only
ACTION5.RUNCOUNT	40404	9DD4	Unsigned32	Read Only
ACTION6.RUNCOUNT	40405	9DD5	Unsigned32	Read Only
ACTION7.RUNCOUNT	40406	9DD6	Unsigned32	Read Only
ACTION8.RUNCOUNT	40407	9DD7	Unsigned32	Read Only
ACTION9.RUNCOUNT	40408	9DD8	Unsigned32	Read Only
ACTION10.RUNCOUNT	40409	9DD9	Unsigned32	Read Only
ACTION11.RUNCOUNT	40410	9DDA	Unsigned32	Read Only
ACTION12.RUNCOUNT	40411	9DDB	Unsigned32	Read Only
ACTION13.RUNCOUNT	40412	9DDC	Unsigned32	Read Only

Name	ID	Hex	Data Type	Access
ACTION14.RUNCOUNT	40413	9DDD	Unsigned32	Read Only
ACTION15.RUNCOUNT	40414	9DDE	Unsigned32	Read Only
ACTION16.RUNCOUNT	40415	9DDF	Unsigned32	Read Only
ACTION17.RUNCOUNT	40416	9DE0	Unsigned32	Read Only
ACTION18.RUNCOUNT	40417	9DE1	Unsigned32	Read Only
ACTION19.RUNCOUNT	40418	9DE2	Unsigned32	Read Only
ACTION20.RUNCOUNT	40419	9DE3	Unsigned32	Read Only
ACTION21.RUNCOUNT	40420	9DE4	Unsigned32	Read Only
ACTION22.RUNCOUNT	40421	9DE5	Unsigned32	Read Only
ACTION23.RUNCOUNT	40422	9DE6	Unsigned32	Read Only
ACTION24.RUNCOUNT	40423	9DE7	Unsigned32	Read Only
ACTION25.RUNCOUNT	40424	9DE8	Unsigned32	Read Only
ACTION26.RUNCOUNT	40425	9DE9	Unsigned32	Read Only
ACTION27.RUNCOUNT	40426	9DEA	Unsigned32	Read Only
ACTION28.RUNCOUNT	40427	9DEB	Unsigned32	Read Only
ACTION29.RUNCOUNT	40428	9DEC	Unsigned32	Read Only
ACTION30.RUNCOUNT	40429	9DED	Unsigned32	Read Only
ACTION31.RUNCOUNT	40430	9DEE	Unsigned32	Read Only
ACTION32.RUNCOUNT	40431	9DEF	Unsigned32	Read Only
ACTION1.SOURCE	40500	9E34	Unsigned8	Read/Write
ACTION2.SOURCE	40501	9E35	Unsigned8	Read/Write
ACTION3.SOURCE	40502	9E36	Unsigned8	Read/Write
ACTION4.SOURCE	40503	9E37	Unsigned8	Read/Write
ACTION5.SOURCE	40504	9E38	Unsigned8	Read/Write
ACTION6.SOURCE	40505	9E39	Unsigned8	Read/Write
ACTION7.SOURCE	40506	9E3A	Unsigned8	Read/Write
ACTION8.SOURCE	40507	9E3B	Unsigned8	Read/Write
ACTION9.SOURCE	40508	9E3C	Unsigned8	Read/Write
ACTION10.SOURCE	40509	9E3D	Unsigned8	Read/Write
ACTION11.SOURCE	40510	9E3E	Unsigned8	Read/Write
ACTION12.SOURCE	40511	9E3F	Unsigned8	Read/Write
ACTION13.SOURCE	40512	9E40	Unsigned8	Read/Write
ACTION14.SOURCE	40513	9E41	Unsigned8	Read/Write
ACTION15.SOURCE	40514	9E42	Unsigned8	Read/Write
ACTION16.SOURCE	40515	9E43	Unsigned8	Read/Write
ACTION17.SOURCE	40516	9E44	Unsigned8	Read/Write
ACTION18.SOURCE	40517	9E45	Unsigned8	Read/Write
ACTION19.SOURCE	40518	9E46	Unsigned8	Read/Write
ACTION20.SOURCE	40519	9E47	Unsigned8	Read/Write
ACTION21.SOURCE	40520	9E48	Unsigned8	Read/Write
ACTION22.SOURCE	40521	9E49	Unsigned8	Read/Write
ACTION23.SOURCE	40522	9E4A	Unsigned8	Read/Write
ACTION24.SOURCE	40523	9E4B	Unsigned8	Read/Write
ACTION25.SOURCE	40524	9E4C	Unsigned8	Read/Write
ACTION26.SOURCE	40525	9E4D	Unsigned8	Read/Write
ACTION27.SOURCE	40526	9E4E	Unsigned8	Read/Write
ACTION28.SOURCE	40527	9E4F	Unsigned8	Read/Write
ACTION29.SOURCE	40528	9E50	Unsigned8	Read/Write
ACTION30.SOURCE	40529	9E51	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
ACTION31.SOURCE	40530	9E52	Unsigned8	Read/Write
ACTION32.SOURCE	40531	9E53	Unsigned8	Read/Write
ACTION1.SOURCEID	40600	9E98	Unsigned8	Read/Write
ACTION2.SOURCEID	40601	9E99	Unsigned8	Read/Write
ACTION3.SOURCEID	40602	9E9A	Unsigned8	Read/Write
ACTION4.SOURCEID	40603	9E9B	Unsigned8	Read/Write
ACTION5.SOURCEID	40604	9E9C	Unsigned8	Read/Write
ACTION6.SOURCEID	40605	9E9D	Unsigned8	Read/Write
ACTION7.SOURCEID	40606	9E9E	Unsigned8	Read/Write
ACTION8.SOURCEID	40607	9E9F	Unsigned8	Read/Write
ACTION9.SOURCEID	40608	9EA0	Unsigned8	Read/Write
ACTION10.SOURCEID	40609	9EA1	Unsigned8	Read/Write
ACTION11.SOURCEID	40610	9EA2	Unsigned8	Read/Write
ACTION12.SOURCEID	40611	9EA3	Unsigned8	Read/Write
ACTION13.SOURCEID	40612	9EA4	Unsigned8	Read/Write
ACTION14.SOURCEID	40613	9EA5	Unsigned8	Read/Write
ACTION15.SOURCEID	40614	9EA6	Unsigned8	Read/Write
ACTION16.SOURCEID	40615	9EA7	Unsigned8	Read/Write
ACTION17.SOURCEID	40616	9EA8	Unsigned8	Read/Write
ACTION18.SOURCEID	40617	9EA9	Unsigned8	Read/Write
ACTION19.SOURCEID	40618	9EAA	Unsigned8	Read/Write
ACTION20.SOURCEID	40619	9EAB	Unsigned8	Read/Write
ACTION21.SOURCEID	40620	9EAC	Unsigned8	Read/Write
ACTION22.SOURCEID	40621	9EAD	Unsigned8	Read/Write
ACTION23.SOURCEID	40622	9EAE	Unsigned8	Read/Write
ACTION24.SOURCEID	40623	9EAF	Unsigned8	Read/Write
ACTION25.SOURCEID	40624	9EB0	Unsigned8	Read/Write
ACTION26.SOURCEID	40625	9EB1	Unsigned8	Read/Write
ACTION27.SOURCEID	40626	9EB2	Unsigned8	Read/Write
ACTION28.SOURCEID	40627	9EB3	Unsigned8	Read/Write
ACTION29.SOURCEID	40628	9EB4	Unsigned8	Read/Write
ACTION30.SOURCEID	40629	9EB5	Unsigned8	Read/Write
ACTION31.SOURCEID	40630	9EB6	Unsigned8	Read/Write
ACTION32.SOURCEID	40631	9EB7	Unsigned8	Read/Write
ACTION1.SOURCEPARAM	40700	9EFC	Signed32	Read/Write
ACTION2.SOURCEPARAM	40701	9EFD	Signed32	Read/Write
ACTION3.SOURCEPARAM	40702	9EFE	Signed32	Read/Write
ACTION4.SOURCEPARAM	40703	9EFF	Signed32	Read/Write
ACTION5.SOURCEPARAM	40704	9F00	Signed32	Read/Write
ACTION6.SOURCEPARAM	40705	9F01	Signed32	Read/Write
ACTION7.SOURCEPARAM	40706	9F02	Signed32	Read/Write
ACTION8.SOURCEPARAM	40707	9F03	Signed32	Read/Write
ACTION9.SOURCEPARAM	40708	9F04	Signed32	Read/Write
ACTION10.SOURCEPARAM	40709	9F05	Signed32	Read/Write
ACTION11.SOURCEPARAM	40710	9F06	Signed32	Read/Write
ACTION12.SOURCEPARAM	40711	9F07	Signed32	Read/Write
ACTION13.SOURCEPARAM	40712	9F08	Signed32	Read/Write
ACTION14.SOURCEPARAM	40713	9F09	Signed32	Read/Write
ACTION15.SOURCEPARAM	40714	9F0A	Signed32	Read/Write

Name	ID	Hex	Data Type	Access
ACTION16.SOURCEPARAM	40715	9F0B	Signed32	Read/Write
ACTION17.SOURCEPARAM	40716	9F0C	Signed32	Read/Write
ACTION18.SOURCEPARAM	40717	9F0D	Signed32	Read/Write
ACTION19.SOURCEPARAM	40718	9F0E	Signed32	Read/Write
ACTION20.SOURCEPARAM	40719	9F0F	Signed32	Read/Write
ACTION21.SOURCEPARAM	40720	9F10	Signed32	Read/Write
ACTION22.SOURCEPARAM	40721	9F11	Signed32	Read/Write
ACTION23.SOURCEPARAM	40722	9F12	Signed32	Read/Write
ACTION24.SOURCEPARAM	40723	9F13	Signed32	Read/Write
ACTION25.SOURCEPARAM	40724	9F14	Signed32	Read/Write
ACTION26.SOURCEPARAM	40725	9F15	Signed32	Read/Write
ACTION27.SOURCEPARAM	40726	9F16	Signed32	Read/Write
ACTION28.SOURCEPARAM	40727	9F17	Signed32	Read/Write
ACTION29.SOURCEPARAM	40728	9F18	Signed32	Read/Write
ACTION30.SOURCEPARAM	40729	9F19	Signed32	Read/Write
ACTION31.SOURCEPARAM	40730	9F1A	Signed32	Read/Write
ACTION32.SOURCEPARAM	40731	9F1B	Signed32	Read/Write
ACTION1.TASK	40800	9F60	Unsigned8	Read/Write
ACTION2.TASK	40801	9F61	Unsigned8	Read/Write
ACTION3.TASK	40802	9F62	Unsigned8	Read/Write
ACTION4.TASK	40803	9F63	Unsigned8	Read/Write
ACTION5.TASK	40804	9F64	Unsigned8	Read/Write
ACTION6.TASK	40805	9F65	Unsigned8	Read/Write
ACTION7.TASK	40806	9F66	Unsigned8	Read/Write
ACTION8.TASK	40807	9F67	Unsigned8	Read/Write
ACTION9.TASK	40808	9F68	Unsigned8	Read/Write
ACTION10.TASK	40809	9F69	Unsigned8	Read/Write
ACTION11.TASK	40810	9F6A	Unsigned8	Read/Write
ACTION12.TASK	40811	9F6B	Unsigned8	Read/Write
ACTION13.TASK	40812	9F6C	Unsigned8	Read/Write
ACTION14.TASK	40813	9F6D	Unsigned8	Read/Write
ACTION15.TASK	40814	9F6E	Unsigned8	Read/Write
ACTION16.TASK	40815	9F6F	Unsigned8	Read/Write
ACTION17.TASK	40816	9F70	Unsigned8	Read/Write
ACTION18.TASK	40817	9F71	Unsigned8	Read/Write
ACTION19.TASK	40818	9F72	Unsigned8	Read/Write
ACTION20.TASK	40819	9F73	Unsigned8	Read/Write
ACTION21.TASK	40820	9F74	Unsigned8	Read/Write
ACTION22.TASK	40821	9F75	Unsigned8	Read/Write
ACTION23.TASK	40822	9F76	Unsigned8	Read/Write
ACTION24.TASK	40823	9F77	Unsigned8	Read/Write
ACTION25.TASK	40824	9F78	Unsigned8	Read/Write
ACTION26.TASK	40825	9F79	Unsigned8	Read/Write
ACTION27.TASK	40826	9F7A	Unsigned8	Read/Write
ACTION28.TASK	40827	9F7B	Unsigned8	Read/Write
ACTION29.TASK	40828	9F7C	Unsigned8	Read/Write
ACTION30.TASK	40829	9F7D	Unsigned8	Read/Write
ACTION31.TASK	40830	9F7E	Unsigned8	Read/Write
ACTION32.TASK	40831	9F7F	Unsigned8	Read/Write

Name	ID	Hex	Data Type	Access
ACTION1.TASKID	40900	9FC4	Unsigned8	Read/Write
ACTION2.TASKID	40901	9FC5	Unsigned8	Read/Write
ACTION3.TASKID	40902	9FC6	Unsigned8	Read/Write
ACTION4.TASKID	40903	9FC7	Unsigned8	Read/Write
ACTION5.TASKID	40904	9FC8	Unsigned8	Read/Write
ACTION6.TASKID	40905	9FC9	Unsigned8	Read/Write
ACTION7.TASKID	40906	9FCA	Unsigned8	Read/Write
ACTION8.TASKID	40907	9FCB	Unsigned8	Read/Write
ACTION9.TASKID	40908	9FCC	Unsigned8	Read/Write
ACTION10.TASKID	40909	9FCD	Unsigned8	Read/Write
ACTION11.TASKID	40910	9FCE	Unsigned8	Read/Write
ACTION12.TASKID	40911	9FCF	Unsigned8	Read/Write
ACTION13.TASKID	40912	9FD0	Unsigned8	Read/Write
ACTION14.TASKID	40913	9FD1	Unsigned8	Read/Write
ACTION15.TASKID	40914	9FD2	Unsigned8	Read/Write
ACTION16.TASKID	40915	9FD3	Unsigned8	Read/Write
ACTION17.TASKID	40916	9FD4	Unsigned8	Read/Write
ACTION18.TASKID	40917	9FD5	Unsigned8	Read/Write
ACTION19.TASKID	40918	9FD6	Unsigned8	Read/Write
ACTION20.TASKID	40919	9FD7	Unsigned8	Read/Write
ACTION21.TASKID	40920	9FD8	Unsigned8	Read/Write
ACTION22.TASKID	40921	9FD9	Unsigned8	Read/Write
ACTION23.TASKID	40922	9FDA	Unsigned8	Read/Write
ACTION24.TASKID	40923	9FDB	Unsigned8	Read/Write
ACTION25.TASKID	40924	9FDC	Unsigned8	Read/Write
ACTION26.TASKID	40925	9FDD	Unsigned8	Read/Write
ACTION27.TASKID	40926	9FDE	Unsigned8	Read/Write
ACTION28.TASKID	40927	9FDF	Unsigned8	Read/Write
ACTION29.TASKID	40928	9FE0	Unsigned8	Read/Write
ACTION30.TASKID	40929	9FE1	Unsigned8	Read/Write
ACTION31.TASKID	40930	9FE2	Unsigned8	Read/Write
ACTION32.TASKID	40931	9FE3	Unsigned8	Read/Write
ACTION1.TASKPARAM	41000	A028	Signed32	Read/Write
ACTION2.TASKPARAM	41001	A029	Signed32	Read/Write
ACTION3.TASKPARAM	41002	A02A	Signed32	Read/Write
ACTION4.TASKPARAM	41003	A02B	Signed32	Read/Write
ACTION5.TASKPARAM	41004	A02C	Signed32	Read/Write
ACTION6.TASKPARAM	41005	A02D	Signed32	Read/Write
ACTION7.TASKPARAM	41006	A02E	Signed32	Read/Write
ACTION8.TASKPARAM	41007	A02F	Signed32	Read/Write
ACTION9.TASKPARAM	41008	A030	Signed32	Read/Write
ACTION10.TASKPARAM	41009	A031	Signed32	Read/Write
ACTION11.TASKPARAM	41010	A032	Signed32	Read/Write
ACTION12.TASKPARAM	41011	A033	Signed32	Read/Write
ACTION13.TASKPARAM	41012	A034	Signed32	Read/Write
ACTION14.TASKPARAM	41013	A035	Signed32	Read/Write
ACTION15.TASKPARAM	41014	A036	Signed32	Read/Write
ACTION16.TASKPARAM	41015	A037	Signed32	Read/Write
ACTION17.TASKPARAM	41016	A038	Signed32	Read/Write

Name	ID	Hex	Data Type	Access
ACTION18.TASKPARAM	41017	A039	Signed32	Read/Write
ACTION19.TASKPARAM	41018	A03A	Signed32	Read/Write
ACTION20.TASKPARAM	41019	A03B	Signed32	Read/Write
ACTION21.TASKPARAM	41020	A03C	Signed32	Read/Write
ACTION22.TASKPARAM	41021	A03D	Signed32	Read/Write
ACTION23.TASKPARAM	41022	A03E	Signed32	Read/Write
ACTION24.TASKPARAM	41023	A03F	Signed32	Read/Write
ACTION25.TASKPARAM	41024	A040	Signed32	Read/Write
ACTION26.TASKPARAM	41025	A041	Signed32	Read/Write
ACTION27.TASKPARAM	41026	A042	Signed32	Read/Write
ACTION28.TASKPARAM	41027	A043	Signed32	Read/Write
ACTION29.TASKPARAM	41028	A044	Signed32	Read/Write
ACTION30.TASKPARAM	41029	A045	Signed32	Read/Write
ACTION31.TASKPARAM	41030	A046	Signed32	Read/Write
ACTION32.TASKPARAM	41031	A047	Signed32	Read/Write
ACTION1.TASKTEXT	41100	A08C	Unknown	Read/Write
ACTION2.TASKTEXT	41101	A08D	Unknown	Read/Write
ACTION3.TASKTEXT	41102	A08E	Unknown	Read/Write
ACTION4.TASKTEXT	41103	A08F	Unknown	Read/Write
ACTION5.TASKTEXT	41104	A090	Unknown	Read/Write
ACTION6.TASKTEXT	41105	A091	Unknown	Read/Write
ACTION7.TASKTEXT	41106	A092	Unknown	Read/Write
ACTION8.TASKTEXT	41107	A093	Unknown	Read/Write
ACTION9.TASKTEXT	41108	A094	Unknown	Read/Write
ACTION10.TASKTEXT	41109	A095	Unknown	Read/Write
ACTION11.TASKTEXT	41110	A096	Unknown	Read/Write
ACTION12.TASKTEXT	41111	A097	Unknown	Read/Write
ACTION13.TASKTEXT	41112	A098	Unknown	Read/Write
ACTION14.TASKTEXT	41113	A099	Unknown	Read/Write
ACTION15.TASKTEXT	41114	A09A	Unknown	Read/Write
ACTION16.TASKTEXT	41115	A09B	Unknown	Read/Write
ACTION17.TASKTEXT	41116	A09C	Unknown	Read/Write
ACTION18.TASKTEXT	41117	A09D	Unknown	Read/Write
ACTION19.TASKTEXT	41118	A09E	Unknown	Read/Write
ACTION20.TASKTEXT	41119	A09F	Unknown	Read/Write
ACTION21.TASKTEXT	41120	A0A0	Unknown	Read/Write
ACTION22.TASKTEXT	41121	A0A1	Unknown	Read/Write
ACTION23.TASKTEXT	41122	A0A2	Unknown	Read/Write
ACTION24.TASKTEXT	41123	A0A3	Unknown	Read/Write
ACTION25.TASKTEXT	41124	A0A4	Unknown	Read/Write
ACTION26.TASKTEXT	41125	A0A5	Unknown	Read/Write
ACTION27.TASKTEXT	41126	A0A6	Unknown	Read/Write
ACTION28.TASKTEXT	41127	A0A7	Unknown	Read/Write
ACTION29.TASKTEXT	41128	A0A8	Unknown	Read/Write
ACTION30.TASKTEXT	41129	A0A9	Unknown	Read/Write
ACTION31.TASKTEXT	41130	A0AA	Unknown	Read/Write
ACTION32.TASKTEXT	41131	A0AB	Unknown	Read/Write
ACTION1.SOURCETEXT	41200	A0F0	Unknown	Read/Write
ACTION2.SOURCETEXT	41201	A0F1	Unknown	Read/Write

Name	ID	Hex	Data Type	Access
ACTION3.SOURCETEXT	41202	A0F2	Unknown	Read/Write
ACTION4.SOURCETEXT	41203	A0F3	Unknown	Read/Write
ACTION5.SOURCETEXT	41204	A0F4	Unknown	Read/Write
ACTION6.SOURCETEXT	41205	A0F5	Unknown	Read/Write
ACTION7.SOURCETEXT	41206	A0F6	Unknown	Read/Write
ACTION8.SOURCETEXT	41207	A0F7	Unknown	Read/Write
ACTION9.SOURCETEXT	41208	A0F8	Unknown	Read/Write
ACTION10.SOURCETEXT	41209	A0F9	Unknown	Read/Write
ACTION11.SOURCETEXT	41210	A0FA	Unknown	Read/Write
ACTION12.SOURCETEXT	41211	A0FB	Unknown	Read/Write
ACTION13.SOURCETEXT	41212	A0FC	Unknown	Read/Write
ACTION14.SOURCETEXT	41213	A0FD	Unknown	Read/Write
ACTION15.SOURCETEXT	41214	A0FE	Unknown	Read/Write
ACTION16.SOURCETEXT	41215	A0FF	Unknown	Read/Write
ACTION17.SOURCETEXT	41216	A100	Unknown	Read/Write
ACTION18.SOURCETEXT	41217	A101	Unknown	Read/Write
ACTION19.SOURCETEXT	41218	A102	Unknown	Read/Write
ACTION20.SOURCETEXT	41219	A103	Unknown	Read/Write
ACTION21.SOURCETEXT	41220	A104	Unknown	Read/Write
ACTION22.SOURCETEXT	41221	A105	Unknown	Read/Write
ACTION23.SOURCETEXT	41222	A106	Unknown	Read/Write
ACTION24.SOURCETEXT	41223	A107	Unknown	Read/Write
ACTION25.SOURCETEXT	41224	A108	Unknown	Read/Write
ACTION26.SOURCETEXT	41225	A109	Unknown	Read/Write
ACTION27.SOURCETEXT	41226	A10A	Unknown	Read/Write
ACTION28.SOURCETEXT	41227	A10B	Unknown	Read/Write
ACTION29.SOURCETEXT	41228	A10C	Unknown	Read/Write
ACTION30.SOURCETEXT	41229	A10D	Unknown	Read/Write
ACTION31.SOURCETEXT	41230	A10E	Unknown	Read/Write
ACTION32.SOURCETEXT	41231	A10F	Unknown	Read/Write

13 Appendix C: Examples of Explicit Messaging

- Example 1: Set an axis-specific non-array parameter
- Example 2: Get an axis-specific non-array parameter
- Example 3: Set an axis-specific array parameter
- Example 4: Get an axis-specific array parameter
- Example 5: Set a drive level (axis-independent) parameter
- Example 6: Get a drive level (axis-independent) parameter

13.1 Example 1: Set an axis-specific non-array parameter

Definition:

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10 (write value)
Class	0x64
Instance Number	Axis Number (1 or 2)
Attribute	Parameter Number (In Hex for Studio 5000 MSG instruction)

Example: AXIS2.GUI.PARAM01

AXIS#.GUI.PARAM01	5400	Signed32	Read/Write	

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10
Class	0x64
Instance Number	Axis Number = 2
Attribute	0x1518 (5400 decimal)

13.2 Example 2: Get an axis-specific non-array parameter

Definition:

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0xe (read value)
Class	0x64
Instance Number	Axis Number (1 or 2)
Attribute	Parameter Number (In Hex for Studio 5000 MSG instruction)

Example: AXIS2.GUI.PARAM01

AXIS#.GUI.PARAM01	5400	Signed32	Read/Write	

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0xE
Class	0x64
Instance Number	Axis Number = 2
Attribute	0x1518 (5400 decimal)

13.3 Example 3: Set an axis-specific array parameter

Definition:

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10 (write value)
Class	0x64
Instance Number	Array Index * 100 + Axis ID
Attribute	Parameter Number (In Hex for Studio 5000 MSG instruction)

Example: Axis 1, MT.P Task 3

AXIS#.MT.P	6307	Signed32[32]	Read/Write	Position

This is an axis-specific parameter but also an array type (of elements 0 to 31).

Axis ID	1
Array Index	Task Number = 3
Instance Number	Array Index * 100 + Axis ID
Instance Number	3 * 100 + 1 = 301
Attribute	0x18A3 (6307 decimal)

13.4 Example 4: Get an axis-specific array parameter

Definition:

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0xE (read value)
Class	0x64
Instance Number	Array Index * 100 + Axis ID
Attribute	Parameter Number (In Hex for Studio 5000 MSG instruction)

Example: Axis 1, MT.P Task 3

AXIS#.MT.P	6307	Signed32[32]	Read/Write	Position

This is an axis-specific parameter but also an array type (of elements 0 to 31).

Axis ID	1
Array Index	Task Number = 3
Instance Number	Array Index * 100 + Axis ID
Instance Number	3 * 100 + 1 = 301
Attribute	0x18A3 (6307 decimal)

13.5 Example 5: Set a drive level (axis-independent) parameter

Definition:

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10 (write value)
Class	0x64
Instance Number	Ignored (Set to 1)
Attribute	0xc80 (3200 decimal)

Example: USERINT1

USERINT1	3200	Signed16	Read/Write	

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10 (write value)
Class	0x64
Instance Number	Ignored (Set to 1)
Attribute	0xc80 (3200 decimal)

13.6 Example 6: Get a drive level (axis-independent) parameter

Definition:

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0xE (read value)
Class	0x64
Instance Number	Ignored (Set to 1)
Attribute	Parameter Number (In Hex for Studio 5000 MSG instruction)

Example: VBUS.VALUE

VBUS.VALUE	2500	Float	Read Only	

Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0xE
Class	0x64
Instance Number	Ignored (Set to 1)
Attribute	0x9c4 (2500 decimal)

14 Appendix D: Software Distribution License

SOFTWARE DISTRIBUTION LICENSE FOR THE
ETHERNET/IP(TM) COMMUNICATION STACK
(ADAPTED BSD STYLE LICENSE)

Redistribution of the Communications Stack Software for EtherNet/IP and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright and trademark notices, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Rockwell Automation, ODVA, nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission from the respective owners.

The Communications Stack Software for EtherNet/IP, or any portion thereof, with or without modifications, may be incorporated into products for sale. However, the software does not, by itself, convey any right to make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute or dispose of any products that implement this software, which products might be covered by valid patents or copyrights of ODVA, Inc., its members or other licensors nor does this software result in any license to use the EtherNet/IP mark owned by ODVA. To make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute or dispose of any products that implement this software, and to use the EtherNet/IP mark, one must obtain the necessary license from ODVA through its Terms of Usage Agreement for the EtherNet/IP technology, available through the ODVA web site at www.odva.org. This license requirement applies equally (a) to devices that completely implement ODVA's Final Specification for EtherNet/IP (“Network Devices”), (b) to components of such Network Devices to the extent they implement portions of the Final Specification for EtherNet/IP, and (c) to enabling technology products, such as any other EtherNet/IP or other network protocol stack designed for use in Network Devices to the extent they implement portions of the Final Specification for EtherNet/IP. Persons or entities who are not already licensed for the EtherNet/IP technology must contact ODVA for a Terms of Usage Agreement.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

About KOLLMORGEN

Kollmorgen is a leading provider of motion systems and components for machine builders. Through world-class knowledge in motion, industry-leading quality and deep expertise in linking and integrating standard and custom products, Kollmorgen delivers breakthrough solutions that are unmatched in performance, reliability and ease-of-use, giving machine builders an irrefutable marketplace advantage.

Join the Kollmorgen Developer Network for product support. Ask the community questions, search the knowledge base for answers, get downloads, and suggest improvements.

North America
KOLLMORGEN
201 West Rock Road
Radford, VA 24141, USA

Web:	www.kollmorgen.com
Mail:	support@kollmorgen.com
Tel.:	+1 - 540 - 633 - 3545
Fax:	+1 - 540 - 639 - 4162

Europe
KOLLMORGEN Europe GmbH
Pempelfurtstr. 1
40880 Ratingen, Germany

Web:	www.kollmorgen.com
Mail:	technik@kollmorgen.com
Tel.:	+49 - 2102 - 9394 - 0
Fax:	+49 - 2102 - 9394 - 3155

South America
KOLLMORGEN
Avenida João Paulo Ablas, 2970
Jardim da Glória, Cotia – SP
CEP 06711-250, Brazil

Web:	www.kollmorgen.com
Mail:	contato@kollmorgen.com
Tel.:	+55 11 4615-6300

China and SEA
KOLLMORGEN
Room 302, Building 5, Lihpao Plaza,
88 Shenbin Road, Minhang District,
Shanghai, China.

Web:	www.kollmorgen.cn
Mail:	sales.china@kollmorgen.com
Tel.:	+86 - 400 668 2802
Fax:	+86 - 21 6248 5367