AKD®

EtherNet/IP Communication

Edition December 2013, Revision D

Valid for firmware version 1.12

Part Number 903-200008-00

Keep all manuals as a product component during the life span of the product.
Pass all manuals to future users/owners of the product.

Record of Document Revisions:

Revision	Remarks
A, 10/2011	Launch version
C, 08/2012	Minor updates to formatting.
D, 12/2012	Add information on using Jog Move and Position Move while in position mode (Communication Profile (pg 9). Data size of IL.FB corrected in Appendix B: Parameter Listing (pg 31).

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

1 Table of Contents

2 About this Manual

This manual describes the installation, setup, range of functions, and software protocol for the AKD EtherNet/IP product series. All AKD 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 disk included with your drive. Manual updates can be downloaded from the Kollmorgen™ website.

Related documents for the AKD series include:

- Using AKD EtherNet/IP with RSLogix. This manual provides easy start guide for RSLogix programs, as well as a reference to the sample add-on instructions.
- AKD Quick Start (also provided in hard copy). This guide provides instructions for initial drive setup and connection to a network.
- AKD Installation Manual (also provided in hard copy). This manual provides instructions for installation and drive setup.
- AKD Parameter and Command Reference Guide. This guide provides documentation for the parameters and commands used to program the AKD.

Additional documentation:

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

3 Overview

EtherNet/IP is an industrial communication protocol based on TCP/IP and UDP/IP. It is used as 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 are used for reading or writing values on-demand. Typically they are used 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. The AKD can be fully configured using Explicit Messages.
- I/O Assembly Messages are data structures usually sent on a timed cyclic basis. These are normally use for drive control and status. The data structure is predetermined and only certain values can be read and written.

Typically, Explicit Messaging is used to configure the amplifier and I/O Assemblies are used to control movement. Most PLC's will support both types of messaging simultaneously.

Explicit Messages allow you to access a single parameter value at a time. The desired parameter is selected by specifying the class object number, instance number and attribute number in an explicit message.

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 configuration may be done through the Parameter Object, which exposes vendor configuration parameters such as those accessible through Workbench.

Motion sequences may be pre-programmed into the amplifier using the AKD 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 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 set points. 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.

4 AKD EtherNet/IP Features

4.1 Supported Features

AKD 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:

- Drive setup and configuration
  - A full range of drive parameters can be accessed
  - 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
- 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

4.2 Expected Packet Rate

The Expected Packet Rate (EPR) is also called the Requested Packet Interval (RPI).

The fastest supported cyclic rate for EtherNet/IP on AKD is 10 milliseconds. For simultaneous operation of Workbench and an EtherNet/IP controller communicating with an AKD, the cycle rate should be reduced to 20 milliseconds.

4.3 Connection Port

The EtherNet/IP network connection port with the AKD is the same RJ45 connector used for the Telnet. This port is numbered as X11 on the AKD side panel.

4.4 Network Topology

AKD can be connected on an EtherNet/IP network in two manners:

- As the last node in the network (since AKD has only one connector) in a line topology
- As another node on the network in star topology (using a switch)

5 Setup

5.1 Setting an IP Address in the Drive

The IP address of the AKD must be configured properly on the same subnet as the controller. The same address is used for both EtherNet/IP and Workbench connections (though they use different TCP/IP ports). See the AKD Quick Start or AKD Use Guide for instructions on setting this address.

5.2 Controller Setup

Some controllers request an EDS file (electronic data sheet) for configuring each EtherNet/IP node. The AKD EtherNet/IP EDS file can be found on the Kollmorgen web site and on the product disk.

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

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

5.3 Setting Expected Packet Rate in the Controller

The controller is responsible for setting the “Expected Packet Rate.” The AKD and controller will each send cyclic messages at this rate.

The fastest supported cyclic rate for EtherNet/IP on AKD is 10 milliseconds. For simultaneous operation of Workbench and an EtherNet/IP controller communicating with an AKD, the cycle rate should be reduced to 20 milliseconds.

If the rate is set to too short of a time, communication may timeout between the drive and controller, resulting in fault F702 Fieldbus Communication Lost. In this case, the EPR should be set to a larger value.

6 Communication Profile

6.1 Explicit Messaging (on-demand)

Typically, Explicit Messages are used to configure the amplifier and setup drive parameters. Explicit Messages allow you to access a single parameter value at a time. The desired parameter is selected by specifying the class object number, instance number and attribute number in an explicit message.

See chapter 2, “Overview” for an overview of Explicit versus IO Assembly messaging.

6.1.1 Supported Services

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

6.1.2 Supported Objects

AKD supports a number of standard and vendor-specific objects for motion control. See the Drive Objects chapter for information about these objects.

Parameter Object
Class Code: 0x0F

Instance: The instance number references the desired parameter. See Appendix B for a list of available parameters.

Description: The parameter object gives direct access to amplifier configuration parameters

Position Controller Supervisor Object
Class Code: 0x24
Instance: 1

Description: The position controller supervisor handles errors for the position controller.

Position Controller Object
Class Code: 0x25
Instance: 1

Description: The position controller object is used to set the operating mode (torque, velocity, position), configure motion profiles, and initiate movement.

AKD 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

6.1.3 Data Types

The table below describes the data type, number of bytes, minimum 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

6.1.4 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

6.2 I/O Assembly Messaging (cyclic)

The cyclic data exchange includes the transmission and reception of data values like set point values (e.g. Position set point, velocity set point or control word) and actual values (actual position value, actual velocity or status word) between the master and the drive.

The data commands and responses contain multiple values in pre-defined data structures, called assemblies.

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

Assemblies are transmitted on a timer according to the Expected Packet Rate.

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

6.2.1 Controller Configuration

A controller must be configured with the correct assembly information in order to open an IO connection to the AKD. This setup will differ depending on the controller type.

See the guide *Using AKD with EtherNet/IP and RSLogix* for information specific to Allen Bradley controllers.

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

Input Assembly (also called Response Assembly or “Target to Originator Connection”)

Instance: 102

Size: 64 bytes

Run/Idle Header: No

Output Assembly (also called Command Assembly or “Originator to Target Connection”)

Instance: 101

Size: 64 bytes

Run/Idle Header: Yes

Configuration Assembly

Instance: 100

Size: 0 bytes

Requested Packet Interval (also called Expected Packet Rate)

20ms or greater for simultaneous use with Workbench, such as during commissioning

10ms or greater if simultaneous Workbench use is not required

IO Connection Type: Multicast, Class 1 Type

6.2.2 Command Assemblies

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, requesting 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 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 fault that must be cleared before amplifier operation can continue.

6.2.2.1 Command Assembly Data Structure

Byte	Data	Comment
0	Control Word	The control word contains bits for enabling, moving, and handshaking with the drive.
1	Block #	The block number is used to start a particular Motion Task, in combination with the Start Block bit in the Control Word.
2	Command Type	Specifies the desired command to execute, such as Set Position or Set Parameter.
3	Response Type	Specifies the desired response data to return in the Response Assembly.
4-7	Data	The command data for most Command Types*
8-11	Position	Position data for Command Type 6 (Position Move)*
12-15	Velocity	Velocity data for Command Type 6 (Position Move) and 7 (Jog)*
16-19	Acceleration	Acceleration data for Command Type 6 (Position Move) and 7 (Jog)*
20-23	Deceleration	Deceleration data for Command Type 6 (Position Move) and 7 (Jog)*
24-31	Parameter/Attribute Data	Command Data for Command Type 0x1B (Set Position Controller Attribute) and 0x1F (Set Parameter)*
32	Attribute to Get	Index of desired Position Controller Attribute value to return in the Response Assembly bytes 24-31)
33-63	Reserved	

*Least significant byte first for all data fields

6.2.2.2 Control Word

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	Reserved	Hard Stop	Smooth Stop	Direction	Relative	Start Block	Load/Start

Enable: Setting this bit enables the amplifier.

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.

Smooth Stop: Setting this bit causes the amplifier to decelerate to a stop.

Direction: This bit is used only in velocity mode. Positive direction=1 and negative

direction=0.

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).

Start Block: Executes a Motion Task sequence previously generated and stored in the drive. Put the starting block number in the Block Number field (byte 1) and transition this bit high (1). The Load/Start flag must be zero (0) while transitioning Start Block.

Load/Start: This bit is used for data handshaking between the controller and amplifier.

To transmit a command to the amplifier, set the Command Type and load data into the data fields, then toggle Load/Start high. The amplifier will accept data only when Load/Start transitions from 0 to 1.

If the command type matches the operating mode (Target Position or Position Move in position mode, Target Velocity or Jog in velocity mode, Torque in torque mode), 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 high.

6.2.2.3 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.

- Put drive in torque mode by sending a message to Position Controller class 0x25, Instance 1, Attribute 3 Operation Mode.
- Load the desired torque value in bytes 4-7.
- Set the Load/Start bit to begin the move.

Torque values are in milliamps [mA].

6.2.2.4 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

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 Motion Task 0 and can be viewed within Workbench.

- Put drive in position mode by sending a message to Position Controller class 0x25, Instance 1, Attribute 3 Operation Mode.
- Load Target Position, Velocity, Acceleration and Deceleration into bytes 8-23 (see Data Structure section).
- Set the Load/Start bit to begin the move.

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

6.2.2.5 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.

- Put drive in velocity mode by sending a message to Position Controller class 0x25, Instance 1, Attribute 3 Operation Mode.
- Load Target Velocity, Acceleration and Deceleration into bytes 12-23 (see Data Structure section).
- Set the Load/Start bit to begin the move.

Velocity and acceleration values are scaled according to EIP.PROFUNIT.

In position mode, a Jog Move may be blended into a Position Move.

6.2.2.6 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 Drive Objects chapter for a listing of available attributes in this object.

- Set Command Type to 0x1B
- Load the Attribute number which you wish to set into bytes 4-5 (first half of the Data field, least significant byte first).
- Load the desired value into bytes 24-31 Parameter/Attribute Data (see Data Structure section).
- Set the Load/Start bit to set the value in the drive.

6.2.2.7 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 for a listing of parameter indexes, data types, and scaling.

Use this command to either read or write the desired parameter. Byte 6 is used to determine whether this is a read or write command.

Some parameters can take a very 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.

- Set Command Type to 0x1F
- Load the parameter Index which you wish to access into bytes 4-5 (first half of the Data field, least significant byte first).
- Set byte 6 according to whether you wish to read or write the parameter. 0=read, 1=write.
- If writing a parameter, load the desired value into bytes 24-31 Parameter/Attribute Data.
- Set the Load/Start bit to execute the command.
- If reading a parameter, the value will be returned in bytes 24-31 of the response.

6.2.2.8 Get Attribute

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, just set byte 32 Attribute to Get to the desired attribute number. The data will be returned in each response assembly in bytes 24-31 Parameter Data with the Attribute to Get mirrored in byte 32 of the response.

Note: Attribute to Get and Command Type 0x1F Read Parameter Value both use bytes 24-31 of the response assembly. When using command 0x1F to read a parameter, set the Attribute to Get field to 0.

- Load the desired attribute number of the Position Controller Object into byte 32 Attribute to Get.
- The value will be updated each communication cycle in bytes 24-31 of the response assembly.

6.2.3 Response Assemblies

In I/O Assembly Messaging, the amplifier transmits a response assembly back to the controller. The response assembly has a number of pre-defined 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.

6.2.3.1 Response Assemly Data Structure

Byte	Data	Comment
0	Status Word 1	Various status bits
1	Executing Block #	The index of the Motion Task which is currently being executed
2	Status Word 2	Various status bits
3	Response Type	Specifies the response type of this assembly, echoing the Response Type set in the command assembly.
4-7	Data	The response data for most Response Types*
8-11	Position	Actual Position*
12-15	Velocity	Actual Velocity*
16-19	Motion Status	Status bits. This provides the status word DRV.MOTIONSTAT. See the Parameter Reference Guide.
20-23	Reserved	
24-31	Parameter/Attribute Data	Response Data for Command Type 0x1F (Set Parameter) and the Attribute to Get*
32	Attribute to Get	Mirrors the Attribute to Get from the Command Assembly. If non-zero, the data will be in the Parameter Data field.
33-63	Reserved	

* 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 will be updated depending on the value of the Response Type.

Parameter/Attribute Data in bytes 24-31 will be updated when Attribute to Get is non-zero or when a Get Parameter command was completed.

6.2.3.2 Status Word 1

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable State	Reserved	Homed	Current Direction	General Fault	In Position	Block in Execution	In Motion

Enable State: This bit reflects the enable state of the amplifier.

Homed: This bit is set when the drive has been successfully homed.

Current Direction: This bit reflects the actual direction of motion.

General Fault: This bit indicates whether or not a fault has occurred.

In Position: This bit indicates whether or not the motor is on the last targeted position (1=On Target).

Block in Execution: When set, indicates the amplifier is running a motion task.

Executing Block # (Byte 1 in Response Assembly): Indicates the index of the currently executing Motion Task when the Block in Execution bit is set.

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.

6.2.3.3 Status Word 2

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2	Load Com-plete	Reserved	Reserve-d	Neg SW Limit	Pos SW Limit	Neg HW Limit	Pos HW Limit	Reserve-d

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 Handshaking.

Negative SW Limit: This bit indicates when the position is less than or equal to the Negative Software Limit Position.

Positive SW Limit: This bit indicates when the position is greater than or equal to the Positive Software Limit Position.

Negative HW Limit: This bit indicates the state of the Negative Hardware Limit Input.

Positive HW Limit: This bit indicates the state of the Positive Hardware Limit Input.

6.2.3.4 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, bits 4-7. Set Response Type = 0x05 in the command assembly to read this value.

6.2.3.5 Response Type 0x14 - Command/Response Error

This I/O response identifies an error that has occurred. This response will always be returned in response to an invalid Command Assembly. The Response Type field of the response assembly usually echoes the matching field from the previous command assembly. But 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
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

Error Code (hex)	Additional Code (hex)	EtherNet IP Error
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

6.2.4 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 high. 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 high. 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 (Response Type = 0x14, byte 4 = Error Code, byte 5 = Additional Code, bytes 6-7 echo command assembly bytes 2-3). See I/O Assembly Messaging 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 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 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 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 high 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 echo command assembly bytes 2-3.
5. Controller waits for either the Load Complete flag to transition high 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

6.3 Velocity Mode

In this mode, the drive is controlled via a speed set point 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.

6.3.1 Setup Velocity Mode

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

- Faults are cleared (query the General Fault bit in Status Word 1 and issue an explicit message to clear faults if necessary)
- Drive is enabled (set Enable bit in the Control Word)
- Drive is in velocity mode (set Attribute 3 Operational Mode of the Position Controller object)
- Smooth Stop and Hard Stop bits are cleared in Status Word 1.
- Position Limits are cleared (check bits in Status Word 2)

6.3.2 Velocity Moves

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

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.

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.

Velocity move values can be verified in Workbench. From the terminal, the affected values are VL.CMD.

6.4 Position Mode

In this mode, the drive 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.

6.4.1 Setup Position Mode

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

- Faults are cleared (query the General Fault bit in Status Word 1 and issue an explicit message to clear faults if necessary)
- Drive is enabled (set Enable bit in the Control Word)
- Drive is in position mode (set Attribute 3 Operational Mode of the Position Controller object)
- Smooth Stop and Hard Stop bits are cleared in Status Word 1.
- Position Limits are cleared (check bits in Status Word 2)
- Drive is homed (check Homed bit in Status Word 1)

6.4.2 Homing

Once all conditions listed under Setup Position Mode have been met (with the exception of homing), the drive 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. See the User Manual 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 1of the response assembly will be set.

6.4.3 Position Moves (point to point)

Once all conditions listed under Setup Position Mode have been met, and the drive 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.

Position moves are loaded into Motion Task 0, which can be viewed in Workbench for test and verification of user programs.

6.4.4 Running a Stored Motion Task Sequence

As an alternative to issuing a single point-to-point position commands, 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.

6.5 Torque

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

6.5.1 Setup Torque Mode

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

- Faults are cleared (query the General Fault bit in Status Word 1 and issue an explicit message to clear faults if necessary)
- Drive is enabled (set Enable bit in the Control Word)
- Drive is in torque mode (set Attribute 3 Operational Mode of the Position Controller object)
- Smooth Stop and Hard Stop bits are cleared in Status Word 1.
- Position Limits are cleared (check bits in Status Word 2)

6.5.2 Torque Moves

Once the drive is setup for torque mode, issue Torque commands (command type 0x05) to set a torque set point in the drive. 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 affected value is IL.CMD.

6.6 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 DRV.FAULT1..DRV.FAULT3 are at indexes 478-480. FAULT1 will always list the highest-priority fault.

Faults may be cleared by sending a message to the DRV.CLRFAULTS index 113 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: 0x0F (Parameter)
Instance: 113 (DRV.CLRFAULTS)
Attribute: 0x01 (Value)
Data Length: 4 bytes
Data Value: 1

6.7 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: 0x0F (Parameter)
Instance: 470 (DRV.NVSAVE)
Attribute: 0x01 (Value)
Data Length: 4 bytes
Data Value: 1

7 Drive Objects

7.1 Position Controller Class 0x25

The following attributes are supported in the Position Controller class. The instance number always equals 1 in the class/instance/attribute mappings for the Position Controller.

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	DRV.FAULTS

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 (DRV.OPMODE 2). 1= velocity (DRV.OPMODE 1). 2=Torque (DRV.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	N/A

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	N/A

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	Double Integer
Range	-2^{31} to 2^{31}
Default	0
Non-Volatile	No
See Also	N/A

Attribute 0x07: Target Velocity	
Description	This attribute specifies the target velocity in counts per second. Use target velocity for position opmode and jog velocity (attribute 22) for velocity opmode. Units are determined by the amplifier setup (VUNIT, Position controller attributes 40-41)
Access Rule	Get / Set
Data Type	Double Integer
Range	Set to a positive number
Default	According to setup
Non-Volatile	Yes
See Also	N/A

Attribute 0x08: Acceleration	
Description	This attribute specifies the acceleration for positioning and homing (HOME.ACC) when in position opmode and the acceleration for constant velocity (DRV.ACC) when in velocity opmode. Units are determined by the amplifier setup (ACCUNIT, Position controller attributes 40-41) 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) requires the motion block to be setup using the amplifier setup software.
Access Rule	Get / Set
Data Type	Double Integer
Range	Set to a positive number
Default	According to setup
Non-Volatile	Yes
See Also	N/A

Attribute 0x09: Deceleration	
Description	This attribute specifies the deceleration for positioning and homing (DECR) when in position opmode and the acceleration for constant velocity (DEC) when in velocity opmode. Units are determined by amplifier setup (ACCUNIT, Position controller attributes 40-41) All position moves initiated through a Command Assembly or Command Block Object use this deceleration rate. To set different deceleration rates for multiple motion blocks (tasks) requires the motion block to be setup using the amplifier setup software.
Access Rule	Get / Set
Data Type	Double Integer
Range	Set to a positive number
Default	According to setup
Non-Volatile	Yes
See Also	N/A

Attribute 0x0A: Move Type	
Description	This bit is used to define the position value as either absolute or incremental in DRV.OPMODE 2.
Access Rule	Get / Set
Data Type	Boolean
Range	0 = Absolute Postion 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 AKD 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	N/A

Attribute 0x19: Torque	
Description	Set a new torque command (IL.CMDU) in torque mode or read the current torque command. The Trajectory Start attribute is used to begin motion.
Access Rule	Get / Set
Data Type	Double Integer
Range	-3280 to 3280 (3280 = peak torque)
Default	0
Non-Volatile	No
See Also	N/A

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	N/A

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	N/A

7.2 Position Controller Supervisor Class 0x24

Attribute 0x05: General Fault	
Description	When active, this indicates that an amplifier related failure has occured (short circuit, over-voltage, ect).
Access Rule	Get
Data Type	U32
Range	1 = Fault condition exists 0 = No fault exists
See Also	DRV.FAULTS

7.3 Parameter Class 0x0F

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 instance number of the desired parameter can be found in Appendix B. See the Appendix for instance numbers, data types, and scaling. Note that Float types are scaled by 1000 to get an integer value.

Attribute 1 of each parameter instance refers to the value of that parameter.

Amplifier commands such as MOVE.HOME 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 (HOME.MOVE, instance = 205):

[class=0x0F, instance=205, attribute=0x01, data length=1, data value=0x01].

7.3.1 Supported Attributes

The following attributes are supported for each parameter index:

1 – Get/Set parameter value

5 – Get data type

6 – Get data size

Attribute 0x01: Parameter Value	
Description	Directly access the parameter value. Check the command reference for the data type and read/write access rule. Float types are multiplied by 1000 to get an integer value. Set the value to 1 to execute an amplifier process (eg Move Home).
Access Rule	Depends on the parameter and is given in ascii.chm in the Type field.
Data Type	Depends on the parameter and is given in ascii.chm in the Format field. The byte length is given by Data Length parameter.
Range	N/A
Default	N/A
Non-Volatile	N/A
See Also	N/A

Attribute 0x05: Data Type	
Description	This data type of this parameter.
Access Rule	Get
Data Type	U8
Range	N/A
Default	N/A
Non-Volatile	Yes
See Also	See table below

Data Type Code	Data Type	Abbreviation
0xC1	Boolean	Bool
0xC2	Short Integer	S8
0xC6	Unsigned Short Integer	U8
0xC3	Integer	S16
0xC7	Unsigned Integer	U16
0xC4	Double Integer	S32
0xC8	Unsigned Double Integer	U32
0xC5	Long Integer	S64
0xC9	Unsigned Long Integer	U64

Attribute 0x06: Data Length	
Description	Length of the parameter in bytes.
Access Rule	Get
Data Type	Unsigned Short Integer

Attribute 0x06: Data Length	
Range	N/A
Default	N/A
Non-Volatile	N/A
See Also	N/A

7.3.2 Read a Parameter Value

To read a parameter value through Explicit messaging, use Service 0x0E (Read Value), Class 0x0F (Parameter class), Attribute 1 (Parameter Value).

The instance number corresponds to the index of the desired parameter. This number may be found in Appendix B.

7.3.3 Write a Parameter Value

To set a parameter value through Explicit messaging, use Service 0x10 (Write Value), Class 0x0F (Parameter class), Attribute 1 (Parameter Value).

The instance number corresponds to the index of the desired parameter. This number may be found in Appendix B.

The length of the data written must match the length of the parameter. Read attribute 0x06 Data Length to determine the correct length to send. In the case of 64 bit parameters, it is also possible to write a 32-bit value.

7.3.4 Execute a Command Parameter

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

The instance number of the desired parameter can be found in Appendix B.

To execute a command parameter through Explicit messaging, use Service 0x10 (Write Value), Class 0x0F (Parameter class), Attribute 1 (Parameter Value), Data=0x01.

8 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.

8.1 Position Units

Position values are scaled according to the EtherNet/IP Position Controller Device standard. One “Position Units” scaling value is defined, 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 in the EtherNet/IP screen or as 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.

8.2 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 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.

8.3 Torque Units

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

8.4 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 VL.KP of 1.200 would be read over EtherNet/IP as 1200.

9 RS Logix Sample Projects

On www.kollmorgen.com, you can find RSLogix sample projects and add-on instructions, which demonstrate an EtherNet/IP network with a CompactLogix controller and the AKD.

The Using AKD EtherNet/IP with RSLogix manual provides easy start guide for RSLogix programs, as well as a reference to the sample add-on instructions.

The sample projects are based on an L32E CompactLogix controller, which easily can be changed to another controller which supports RSLogix.

9.1 Add On Instructions

A set of Add On instructions are provided for easy creation of AKD programs with RSLogix. The instructions are written to mirror the native instructions, leveraging existing knowledge of the software. They provide easy control of IO Assembly messages.

The native MSG instruction is used in RSLogix for sending Explicit Messages.

Add-On Instructions include:

- AKD_Enable
- AKD_Disable
- AKD_Home
- AKD_Jog
- AKD_Move
- AKD_Set_Home_Mode
- AKD_Set_Mode
- AKD_Shutdown
- AKD_Shutdown_Reset
- AKD_Stop_Smooth
- AKD_Get_Attribute
- AKD_Get_Parameter
- AKD_Set_Attribute
- AKD_Set_Parameter
- AKD_Set_Units

9.2 AKD Sample Project

The sample project can help you to learn:

- how to enable the drive
- how to write/read a parameter via the acyclic channel
- how the cyclic data exchange is done
- how to run motion in position or velocity mode
- how to clear faults
- how to load and execute motion task sequences

9.3 "Registration Example" Project

This sample project can help you learn:

- How to configure the drive for registration using only EtherNet/IP (no Workbench setup required).
- How to start a motion task sequence
- How to control digital I/O

10 Appendix A: Supported EtherNet/IP Objects and Attributes

10.1 Position Controller Object 0x25

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.
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).
5	Profile Units	Get/Set	DINT	Profile Units ratio value is the number of actual position feedback counts per second or second2 equal to one velocity, acceleration or deceleration unit (default 1).
6	Target Position	Get/Set	DINT	Specifies the target position in counts.
7	Target Velocity	Get/Set	DINT	Specifies the Target Velocity in counts per second.
8	Acceleration	Get/Set	DINT	Not used yet.
9	Deceleration	Get/Set	DINT	Not used yet.
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 DRV.EN).
25	Torque	Get/Set	DINT	Output torque.
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 home mode section of the AKD User Manual
101	Home Move	Set	BOOL	Initiate a home move.

11 Appendix B: Parameter Listing

The parameters in this list correspond to drive parameters available in Workbench and are described in the Workbench help documentation and the AKD User's Guide.

Position values are scaled according to EIP.PROSUNIT.

Velocity and Acceleration values are scaled according to 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.

Instance	Parameter	Data Size	Data Type
1	AIN.CUTOFF	4 Byte	Float
2	AIN.DEADBAND	2 Byte	Float
3	AIN.ISCALE	4 Byte	Float
4	AIN.OFFSET	2 Byte Signed	Float
5	AIN.PSCALE	8 Byte	Position
7	AIN.VALUE	2 Byte	Float
8	AIN.VSCALE	4 Byte	Velocity
9	AIN.ZERO	Command	None
10	AOUT.ISCALE	4 Byte	Float
11	AOUT.MODE	2 Byte	Integer
12	AOUT.OFFSET	2 Byte Signed	Float
13	AOUT.PSCALE	8 Byte	Position
15	AOUT.VALUE	8 Byte Signed	Float
17	AOUT.VALUEU	8 Byte Signed	Float
19	AOUT.VSCALE	4 Byte	Velocity
20	BODE.EXCITEGAP	1 Byte	Integer
21	BODE.FREQ	4 Byte	Float
22	BODE.IAMP	2 Byte Signed	Float
23	BODE.INJECTPOINT	1 Byte	Integer
24	BODE.MODE	1 Byte	Integer
25	BODE.MODETIMER	4 Byte	Integer
26	BODE.PRBDEPTH	1 Byte	Integer
27	BODE.VAMP	8 Byte Signed	Velocity
28	CAP0.EDGE	1 Byte	Integer
29	CAP0.EN	1 Byte	Integer
30	CAP0.EVENT	1 Byte	Integer
31	CAP0.FILTER	1 Byte	Integer
32	CAP0.MODE	1 Byte	Integer
33	CAP0.PLFB	8 Byte Signed	Position
35	CAP0.PREEDGE	1 Byte	Integer
36	CAP0.PREFILTER	1 Byte	Integer
37	CAP0.PRESLECT	1 Byte	Integer
38	CAP0.STATE	1 Byte	Integer
39	CAP0.T	4 Byte	Integer
40	CAP0.TRIGGER	1 Byte	Integer
41	CAP1.EDGE	1 Byte	Integer

Instance	Parameter	Data Size	Data Type
42	CAP1.EN	1 Byte	Integer
43	CAP1.EVENT	1 Byte	Integer
44	CAP1.FILTER	1 Byte	Integer
45	CAP1.MODE	1 Byte	Integer
46	CAP1.PLFB	8 Byte Signed	Position
48	CAP1.PREEDGE	1 Byte	Integer
49	CAP1.PREFILTER	1 Byte	Integer
50	CAP1.PRESELECT	1 Byte	Integer
51	CAP1.STATE	1 Byte	Integer
52	CAP1.T	4 Byte	Integer
53	CAP1.TRIGGER	1 Byte	Integer
54	CS.DEC	8 byte	Acceleration
56	CS.STATE	1 Byte	Integer
57	CS.TO	4 Byte	Integer
58	CS.VTHRESH	8 Byte	Velocity
59	DIN.ROTARY	1 Byte	Integer
60	DIN.STATES	1 Byte	Array
61	DIN1.INV	1 Byte	Integer
62	DIN1.MODE	2 Byte	Integer
63	DIN1.PARAM	8 Byte Signed	Varies
65	DIN1.STATE	1 Byte	Integer
66	DIN2.INV	1 Byte	Integer
67	DIN2.MODE	2 Byte	Integer
68	DIN2.PARAM	8 Byte Signed	Varies
70	DIN2.STATE	1 Byte	Integer
71	DIN3.INV	1 Byte	Integer
72	DIN3.MODE	2 Byte	Integer
73	DIN3.PARAM	8 Byte Signed	Varies
75	DIN3.STATE	1 Byte	Integer
76	DIN4.INV	1 Byte	Integer
77	DIN4.MODE	2 Byte	Integer
78	DIN4.PARAM	8 Byte Signed	Varies
80	DIN4.STATE	1 Byte	Integer
81	DIN5.INV	1 Byte	Integer
82	DIN5.MODE	2 Byte	Integer
83	DIN5.PARAM	8 Byte Signed	Varies
85	DIN5.STATE	1 Byte	Integer
86	DIN6.INV	1 Byte	Integer
87	DIN6.MODE	2 Byte	Integer
88	DIN6.PARAM	8 Byte Signed	Varies
90	DIN6.STATE	1 Byte	Integer
91	DIN7.INV	1 Byte	Integer
92	DIN7.MODE	2 Byte	Integer
93	DIN7.PARAM	8 Byte Signed	Varies

Instance	Parameter	Data Size	Data Type
95	DIN7.STATE	1 Byte	Integer
96	DOUT.CTRL	1 Byte	Integer
97	DOUT.RELAYMODE	1 Byte	Integer
98	DOUT.STATES	1 Byte	Array
99	DOUT1.MODE	1 Byte	Integer
100	DOUT1.PARAM	8 Byte Signed	Float
102	DOUT1.STATE	1 Byte	Integer
103	DOUT1.STATEU	1 Byte	Integer
104	DOUT2.MODE	1 Byte	Integer
105	DOUT2.PARAM	8 Byte Signed	Float
107	DOUT2.STATE	1 Byte	Integer
108	DOUT2.STATEU	1 Byte	Integer
109	DRV.ACC	8 Byte	Acceleration
111	DRV.ACTIVE	1 Byte	Integer
112	DRV.CLRFAULTHIST	Command	None
113	DRV.CLRFAULTS	Command	None
114	DRV.CMDSOURCE	1 Byte	Integer
115	DRV.DBILIMIT	2 Byte	Float
116	DRV.DEC	8 Byte	Acceleration
118	DRV.DIR	1 Byte	Integer
119	DRV.DI	Command	None
120	DRV.DISMODE	1 Byte	Integer
121	DRV.DISSOURCES	2 Byte	Integer
122	DRV.DISTO	4 Byte	Integer
123	DRV.EMUEDIR	1 Byte	Integer
124	DRV.EMUEMODE	2 Byte	Integer
125	DRV.EMUEMTURN	4 Byte	Integer
126	DRV.EMUERES	4 Byte	Integer
127	DRV.EMUEZOFFSET	2 Byte	Integer
128	DRV.EN	Command	None
129	DRV.ENDEFAULT	1 Byte	Integer
130	DRV.HANDWHEEL	4 Byte	Integer
131	DRV.HWENMODE	1 Byte	Integer
132	DRV.ICONT	2 Byte Signed	Float
133	DRV.IPEAK	2 Byte Signed	Float
134	DRV.IZERO	2 Byte	Float
135	DRV.MOTIONSTAT	4 Byte	Integer
136	DRV.OPMODE	1 Byte	Integer
137	DRV.RSTVAR	Command	None
138	DRV.STOP	Command	None
139	DRV.TYPE	1 Byte	Integer
140	DRV.ZERO	1 Byte	Integer
141	FB1.BISSBITS	1 Byte	Integer
142	FB1.ENCRES	4 Byte	Integer

Instance	Parameter	Data Size	Data Type
143	FB1.IDENTIFIED	1 Byte	Integer
144	FB1.INITSIGNED	1 Byte Signed	Integer
145	FB1.MECHPOS	4 Byte	Integer
146	FB1.OFFSET	8 Byte Signed	Position
148	FB1.ORIGIN	8 Byte	Position
150	FB1.PFIND	1 Byte	Integer
151	FB1.PFINDCMDU	2 Byte	Float
152	FB1.POLES	2 Byte	Integer
153	FB1.PSCALE	1 Byte	Integer
154	FB1.RESKTR	2 Byte	Float
155	FB1.RESREFPHASE	4 Byte Signed	Float
156	FB1.SELECT	1 Byte Signed	Integer
157	FB1.TRACKINGCAL	1 Byte	Integer
158	FBUS.PARAM01	4 Byte	Integer
159	FBUS.PARAM02	4 Byte	Integer
160	FBUS.PARAM03	4 Byte	Integer
161	FBUS.PARAM04	4 Byte	Integer
162	FBUS.PARAM05	4 Byte	Integer
163	FBUS.PARAM06	4 Byte	Integer
164	FBUS.PARAM07	4 Byte	Integer
178	FBUS.PLLTHRESH	2 Byte	Integer
179	FBUS.SAMPLEPERIOD	1 Byte	Integer
180	FBUS.SYNCACT	4 Byte	Integer
181	FBUS.SYNCDIST	4 Byte	Integer
182	FBUS.SYNCWND	4 Byte	Integer
183	FBUS.TYPE	1 Byte	Integer
184	GEAR.ACCMAX	8 Byte	Acceleration
186	GEAR.DECMAX	8 Byte	Acceleration
188	GEAR.IN	2 Byte	Integer
189	GEAR.MODE	2 Byte	Integer
190	GEAR.MOVE	Command	None
191	GEAR.OUT	2 Byte signed	Integer
192	GEAR.VMAX	8 Byte	Velocity
193	HOME.ACC	8 Byte	Acceleration
195	HOME.AUTOMOVE	1 Byte	Integer
196	HOME.DEC	8 Byte	Acceleration
198	HOME.DIR	2 Byte	Integer
199	HOME.DIST	8 Byte Signed	Position
201	HOME.FEEDRATE	2 Byte	Integer
202	HOME.IPEAK	4 Byte Signed	Float
204	HOME.MODE	2 Byte	Integer
205	HOME.MOVE	Command	None
206	HOME.P	8 Byte Signed	Position
208	HOME.PERRTHRESH	8 Byte Signed	Position

Instance	Parameter	Data Size	Data Type
210	HOME.SET	Command	None
211	HOME.V	8 Byte	Velocity
212	HWLS.NEGSTATE	1 Byte	Integer
213	HWLS.POSSTATE	1 Byte	Integer
214	IL.BUSFF	2 Byte Signed	Float
215	IL.CMD	2 Byte Signed	Float
217	IL.FB	8 Byte Signed	Float
218	IL.FF	2 Byte	Float
219	IL.FOLDFTHRESH	2 Byte	Float
220	IL.FOLDFTHRESHU	4 Byte Signed	Float
221	IL.FOLDWTHRESH	4 Byte Signed	Float
222	IL.FRICTION	4 Byte	Float
223	IL.IFOLDS	4 Byte	Float
224	IL.IUFB	2 Byte Signed	Float
225	IL.IVFB	2 Byte Signed	Float
226	IL.KACCFF	4 Byte Signed	Float
227	IL.KBUSFF	4 Byte	Float
228	IL.KP	2 Byte	Float
229	IL.KPDRATIO	4 Byte	Float
230	IL.KVFF	4 Byte Signed	Float
231	IL.LIMITN	4 Byte Signed	Float
232	IL.LIMITP	4 Byte Signed	Float
233	IL.MFOLDD	4 Byte	Float
234	IL.MFOLDR	4 Byte	Float
235	IL.MFOLDT	4 Byte	Float
236	IL.MIFOLD	4 Byte	Float
237	IL.OFFSET	4 Byte Signed	Float
238	IL.VCMD	2 Byte Signed	Integer
239	IL.VUFB	2 Byte Signed	Integer
240	IL.VVFB	2 Byte Signed	Integer
241	MOTOR.AUTOSET	1 Byte	Integer
242	MOTOR.BRAKE	1 Byte	Integer
243	MOTOR.BRAKERLS	1 Byte	Integer
244	MOTOR.CTF0	4 Byte	Float
245	MOTOR.ICONT	4 Byte	Float
246	MOTOR.IDDATAVALID	1 Byte	Integer
247	MOTOR.INTERTIA	4 Byte	Float
248	MOTOR.IPEAK	4 Byte	Float
249	MOTOR.KT	4 Byte	Float
250	MOTOR.LQLL	4 Byte	Float
251	MOTOR.PHASE	2 Byte	Integer
252	MOTOR.PITCH	4 Byte	Float
253	MOTOR.POLES	2 Byte	Integer
254	MOTOR.R	4 Byte	Float

Instance	Parameter	Data Size	Data Type
255	MOTOR.RTYPE	1 Byte	Integer
256	MOTOR.TBRAKEAPP	2 Byte	Integer
257	MOTOR.TBRAKERLS	2 Byte	Integer
258	MOTOR.TEMP	4 Byte	Integer
259	MOTOR.TEMPFAULT	4 Byte	Integer
260	MOTOR.TEMPWARN	4 Byte	Integer
261	MOTOR.TYPE	1 Byte	Integer
262	MOTOR.VMAX	2 Byte	Integer
263	MOTOR.VOLTMAX	2 Byte	Integer
264	MT,ACC	8 Byte	Acceleration
266	MT.CLEAR	2 Byte Signed	Integer
267	MT.CNTL	4 Byte	Integer
268	MT.CONTINUE	Command	None
269	MT.DEC	8 Byte	Acceleration
271	MT.EMERGMT	2 Byte Signed	Integer
272	MT.LOAD	Command	None
273	MT.MOVE	2 Byte Command	None
274	MT.MTNEXT	1 Byte	Integer
275	MT.NUM	1 Byte	Integer
276	MT.P	8 Byte Signed	Position
278	MT.SET	1 Byte Command	None
279	MT.TNEXT	2 Byte	Integer
280	MT.TNUM	1 Byte	Integer
281	MT.TPOSWND	8 Byte Signed	Position
283	MT.TVELWND	8 Byte	Velocity
284	MT.V	8 Byte	Velocity
285	MT.VCMD	8 Byte Signed	Velocity
286	PL.CMD	8 Byte	Position
288	PL.ERR	8 Byte	Position
290	PL.ERRMODE	1 Byte	Integer
291	PL.ERRFTHRESH	8 Byte	Position
293	PL.ERRWTHRESH	8 Byte	Position
295	PL.FB	8 Byte Signed	Position
297	PL.FBSOURCE	1 Byte	Integer
298	PL.INTINMAX	8 Byte	Position
300	PL.INTOUTMAX	8 Byte	Position
302	PL.KI	4 Byte	Float
303	PL.KP	4 Byte	Float
304	PL.MODP1	8 Byte Signed	Position
306	PL.MODP2	8 Byte Signed	Position
308	PL.MODPDIR	1 Byte	Integer
309	PL.MODPEN	1 Byte	Integer
310	PLS.EN	2 Byte	Integer
311	PLS.MODE	2 Byte	Integer

Instance	Parameter	Data Size	Data Type
312	PLS.P1	8 Byte Signed	Position
314	PLS.P2	8 Byte Signed	Position
316	PLS.P3	8 Byte Signed	Position
318	PLS.P4	8 Byte Signed	Position
320	PLS.P5	8 Byte Signed	Position
322	PLS.P6	8 Byte Signed	Position
324	PLS.P7	8 Byte Signed	Position
326	PLS.P8	8 Byte Signed	Position
328	PLS.RESET	2 Byte	Integer
329	PLS.STATE	2 Byte	Integer
330	PLS.T1	2 Byte	Integer
331	PLS.T2	2 Byte	Integer
332	PLS.T3	2 Byte	Integer
333	PLS.T4	2 Byte	Integer
334	PLS.T5	2 Byte	Integer
335	PLS.T6	2 Byte	Integer
336	PLS.T7	2 Byte	Integer
337	PLS.T8	2 Byte	Integer
338	PLS.UNITS	1 Byte	Integer
339	PLS.WIDTH1	8 Byte Signed	Position
341	PLS.WIDTH2	8 Byte Signed	Position
343	PLS.WIDTH3	8 Byte Signed	Position
345	PLS.WIDTH4	8 Byte Signed	Position
347	PLS.WIDTH5	8 Byte Signed	Position
349	PLS.WIDTH6	8 Byte Signed	Position
351	PLS.WIDTH7	8 Byte Signed	Position
353	PLS.WIDTH8	8 Byte Signed	Position
355	REC.ACTIVE	1 Byte	Integer
356	REC.DONE	1 Byte	Integer
357	REC.GAP	2 Byte	Integer
358	REC.NUMPOINTS	2 Byte	Integer
359	REC.OFF	Command	None
360	REC.STOPTYPE	1 Byte	Integer
361	REC.TRIG	Command	None
362	REC.TRIGPOS	1 Byte	Integer
363	REC.TRIGPRMLIST	-	String
364	REC.TRIGSLOPE	1 Byte	Integer
365	REC.TRIGTYPE	1 Byte	Integer
366	REC.TRIGVAL	8 Byte Signed	Varies
368	REGEN.POWER	8 Byte	Integer
370	REGEN.REXT	2 Byte	Integer
371	REGEN.TEXT	4 Byte	Float
372	REGEN.TYPE	1 Byte Signed	Integer
373	REGEN.WATTEXT	2 Byte	Integer

Instance	Parameter	Data Size	Data Type
374	SM.I1	2 Byte Signed	Float
375	SM.I2	2 Byte Signed	Float
376	SM.MODE	2 Byte	Integer
377	SM.MOVE	Command	None
378	SM.T1	2 Byte	Integer
379	SM.T2	2 Byte	Integer
380	SM.V1	8 Byte Signed	Velocity
381	SM.V2	8 Byte Signed	Velocity
382	STO.STATE	1 Byte	Integer
383	SWLS.EN	2 Byte	Integer
384	SWLS.LIMIT0	8 Byte Signed	Position
386	SWLS.LIMIT1	8 Byte Signed	Position
388	SWLS.STATE	2 Byte	Integer
389	UNIT.ACCLINEAR	1 Byte	Integer
390	UNIT.ACCROTARY	1 Byte	Integer
391	UNIT.PIN	4 Byte	Integer
392	UNIT.PLINEAR	1 Byte	Integer
393	UNIT.POUT	4 Byte	Integer
394	UNIT.PROTARY	1 Byte	Integer
395	UNIT.VLINEAR	1 Byte	Integer
396	UNIT.VROTARY	1 Byte	Integer
397	VBUS.CALGAIN	4 Byte	Float
398	VBUS.OVFTHRESH	2 Byte	Integer
399	VBUS.OVWTHRESH	2 Byte	Integer
400	VBUS.RMSLIMIT	1 Byte	Integer
401	VBUS.UVFTHRESH	2 Byte	Integer
402	VBUS.UVMODE	1 Byte	Integer
403	VBUS.UVWTHRESH	2 Byte	Integer
404	VBUS.VALUE	4 Byte	Float
405	VL.ARPF1	4 Byte	Float
406	VL.ARPF2	4 Byte	Float
407	VL.ARPF3	4 Byte	Float
408	VL.ARPF4	4 Byte	Float
409	VL.ARPQ1	4 Byte	Float
410	VL.ARPQ2	4 Byte	Float
411	VL.ARPQ3	4 Byte	Float
412	VL.ARPQ4	4 Byte	Float
413	VL.ARTYPE1	1 Byte	Integer
414	VL.ARTYPE2	1 Byte	Integer
415	VL.ARTYPE3	1 Byte	Integer
416	VL.ARTYPE4	1 Byte	Integer
417	VL.ARZF1	4 Byte	Float
418	VL.ARZF2	4 Byte	Float
419	VL.ARZF3	4 Byte	Float

Instance	Parameter	Data Size	Data Type
420	VL.ARZF4	4 Byte	Float
421	VL.ARZQ1	4 Byte	Float
422	VL.ARZQ2	4 Byte	Float
423	VL.ARZQ3	4 Byte	Float
424	VL.ARZQ4	4 Byte	Float
425	VL.BUSFF	8 Byte Signed	Velocity
426	VL.CMD	8 Byte Signed	Velocity
427	VL.CMDU	8 Byte Signed	Velocity
428	VL.ERR	8 Byte Signed	Velocity
429	VL.FB	8 Byte Signed	Velocity
430	VL.FBFILTER	8 Byte Signed	Velocity
431	VL.FBSOURCE	1 Byte	Integer
432	VL.FF	8 Byte Signed	Velocity
433	VL.GENMODE	2 Byte	Velocity
434	VL.KBUSFF	4 Byte	Float
435	VL.KI	4 Byte	Float
436	VL.KO	4 Byte	Float
437	VL.KP	4 Byte	Float
438	VL.KVFF	4 Byte	Float
439	VL.LIMITN	8 Byte Signed	Velocity
440	VL.LIMITP	8 Byte	Velocity
441	VL.LMJR	4 Byte	Float
442	VL.MODEL	8 Byte Signed	Velocity
443	VL.OBSBW	4 Byte	Float
444	VL.OBSMODE	4 Byte	Integer
445	VL.THRESH	8 Byte Signed	Velocity
446	WS.ARM	Command	None
447	WS.DISTMAX	8 Byte Signed	Position
449	WS.DISTMIN	8 Byte Signed	Position
451	WS.IMAX	2 Byte Signed	Float
452	WS.MODE	1 Byte	Integer
453	WS.NUMLOOPS	1 Byte	Integer
454	WS.STATE	1 Byte	Integer
455	WS.T	2 Byte	Integer
456	WS.TDELAY1	2 Byte	Integer
457	WS.TDELAY2	2 Byte	Integer
458	WS.TDELAY3	2 Byte	Integer
459	WS.VTHRESH	8 Byte Signed	Velocity
460	DIN1.FILTER	2 Byte	Integer
461	DIN2.FILTER	2 Byte	Integer
462	DIN3.FILTER	2 Byte	Integer
463	DIN4.FILTER	2 Byte	Integer
464	DIN5.FILTER	2 Byte	Integer
465	DIN6.FILTER	2 Byte	Integer

Instance	Parameter	Data Size	Data Type
466	DIN7.FILTER	2 Byte	Integer
467	FB1.HALLSTATEU	1 Byte	Integer
468	FB1.HALLSTATEV	1 Byte	Integer
469	FB1.HALLSTATEW	1 Byte	Integer
470	DRV.NVSAVE	Command	None
471	MODBUS.DIO	4 Byte	Integer
472	MODBUS.DRV	4 Byte	Integer
473	MODBUS.DRVSTAT	4 Byte	Integer
474	MODBUS.HOME	4 Byte	Integer
475	MODBUS.MOTOR	4 Byte	Integer
476	MODBUS.MT	2 Byte	Integer
477	MODBUS.SM	4 Byte	Integer
478	DRV.FAULT1	2 Byte	Integer
479	DRV.FAULT2	2 Byte	Integer
480	DRV.FAULT3	2 Byte	Integer
481	DRV.FAULT4	2 Byte	Integer
482	DRV.FAULT5	2 Byte	Integer
483	DRV.FAULT6	2 Byte	Integer
484	DRV.FAULT7	2 Byte	Integer
485	DRV.FAULT8	2 Byte	Integer
486	DRV.FAULT9	2 Byte	Integer
487	DRV.FAULT10	2 Byte	Integer
488	MODBUS.PIN	4 Byte	Integer
489	MODBUS.POUT	4 Byte	Integer
490	MODBUS.PSCALE	2 Byte	Integer
491	MODBUS.UNITLABEL	-	String
492	MOTOR.HFPHASEREAD	2 Byte	Integer
493	FB2.ENCRES	4 Byte	Integer
494	FB2.MODE	2 Byte	Integer
495	FB2.SOURCE	2 Byte	Integer
496	MOTOR.TBRAKETO	2 Byte	Integer
497	MODBUS.MSGLOG	1 Byte	Integer
498	USER.INT1	4 Byte Signed	Integer
499	USER.INT2	4 Byte Signed	Integer
500	USER.INT3	4 Byte Signed	Integer
501	USER.INT4	4 Byte Signed	Integer
502	USER.INT5	4 Byte Signed	Integer
503	USER.INT6	4 Byte Signed	Integer
504	USER.INT7	4 Byte Signed	Integer
505	USER.INT8	4 Byte Signed	Integer
506	USER.INT9	4 Byte Signed	Integer
507	USER.INT10	4 Byte Signed	Integer
508	USER.INT11	4 Byte Signed	Integer
509	USER.INT12	4 Byte Signed	Integer

Instance	Parameter	Data Size	Data Type
510	USER.INT13	4 Byte Signed	Integer
511	USER.INT14	4 Byte Signed	Integer
512	USER.INT15	4 Byte Signed	Integer
513	USER.INT16	4 Byte Signed	Integer
514	USER.INT17	4 Byte Signed	Integer
515	USER.INT18	4 Byte Signed	Integer
516	USER.INT19	4 Byte Signed	Integer
517	USER.INT20	4 Byte Signed	Integer
518	USER.INT21	4 Byte Signed	Integer
519	USER.INT22	4 Byte Signed	Integer
520	USER.INT23	4 Byte Signed	Integer
521	USER.INT24	4 Byte Signed	Integer
522	DRV.NVCHECK	8 Byte	Integer
523	FB3.MODE	2 Byte	Integer
524	FB3.P	8 Byte	Integer
525	MODBUS.SCALING	1 Byte	Integer
526	DRV.EMUEPULSEWIDTH	4 Byte	Float
527	DRV.EMUECHECKSPEED	1 Byte	Integer
528	DRV.HWENABLE	1 Byte	Integer
529	DRV.SWENABLE	1 Byte	Integer
530	DRV.TIME	4 Byte	Integer
531	EGEAR.ACCLIMIT	8 Byte	Float
532	EGEAR.DECLIMIT	8 Byte	Float
533	EGEAR.ERROR	8 Byte Signed	Integer
534	EGEAR.LOCK	1 Byte	Integer
535	EGEAR.ON	1 Byte	Integer
536	EGEAR.PULSESIN	2 Byte	Integer
537	EGEAR.PULSESOUT	2 Byte Signed	Integer
538	EGEAR.RATIO	4 Byte Signed	Float
539	EGEAR.TYPE	1 Byte	Integer
540	EXTENCODER.FREQ	4 Byte	Float
541	EXTENCODER.POSITION	8 Byte Signed	Integer
543	EXTENCODER.POSMODULO	8 Byte	Integer
545	MOVE.ACC	8 Byte	None
547	MOVE.DEC	8 Byte	None
549	MOVE.DIR	4 Byte	Integer
550	MOVE.GOABS	Command	None
551	MOVE.GOABSREG	Command	None
552	MOVE.GOHOME	Command	None
553	MOVE.GORELREG	Command	None
554	MOVE.GOREL	Command	None
555	MOVE.GOUPDATE	Command	None
556	MOVE.GOVEL	Command	None
557	MOVE.INPOSITION	4 Byte	Integer

Instance	Parameter	Data Size	Data Type
558	MOVE.INPOSLIMIT	8 Byte Signed	Position
560	MOVE.MOVING	4 Byte	Integer
561	MOVE.POSCOMMAND	8 Byte Signed	Position
566	MOVE.REGOFFSET	8 Byte Signed	Position
568	MOVE.RELATIVEDIST	8 Byte Signed	Position
570	MOVE.RUNSPEED	8 Byte	None
572	MOVE.SCURVETIME	4 Byte	Float
573	MOVE.ABORT	Command	None
574	MOVE.TARGETPOS	8 Byte Signed	Position
576	MOVE.VCMD	4 Byte Signed	None
577	VM.AUTOSTART	4 Byte	Integer
578	VM.RESTART	Command	None
579	VM.START	Command	None
580	VM.STATE	1 Byte	Integer
581	VM.STOP	Command	None
582	VM.ERR	4 Byte	Integer
583	WHEN.FB1MECHPOS	4 Byte	Integer
584	WHEN.FB3P	8 Byte Signed	Integer
586	WHEN.DRVHANDWHEEL	4 Byte	Integer
587	WHEN.DRVTIME	4 Byte	Integer
588	WHEN.PLCMD	8 Byte Signed	Integer
590	WHEN.PLFB	8 Byte Signed	Integer
592	MOVE.DWELLTIME	4 Byte	Integer
593	IL.MI2T	2 Byte	Float
594	AIN.DEADBANDMODE	2 Byte	Integer
595	AIN.MODE	1 Byte	Integer
596	DIO10.DIR	1 Byte	Integer
597	DIO10.INV	1 Byte	Integer
598	DIO11.DIR	1 Byte	Integer
599	DIO11.INV	1 Byte	Integer
600	DIO9.DIR	1 Byte	Integer
601	DIO9.INV	1 Byte	Integer
602	FAULT130.ACTION	1 Byte	Integer
603	FAULT131.ACTION	1 Byte	Integer
604	FAULT132.ACTION	1 Byte	Integer
605	FAULT134.ACTION	1 Byte	Integer
606	FAULT702.ACTION	1 Byte	Integer
607	IP.MODE	2 Byte	Integer
608	LOAD.INERTIA	4 Byte	Float
609	MOTOR.KE	4 Byte	Float
610	VBUS.HALFVOLT	1 Byte	Integer
611	FB2.DIR	1 Byte	Integer
612	FAULT451.ACTION	1 Byte	Integer
613	DRV.HWENDELAY	1 Byte	Integer

Instance	Parameter	Data Size	Data Type
614	DRV.HANDWHEELSRC	1 Byte	Integer
615	IL.KPLOOKUPINDEX	2 Byte	Integer
616	IL.KPLOOKUPVALUE	4 Byte	Float
617	MOTOR.BRAKEIMM	1 Byte	Integer
618	AIN2.CUTOFF	4 Byte	Float
619	AIN2.DEADBAND	2 Byte	Float
620	AIN2.DEADBANDMODE	2 Byte	Integer
621	AIN2.ISCALE	4 Byte	Float
622	AIN2.MODE	1 Byte	Integer
623	AIN2.OFFSET	2 Byte Signed	Float
624	AIN2.PSCALE	8 Byte	Position
626	AIN2.VALUE	2 Byte	Float
627	AIN2.VSCALE	8 Byte	Velocity
630	AIN2.ZERO	Command	None
636	AOUT.CUTOFF	4 Byte	Float
637	AOUT2.CUTOFF	4 Byte	Float
638	AOUT2.ISCALE	4 Byte	Float
639	AOUT2.MODE	2 Byte	Integer
640	AOUT2.OFFSET	2 Byte Signed	Float
641	AOUT2.PSCALE	8 Byte	Position
643	AOUT2.VALUE	8 Byte Signed	Float
645	AOUT2.VALUEU	8 Byte Signed	Float
647	AOUT2.VSCALE	8 Byte	Velocity
649	BODE.IFLIMIT	4 Byte Signed	Float
650	BODE.IFTHRESH	4 Byte Signed	Float
651	BODE.VFLIMIT	4 Byte Signed	Float
652	BODE.VFTHRESH	8 Byte Signed	Velocity
654	DIN10.STATE	1 Byte	Integer
655	DIN11.STATE	1 Byte	Integer
656	DIN21.FILTER	2 Byte	Integer
657	DIN21.INV	1 Byte	Integer
658	DIN21.MODE	2 Byte	Integer
659	DIN21.PARAM	8 Byte Signed	Varies
661	DIN21.STATE	1 Byte	Integer
662	DIN22.FILTER	2 Byte	Integer
663	DIN22.INV	1 Byte	Integer
664	DIN22.MODE	2 Byte	Integer
665	DIN22.PARAM	8 Byte Signed	Varies
667	DIN22.STATE	1 Byte	Integer
668	DIN23.FILTER	2 Byte	Integer
669	DIN23.INV	1 Byte	Integer
670	DIN23.MODE	2 Byte	Integer
671	DIN23.PARAM	8 Byte Signed	Varies
673	DIN23.STATE	1 Byte	Integer

Instance	Parameter	Data Size	Data Type
674	DIN24.FILTER	2 Byte	Integer
675	DIN24.INV	1 Byte	Integer
676	DIN24.MODE	2 Byte	Integer
677	DIN24.PARAM	8 Byte Signed	Varies
679	DIN24.STATE	1 Byte	Integer
680	DIN25.FILTER	2 Byte	Integer
681	DIN25.INV	1 Byte	Integer
682	DIN25.MODE	2 Byte	Integer
683	DIN25.PARAM	8 Byte Signed	Varies
685	DIN25.STATE	1 Byte	Integer
686	DIN26.FILTER	2 Byte	Integer
687	DIN26.INV	1 Byte	Integer
688	DIN26.MODE	2 Byte	Integer
689	DIN26.PARAM	8 Byte Signed	Varies
691	DIN26.STATE	1 Byte	Integer
692	DIN27.FILTER	2 Byte	Integer
693	DIN27.INV	1 Byte	Integer
694	DIN27.MODE	2 Byte	Integer
695	DIN27.PARAM	8 Byte Signed	Varies
697	DIN27.STATE	1 Byte	Integer
698	DIN28.FILTER	2 Byte	Integer
699	DIN28.INV	1 Byte	Integer
700	DIN28.MODE	2 Byte	Integer
701	DIN28.PARAM	8 Byte Signed	Varies
703	DIN28.STATE	1 Byte	Integer
704	DIN29.FILTER	2 Byte	Integer
705	DIN29.INV	1 Byte	Integer
706	DIN29.MODE	2 Byte	Integer
707	DIN29.PARAM	8 Byte Signed	Varies
709	DIN29.STATE	1 Byte	Integer
710	DIN30.FILTER	2 Byte	Integer
711	DIN30.INV	1 Byte	Integer
712	DIN30.MODE	2 Byte	Integer
713	DIN30.PARAM	8 Byte Signed	Varies
715	DIN30.STATE	1 Byte	Integer
716	DIN31.FILTER	2 Byte	Integer
717	DIN31.INV	1 Byte	Integer
718	DIN31.MODE	2 Byte	Integer
719	DIN31.PARAM	8 Byte Signed	Varies
721	DIN31.STATE	1 Byte	Integer
722	DIN32.FILTER	2 Byte	Integer
723	DIN32.INV	1 Byte	Integer
724	DIN32.MODE	2 Byte	Integer
725	DIN32.PARAM	8 Byte Signed	Varies

Instance	Parameter	Data Size	Data Type
727	DIN32.STATE	1 Byte	Integer
728	DIN9.STATE	1 Byte	Integer
729	DOUT10.STATE	1 Byte	Integer
730	DOUT10.STATEU	1 Byte	Integer
731	DOUT11.STATE	1 Byte	Integer
732	DOUT11.STATEU	1 Byte	Integer
733	DOUT21.MODE	1 Byte	Integer
734	DOUT21.PARAM	8 Byte Signed	Float
736	DOUT21.STATE	1 Byte	Integer
737	DOUT21.STATEU	1 Byte	Integer
738	DOUT22.MODE	1 Byte	Integer
739	DOUT22.PARAM	8 Byte Signed	Float
741	DOUT22.STATE	1 Byte	Integer
742	DOUT22.STATEU	1 Byte	Integer
743	DOUT23.MODE	1 Byte	Integer
744	DOUT23.PARAM	8 Byte Signed	Float
746	DOUT23.STATE	1 Byte	Integer
747	DOUT23.STATEU	1 Byte	Integer
748	DOUT24.MODE	1 Byte	Integer
749	DOUT24.PARAM	8 Byte Signed	Float
751	DOUT24.STATE	1 Byte	Integer
752	DOUT24.STATEU	1 Byte	Integer
753	DOUT25.MODE	1 Byte	Integer
754	DOUT25.PARAM	8 Byte Signed	Float
756	DOUT25.STATE	1 Byte	Integer
757	DOUT25.STATEU	1 Byte	Integer
758	DOUT26.MODE	1 Byte	Integer
759	DOUT26.PARAM	8 Byte Signed	Float
761	DOUT26.STATE	1 Byte	Integer
762	DOUT26.STATEU	1 Byte	Integer
763	DOUT27.MODE	1 Byte	Integer
764	DOUT27.PARAM	8 Byte Signed	Float
766	DOUT27.STATE	1 Byte	Integer
767	DOUT27.STATEU	1 Byte	Integer
768	DOUT28.MODE	1 Byte	Integer
769	DOUT28.PARAM	8 Byte Signed	Float
771	DOUT28.STATE	1 Byte	Integer
772	DOUT28.STATEU	1 Byte	Integer
773	DOUT29.MODE	1 Byte	Integer
774	DOUT29.PARAM	8 Byte Signed	Float
776	DOUT29.STATE	1 Byte	Integer
777	DOUT29.STATEU	1 Byte	Integer
778	DOUT30.MODE	1 Byte	Integer
779	DOUT30.PARAM	8 Byte Signed	Float

Instance	Parameter	Data Size	Data Type
781	DOUT30.STATE	1 Byte	Integer
782	DOUT30.STATEU	1 Byte	Integer
783	DOUT9.STATE	1 Byte	Integer
784	DOUT9.STATEU	1 Byte	Integer
785	DRV.BLINKDISPLAY	Command	None
786	DRV.CLRCRASHDUMP	Command	None
787	DRV.CMDDELAY		Float
789	DRV.NVLOAD	Command	None
790	DRV.RUNTIME		String
791	DRV.SETUPREQBITS	4 Byte	Integer
792	DRV.WARNING1	4 Byte	Integer
793	DRV.WARNING2	4 Byte	Integer
794	DRV.WARNING3	4 Byte	Integer
795	EIP.CONNECTED	1 Byte	Integer
796	EIP.POSUNIT	4 Byte	Integer
797	EIP.PROFUNIT	4 Byte	Integer
798	FAULT139.ACTION	1 Byte	Integer
803	FB1.CALTHRESH	8 Byte	Integer
806	FB1.P	8 Byte Signed	Position
808	FB1.PDIR	1 Byte	Integer
809	FB1.PIN	4 Byte	Integer
810	FB1.POFFSET	8 Byte Signed	Position
812	FB1.POUT	4 Byte	Integer
813	FB1.PUNIT	4 Byte	Integer
814	FB1.USERBYTE	1 Byte	Integer
815	FB1.USERDWORD	4 Byte	Integer
816	FB1.USERWORD	2 Byte	Integer
817	FB2.P	8 Byte Signed	Integer
819	FB2.PIN	4 Byte	Integer
820	FB2.POFFSET	8 Byte Signed	Position
822	FB2.POUT	4 Byte	Integer
823	FB2.PUNIT	4 Byte	Integer
824	FB3.P	8 Byte Signed	Integer
826	FB3.PDIR	1 Byte	Integer
827	FB3.PIN	4 Byte	Integer
828	FB3.POFFSET	8 Byte Signed	Position
830	FB3.POUT	4 Byte	Integer
831	FB3.PUNIT	4 Byte	Integer
832	HOME.MAXDIST	8 Byte Signed	Position
834	IL.DIFOLD	4 Byte	Float
835	IL.MI2TWTHRESH	1 Byte	Integer
836	IL.MIMODE	1 Byte	Integer
837	IP.RESET	Command	None
838	MOTOR.VOLTMIN	2 Byte	Integer

Instance	Parameter	Data Size	Data Type
839	MOTOR.VOLTRATED	2 Byte	Integer
840	MOTOR.VRATED	8 Byte Signed	Float
843	SD.LOAD	Command	None
844	SD.SAVE	Command	None
845	SD.STATUS	1 Byte	Integer
846	VL.FBUNFILTERED	8 Byte Signed	Velocity
848	WS.DISARM	Command	None
849	WS.FREQ	4 Byte	Float
850	WS.TDELAY4	2 Byte	Integer
851	WS.CHECKT	2 Byte	Integer
852	WS.CHECKV	8 Byte Signed	Velocity
859	AOUT.VSCALE	8 Byte	Velocity
861	WS.TSTANDSTILL	2 Byte	Integer
862	WS.TIRAMP	2 Byte	Integer
863	FB1.PMTSAVEEN	1 Byte	None
864	FB1.PMTBITS	1 Byte	None
865	MOTOR.IMTR	2 Byte	Integer
866	IL.FBSOURCE	1 Byte	Integer
867	MOTOR.IMID	4 Byte	Float
868	WS.CHECKMODE	1 Byte	Integer
869	REGEN.POWERFILTERED	8 Byte	Integer

12 Appendix C: 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.

For assistance with your application needs, visit www.kollmorgen.com or contact us at:

North America
KOLLMORGEN
203A West Rock Road
Radford, VA 24141 USA

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

Europe
KOLLMORGEN Europe GmbH
Pempelfurtstraße 1
40880 Ratingen, Germany

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

Asia
KOLLMORGEN
Rm 2205, Scitech Tower, China
22 Jianguomen Wai Street

Internet	www.kollmorgen.com
E-Mail	sales.asia@kollmorgen.com
Tel.:	+86 - 400 666 1802
Fax:	+86 - 10 6515 0263