AKD® and AKD®2G

EtherNet/IP with Studio 5000 Manual

Edition: A, March 2022

Part Number 907-200009-00

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

Record of Document Revisions:

Revision	Remarks
A, 03/2022	Launch version

- AKD is a registered trademark of Kollmorgen Corporation
- Allen-Bradley® CompactLogix® and ControlLogix® control systems are registered trademarks of Rockwell Automation
- Allen-Bradley® Micro800™ controllers are registered trademarks of Rockwell Automation
- Allen-Bradley® MicroLogix™ 1100 controllers are registered trademarks of Rockwell Automation
- Allen-Bradley® MicroLogix™ 1400 controllers are registered trademarks of Rockwell Automation
- 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 Introduction

This manual provides an easy start guide for using Studio 5000®, an overview on how to import and configure the AKD Add-On Instructions (AOI) using Studio 5000, as well as a reference to the Add-On Instructions.

NOTE

In this document "AKD" refers to the AKD family of servo drives, while references to specific drive models are "AKD1G" or "AKD2G".

Studio 5000 sample projects and Add-On Instructions, which demonstrate an EtherNet/IP network with a CompactLogix controller and the AKD and AKD2G EtherNet/IP drives are available on the Kollmorgen website.

Sample projects are based on a CompactLogix controller, which can be changed to another controller which supports Studio 5000.

This document assumes that the reader has a basic knowledge of EtherNet/IP protocols, AKD or AKD2G drives, and Rockwell Studio 5000.

For additional information on the Command Assembly, Response Assembly, Response Types, and other information related to the AKD1G EtherNet/IP and AKD2G EtherNet/IP drives see the AKD1G EtherNet/IP Communication Manual and AKD2G EtherNet/IP online help.

A list of Add-On Instructions can be found under Add-On Instructions List and a table of AOI revisions and compatibility can be found in AOI revisions library version 6.0. A more detailed comparision of features can be found in the AKD vs AKD2G EtherNet/IP Comparison Chart.

1.1 Abbreviations and Terminology

.DN:	Done bit
.ER:	Error bit
.IP:	In Process bit
.PC:	Process Complete bit
AOI:	Add-On Instruction
Instance (ID):	AKD1G uses the term Instance = ID, AKD2G uses ID.
N.C. Contact:	Normally Closed contact
N.O. Contact:	Normally Open contact
PLC:	Programmable Logic Controller
Profile_In_Progress:	Based on the Response Assembly Data Structure's bit 0 in Status Word 1. Profile_In_Progress is the term used inside the AOI with AKD1G drives. Known as In Motion with AKD2G drives.
RPI:	Request Packet Interval, also known as Expected Packet Rate
Step Number:	Also known as a Step or Function_Step in the code depending on the AOI. Referred to as Step Number in the documentation for consistency.
Tag name:	Term for a variable. The terms Tag or Tag name may be preferred by other softwares.
ONS:	One Shot in Studio 5000

1.2 Controller Support and Add-On Instructions

The Add-On Instructions (AOI) described in this manual are only compatible with CompactLogix and ControlLogix controllers.

The Add-On Instructions are Studio 5000 instructions on defining AKD and AKD2G drives and axis configurations. Data Types and Add-On Instructions library rev 6.0 or later must be used with AKD2G drives and can also be used with AKD1G drives. Some AOIs are exclusive to either the AKD or AKD2G drive in compatibility, but most AOIs are compatible with both. This manual will detail the compatibility of each AOI definition and its description. The existing Data Types, Add-On Instructions library rev 5.0, and Sample project will be archived and still available for AKD1G only applications.

These instructions are intended to be imported into a Studio 5000 project. Once defined in a project, they function just like native Studio 5000 instructions. The Add-On Instructions encapsulate the most commonly used logic and EtherNet/IP commands for AKD and AKD2G axes providing easily reusable tools to operate drives and axes while promoting consistency across different projects.

- Studio 5000 uses the native MSG instruction for sending Explicit Messages.
- MicroLogix 1400 controllers are supported using Explicit Messaging only. The Add-On Instructions provided with Studio 5000 cannot be used with these controllers. At the time of this publication, the MicroLogix 1400 has only been tested with the AKD1G Drive.
- MicroLogix 1100 and SLC500 controllers are not supported.
- No testing for compatibility with the Allen-Bradley Micro800 series has been conducted to date.

1.3 What's New For AOI library v6.0?

AOI library v6.0 is a major overhaul from AOI library v5.0 with changes required to accommodate the AKD2G EtherNet/IP drive and changes to allow most of the existing AOIs from v5.0 to be compatible with both the AKD1G and AKD2G EtherNet/IP drives.

- New data type **AKD_Data** and its implementation into the AKD_Axis Data Type to detach the Axis structure declaration to the Generic Ethernet Module declaration and data type in order to facilitate the importation of the AOIs in version 6.0.
- For existing users of the AKD1G EtherNet/IP drive, AOI library v5.0 (Data Types, AOIs, and Sample project) will be archived and can still be used for exclusive AKD1G EtherNet/IP systems, but moving forward it will no longer be maintained.
- For new applications of the AKD drive, AKD2G EtherNet/IP drive, or a combination of the two drives, it is recommended to use AOI library v6.0 and later data types, AOIs, and Sample projects.
- New **AKD2G_Drive AOI** required for EtherNet/IP communications and AOI usage with the AKD2G EtherNet/IP Drive.
- New **AKD2G_Set_Motion_Task AOI** which uses the new **Command Type 0x09 Motion Task** and is exclusive to the AKD2G EtherNet/IP drive.
- New **AKD2G_Motion_Status** diagnostic AOI, which is exclusive to the AKD2G EtherNet/IP drive.

1.4 AOI revisions library version 6.0

The table below provides a full list of all AOIs available in version 6.0 and lists the compatibility of each AOI.

AOI	Version	AKD	AKD2G
AKD2G_Drive	1.0		✓
AKD2G_Motion_Status	1.1		✓
AKD2G_Set_Motion_Task	1.0		✓
AKD_Command_Assembly	3.1	✓	✓
AKD_Command_Control_Word	2.0	✓	✓
AKD_Disable	3.0	✓	✓
AKD_Drive	3.0	✓	
AKD_Enable	3.0	✓	✓
AKD_Fault_Reset	5.0	✓	✓
AKD_Get_Attribute	3.1	✓	✓
AKD_Get_Parameter	3.0	✓	✓
AKD_Home	5.0	✓	✓
AKD_Jog	3.0	✓	✓
AKD_Motion_Status	3.1	✓	
AKD_Move	4.1	✓	✓
AKD_Response_Assembly	3.1	✓	✓
AKD_Response_Status_Words	3.0	✓	✓
AKD_Set_Attribute	3.0	✓	✓
AKD_Set_Home_Mode	3.0	✓	✓
AKD_Set_Mode	5.0	✓	✓
AKD_Set_Motion_Task	3.0	✓	
AKD_Set_Parameter	3.0	✓	✓
AKD_Shutdown	5.0	✓	✓
AKD_Shutdown_Reset	3.0	✓	✓
AKD_Start_MotionTask	3.0	✓	✓
AKD_Stop_Smooth	5.0	✓	✓
AKD_Torque_Move	3.0	✓	✓

2 AKD1G Installation and Setup

The following procedures pertain to the Quick Start and Adding the Generic Ethernet Module for the AKD1G EtherNet/IP Communications sections prior to importing data types and the Add-On Instructions.

For procedures pertaining to the AKD2G drive, see the AKD2G Installation and Setup section of this manual.

See the following manuals for installation and setup of an AKD EtherNet/IP drive:

- **AKD Installation Manual**: This manual provides instructions for installation and drive setup.
- **AKD Online Help**: The help system describes how to use your drive in common applications. It also provides tips for maximizing your system performance with the AKD. Additionally, the Online Help provides documentation for the parameters and commands used to program the drive.
- **Accessories Manual**: This manual provides documentation for accessories like cables and regen resistors used with AKD. Regional versions of this manual are available.
- **AKD EtherNet/IP Communications Manual**: This manual describes the installation, setup, range of functions, and software protocol for the AKD EtherNet/IP product series.

2.1 Quick Start with the AKD1G Sample project

The **Add On Instruction Library for AKD EtherNet/IP** can be downloaded in .ZIP file format from kollmorgen.com.

This library contains:

- The most recent library of Add-On Instruction (AOI) and Data Types (see the section, Importing Data Types and AOIs Into a Project).
- The Getting Started With AKD EtherNet/IP application note.
- The AKD Simple Example Program *.ACD project.

For application notes, FAQ, and other supplementary AKD EtherNet/IP support, see the AKD EtherNet/IP Support Landing Page on the Kollmorgen website.

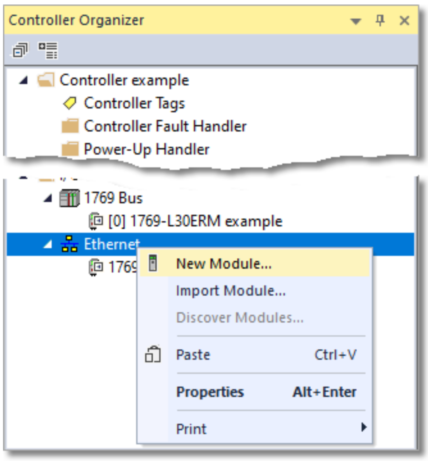
2.2 Adding the Generic Ethernet Module for AKD Communication

NOTE

This section applies to AKD1G. For AKD2G see the Adding the Ethernet I/O Module for AKD2G Communication section of this manual.

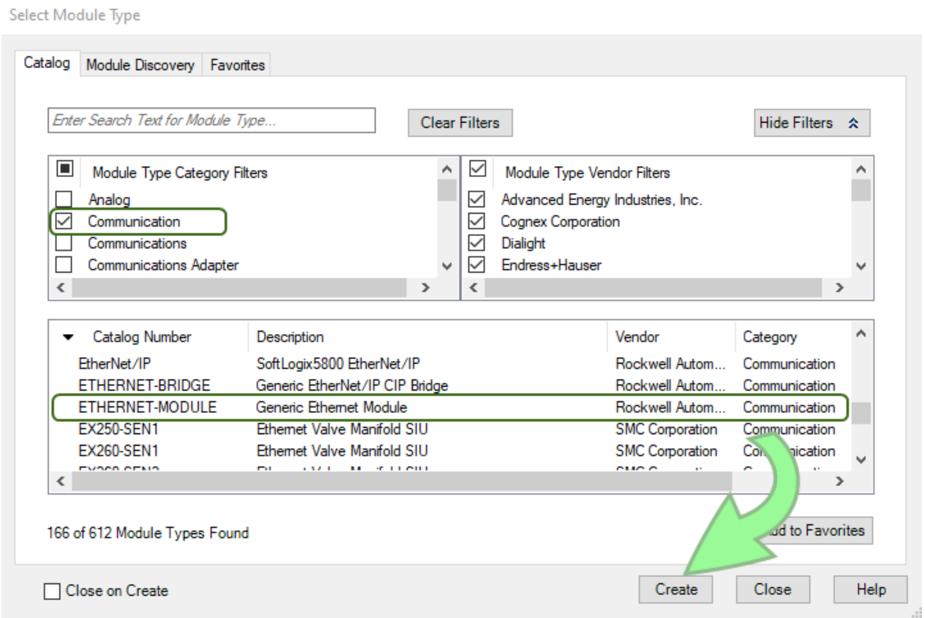
These basic instructions can be used for any Rockwell PLC that uses Studio 5000 and supports EtherNet/IP.

1. Start **Studio 5000** and open the project to be used with the AKD EtherNet/IP drive.
2. In the **Controller Organizer**, right-click on **Ethernet** under **I/O Configuration** and select **New Module...**.

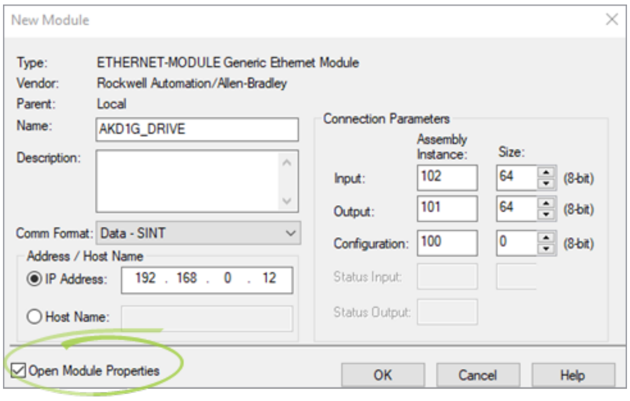


Clicking New Module opens the Select Module Type dialog.

3. Under the **Catalog** tab, select **Communication**.
4. In the Catalog Number column, select **ETHERNET-MODULE**.
5. Click **Create** to open the New Module dialog.

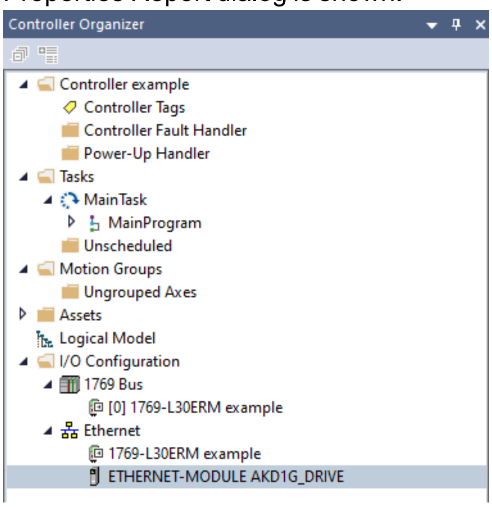


6. Enter the settings for the new module as shown below, ensuring the **Open Module Properties** checkbox is selected and click **OK**.



Field		Value
Name		AKD1G_Drive (Example only. Enter the name you wish to use.)
Description		Optional. Enter a text description for the drive.
Comm Format		Data-SINT
IP Address		Set to the same IP Address as the target AKD Drive. For more details see the section Setting the IP Address of the AKD drive using WorkBench.
Connection Parameters	Input Assembly Instance	102
	Input Size	64
	Output Assembly Instance	101
	Output Size	64
	Configuration Assembly Instance	100
	Configuration Size	0

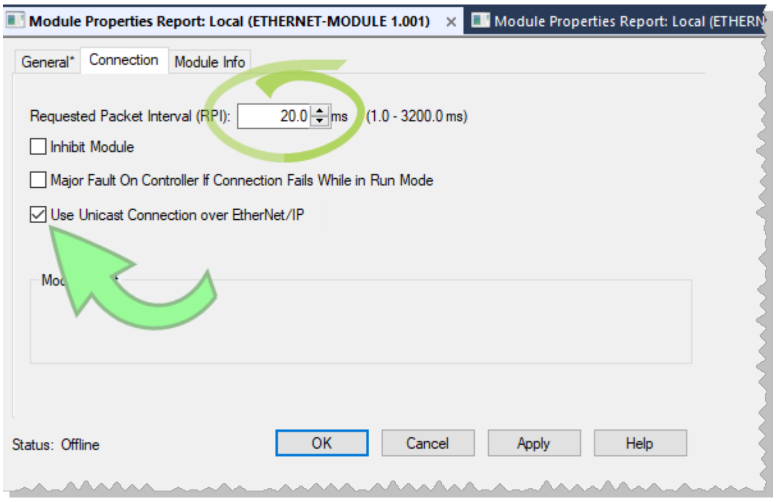
7. The ETHERNET-MODULE is added to the Controller Organizer tree under Ethernet and the Module Properties Report dialog is shown.



8. Under the **Connection** tab, set the **Requested Packet Interval (RPI)** values to 20.0 ms and above as required.

   NOTE

   When using WorkBench and EtherNet/IP simultaneously through the service port of the AKD drive, 40.0 ms or above is recommended.
9. Select **Use Unicast Connection over EtherNet/IP**.
10. Click **Apply**.
11. Click **OK**.



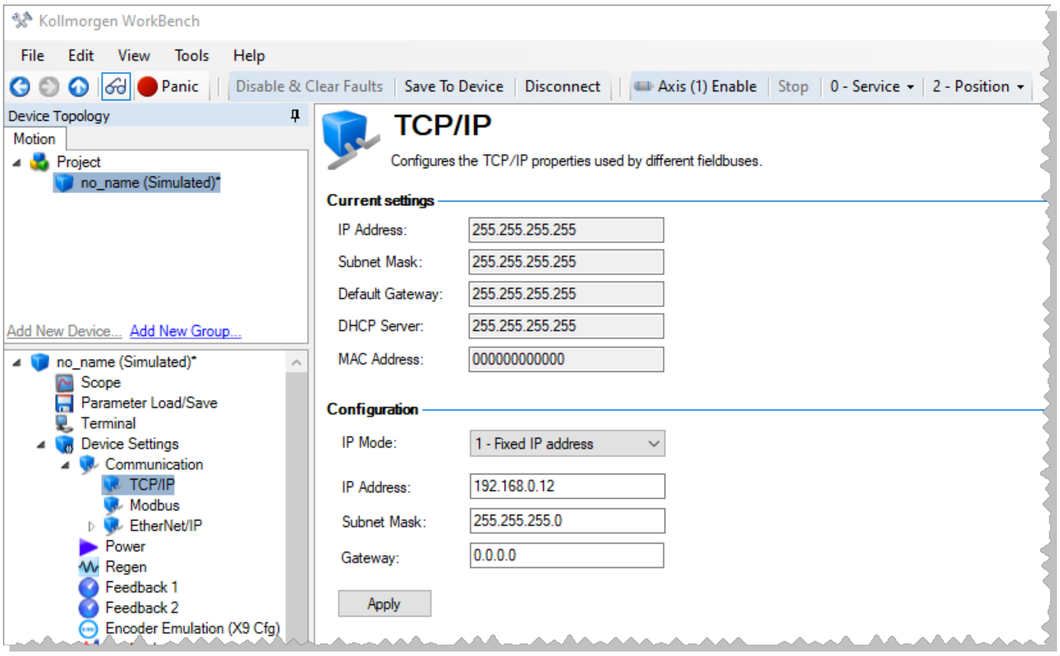
The AKD EtherNet/IP drive should now be configured and displayed under Ethernet in the Controller Organizer as **ETHERNET-MODULE AKD1G_DRIVE** or the name chosen during configuration.

Ensure that the Ethernet port for your controller is setup with a compatible IP address on the same subnet as the AKD drive IP address. See your controller's user manual for more information.

2.2.1 Setting the IP Address of the AKD drive using WorkBench

The Kollmorgen WorkBench screen capture below shows the TCP/IP screen with an example IP Address setup for the AKD drive using **IP Mode: 1 - Fixed IP Address**, which is typical with EtherNet/IP systems. IP Mode 0 - Rotary switches may also be used. Atypical with EtherNet/IP systems, but also available, is IP Mode 2 - DHCP/Auto IP. The settings for IP Mode 0 and IP Mode 2 are network- and application-specific and are to be determined by the user.

Note that the IP Address in the WorkBench screen capture below matches the IP Address shown in the Studio 5000 New Module dialog in step 5 of Adding the Generic Ethernet Module for AKD Communication.



3 AKD2G Installation and Setup

The following procedures pertain to the Quick Start and Adding the Ethernet I/O Module for AKD2G Communication sections prior to importing data types and the Add-On Instructions.

For procedures pertaining to the AKD1G drive, see the AKD1G Installation and Setup section of this manual.

See the following manuals for guidance on the installation and setup of an AKD2G EtherNet/IP drive:

- **AKD2G Installation Manual:** This manual provides instructions for installation and drive setup.
- **AKD2G Online Help:** The help system describes how to use your drive in common applications. It also provides tips for maximizing your system performance with the AKD2G. Additionally, the Online Help provides documentation for the parameters and commands used to program the drive.
- **Accessories Manual:** This manual provides documentation for accessories like cables and regen resistors used with AKD2G. Regional versions of this manual are available on the Kollmorgen website.
- **AKD2G EtherNet/IP Communications Manual**: This manual describes the installation, setup, range of functions, and software protocol for the AKD2G EtherNet/IP product series.

3.1 Quick Start with the AKD2G Sample project

The **Add On Instruction Library for AKD EtherNet/IP** can be downloaded in .ZIP file format from kollmorgen.com.

This library contains:

1. The most recent library of Add-On Instructions and data types (see the section, Importing Data Types and AOIs Into a Project).
2. The Getting Started With AKD2G EtherNet/IP application note.
3. The AKD2G Simple Example Program *.ACD project.

For application notes, FAQ, and other supplementary AKD EtherNet/IP support, see the AKD2G EtherNet/IP Support Landing Page on the Kollmorgen website.

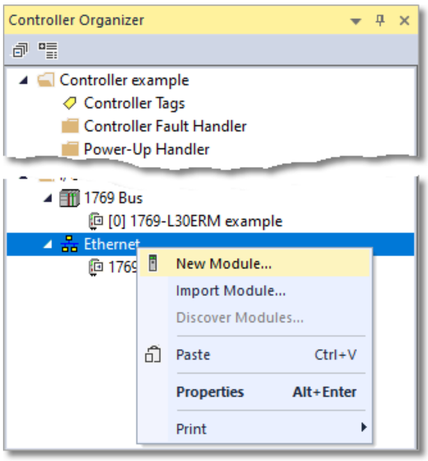
3.2 Adding the Ethernet I/O Module for AKD2G Communication

NOTE

This section applies to AKD2G. For AKD1G, see Adding the Generic Ethernet Module for AKD Communication section of this manual.

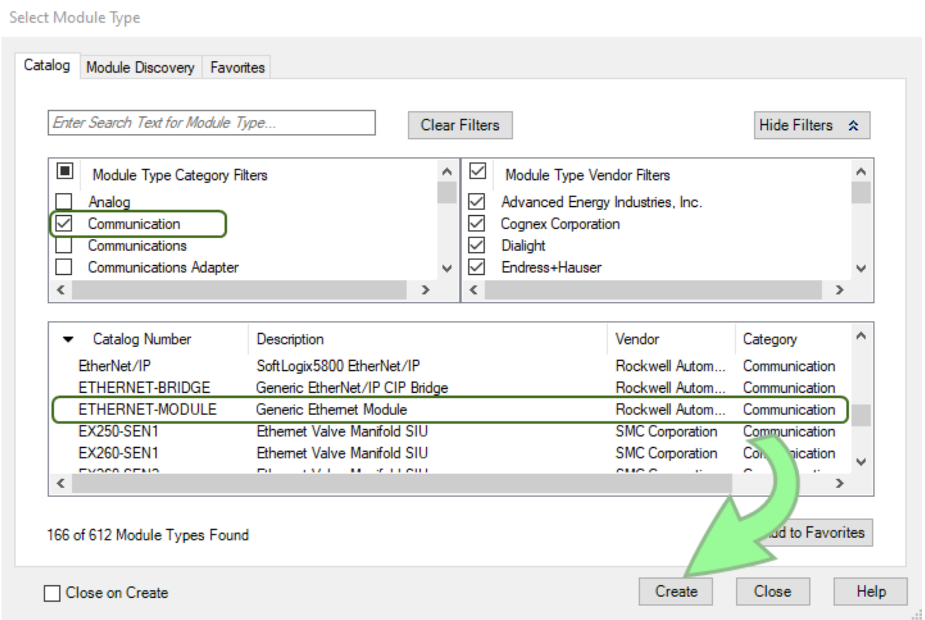
These basic instructions can be used for any Rockwell PLC that uses Studio 5000 and supports EtherNet/IP.

1. Start **Studio 5000** and open the project to be used with the AKD2G EtherNet/IP drive.
2. In the **Controller Organizer**, right-click on **Ethernet** under **I/O Configuration** and select **New Module…**.



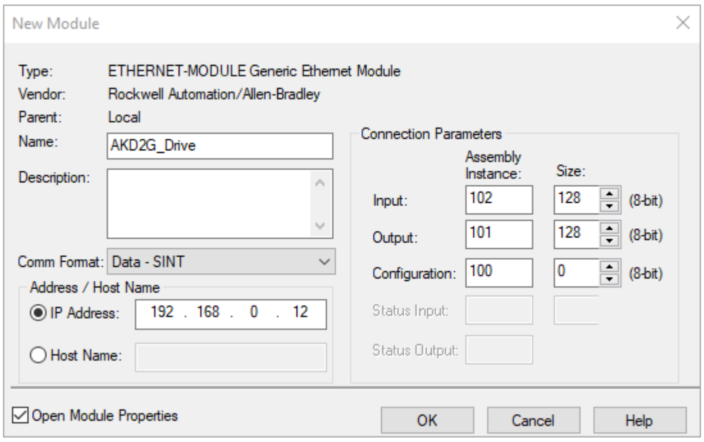
Clicking New Module opens the Select Module Type dialog.

3. Under the **Catalog** tab, select **Communication**.
4. In the Catalog Number column, select **ETHERNET-MODULE**.
5. Click **Create**.



Clicking Create opens the New Module dialog.

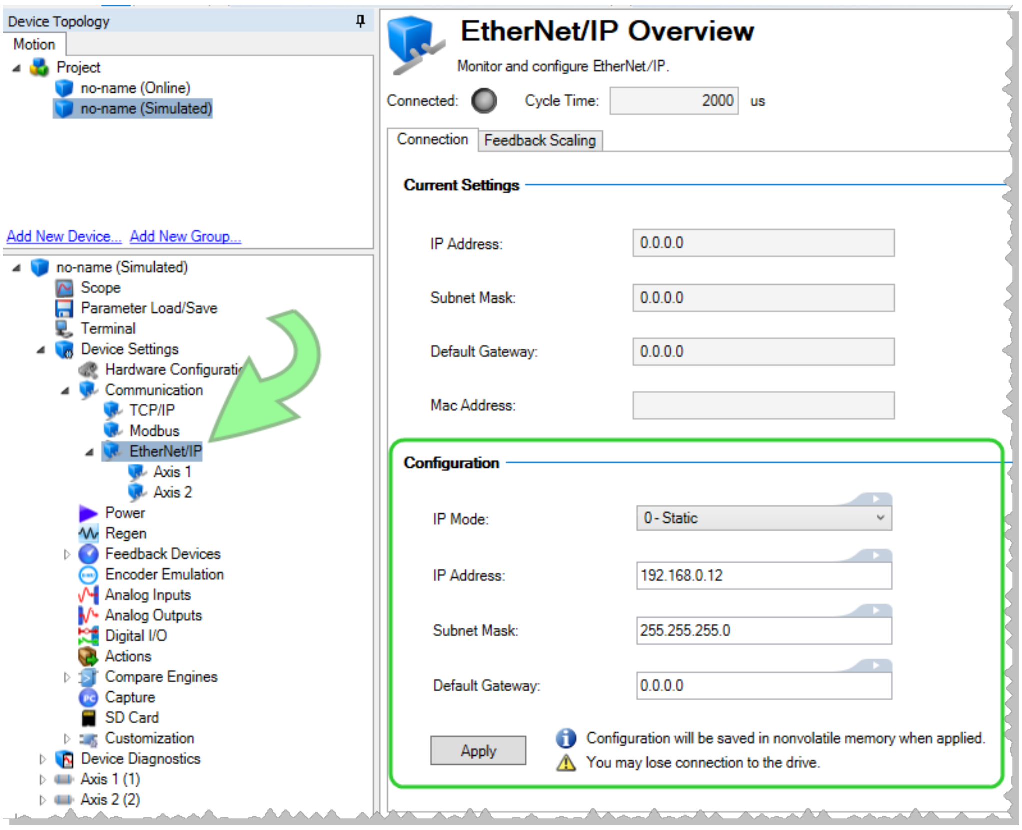
6. Enter the settings for the new module as shown below, ensuring the **Open Module Properties** checkbox is selected and click **OK**.



Field		Value
Name		AKD2G_Drive (Example only. Enter the name you wish to use.)
Description		Optional. Enter a text description for the drive.
Comm Format		Data-SINT
IP Address		Set to the same IP Address as set in the target AKD2G drive (EIP.IPADDRESS)
Connection Parameters	Input Assembly Instance	102
	Input Size	128
	Output Assembly Instance	101
	Output Size	128
	Configuration Assembly Instance	100
	Configuration Size	0

The AKD2G has a dedicated IP Address for EtherNet/IP (Fieldbus ports X11 [Port 2] and X12 [Port 1]).

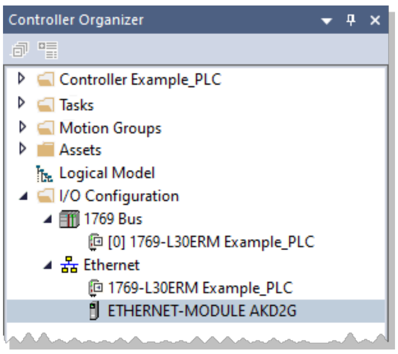
The Fieldbus port settings can be accessed in WorkBench from **Project** → **drive_name (Online)** → **Device Settings** → **Communication** → **EtherNet/IP** → **Connection** tab.



NOTE

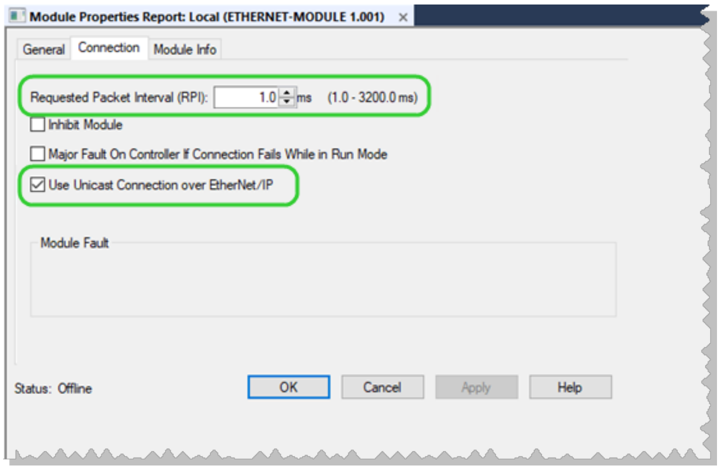
WorkBench can access the drive over Ethernet via the fieldbus port(s), but the RPI may need to be extended while polling the drive with WorkBench and EtherNet/IP simultaneously. Optionally, the Service/HMI X20 port may be used for a dedicated WorkBench connection and/or to an HMI (i.e., Modbus TCP) independent of EtherNet/IP.
Commissioning and tuning via WorkBench should be conducted using the X20 Service Port. X11/X12 do not support the PST (Performance Autotuner).

7. The ETHERNET-MODULE is added to the Controller Organizer tree under Ethernet.

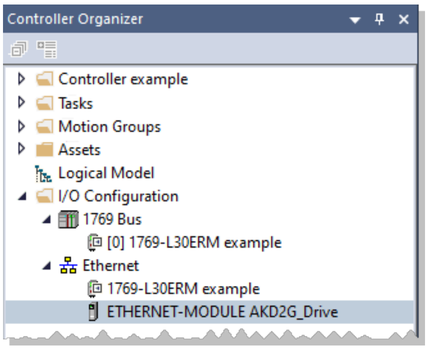


The Module Properties Report dialog displays.

8. Under the **Connection** tab, set the **Requested Packet Interval (RPI)** values to 1 ms and above as required.
9. Select **Use Unicast Connection over EtherNet/IP**.
10. Click **Apply**.
11. Click **OK**.



The AKD2G EtherNet/IP drive as a Generic Ethernet Module should now be configured and shown under Ethernet in the Controller Organizer.



Ensure the Ethernet port for your controller is setup with a compatible IP address on the same subnet as the AKD2G drive's EtherNet/IP address. See your controller's user manual for more information.

4 Importing Data Types and AOIs Into a Project

NOTE

Add-On Instruction Library v6.0 or later is required in systems with AKD2G drives.

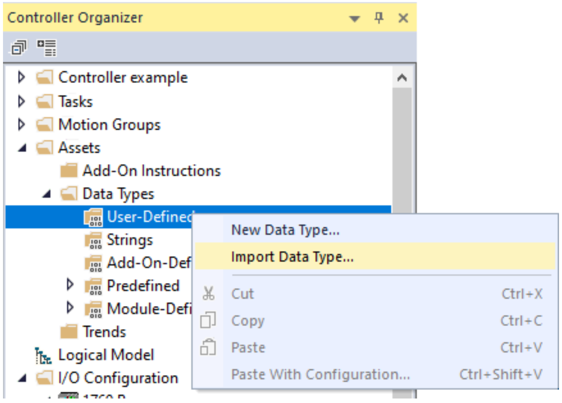
4.1 Importing Data Types

The Generic Ethernet Modules for the AKD and/or AKD2G drives should be added and configured prior to importing the data types. The following steps guide you through importing data types.

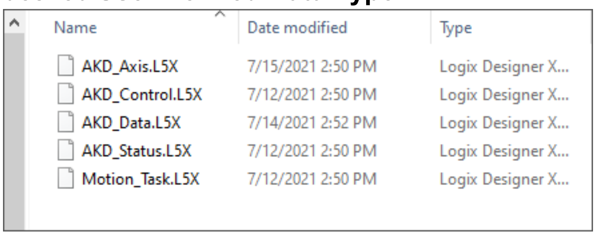
ⓘIMPORTANT

- The User Defined Data Types must be imported before importing Add-On Instructions into your project.
- It is very important to import the data types into your project in the order shown in the UDT Import Order table in step 6.

1. In the **Controller Organizer**, under **Data Types**, right-click the **User-Defined** folder.
2. Select **Import Data Type…**



3. Browse to the location of the **AKD User Defined Data Type library** on your PC and select the desired **User Defined Data Type**.



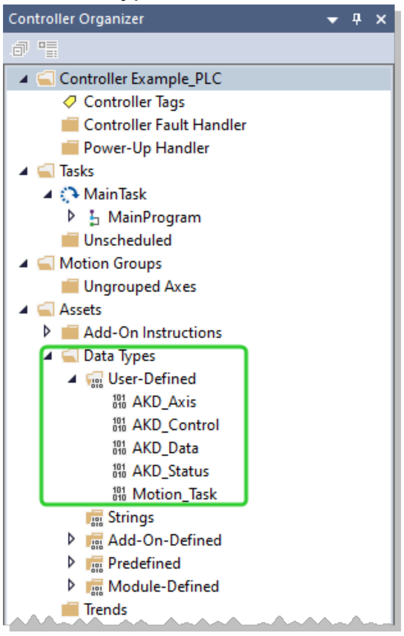
4. Click **Import…**
5. Click **OK** in the **Import Configuration** dialog.

6. Repeat the steps above to import all of the needed data types.

UDT Import Order

Order	File	Description
1	AKD_Control.L5X	Control message for sending to axis
2	AKD_Status.L5X	Status message for updating from axis
3	AKD_Data	Array to hold the data
4	AKD_Axis.L5X	Axis definition
5	Motion_Task.L5X	Motion Task data table structure

The data types should now be listed under **Data Types** → **User-Defined**.



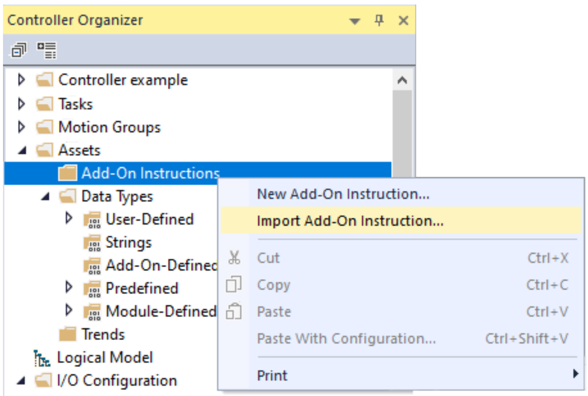
4.2 Importing Add-On Instructions

ⓘIMPORTANT

The User Defined Data Types must be imported before importing Add-On Instructions into your project.

The next step after importing the Data Types is to import the Add-On Instructions; the following steps guide you through the process.

1. Under **Assets**, right-click on the **Add-On Instructions** folder and select **Import Add-On Instruction…**



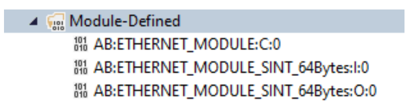
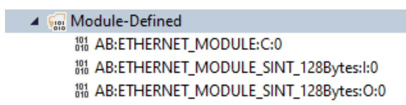
2. Browse to the location of the **AKD Add-On Instruction library** and select the desired AOI.
3. Click **Import…**

NOTE

To support a system running both AKD and AKD2G drives and ensure complete functionality, import all of the files associated with your drive and application. A list of AOIs and which drives they are associated with can be found below.

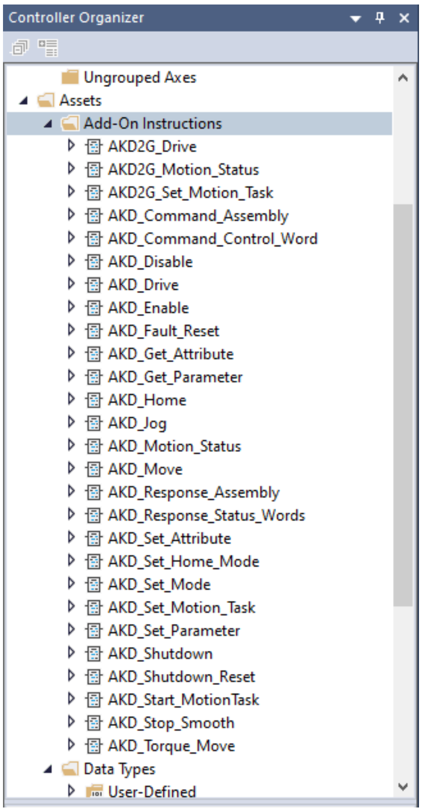
ⓘIMPORTANT

The Generic Ethernet Modules must be declared prior to importing the AKD_Drive or AKD2G_Drive AOIs into Studio 5000. This is due to these AOIs being codependent on the data types for the Ethernet Module (128 Bytes I/O for AKD2G and 64 Bytes I/O for AKD). The data types for the Ethernet Module are created when the Generic Ethernet Module is added.

AKD	AKD2G
	

4. Click **OK** on the **Import Configuration** dialog.
5. Repeat for all files in the AOI revisions library version 6.0 table to import all of the AOIs and have full functionality.

Any imported AOIs should now be listed under the Add-On Instructions folder under Assets, as shown below.



4.3 Exclusive Add-On Instructions Usage

Exclusive Add-On Instructions Usage

The AOIs listed below are compatible with only their corresponding drive. All other AOIs are compatible for usage with both AKD and AKD2G EtherNet/IP drives.

AKD	AKD2G
AKD_Drive	AKD2G_Drive
AKD_Motion_Status	AKD2G_Motion_Status
AKD_Set_Motion_Task	AKD2G_Set_Motion_Task

4.4 Scaling for AKD1G

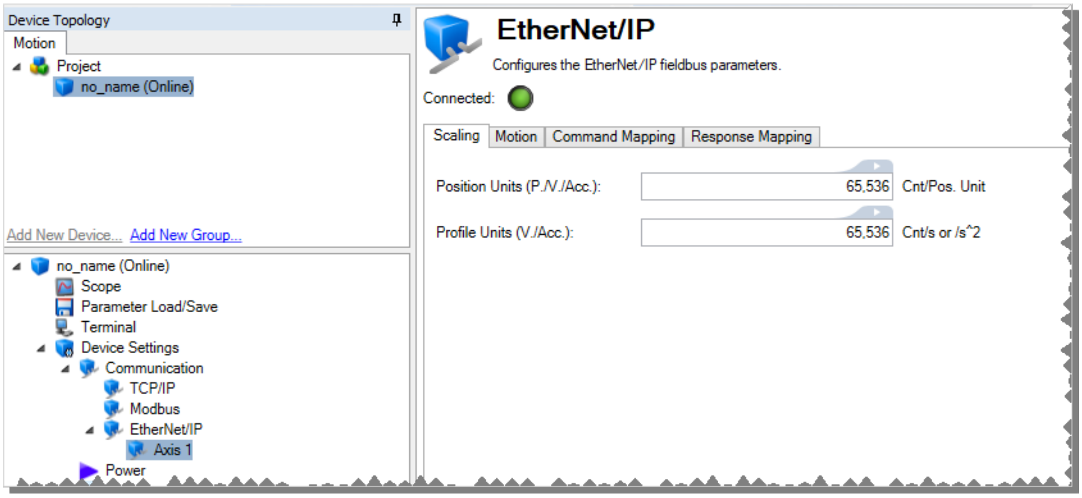
It is important for first time users to understand how EtherNet/IP scaling works. Data sent and received over EtherNet/IP are in counts/position unit, counts/s or counts/s^2 for Position, Velocity, and Acceleration respectively.

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.

4.4.1 EtherNet/IP Scaling in WorkBench

To view the default scaling as shown below, click **Start Page** → ***drive name*** **(Online)** → **Settings** → **Communication** → **EtherNet/IP** and click the **Scaling** tab.



4.4.2 Position Unit Scaling

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

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

Definition

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

Default scaling

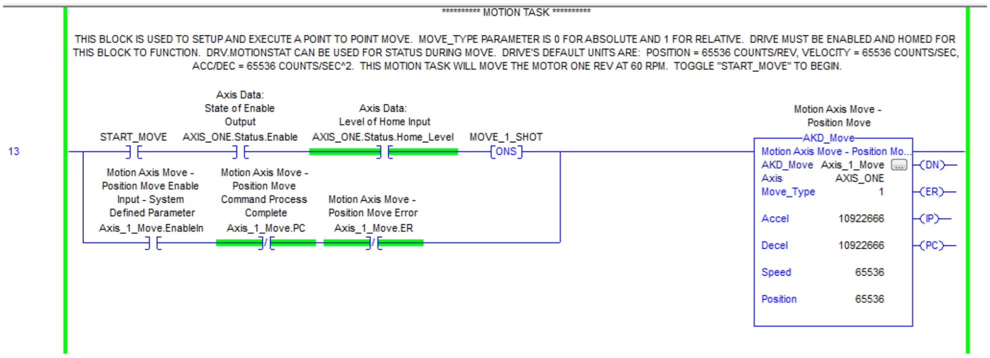
$$AXIS\#.EIP.POSUNIT = 2^{16}\ or\ 65536\ counts$$

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

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

This means if the AKD_Move block 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.

From the Sample project the AKD_Move AOI is setup to make a Relative Move of 1 motor revolution.



NOTE

The PLC/HMI must do the conversions from real-world units (i.e. inches, inches/sec, etc.) to motor revolutions and motor 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.

$$1\ inch \cdot \frac{1\ rev.\ of\ ballscrew}{0.2\ inch} \cdot \frac{5\ motor\ revs.}{1\ rev.\ of\ ballscrew} \cdot \frac{65536\ counts}{1\ motor\ rev.} = 1638400\ counts$$

4.4.3 Velocity and Acceleration Unit Scaling

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

The following example demonstrates that when the default value of 65536 is used for EIP.PROFUNIT then 65536 counts/sec is equal to 1 rev/sec or 60 RPM. For 10 rps, 655360 is used to set the Speed value for AKD_Move.

Definition

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

Default scaling:

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

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

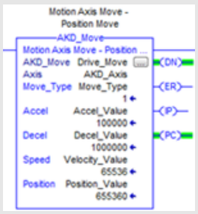
$$= \frac{2^{32}}{2^{16}} = 2^{16}\ or\ 65536\ EIP\ Counts/s\ per\ motor\ rev/s$$

NOTE

Assuming default EtherNet/IP scaling, 65536 counts/s in PLC = 1 rev/s of actual motor speed = 60 RPM.

- PLC value = motor speed in revs/s * 65536
- PLC value = motor speed in rpm * 65536 / 60

Incremental Move of 10 revs (655360 counts) at 60 RPM (65536 counts /sec)



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.

$$\frac{5\ inches}{1\ sec.} \cdot \frac{1\ rev.\ of\ ballscrew}{0.2\ inch} \cdot \frac{5\ motor\ revs.}{1\ rev.\ of\ ballscrew} \cdot \frac{65536\ counts}{1\ motor\ rev.} = \frac{8192000\ counts}{sec.}$$

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

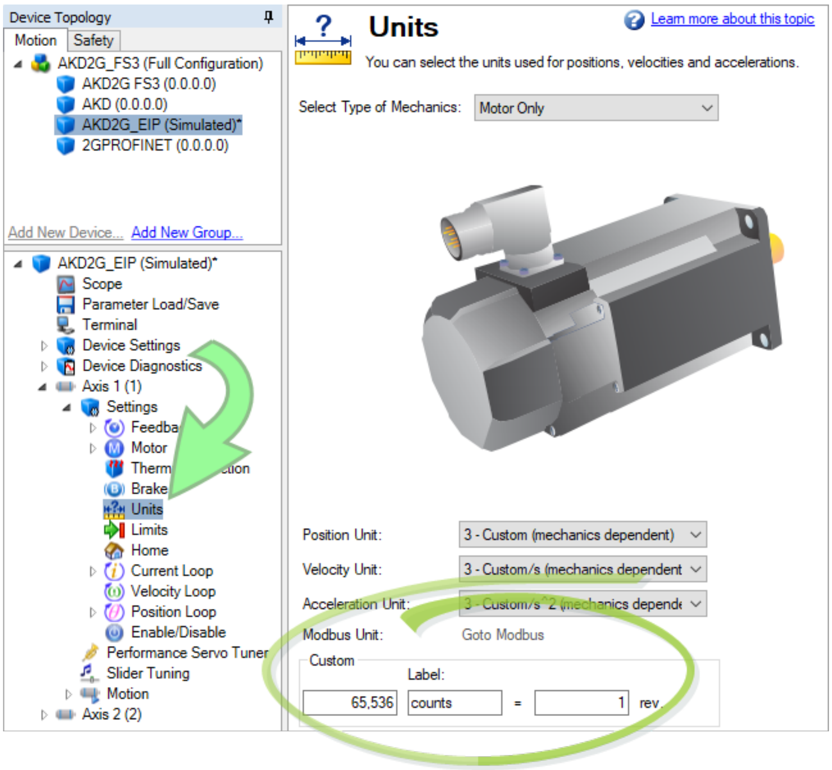
As mentioned previously, WorkBench Units can be set to match EtherNet/IP Units.

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.

4.4.4 WorkBench Units

The user units in WorkBench can be set to match the EtherNet/IP units. It is important to keep in mind that the WorkBench units are unrelated to EtherNet/IP units.

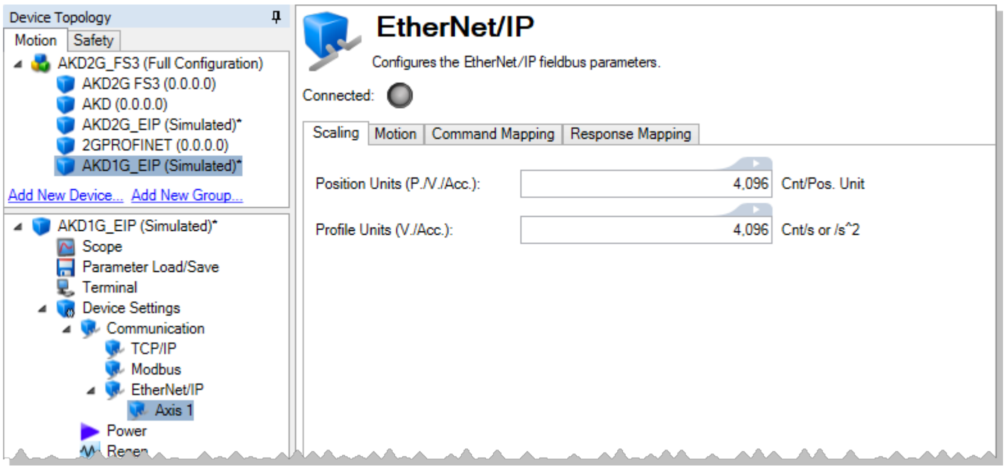


4.4.5 Higher Resolution Scaling - Velocity Units

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 EIP.POSUNIT and 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 in the Higher Resolution Scaling Example.

4.4.6 Changing the EtherNet/IP Scaling



4.4.7 Higher Resolution Scaling Example

EtherNet/IP Position Unit

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

$$EIP\ counts\ per\ motor\ rev = \frac{AKD2G\ internal\ feedback\ counts}{AXIS\#.EIP.POSUNIT}$$

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

EtherNet/IP Velocity and Acceleration Unit scaling

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

$$EIP\ counts\ per\ motor\ rev\ /\ s = \frac{AKD2G\ internal\ feedback\ counts}{AXIS\#.EIP.PROFUNIT}$$

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

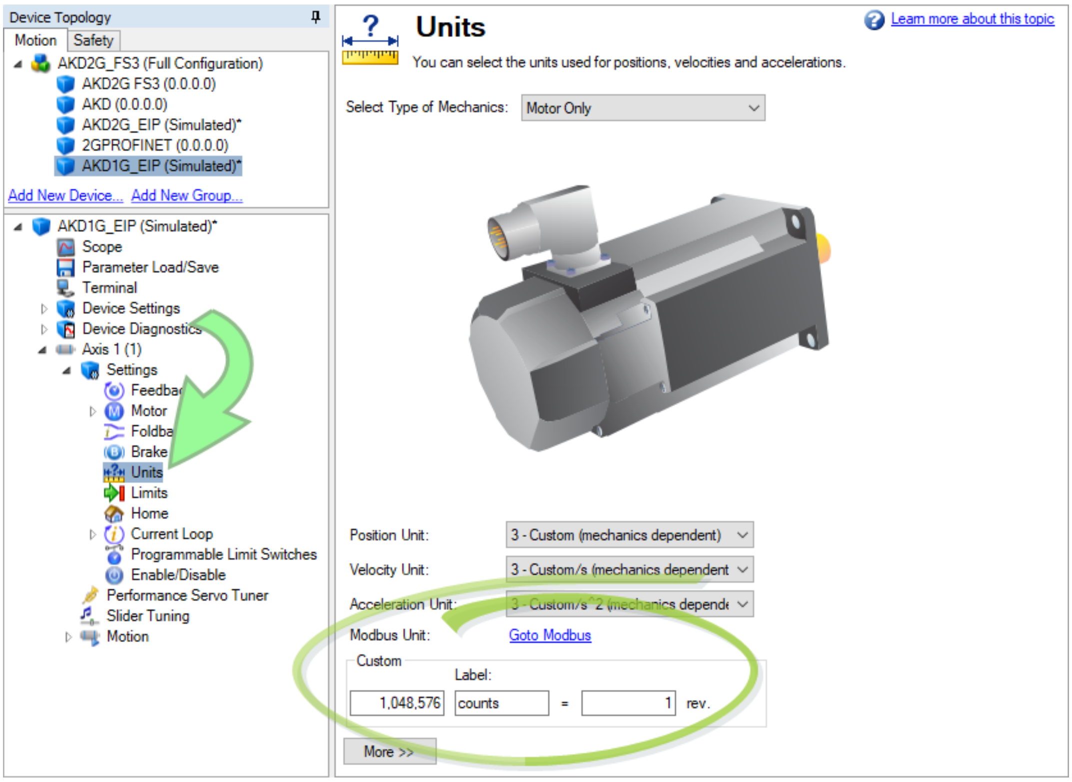
NOTE

Using the Higher Resolution Scaling example above:

- 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

4.4.8 Setting WorkBench Units to Match EtherNet/IP

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



4.5 Scaling for AKD2G

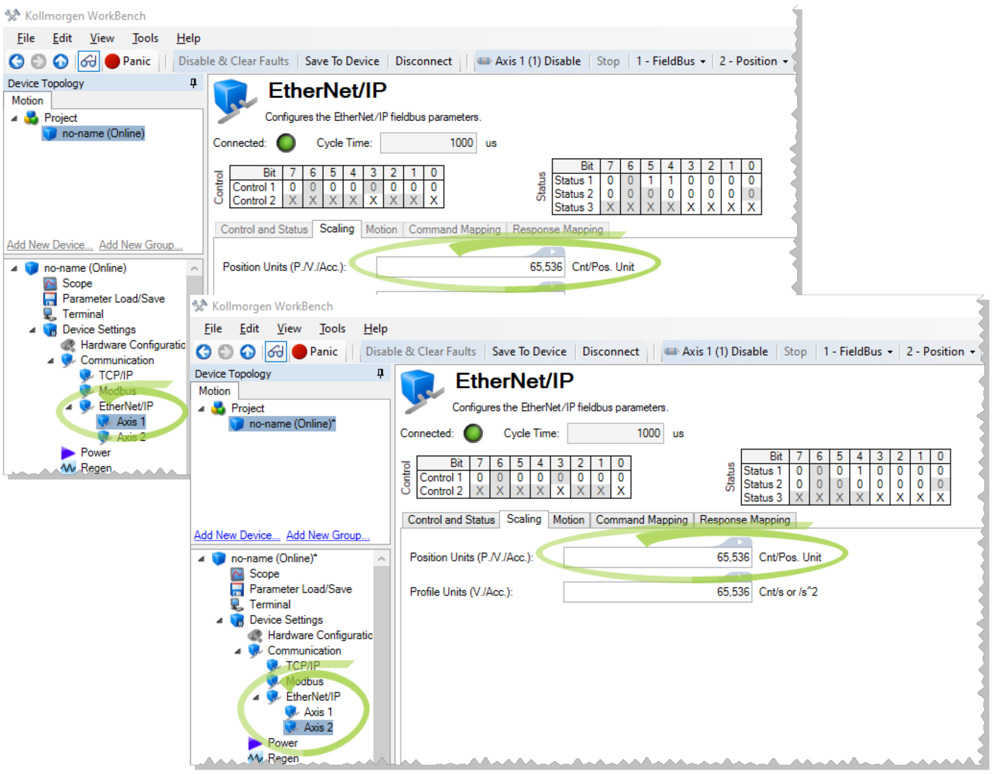
4.5.1 Scaling Best Practices

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.)

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.

4.5.2 EtherNet/IP Scaling in WorkBench

The figure below shows the default scaling for Axis 1 and Axis 2.



4.5.3 EtherNet/IP Position Unit Scaling

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 scaling is 65536 for AXIS1.EIP.POSUNIT is 65536, as shown above.

65536 is 2^{16}, so the actual counts per revolution of the motors is found using the fomula:

Definition

$$EIP\, Counts\, per\, motor\, rev\; = \frac{AKD2G\, internal\, feedback\, counts}{AXIS\#.\, EIP.\, POSUNIT} = \frac{2^{32}}{AXIS\#.\, EIP.\, POSUNIT}$$

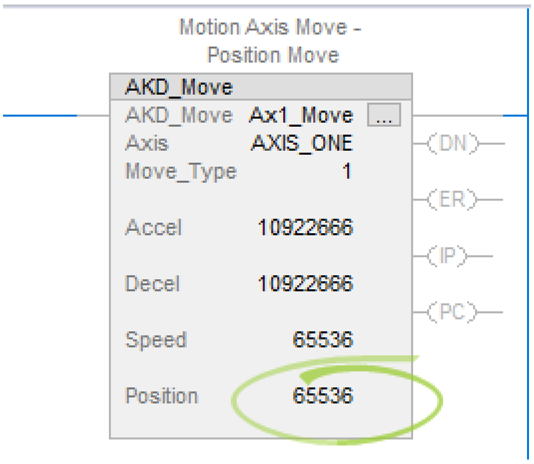
Default scaling

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

$$EIP\; Counts\; per\; motor\; rev\; = \frac{AKD2G\; internal\; feedback\; counts}{AXIS\#.\, EIP.\, POSUNIT}$$

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

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



The PLC/HMI must do the conversions from real-world units (i.e. inches, inches/sec, etc.) to motor revolutions and motor 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.

$$1\,inch \cdot \frac{1\,rev.\;of\;ballscrew}{0.2\,inch} \cdot \frac{5\,motor\,revs.}{1\,rev.\;of\;ballscrew} \cdot \frac{65536\,counts}{1\,motor\,rev.} = 1638400\,counts$$

EtherNet/IP Velocity and Acceleration Unit Scaling

Velocity and Acceleration values are scaled according to the EtherNet/IP Position Controller Device standard. When "Profile Units" scaling is defined both Velocity and Acceleration are affected.

- 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. The # in the parameter will be replaced with either a 1 or 2 indicating the axis being used.
- From EtherNet/IP, this value can be accessed at Attribute 0x05 Profile Units of the Position Controller Object.

Definition

$$EIP\ Counts\ per\ motor\ rev/s\ =\ \frac{AKD2G\ internal\ feedback\ counts/s}{AXIS\#.EIP.PROFUNIT}$$

$$=\frac{2^{32}}{AXIS\#.EIP.PROFUNIT}$$

Default scaling:

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

$$EIP\ Counts/s\ per\ motor\ rev/s\ =\ \frac{AKD2G\ internal\ feedback\ counts/s}{AXIS\#.EIP.PROFUNIT}$$

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

NOTE

Using the Higher Resolution Scaling example above:
- 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.

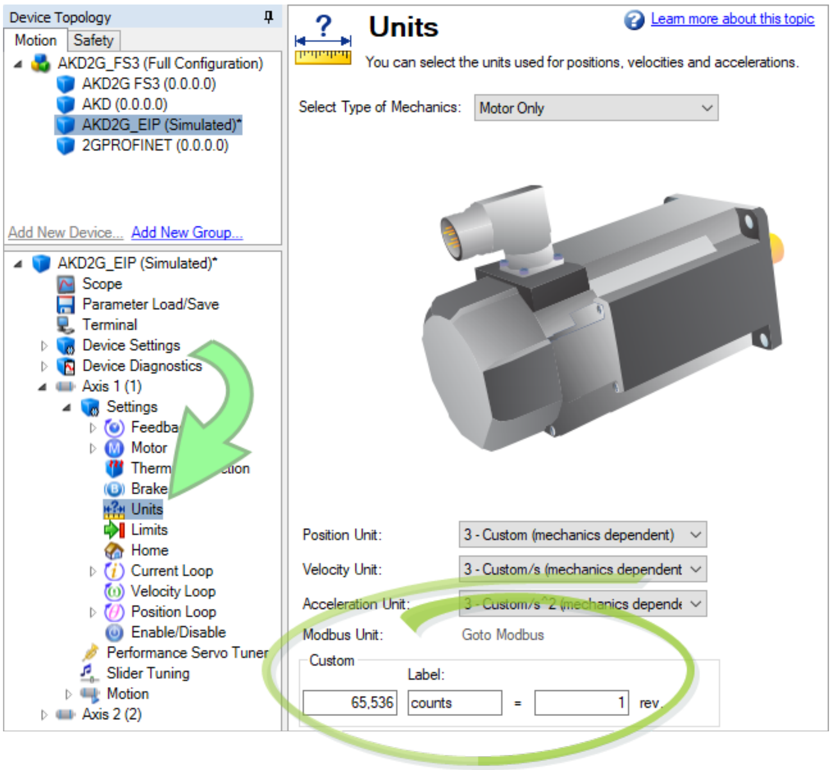
$$\frac{5\ inches}{1\ sec.}\cdot\frac{1\ rev.\ of\ ballscrew}{0.2\ inch}\cdot\frac{5\ motor\ revs.}{1\ rev.\ of\ ballscrew}\cdot\frac{65536\ counts}{1\ motor\ rev.}=\frac{8192000\ counts}{sec.}$$

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/IP Units.

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.


Device Topology
Motion
Safety
AKD2G_FS3 (Full Configuration)
AKD2G FS3 (0.0.0.0)
AKD (0.0.0.0)
AKD2G_EIP (Simulated)*
2GPROFINET (0.0.0.0)
Add New Device...
Add New Group...
AKD2G_EIP (Simulated)*
Scope
Parameter Load/Save
Terminal
Device Settings
Device Diagnostics
Axis 1 (1)
Settings
Motor
Brake
Units
Limits
Home
Current Loop
Velocity Loop
Position Loop
Enable/Disable
Performance Servo Tuner
Slider Tuning
Motion
Axis 2 (2)
Units
Learn more about this topic
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:
Acceleration Unit:
Modbus Unit:
Goto Modbus
Custom
Label:
65,536
counts
=
1
rev.

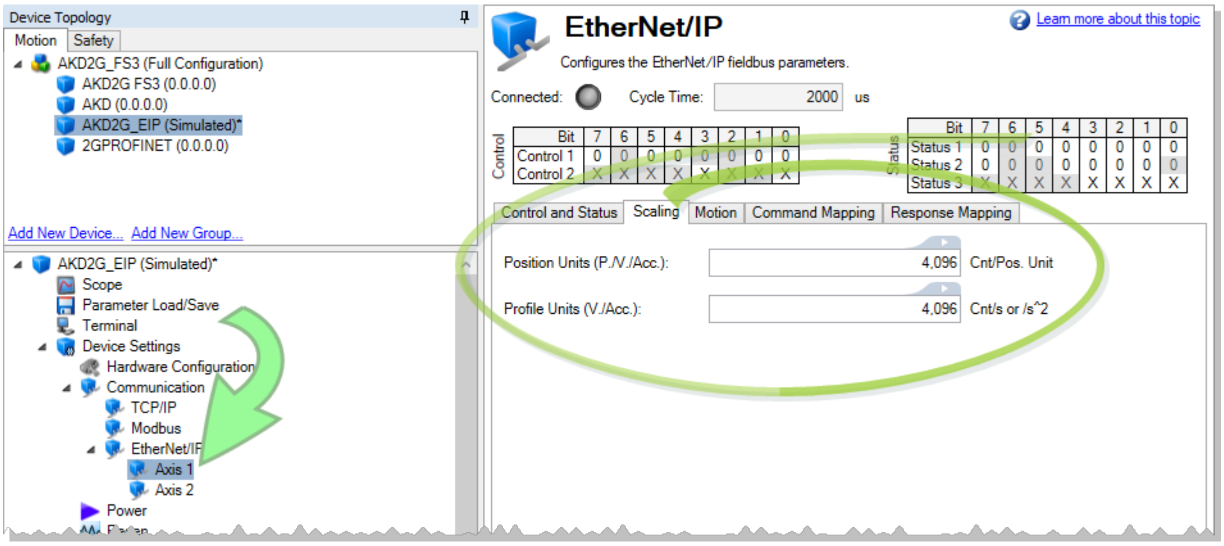
4.5.4 Higher Resolution Scaling

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 desire is to increase the resolution from 65536 (2^{16}) counts to 1,048,576 (2^{20}) counts per revolution. This is done by setting AXIS#.EIP.POSUNIT and AXIS#.EIP.PROFUNIT to 4096 (2^{12}) will increase the resolution.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:

Example: Axis 1



4.5.5 Higher Resolution Scaling Example

EtherNet/IP Position Unit

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

$$EIP \ counts \ per \ motor \ rev = \frac{AKD2G \ internal \ feedback \ counts}{AXIS\#.EIP.POSUNIT}$$

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

EtherNet/IPVelocity and Acceleration Unit scaling

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

$$EIP \ counts \ per \ motor \ rev \ / \ s = \frac{AKD2G \ internal \ feedback \ counts}{AXIS\#.EIP.PROFUNIT}$$

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

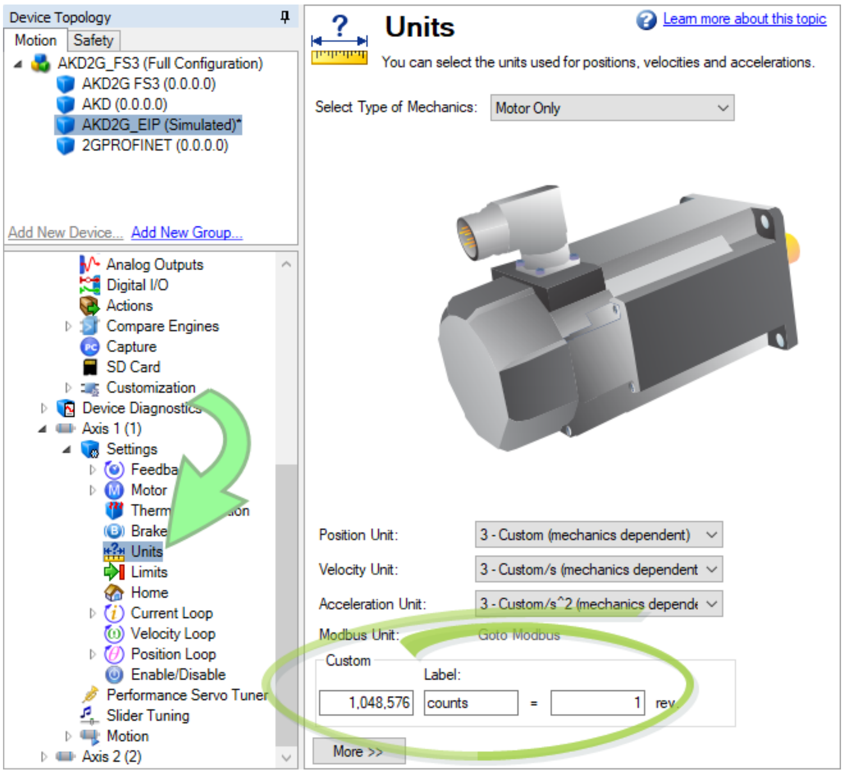
NOTE

Based on the scaling above:

- 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

Example: Axis 1



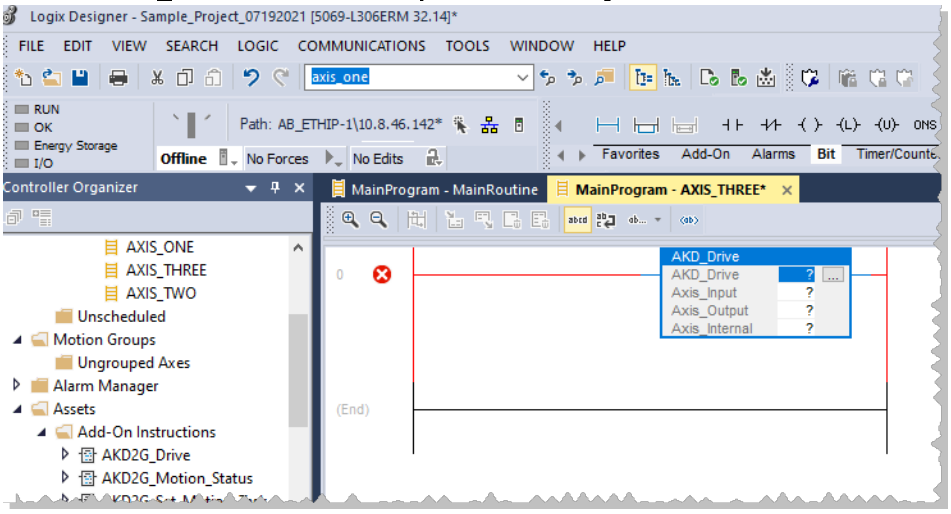
5 AKD1G - Using AOIs in a Project

Any projects requring Add-On Instructions with an AKD EtherNet/IP drive must include one instance of the AKD_Drive instruction for each AKD EtherNet/IP drive and its associated Generic Ethernet Module.

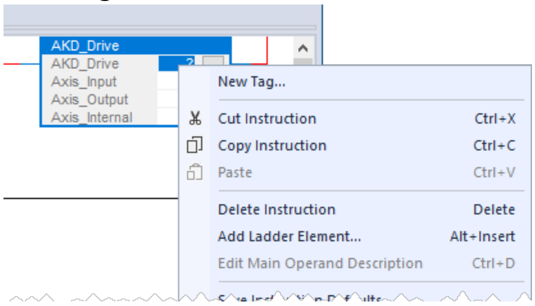
IMPORTANT

It is important that the AKD_Drive AOI is unconditionally tied to the left rail of your Ladder so it executes on every scan. Ensure the AKD_Drive instance is placed in your Ladder execution so that it is scanned prior to any AOIs being declared with the associated Axis_Internal name of the associated AKD_Drive instruction.

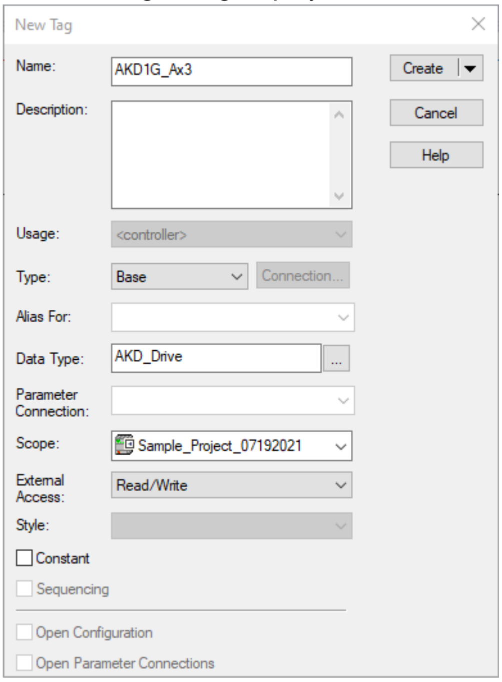
1. Add the **AKD_Drive instruction** to your Ladder diagram.



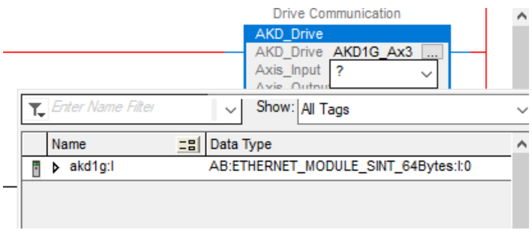
2. Right-click the **AKD_Drive parameter** (first question mark) in the AKD_Drive instruction and select **New Tag...**



The New Tag dialog displays.



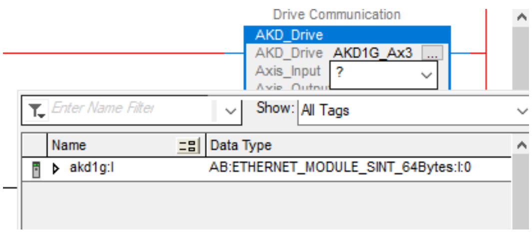
3. Enter a **Name** for the Tag.
4. Enter a **Description** (optional).
5. Ensure the **Data Type** is set to **AKD_Drive**. This will define the instance of the AKD_Drive AOI.
6. Click **Create** at the top of the dialog to create your Tag.
   The Tag will be displayed on both the block and in the controller under **Controller Tags**.



7. Click the down arrow to the right of the **Axis_Input** field of the AKD_Drive AOI.

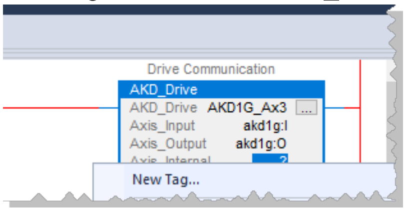
   Any valid Tags in your database with the Data Type AB:ETHERNET_MODULE_SINT_64Bytes:I:0 should be shown in the list.

   The only drive declared at this point in the process is the Generic Ethernet Module declared as AKD1G.

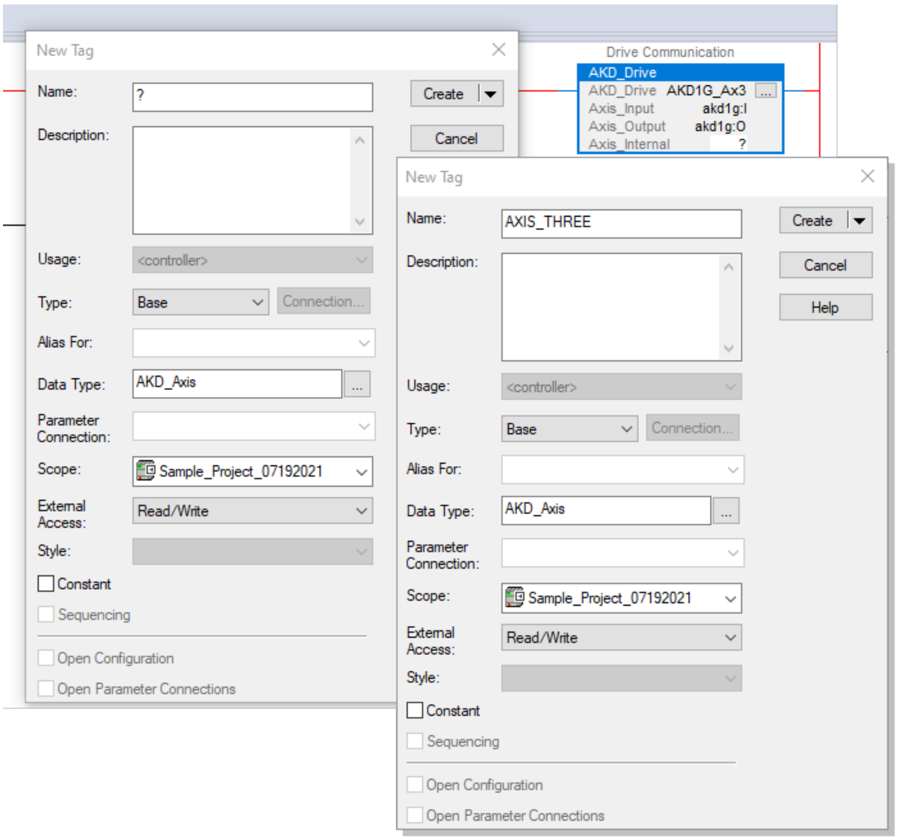


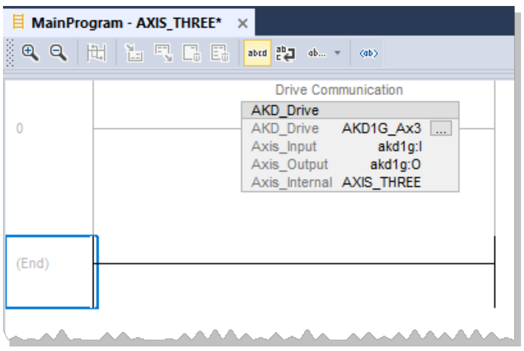
8. Repeat the steps above for the **Axis_Output** field and choose the appropriate Tag for the **AKD Generic Ethernet Module** (i.e. axis) the AKD_Drive AOI is being declared for.

9. Next, right-click on the **Axis_Internal** field and select **New Tag...**



10. Set the **Axis_Internal** Tag name to the name the axis will be identified by. For example: Conveyor, Axis 2, Z Axis, Filler, etc.
The AXIS fields of other AOIs will display this name and correlate the axis and its associated AKD_Drive instructions.





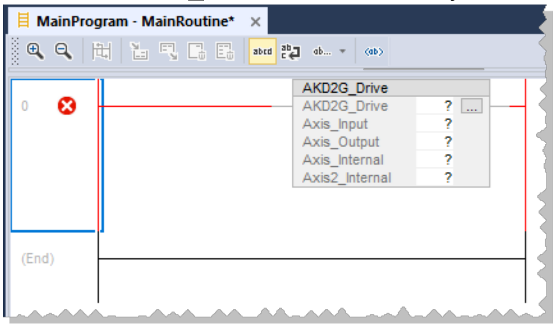
6 AKD2G - Using AOIs in a Project

Any projects requring Add-On Instructions with the AKD2G EtherNet/IP drive must include one instance of the AKD_Drive instruction for each AKD2G EtherNet/IP drive and its associated Generic Ethernet Module.

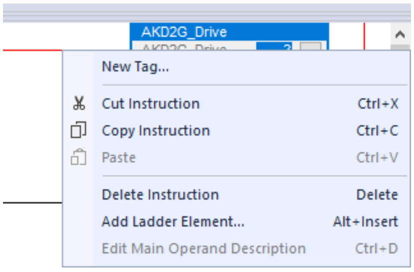
ⓘIMPORTANT

It is important that the AKD2G_Drive AOI is unconditionally tied to the left rail of your Ladder so it executes on every scan. Ensure the AKD2G_Drive instance is placed in your Ladder execution so that it is scanned prior to any AOIs being declared with the associated Axis_Internal or Axis2_Internal Tag name of the associated AKD2G_Drive instruction.

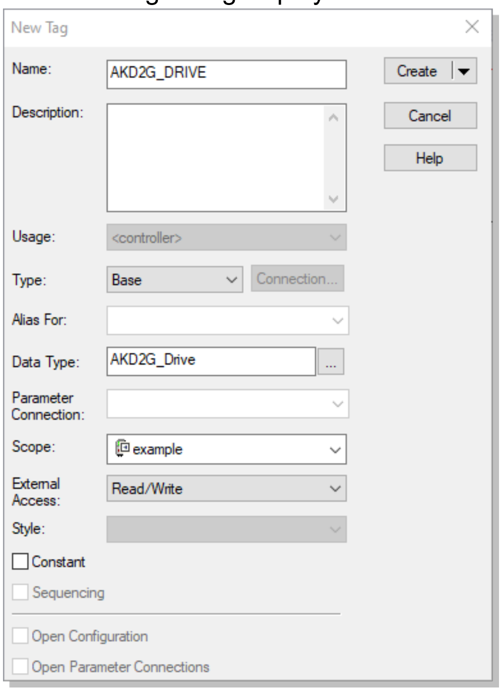
1. Add the **AKD2G_Drive instruction** to your Ladder diagram.



2. Right-click the **AKD2G_Drive parameter** (first question mark) in the AKD2G_Drive instruction and select **New Tag...**

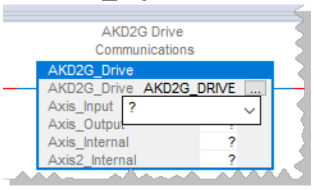


The New Tag dialog displays.



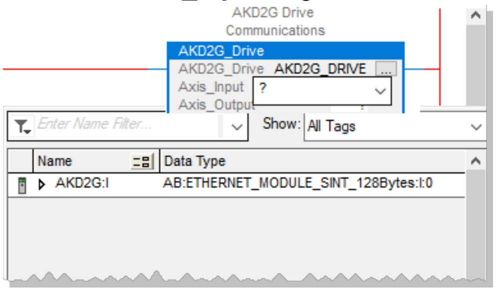
3. Enter a **Name**. This will define the instance of the AKD2G_Drive AOI.

4. Enter a **Description** (optional).
5. Ensure the **Data Type** is set to **AKD2G_Drive**.
6. Click **Create** at the top of the dialog to create your Tag.
7. Double-click the **Axis_Input** field of the AKD2G_Drive AOI and select the **down arrow** to the right of the **Axis_Input** field.

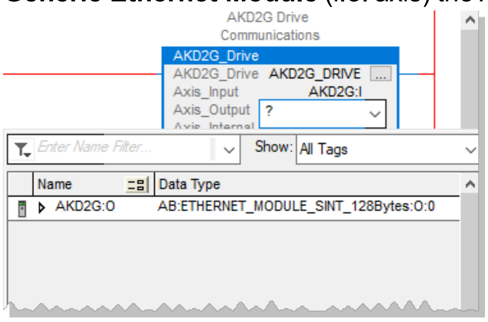


The Controller Tags appears with a list of any valid Tags in your database with the Data Type AB:ETHERNET_MODULE_SINT_128Bytes:I:0 should be shown in the list. The only drive declared at this point in the process is the Generic Ethernet Module declared as **AKD2G**.

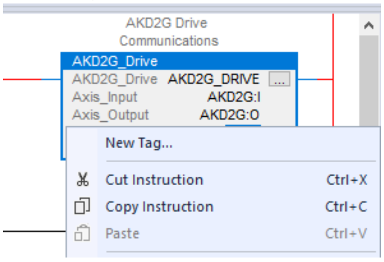
8. Select the **Axis_Input** Tag from the list. In this example the Tag is AKD2G:I.



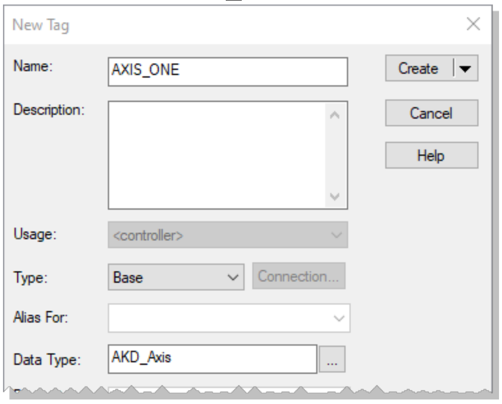
9. Repeat the steps above for the **Axis_Output** field and choose the appropriate Tag for the **AKD Generic Ethernet Module** (i.e. axis) the AKD2G_Drive AOI is being declared for.



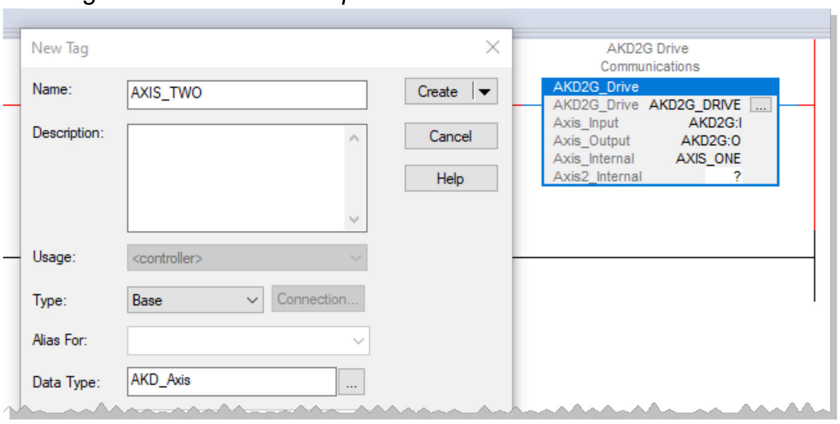
10. Next, right-click on the **Axis_Internal** field and select **New Tag...**.

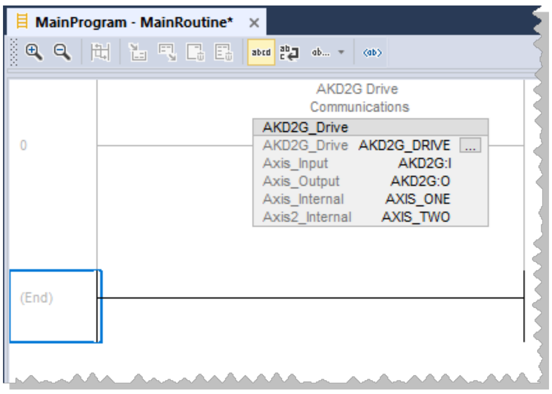


11. Set the **Axis_Internal** Tag name to the name Axis 1 in the AKD2G drive will be identified by. For example: Conveyor, Axis 1, Z Axis, Filler, etc.
    This name will be entered into other AOI's AXIS fields to correlate those AOIs to the given axis and associated AKD2G_Drive instruction.

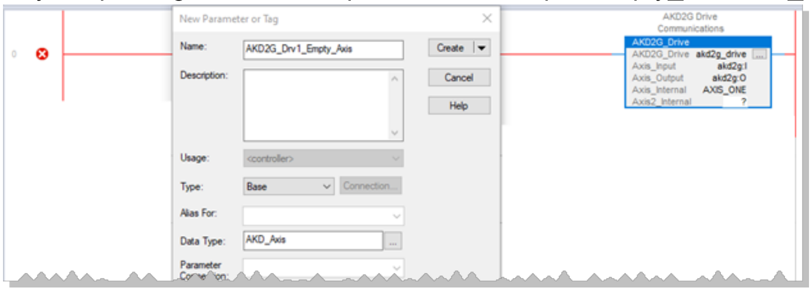


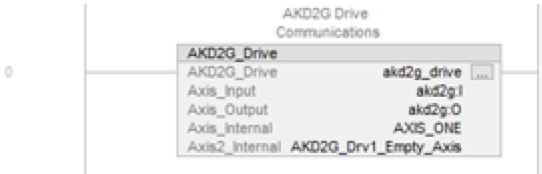
12. For a dual-axes AKD2G EtherNet/IP drive, set the **Axis2_Internal** Tag name to the name the axis will be identified by. For example: Conveyor, Axis 2, Z Axis, Filler, etc.
    This name will be entered into the Axis field of any correlated AOIs in order to correlate those AOIs to the given axis and associated AKD2G_Drive instruction.
    For single-axis drives see step 13.





13. For a single-axis AKD2G EtherNet/IP drive the Axis2_Internal Tag entry is required by the AKD2G_Drive Add-On Instruction, but the entry will not be used in the PLC program (Do not reference this as an "AXIS" entry for any AOIs in the Ladder). Any attempts to use the Axis2_Internal Tag name in AOIs and Ladder logic will result in the given AOI's .ER bit being set on attempt to execute.
    Any unique Tag name is acceptable. For example, Empty_Axis, Not_Used, etc.





The structure is still created in the Tag database but all data exchanged will be zero.

◢ AKD2G_Drv1_Empty_Axis	{...}	{...}
▷ AKD2G_Drv1_Empty_Axis.Control	{...}	{...}
▷ AKD2G_Drv1_Empty_Axis.Status	{...}	{...}
▷ AKD2G_Drv1_Empty_Axis.Input	{...}	{...}
▷ AKD2G_Drv1_Empty_Axis.Output	{...}	{...}
▷ AKD2G_Drv1_Empty_Axis.ResponseMsgType	0	
▷ AKD2G_Drv1_Empty_Axis.CommandTimeout	0	
▷ AKD2G_Drv1_Empty_Axis.PositionFeedback	0	
▷ AKD2G_Drv1_Empty_Axis.VelocityFeedback	0	
AKD2G_Drv1_Empty_Axis.IsAKD2G	0	

7 AOI Library, Definitions, & Functionality

The section below provides an outline of each AOI's content in this manual. A list of Add-On Instructions can be found in the next section.

7.1 Format Of Each Entry

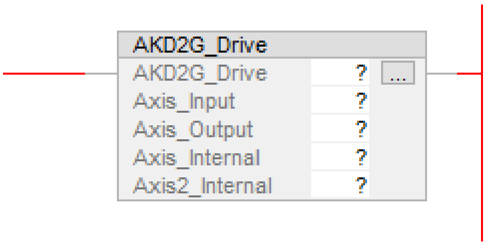
- **AOI name**
- **Unconfigured AOI block** Default image of what the block looks like and any entries and/or outputs
- **Description**
  - Compatibility (AKD1G, AKD2G, or both)
  - Expected Command Source and Operation Mode for successful operation
  - How the AOI works, its expected behavior, what the AOI is intended for, etc.
- **Operands** This includes:
  - **Instance Tag name:** Required for every AOI added to the Ladder logic.
  - **Axis Name:** Must be set to the desired axis' unique Axis_Internal or Axis2_Internal Tag name entered in the **AKD_Drive AOI** or **AKD2G_Drive AOI** for the given axis, which points to the Generic Ethernet Module communications for a given IP address.
  - Any additional field entries required for the AOI's configuration to be complete and valid.
- **Structure** This indicates information about the AOI specific to its inputs and outputs.
- **Execution** Description of the internal logic in each Ladder of the AOI which includes, if applicable:
  - Prescan
  - .EnableIn False
  - Main Logic Executing
- Changes to Axis Status bits
- Example of Usage/Programming Guidelines
- Troubleshooting: Summary of Reasons for Errors
- Step Summary
- Revision History

7.2 Add-On Instructions List

The list below includes all 27 AOIs, the version of the AOI included in AOI Library version 6.0, and its drive compatibility.

AOI	Version	AKD	AKD2G
AKD2G_Drive	1.0		✓
AKD2G_Motion_Status	1.1		✓
AKD2G_Set_Motion_Task	1.1		✓
AKD_Command_Assembly	3.1	✓	✓
AKD_Command_Control_Word	2.0	✓	✓
AKD_Disable	3.0	✓	✓
AKD_Drive	3.0	✓	
AKD_Enable	3.0	✓	✓
AKD_Fault_Reset	5.0	✓	✓
AKD_Get_Attribute	3.1	✓	✓
AKD_Get_Parameter	3.0	✓	✓
AKD_Home	5.0	✓	✓
AKD_Jog	3.0	✓	✓
AKD_Motion_Status	3.1	✓	
AKD_Move	4.1	✓	✓
AKD_Response_Assembly	3.1	✓	✓
AKD_Response_Status_Words	3.0	✓	✓
AKD_Set_Attribute	3.0	✓	✓
AKD_Set_Home_Mode	3.0	✓	✓
AKD_Set_Mode	5.0	✓	✓
AKD_Set_Motion_Task	3.0	✓	
AKD_Set_Parameter	3.0	✓	✓
AKD_Shutdown	5.0	✓	✓
AKD_Shutdown_Reset	3.0	✓	✓
AKD_Start_MotionTask	3.0	✓	✓
AKD_Stop_Smooth	5.0	✓	✓
AKD_Torque_Move	3.0	✓	✓

7.3 AKD2G_Drive



Description

Use the drive communication (AKD2G_Drive) instruction to initiate communication for an axis.

Compatibility

The AKD2G_Drive AOI is only compatible with AKD2G drives. Use the AKD_Drive AOI with AKD1G drives.

Required Command Source and Operation Mode

AKD2G

AXIS#.CMDSOURCE = ANY/ALL

AXIS#.OPMODE = ANY/ALL

This instruction is tied to the left rail of the Ladder so that it executes on every scan and should be placed in the Main program, or at the top of the axis' subroutine, prior to scanning the other codependent AOIs for that axis.

The Axis_Input and Axis_Output entries bind the data exchange between the declared Generic Ethernet Module of the AKD2G drive for both the Command Assembly and Response Assembly to the axis names in the AKD2G_Drive instruction (Axis_Internal and Axis2_Internal).

Structures in the Controller Tags are created when the Axis_Internal and Axis2_Internal Tags' names are declared. These internal names are then used with all other AOIs which require an Axis field entry so any AOIs used with a given axis are correlated to the AKD2G_Drive AOI.

Operands

These entries are required by the user.

Operand	Data Type	Format	Description
AKD2G_Drive	AKD2G_Drive	Tag	Tag name for this instance of the AOI.
Axis_Input	AB:ETHERNET_MODULE_SINT_128Bytes:I:0	Tag	Must point to the Generic Ethernet Module for the given AKD2G drive. The syntax will follow NAME:I where NAME is the name of the Generic Ethernet Module defined when it is first declared. I indicates input.
Axis_Output	AB:ETHERNET_MODULE_SINT_128Bytes:O:0	Tag	Must point to the Generic Ethernet Module for the given AKD2G drive. The syntax will follow NAME:O where NAME is the name of the Generic Ethernet Module defined when it is first declared. O indicates output.
Axis_Internal	AKD_Axis	Tag	User Tag defining the name of Axis 1 of the AKD2G drive. This name is used in all other AOIs to be used with and correlated to this axis.
Axis2_Internal	AKD_Axis	Tag	User Tag defining the name of Axis 2 of the AKD2G drive. This name is used by all other AOIs that correlate to this axis. For single-axis drives, declare a placeholder name such as Empty_Axis or Not_Used, etc. for the second axis entry to indicate this axis does not exist.

Structure

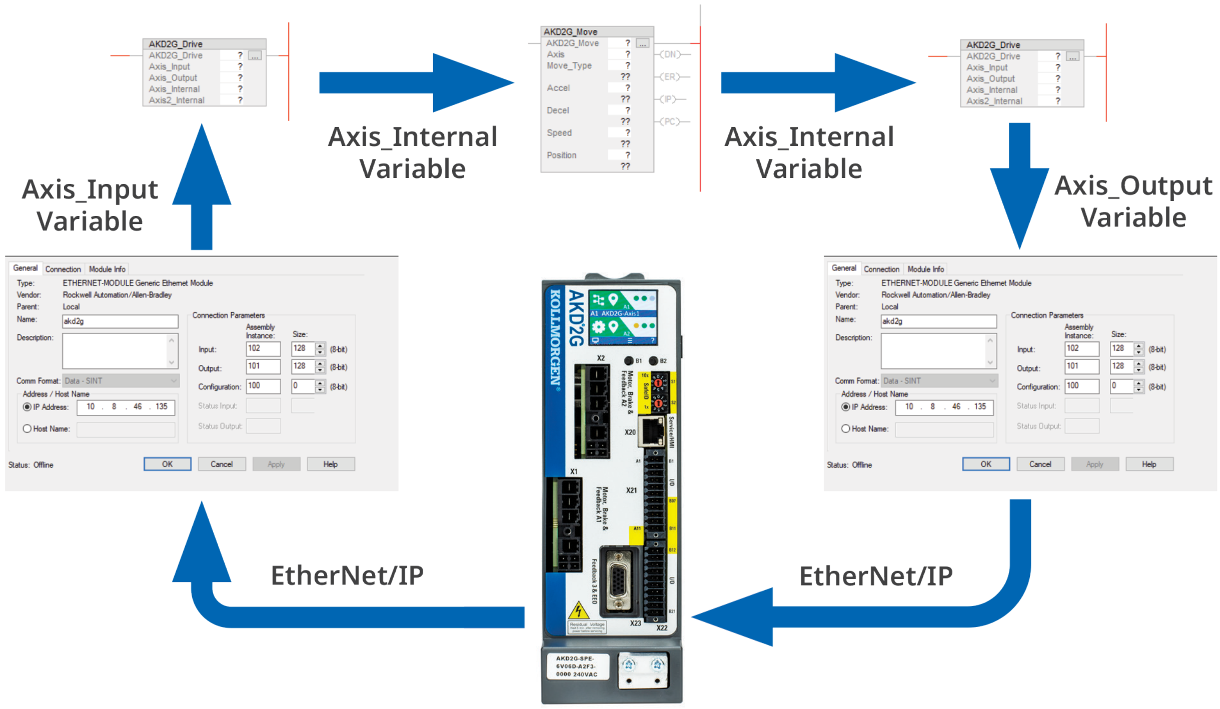
Mnemonic	Data Type	Description
.EnableIn	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.

Execution

Condition	Ladder Diagram Action
Prescan	• Unlatch the following bits of the Control Word: **Bit 0:** Load Command Data/Start Profile Move **Bit 1:** Start Block **Bit 2:** Incremental (Relative) (Select Absolute; 0 = Absolute) **Bit 3:** Direction in Velocity Mode (Select Negative; 0 = Negative) **Bit 4:** Smooth Stop **Bit 5:** Hard Stop **Bit 6:** Registration Input Armed (Bit 6 is Reserved. There is a Tag in the axis structure for this.) **Bit 7:** Enable • Clear the entire Command Assembly Data Structure.
Logic	While the rung is True (.EnableIn = 1; True), the AOI outputs are updated with axis data on every PLC scan. RPI scan time may also affect how often the data is updated.

The following example shows how an AOI is codependent on the AKD_Drive AOI.

In a normal EtherNet/IP application, the Move data is transferred from one variable to another six times before the answer arrives back at the MOVE block. The Move data is then transferred four times inside the PLC and two times outside the PLC.



All 128 Bytes of the Generic Ethernet Module Inputs are copied (0-63 for Axis 1 and 64-127 for Axis 2) into the Axis_Internal structure for each respective axis. (Axis1 = Axis_Internal.Input; Axis2 = Axis2_Internal.Input).

1. Copy the **input data** from the Generic Ethernet Module to the declared Axis_Internal name's **Input Bytes** (either Axis_Internal.Input or Axis2_Internal.Input) which are part of the structure when the AKD2G_Drive AOI is declared.
   This is the part of the structure when the AKD2G_Drive, Axis_Internal, or Axis2_Internal Tag is declared. This example shows an axis declared as **AXIS_ONE**.

AXIS_ONE	{...}	{...}	AKD_Axis
AXIS_ONE.Control	{...}	{...}	AKD_Control
AXIS_ONE.Status	{...}	{...}	AKD_Status
AXIS_ONE.Input	{...}	{...}	AKD_Data
AXIS_ONE.Output	{...}	{...}	AKD_Data
AXIS_ONE.ResponseMsgType	0	Decimal	SINT
AXIS_ONE.CommandTimeout	0	Decimal	INT
AXIS_ONE.PositionFeedback	-5586956	Decimal	DINT
AXIS_ONE.VelocityFeedback	-2405	Decimal	DINT
AXIS_ONE.IsAKD2G	1	Decimal	BOOL

2. Move from **Input** (Response Assembly data) to **Status** structure, which is part of the structure when the AKD2G_Drive AOI's Axis_Internal and Axis2_Internal Tags are declared.

Scope: Example_PLC Show: All Tags Enter Name Filter...

Name	Alias For	Base Tag	Data Type	Description	External Access
AXIS_ONE.Control			AKD_Control	Axis Data: Control bi...	Read/Write
AXIS_ONE.Status			AKD_Status	Axis Data: Status bits...	Read/Write
AXIS_ONE.Input			AB:ETHERNET_MODULE_SINT_64Bytes:I:0	Axis Data: Data from ...	Read/Write
AXIS_ONE.Output			AB:ETHERNET_MODULE_SINT_64Bytes:O:0	Axis Data: Data to th...	Read/Write
AXIS_ONE.ResponseMsgType			SINT	Axis Data: Response ...	Read/Write
AXIS_ONE.CommandTimeout			INT	Axis Data: Time to all...	Read/Write
AXIS_ONE.PositionFeedback			DINT	Axis Data: Actual Pos...	Read/Write
AXIS_ONE.VelocityFeedback			DINT	Axis Data: Actual Vel...	Read/Write
AXIS_ONE.IsAKD2G			BOOL	Axis Data: Whether t...	Read/Write
AXIS_TWO			AKD_Axis	Axis Data:	Read/Write
AXIS_TWO.Control			AKD_Control	Axis Data: Control bi...	Read/Write
AXIS_TWO.Status			AKD_Status	Axis Data: Status bits...	Read/Write
AXIS_TWO.Input			AB:ETHERNET_MODULE_SINT_64Bytes:I:0	Axis Data: Data from ...	Read/Write
AXIS_TWO.Output			AB:ETHERNET_MODULE_SINT_64Bytes:O:0	Axis Data: Data to th...	Read/Write
AXIS_TWO.ResponseMsgType			SINT	Axis Data: Response ...	Read/Write
AXIS_TWO.CommandTimeout			INT	Axis Data: Time to all...	Read/Write
AXIS_TWO.PositionFeedback			DINT	Axis Data: Actual Pos...	Read/Write
AXIS_TWO.VelocityFeedback			DINT	Axis Data: Actual Vel...	Read/Write
AXIS_TWO.IsAKD2G			BOOL	Axis Data: Whether t...	Read/Write

Axis 1 for example:

AXIS_ONE.Status	{...}	{...}	AKD_Status
AXIS_ONE.Status.Profile_In_Progress	0	Decimal	BOOL
AXIS_ONE.Status.Block_In_Execution	0	Decimal	BOOL
AXIS_ONE.Status.On_Target_Position	0	Decimal	BOOL
AXIS_ONE.Status.General_Fault	1	Decimal	BOOL
AXIS_ONE.Status.Current_Direction	1	Decimal	BOOL
AXIS_ONE.Status.Home_Level	0	Decimal	BOOL
AXIS_ONE.Status.Reg_Level	0	Decimal	BOOL
AXIS_ONE.Status.Enable	0	Decimal	BOOL
AXIS_ONE.Status.Fault_Input_Fault	0	Decimal	BOOL
AXIS_ONE.Status.Fwd_Limit	0	Decimal	BOOL
AXIS_ONE.Status.Rev_Limit	0	Decimal	BOOL
AXIS_ONE.Status.Positive_Limit	0	Decimal	BOOL
AXIS_ONE.Status.Negative_Limit	0	Decimal	BOOL
AXIS_ONE.Status.FE_Fault	0	Decimal	BOOL
AXIS_ONE.Status.Block_Fault	0	Decimal	BOOL
AXIS_ONE.Status.Load_Complete	0	Decimal	BOOL

3. Copy Byte 3 **Axis_Internal.Input.Data** and **Axis2_Internal.Input.Data** to the **Axis.ResponseMsgType** Controller Tag.

Scope: Sample_Project_ Show: All Tags Enter Name Filter

Name	Value	Force Mask	Style	Data Type
AXIS_ONE	{...}	{...}		AKD_Axis
AXIS_ONE.Control	{...}	{...}		AKD_Control
AXIS_ONE.Status	{...}	{...}		AKD_Status
AXIS_ONE.Input	{...}	{...}		AKD_Data
AXIS_ONE.Output	{...}	{...}		AKD_Data
AXIS_ONE.ResponseMsgType	0		Decimal	SINT
AXIS_ONE.CommandTimeout	0		Decimal	INT
AXIS_ONE.PositionFeedback	-5586956		Decimal	DINT
AXIS_ONE.VelocityFeedback	11936		Decimal	DINT
AXIS_ONE.IsAKD2G	1		Decimal	BOOL
AXIS_TWO	{...}	{...}		AKD_Axis
AXIS_TWO.Control	{...}	{...}		AKD_Control
AXIS_TWO.Status	{...}	{...}		AKD_Status
AXIS_TWO.Input	{...}	{...}		AKD_Data
AXIS_TWO.Output	{...}	{...}		AKD_Data
AXIS_TWO.ResponseMsgType	0		Decimal	SINT
AXIS_TWO.CommandTimeout	0		Decimal	INT
AXIS_TWO.PositionFeedback	340645		Decimal	DINT
AXIS_TWO.VelocityFeedback	-1064		Decimal	DINT
AXIS_TWO.IsAKD2G	1		Decimal	BOOL

4. Copy Position and Velocity feedback to the **AxisName.PositionFeedback** and **AxisName.VelocityFeedback** for each axis.

Scope: Sample_Project_ Show: All Tags Enter Name Filter...

Name	Value	Force Mask	Style	Data Type
AXIS_ONE	{...}	{...}		AKD_Axis
AXIS_ONE.Control	{...}	{...}		AKD_Control
AXIS_ONE.Status	{...}	{...}		AKD_Status
AXIS_ONE.Input	{...}	{...}		AKD_Data
AXIS_ONE.Output	{...}	{...}		AKD_Data
AXIS_ONE.ResponseMsgType	0		Decimal	SINT
AXIS_ONE.CommandTimeout	0		Decimal	INT
AXIS_ONE.PositionFeedback	-5586956		Decimal	DINT
AXIS_ONE.VelocityFeedback	11936		Decimal	DINT
AXIS_ONE.IsAKD2G	1		Decimal	BOOL
AXIS_TWO	{...}	{...}		AKD_Axis
AXIS_TWO.Control	{...}	{...}		AKD_Control
AXIS_TWO.Status	{...}	{...}		AKD_Status
AXIS_TWO.Input	{...}	{...}		AKD_Data
AXIS_TWO.Output	{...}	{...}		AKD_Data
AXIS_TWO.ResponseMsgType	0		Decimal	SINT
AXIS_TWO.CommandTimeout	0		Decimal	INT
AXIS_TWO.PositionFeedback	340645		Decimal	DINT
AXIS_TWO.VelocityFeedback	-1064		Decimal	DINT
AXIS_TWO.IsAKD2G	1		Decimal	BOOL

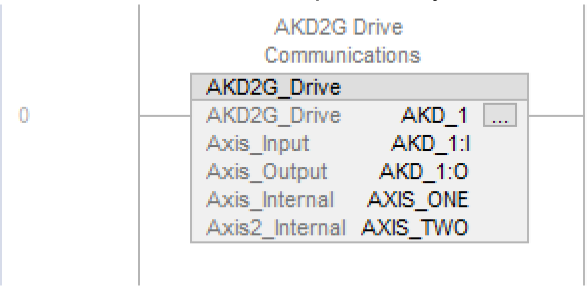
5. Latch the axis structure's Tags **axis_name.IsAKD2G** and set it to 1 to indicate the Axis is correlated to an AKD2G drive.
6. Copy the internal Control Word Byte (Bytes 0-7) for each axis to the **Axis_Output.Data** which is represented by Axis_Internal.Output.Data[0].x and Axis2_Internal.Output.Data[0].
7. Copy the **Axis_Internal.Output data** in the structure for each axis to the **Axis_Output.Data** (Generic Ethernet Module) for Axis 1 (Bytes 0-63) and Axis 2 (Bytes 64-127).

Changes to Axis Status Bits

All status bits are updated from the drive. See Status Word 1 and Status Word 2 for more information.

Example of Usage/Programming Guidelines

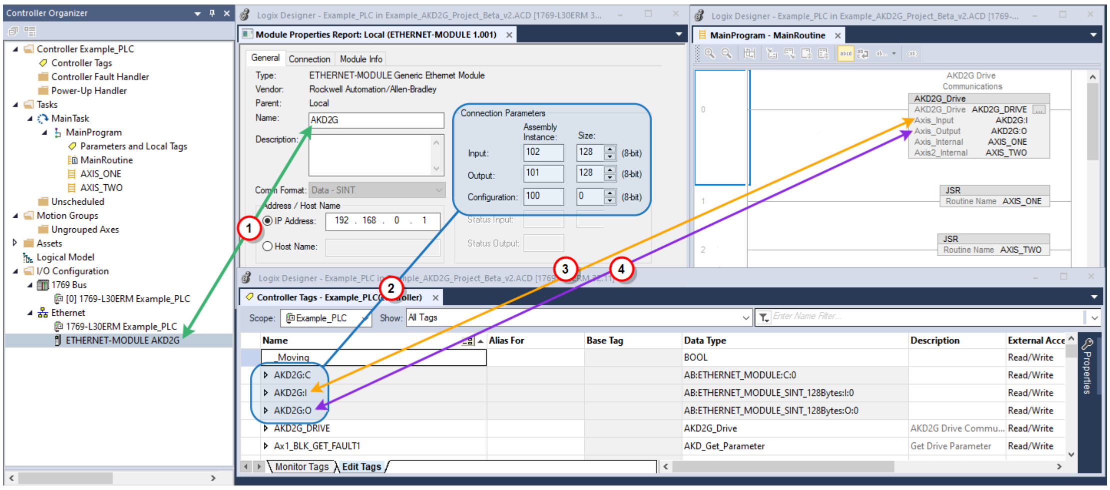
The screen capture below, taken from the Sample project, demonstrates usage. The AKD2G_Drive (AKD2G Drive Communications) AOI is always Enabled in the Ladder program and is executed on every scan. All other AOIs pointing to axis names declared in the AKD2G_Drive AOI Axis_Internal or Axis2_Internal entries are codependent on the AKD2G_Drive AOI. Therefore, the AKD2G_Drive AOI should be scanned first in the Ladder prior to any other AOIs for that axis.

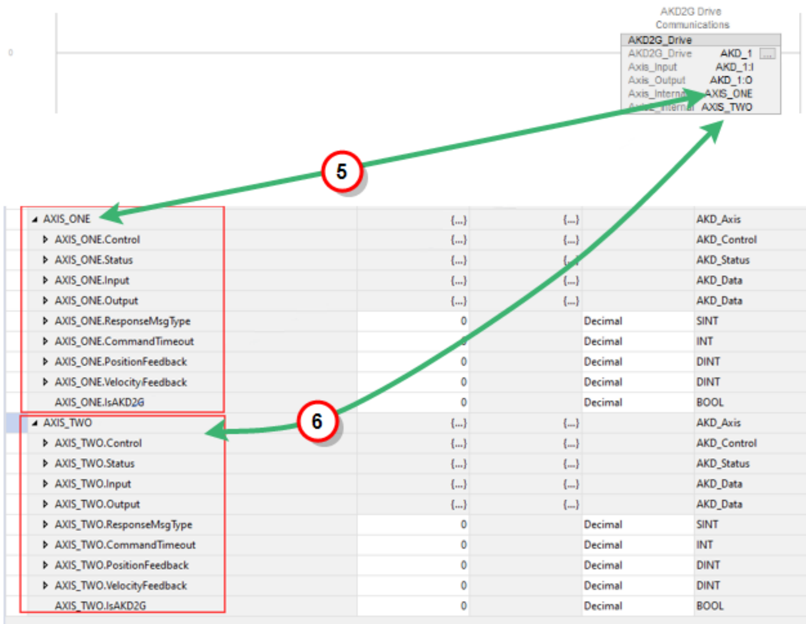


The screen capture below shows:

- the Generic Ethernet Module properties for the AKD2G axis
- that the AKD2G_Drive (Drive Communication) Add-On Instruction is in rung 0 of the routine
- the Controller Tags and structures declared when the Generic Ethernet Module was added and configured
- the Controller Tags and structures when the AKD2G_Drive AOI instance was declared.

The name of the Generic Ethernet Module is up to the programmer. Since this is a Sample project, the name AKD2G was used. This name will be displayed under Ethernet in the Controller Organizer. (See arrow 1 below.)





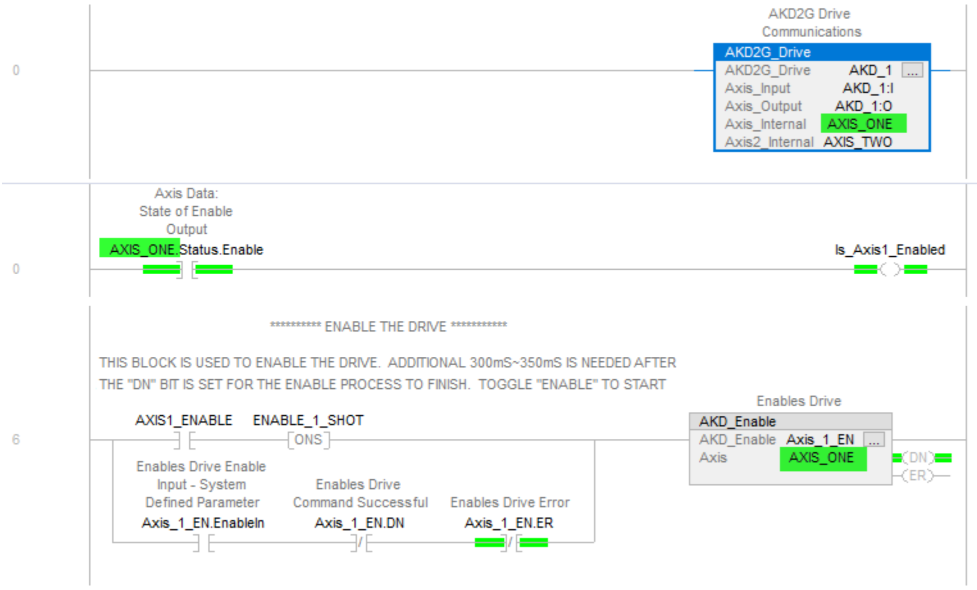
A few important points about the screen captures above:

- When the Generic Ethernet Module is added under Ethernet and configured, including declaring a Name and the Connection Parameters, the Tag structure in the Controller database is created. (See arrow 2.)
- In the AKD2G_Drive AOI, the Axis_Input and Axis_Output entries point to the Name.I and Name:0 Tags to correlate the AKD2G_Drive AOI with the Generic Ethernet Module to be associated with the Axis_Internal and Axis2_Internal Tag names. (See arrow 3 and arrow 4.)

In this way, the AKD2G_Drive correlates and acts as a liaison between the AOI instances that are declared later with the same Axis name as the AKD2G_Drive's Axis_Internal and Axis2_Internal Tag names and the communications of the Generic Ethernet Module (AKD or AKD2G drive with the respective IP Address). (See arrow 3 and arrow 4.)

Also shown are the axis structures in the Controller Tags once the Axis_Internal and Axis2_Internal Tags are declared in the AKD2G_Drive AOI. (See arrows 5 and 6.)

The Ladder shown below is an example of a given axis declared in the AKD2G_Drive AOI and the codependent AOIs for that axis:



Troubleshooting

- None

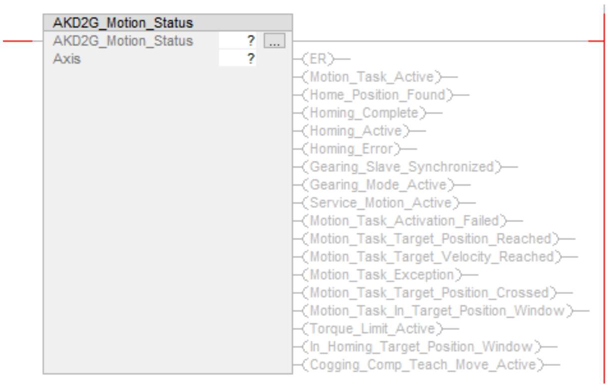
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	• Created to accommodate drive communications with the AKD2G drive. • AOI library v6.0 revision	07/16/2021

7.4 AKD2G_Motion_Status



Description

The AKD2G_Motion_Status AOI is intended for troubleshooting purposes and also to make the AXIS#.MOTIONSTAT data (bits) from the communications and controller Tags visible in the Ladder.

Compatibility

The AKD2G_Motion_Status AOI is only compatible with AKD2G drives. Use the AKD_Motion_Status AOI for AKD1G drives.

Command Source and Operation Mode Requirements

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

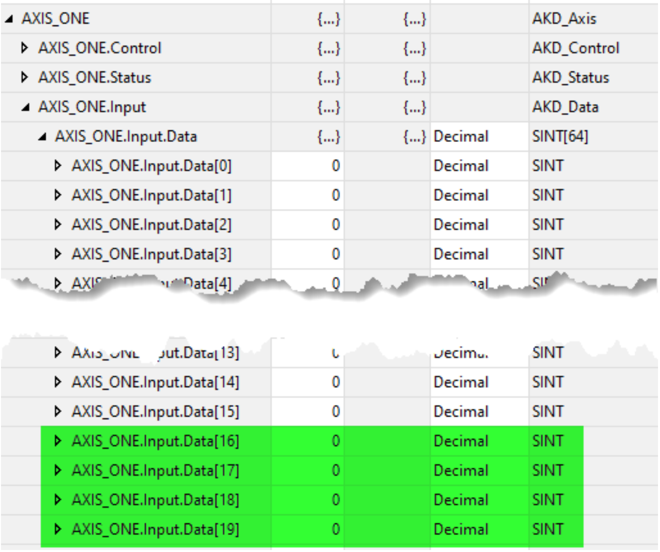
The AKD2G_Motion_Status AOI is intended for troubleshooting purposes and also to make the AXIS#.MOTIONSTAT data (bits) from the communications and Controller Tags visible in the Ladder. When the AKD2G_Motion_Status AOI's .EnableIn is True, the status bits for the given axis are copied from the Axis.Input data (Bytes 16-19) of the axis structure to the outputs of the AKD2G_Motion_Status AOI. Note the raw EtherNet/IP data will come from the input data (Bytes 16-19 for Axis 1; Bytes 80-83 for Axis 2). See the AKD2G Response Assembly Data Structure for more information on AXIS#.MOTIONSTAT.

ⓘIMPORTANT

It is important to check your drive's firmware version and AKD WorkBench Help for the description of AXIS#.MOTIONSTAT and the definition of each bit as they may be subject to change with firmware revision. The output names of the AOI is based on firmware 2-7-0-0. The implementation of the AOI requires the .EnableIn of the AOI to be tied unconditionally to the left rail in the Ladder logic so the values are updated with every scan. See Example of Usage/Programming Practices for more detail.

Using AXIS_ONE from the Sample project as an example (see screen capture below), the data for AKD2G_Motion_Status is sourced from here and is updated by the AKD2G_Drive Communication block.

In the example below, AXIS_ONE.Input.Data[0] through [19] are all set to a Value of 0, a Style of Decimal, and Data Type of SINT.



Note if the associated axis is not identified as an AKD2G drive then the following steps occur:

- The outputs of AOI are not updated and are cleared; their state would be invalid otherwise.
- The .ER bit of the AOI output is turned ON.

Operands

These entries are required by the user.

Operand	Data Type	Format	Description
AKD2G_ Motion_ Status	AKD2G_ Motion_ Status	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Name of axis. Points to Axis_Internal or Axis2_Internal Tag name declared in the AKD2G_Drive AOI field entries.

Structure

Mnemonic	Data Type	Format	Description	Read/Write
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.	Read Only
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.	Read Only
.ER	Output	BOOL	Turns ON if axis is not an AKD2G drive.	Read Only
Motion_Task_Active	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Home_Position_Found	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Homing_Complete	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Homing Active	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Homing_Error	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Gearing_Slave_ Synchronized	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Gearing_Mode_Active	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Service_Motion_Active	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Motion_Task_ Activation_Failed	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Motion_Task_Target_ Position_Reached	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Motion_Task_Target_ Velocity_Reached	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Motion_Task_ Exception	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Motion_Task_Target_ Position_Crossed	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Motion_Task_In_ Target_Position_ Window	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details	Read Only
Torque_Limit_Active	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
In_Homing_Target_ Position_Window	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only
Cogging_Comp_ Teach_Move_Active	Output	BOOL	See WorkBench Help and AXIS#.MOTIONSTAT for details.	Read Only

Execution

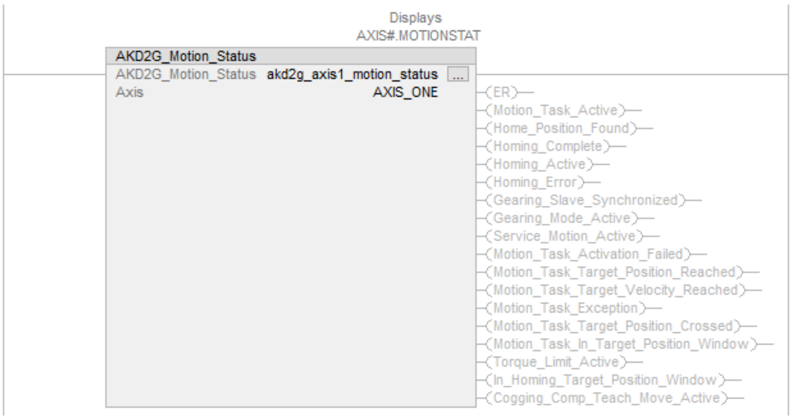
Condition	Ladder Diagram Action
Prescan	None
.EnableIn False	The output data of the AKD2G_Motion_Status AOI is not updated and retains their last value.
Instruction Execution	The output data of the AKD2G_Motion_Status AOI is updated with every scan.

Changes to Axis Status Bits

- None

Example of Usage/Programming Practices

In the following example, the AKD2G_Motion_Status AOI is added to a rung where the AOI's .EnableIn is always Enabled in the Ladder program and is executed on every scan. The AKD_Motion_Status entry is given a unique Tag name for that instance and the Axis name is entered based on the Axis_Internal or Axis2_Internal name entered in the AKD2G_Drive Communications Add-On Instruction for the desired axis.



Troubleshooting

Note if the associated axis is not identified as an AKD2G drive then the following steps occur:

- The outputs of the AOI are not updated and are cleared as their state would be invalid otherwise.
- The .ER bit of the AOI output is turned ON.

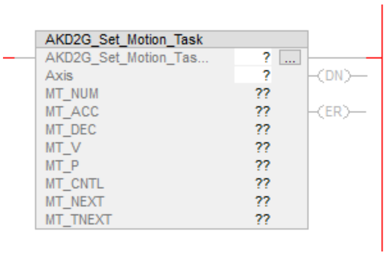
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	• Converted AKD_Motion_Status to AKD2G_Motion_Status to accommodate AXIS#.MOTIONSTAT. • Eliminated bit 7 and bit 8 status (Reserved in AKD2G). • Added bit 18 for Torque Limitation Active (i.e. current limit reached, I2t). Note bit 22 currently is not used in the AKD2G, but cogging compensation will be added in the future. • Added .ER output bit to show error if the user attempts to use the AKD2G_Motion_Status AOI with an AKD drive.	07/16/2021
v1.1	• Changed internal logic to eliminate unnecessary warnings when compiling the project. • AOI library v6.0 revision	09/20/2021

7.5 AKD2G_Set_Motion_Task



Description

The AKD2G_Set_Motion_Task AOI provides a method for editing or creating motion tasks in the Motion Task table from the PLC.

Compatibility

The AKD2G_Set_Motion_Task AOI is only compatible with AKD2G drives. Use the AKD_Set_Motion_Task AOI for AKD1G drives.

Command Source and Operation Mode Requirements

AKD2G

Required AXIS#.CMDSOURCE = Fieldbus

Required AXIS#.OPMODE = Any/All

NOTE

Position mode is not required to successfully execute the AKD2G_Set_Motion_Task AOI but the axis must be in Position mode prior to using the AKD2G_Start_Motion_Task.

The AKD2G_Set_Motion_Task AOI utilizes Command Type 0x09 – Motion Task internally, which is unique and exclusive to the AKD2G EtherNet/IP drive.

Tell me more about **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 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. Note, this is not a requirement to set the Motion Task but it is required prior to the Start Block bit being triggered (Step 6).
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. When using AOIs this is automated by using the AKD_Start_MotionTask AOI.

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.

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.

The AKD2G_Set_Motion_Task does not initiate the Motion Task; it merely changes or creates one. To start a defined Motion Task in the motion task table, utilize the AKD_Start_MotionTask AOI.

Note if the associated axis is not identified as an AKD2G drive then the following steps occur:

- .ER bit of the AOI output is turned ON.

Operands

These entries are required by the user. In the case of the MT entries, either a Constant (hardcoded) value or a variable (Tag) may be entered.

Operand	Data Type	Format	Description
AKD2G_Set_Motion_Task	AKD2G_Set_Motion_Task	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Name of axis. Points to Axis_Internal or Axis2_Internal Tag name declared in the AKD2G_Drive AOI field entries.
MT_NUM	SINT	Constant or Tag	Loads into Byte 1 (Block Number) of axis structure's output data (Command Assembly).
MT_ACC	DINT	Constant or Tag	Loads into Byte 16-19 (Acceleration) of axis structure's output data (Command Assembly) .
MT_DEC	DINT	Constant or Tag	Loads into Byte 20-23 (Deceleration) of axis structure's output data (Command Assembly).
MT_V	DINT	Constant or Tag	Loads into Byte 12-15 (Velocity) of axis structure's output data (Command Assembly).
MT_P	DINT	Constant or Tag	Loads into Byte 8-11 (Position) of axis structure's output data (Command Assembly).
MT_CNTL	DINT	Constant or Tag	This value is written to Instance 6302 (AXIS#.MT.CNTL) of the specified axis and the parameter index array is the MT_NUM entry of this AOI.
MT_NEXT	DINT	Constant or Tag	This value is written to ID 6306 (AXIS#.MT.MTNEXT) of the specified axis and the parameter index array is the MT_NUM entry of this AOI.
MT_MTNEXT	DINT	Constant or Tag	This value is written to ID 6308 (AXIS#.MT.TNEXT) of the specified axis and the parameter index array is the MT_NUM entry of this AOI.

Structure

Mnemonic	Data Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the AOI execution completes successfully.
.ER	Output	BOOL	Turns ON if axis is not an AKD2G drive or if execution fails. See Troubleshooting for more information.

Execution

Condition	Ladder Diagram Action
Prescan	• Unlatch the OS_Start_Sequence bit. • Move a 0 into the Step Number.
.EnableIn False	• Unlatch the OS_Start_Sequence bit. • Reset the Command_Timeout timer. • Clear the internal Tag Command_Bytes 8 bits (0-7)

Condition	Ladder Diagram Action
Logic	**Enable In:** • On .EnableIn unlatch the .DN and .ER bits, set the Command_Timeout.PRE to 5000 ms, and set the Step Number to 0. • If the Axis is not identified to be associated with an AKD2G drive then set the .ER bit of the AKD2G_Set_Motion_Task AOI and set the Step Number to -3. **Step 0:** • Command Type is Set to 16#09 (Command Type 0x09 – Motion Task) in Byte 1 (Block #) of the command structure's output data. This is copied to Byte 2 or Byte 66 depending on which axis is specified. • MT_NUM is loaded into the Block Number (Byte 1 of the command structure's output data). This is copied to Byte 1 for Axis 1 or Byte 65 for Axis 2 depending on which axis is specified. • MT_P, MT_V, MT_ACC, and MT_DEC are moved into the Command Assembly Data Structure in the correlated Bytes for Position, Velocity, Accel, and Deceleration. **Load/Start:** Load/Start (bit 0 in Control Word) is unlatched and the Step Number is set to 1. **Step 1:** • The Load/Start bit (bit 0 in Control Word) is latched, the Step Number is set to 2, and the Command_Timeout timer runs for as long as the Load Complete (bit 7 of Status Word 2) is OFF. • If the Command_Timeout timer reaches the preset of 5000 ms then the Load command failed and the Load/Start bit 0 in Control Word is unlatched, the .ER bit of the AOI is set, and the Step Number is set to -2. **Step 2:** On confirmation from Load Complete (bit 7 in Status Word 2) (ON), set the step number to 3. **Step 3:** Unlatch the .DN and .ER bits of the internal Mt_Cntl_Parameter (AKD_Set_Parameter) AOI and set the Step Number to 4. **Step 4:** • Move the MT_NUM value into the internal Mt_Cntl_ Parameter (AKD_Set_Parameter) AOI's Parameter_Array_ Index field and enable the Mt_Cntl_Parameter AOI. The MT_CNTL value is written to parameter 6302 (AXIS#.MT.CNTL). On Success (.DN) set the Step Number to 5. • On Fail (.ER), set the AKD2G_Set_Motion_Task AOI's .ER bit and set the Step Number to -4. **Step 5:** Unlatch the .DN and .ER bits of the internal Mt_Next_Parameter (AKD_Set_Parameter) AOI and set the Step Number to 6. **Step 6:** • Move the MT_NUM value into the internal Mt_Next_ Parameter (AKD_Set_Parameter) AOI's Parameter_Array_ Index field and enable the Mt_Next_Parameter AOI. The

Condition	Ladder Diagram Action
	MT_NEXT value is written to parameter number 6306 (AXIS#.MT.MTNEXT). • On Success (.DN) set the Step Number to 7. • On Fail (.ER), set the AKD2G_Set_Motion_Task AOI's .ER bit and set the Step Number to -5. **Step 7:** Unlatch the .DN and .ER bits of the internal Mt_TNext_Parameter (AKD_Set_Parameter) AOI and set the Step Number to 8. **Step 8:** • Move the MT_NUM value into the internal Mt_TNext_Parameter (AKD_Set_Parameter) AOI's Parameter_Array_Index field and enable the Mt_TNext_Parameter AOI. The MT_TNEXT value is written to parameter number 6308 (AXIS#.MT.TNEXT). • On success (DN) set the Step number to 9. • On Fail (.ER), set the AKD_Set_Motion_Task AOI's ER bit and set the Step Number to -6. **Step 9:** On completion the .DN bit is set on the output of the AKD2G_Set_Motion_Task AOI. If at any time during the execution of the AKD2G_Set_Motion_Task AOI an I/O Response Error occurs (Response Type 16#14; Response Type 0x14 - Command/Response Error), set the AKD2G_Set_Motion_Task AOI's .ER bit, copy the first 8 Bytes of input data from the Response Assembly, and set the Step Number to -1.

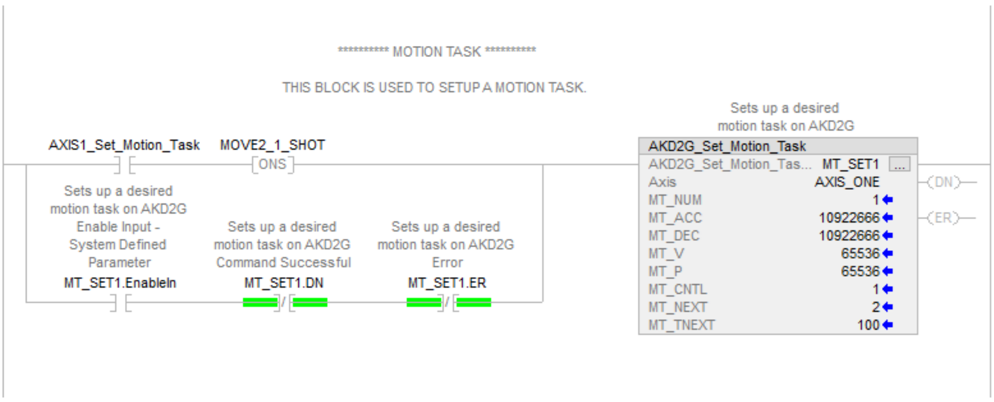
Changes to Axis Status Bits

- None

Example of Usage/Programming Practices

In the following example, the AKD2G_Set_Motion_Task entry is given a unique Tag name for that instance and the Axis name is entered based on the Axis_Internal or Axis2_Internal name entered in the AKD2G_Drive communications Add-On Instruction for the desired axis. The rest of the AOI's field entries are either Constant (hardcoded) values or Tags (variables). The logic for triggering and executing the AOI in the example below uses the **AXIS1_Set_Motion_Task** contact which is then One Shot to the AOI's .EnableIn. A seal-in parallel branch uses the .EnableIn, .DN, and .ER bits of the AOI to keep the AOI enabled until 1) .DN (Success of the AOI's execution) or 2) .ER (AOI execution failed).

Note the MT field entries of the AOI may be Constant (hardcoded) or variable (Tags).



Troubleshooting

The following conditions will result in the .ER bit being set on the AKD2G_Set_Motion_Task AOI:

1. Error from the Response Type.
2. Command Timeout
3. Axis entered for the AOI is not associated with an AKD2G_Drive communication block.
4. Internal AKD_Set_Parameter write to the AXIS#.MT.CNTL failed.
5. Internal AKD_Set_Parameter write to the AXIS#.MT.MTNEXT failed.
6. Internal AKD_Set_Parameter write to the AXIS#.MT.TNEXT failed.

Step Summary

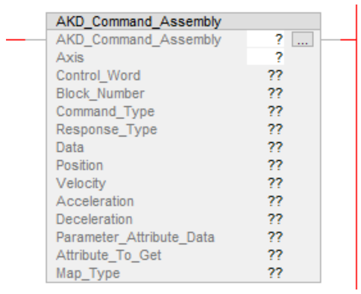
Step Number	Operation/Result
0	Enabled. Load/Start bit unlatched. Set Step Number to 1.
1	Once Load Complete bit is cleared, set the Load/Start bit and set Step Number to 2.
2	When Load Complete is confirmed set the Step Number to 3.
3	Unlatch .ER and .DN bits of the internal AKD_Set_Parameter AOI used to write to AXIS#.MT.CNTL. Set the Step Number to 4.
4	Execute internal AKD_Set_Parameter AOI to write to AXIS#.MT.CNTL. On .DN (Success) set Step Number to 5.
5	Unlatch .ER and .DN bits of the internal AKD_Set_Parameter AOI used to write to AXIS#.MT.MTNEXT. Set the Step Number to 6.
6	Execute internal AKD_Set_Parameter AOI to write to AXIS#.MT.MTNEXT. On .DN (Success) set the Step Number to 7.
7	Unlatch .ER and .DN bits of the internal AKD_Set_Parameter AOI used to write to AXIS#.MT.TNEXT. Set the Step Number to 8.
8	Execute the internal AKD_Set_Parameter AOI to write to AXIS#.MT.TNEXT . On .DN (Success) set the Step Number to 9.
9	Step 9 indicates success and completion of all steps in execution of the AOI. Set the Done (.DN) bit.

Step Number	Operation/Result
-1	Error from Response Type. Latch .ER bit.
-2	Command Timeout. Latch .ER bit.
-3	Axis entered for AOI is not associated with an AKD2G_Drive communication block. Latch .ER bit.
-4	Internal AKD_Set_Parameter write to the AXIS#.MT.CNTL failed (.ER). Latch .ER bit.
-5	Internal AKD_Set_Parameter write to AXIS#.MT.TNEXT failed (.ER). Latch .ER bit.
-6	Internal AKD_Set_Parameter write to AXIS#.MT.TNEXT failed. Latch .ER bit.

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	Initial implementation of the AKD2G_Set_Motion_Task AOI	07/16/2021
v1.1	• Changed definition to allow Tag assignment to the MT attributes of the AOI. • AOI library v6.0 revision.	09/30/2021

7.6 AKD_Command_Assembly



Description

The AKD_Command_Assembly AOI can be used in the Ladder program to allow the static mapping of the given axis' Command Assembly to be monitored in the Ladder during runtime. This can also be useful for troubleshooting.

Compatibility

The AKD_Command_Assembly AOI is compatible with both AKD1G and AKD2G drives.

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The AKD_Command_Assembly AOI can be used in the Ladder program to allow the static mapping of the given axis' Command Assembly to be monitored in the Ladder during runtime. This can also be useful for troubleshooting. Without the AOI, the user must access the Bytes directly in the Controller Tags for the given axis. The AKD_Command_Assembly AOI copies the values from each scan of the Axis.Output.Data.x Bytes from the structure of the specified axis and populates the output values of the AOI. The values are read-only and the Tag values cannot be changed.

Example of an axis named AXIS_ONE

◢ AXIS_ONE	{...}	{...}		AKD_Axis
▷ AXIS_ONE.Control	{...}	{...}		AKD_Control
▷ AXIS_ONE.Status	{...}	{...}		AKD_Status
▷ AXIS_ONE.Input	{...}	{...}		AKD_Data
◢ AXIS_ONE.Output	{...}	{...}		AKD_Data
▷ AXIS_ONE.Output.Data	{...}	{...}	Decimal	SINT[64]
▷ AXIS_ONE.ResponseMsgType	0		Decimal	SINT
▷ AXIS_ONE.CommandTimeout	0		Decimal	INT
▷ AXIS_ONE.PositionFeedback	-5586956		Decimal	DINT
▷ AXIS_ONE.VelocityFeedback	10597		Decimal	DINT
AXIS_ONE.IsAKD2G	1		Decimal	BOOL

Operands

These entries are required by the user.

Operand	Data Type	Format	Description
AKD_ Command_ Assembly	AKD_ Command_ Assembly	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI for AKD1G and either Axis_ Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for AKD2G for the given axis.

Structure

The following fields are not entered by the user and are populated automatically with Read Only data once the Operands above are entered.

Mnemonic	Data Type	Format	Description	Read/Write
Control Word	Output	SINT	Displays the value and monitors Byte 0 of Axis.Output.Data. Displayed in the AOI as a binary value.	Read Only
Block_ Number	Output	SINT	Displays the value and monitors Byte 1 of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only
Command_ Type	Output	SINT	Displays the value and monitors Byte 2 of Axis.Output.Data. Displayed in the AOI as a hex value.	Read Only
Response Type	Output	SINT	Displays the value and monitors Byte 3 of Axis.Output.Data. Displayed in the AOI as a hex value.	Read Only
Data	Output	DINT	Displays the value and monitors Bytes 4-7 of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only
Position	Output	DINT	Displays the value and monitors Bytes 8-11 of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only
Velocity	Output	DINT	Displays the value and monitors Bytes 12-15 of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only
Acceleration	Output	DINT	Displays the value and monitors Bytes 16-19 of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only
Deceleration	Output	DINT	Displays the value and monitors Byte 20-23 of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only
Parameter_ Attribute_Data	Output	DINT	Displays the value and monitors Bytes 24-27 of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only

Mnemonic	Data Type	Format	Description	Read/Write
Attribute_To_Get	Output	DINT	Displays the value and monitors Byte 32 (AKD1G) or Byte 28 (AKD2G) of Axis.Output.Data. Displayed in the AOI as a decimal value.	Read Only
Map_Type	Output	SINT	Displays the value and monitors Byte 33 of Axis.Output.Data for the AKD1G. Displayed in the AOI as a decimal value. AKD1G values can be 0 (Static), 1 (Custom), 2 (Dynamic). The Map Type will always report 0 for the AKD2G.	Read Only

Execution

Condition	Ladder Diagram Action
.EnableIn False	Not applicable. While rung is False, data in the AOI fields (outputs) are not updated and are frozen in their last state at the time when the rung went False.
Logic	While the rung is True (.EnableIn = 1; True) the data in the AOI fields (outputs) are updated with axis data every PLC scan. RPI scan time may affect how often the data is updated as well.

Changes to Axis Status Bits

- None

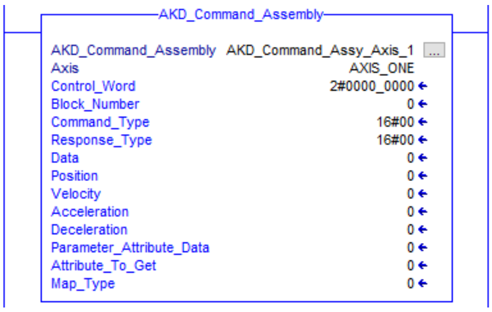
Example of Usage/Programming Guidelines

The AKD_Command_Assembly AOI is always Enabled in the Ladder program and is executed on every scan.

The AKD_Command_Assembly field must have a declared Tag for that instance and the Axis must be the same Tag for the given axis as the Axis_Internal field in the AKD_Drive or the Axis_Internal or Axis2_

Internal field in the AKD2G_Drive Add-On Instruction for that axis.

All other fields (outputs) in the AOI are Read-Only.



Troubleshooting

- None

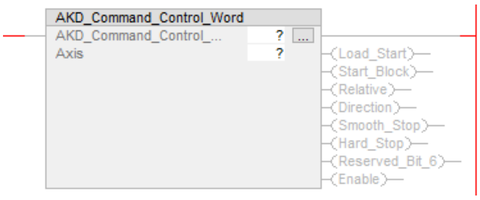
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	Initial release	08/13/2015
v2.0	• Output Data Bytes 12-15 changed to velocity Tag (was position; a duplicate). • Masked upper bits of command byte so the command type displayed matches the AKD EtherNet/IP Communication Manual. • AOI 5.0 library revision	01/18/2018
v3.0	• Added logic based on whether AKD_Drive or AKD2G_Drive AOI is being used for the given axis. The logic either reports Map Type as given by the AKD drive as 0,1, or 2 or in the case of the AKD2G it will always report 0. • Initial Beta Revision to accomodate AKD2G.	06/18/2021
v3.1	• Added code to handle the difference in Byte allocation for the Attribute To Get between the AKD1G and AKD2G. • The Attribute To Get is Byte 32 for the AKD1G and Byte 28 for the AKD2G. • Also updated to sync with the Axis and new data type AKD_ Data. • AOI library v6.0 revision.	07/27/2021

7.7 AKD_Command_Control_Word



Description

The AKD_Command_Control_Word AOI can be used in the Ladder program to allow the Control Word status of the given axis' Command Assembly to be monitored in the Ladder during runtime. This can be also useful for troubleshooting.

Compatibility

The AKD_Command_Control_Word AOI can be used with either AKD1G or AKD2G drives.

Command Source and Operation Mode Requirements

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The AKD_Command_Control_Word AOI can be used in the Ladder program to allow the Control Word status of the given axis' Command Assembly to be monitored in the Ladder during runtime. This can be also useful for troubleshooting. Without this AOI, the user must access the Byte directly in the Controller Tags for the given axis. The AKD_Command_Control_Word AOI copies the state of each bit in the Control Word of the Axis structure's Axis.Control Tag for the specified axis and displays the state as outputs in the AOI. The outputs are Read-Only.

Example of an axis named AXIS_ONE

AXIS_ONE	{...}	{...}		AKD_Axis
AXIS_ONE.Control	{...}	{...}		AKD_Control
AXIS_ONE.Control.Load_Data_Start_Pr...	0		Decimal	BOOL
AXIS_ONE.Control.Start_Block	0		Decimal	BOOL
AXIS_ONE.Control.Incremental	0		Decimal	BOOL
AXIS_ONE.Control.Direction	0		Decimal	BOOL
AXIS_ONE.Control.Smooth_Stop	0		Decimal	BOOL
AXIS_ONE.Control.Hard_Stop	0		Decimal	BOOL
AXIS_ONE.Control.Reg_Arm	0		Decimal	BOOL
AXIS_ONE.Control.Enable	1		Decimal	BOOL

Operands

These entries are required by the user.

Operand	Data Type	Format	Description
AKD_Command_Control_Word	AKD_Command_Control_Word	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

Mnemonic	Data Type	Format	Description	Read Write
Load_Start	Output	BOOL	Status of Load/Start bit in Control Word.	Read Only
Start_Block	Output	BOOL	Status of Start Block bit in Control Word.	Read Only
Relative	Output	BOOL	Status of Relative bit in Control Word. 0 = Absolute; 1 = Relative	Read Only
Direction	Output	BOOL	Status of Direction bit in Control Word. 0 = Negative; 1 = Positive	Read Only
Smooth_Stop	Output	BOOL	Status of Smooth Stop bit in Control Word.	Read Only
Hard_Stop	Output	BOOL	Status of Hard Stop bit in Control Word.	Read Only
Reserved_Bit_6	Output	BOOL	Reserved.	Read Only
Enable	Output	BOOL	Status of Enable bit in Control Word.	Read Only

Execution

Condition	Ladder Diagram Action
.EnableIn False	Not applicable. While the rung is False, data in the fields (outputs) are not updated and are frozen in their last state at the time when the rung went False.
Logic	While the rung is True (.EnableIn = 1; True) the data in the AOI fields (outputs) are updated with axis data every PLC scan. RPI scan time may affect how often the data is updated as well.

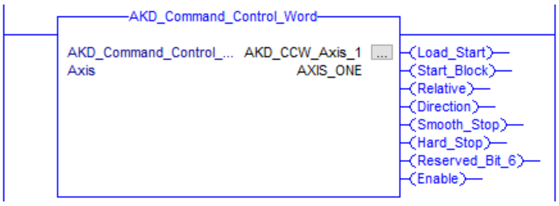
Changes to Axis Status Bits

- None

Example of Usage/Programming Guidelines

The AKD_Command_Control_Word AOI is always Enabled in the Ladder program and is executed on every scan.

The AKD_Command_Control_Word field must have a declared Tag for that instance and the Axis must be the same Tag for the given axis as the Axis_Internal field in the AKD_Drive Add-On Instruction. For AKD2G_Drive the Axis_Internal or Axis2_Internal field must be the same.



Troubleshooting

- None

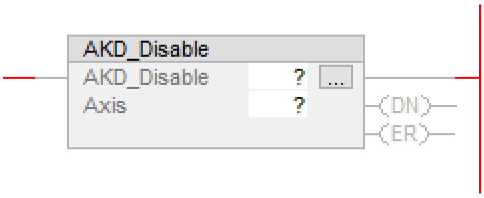
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	• Initial release • AOI library v5.0 revision	08/13/2015
v2.0	• Updated to sync with new AKD_Data type in the Axis structure. • AOI library v6.0 revision.	07/16/2021

7.8 AKD_Disable



Description

The AKD_Disable instruction disables the axis' software enable via Fieldbus Enable. This turns off the axis' output and using DRV.DISMODE (AKD1G) or AXIS#.DISMODE (AKD2G) disables the axis and puts it in the Disabled state. When executed, the AKD_Disable instruction also disables any motion active for the given axis.

Compatibility

The AKD_Disable AOI is compatible with both AKD1G and AKD2G drives.

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The AKD_Disable instruction disables the axis' software enable via Fieldbus Enable. This turns off the axis' output and using DRV.DISMODE (AKD1G) or AXIS#.DISMODE (AKD2G) disables the axis and puts it in the Disabled state. When executed, the AKD_Disable instruction also disables any motion active for the given axis.

The AKD_Disable instruction only requires the Axis Name (Tag) to be declared in the Axis_Internal field for the given axis' AKD_Drive AOI or the Axis_Internal or Axis2_Internal fields for the given AKD2G_Drive AOI.

NOTE

The AKD_Disable instruction execution time may require multiple scans to execute (additional 300-350 ms) due to required fieldbus communication time and time for the drive output and servo loop to be deactivated.

The Done (.DN) bit is set when the AKD_Disable AOI successfully executes and the Enable State (bit 7 in Status Word 1) turns OFF, thereby confirming the axis has transitioned to a disabled state.

Operands

These entries are required by the user.

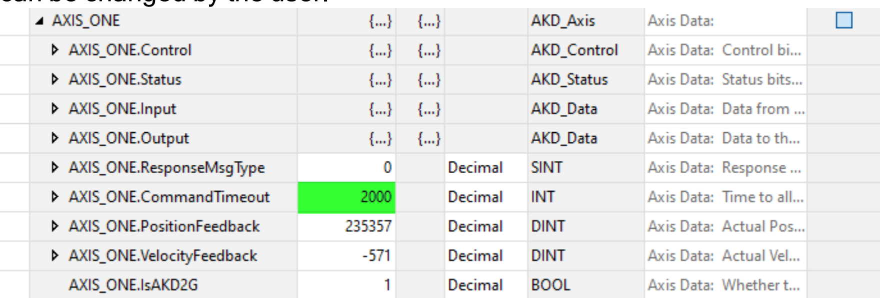
Operand	Type	Format	Description
AKD_ Disable	AKD_ Disable	Tag	Tag name for this instance of the AOI.
Axis	AKD_ Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the AOI execution completes successfully (Disable state acknowledged).
.ER	Output	BOOL	The .ER bit is set if the Enable OFF in Status Word 1 is not confirmed in the time set by the Command Timeout preset.

Execution

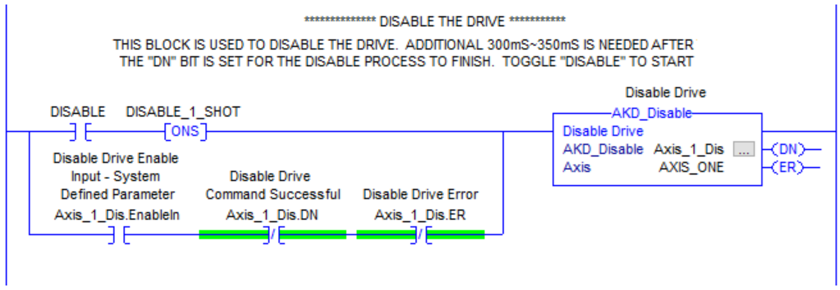
<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>Reset the Command Timeout timer and set the preset to the CommandTimeout defined in the Controller Tag structure for the given axis.

Example: AXIS_ONE
The screencapture below shows the Controller Tags of the AXIS_ONE structure of the Sample Project. The AXIS_ONE.CommandTimeout value is set to 2000 ms. This value can be changed by the user.

<table>
<tr><td>AXIS_ONE</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Axis</td><td>Axis Data:</td></tr>
<tr><td>AXIS_ONE.Control</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Control</td><td>Axis Data: Control bi...</td></tr>
<tr><td>AXIS_ONE.Status</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Status</td><td>Axis Data: Status bits...</td></tr>
<tr><td>AXIS_ONE.Input</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data from ...</td></tr>
<tr><td>AXIS_ONE.Output</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data to th...</td></tr>
<tr><td>AXIS_ONE.ResponseMsgType</td><td>0</td><td></td><td>Decimal</td><td>SINT</td><td>Axis Data: Response ...</td></tr>
<tr><td>AXIS_ONE.CommandTimeout</td><td>2000</td><td></td><td>Decimal</td><td>INT</td><td>Axis Data: Time to all...</td></tr>
<tr><td>AXIS_ONE.PositionFeedback</td><td>235357</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Pos...</td></tr>
<tr><td>AXIS_ONE.VelocityFeedback</td><td>-571</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Vel...</td></tr>
<tr><td>AXIS_ONE.IsAKD2G</td><td>1</td><td></td><td>Decimal</td><td>BOOL</td><td>Axis Data: Whether t...</td></tr>
</table></td></tr>
<tr><td>.EnableIn False</td><td>Reset the Command Timeout timer.</td></tr>
<tr><td>Logic</td><td>Unlatch Enable: Unlatch the Enable bit (bit 7 of the Control Word) of the Command Assembly. Byte 0 for Axis 1 (AKD1G or AKD2G) and Byte 64 for Axis 2 (AKD2G)

Enable State OFF: The AOI's internal logic checks for confirmation that the Enable State is OFF by checking bit 7 of the Response Assembly's Status Word 1. (Byte 0 for Axis 1 [AKD1G or AKD2G] and Byte 64 for Axis 2 [AKD2G])
If the Enable State is OFF then the Done (.DN) bit in the AOI is set.
If the acknowledgement does not happen in a timely fashion (200 ms), set the .ER bit in the AOI.</td></tr>
</table>

Example of Usage/Programming Guidelines

As demonstrated in the Sample project, the best practice for the AKD_Disable AOI is to use a conditional N.O. Contact as a trigger (DISABLE in the example below) and then use a One Shot (ONS) to trigger the AKD_Disable AOI .EnableIn. A parallel branch is implemented around the N.O. Contact and One Shot to seal-in the .EnableIn of the AOI until execution completes (.DN; Done) or fails (.ER; Error).



Troubleshooting

The condition for the Error (.ER) bit to be set for the AKD_Disable AOI:

- Enable OFF in Status Word 1 is not confirmed within 200 ms.

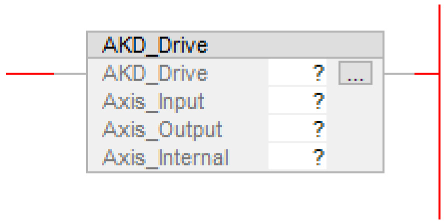
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	Initial version	02/23/2011
v2.1	• Don't error if Command Timeout = 0 • AOI library v5.0 revision	10/29/2014
v3.0	• Updated to sync with the new data type AKD_Data in the Axis structure. • AKD_Data in the Axis structure • AOI library v6.0 revision	07/16/2021

7.9 AKD_Drive



Description

The AKD_Drive instruction initiates communication for an axis.

Compatibility

The AKD_Drive AOI is only compatible with the AKD drive. For AKD2G drives use the AKD2G_Drive AOI.

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

Use the AKD_Drive instruction to initiate communication for an axis. The Axis_Input and Axis_Output bind the data exchange between the declared Generic Ethernet Module of the AKD drive for both the Command Assembly and Response Assembly to the Axis name in the AKD_Drive instruction (Axis_Internal). A structure in the Controller Tags is created when the Axis_Internal Tag (name) is declared. This internal name is then used with all the other AOIs which require an Axis field entry so the AOIs used with a given axis is tied or correlated to the AKD_Drive AOI.

The AKD_Drive AOI must be tied to the left rail of the Ladder so it executes on every scan and should be placed in the Main program or at the top of the Axis' subroutine prior to scanning the other codependent AOIs for that Axis.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Drive	AKD_Drive	Tag	Tag name for this Instance of the AOI.
Axis_Input	AB:ETHERNET_MODULE_SINT_64Bytes:I:0	Tag	Must point to the Generic Ethernet Module for the given AKD drive. The syntax will follow NAME:I where NAME is the name of the Generic Ethernet Module defined when it is first declared.
Axis_Output	AB:ETHERNET_MODULE_SINT_64Bytes:O:0	Tag	Must point to the Generic Ethernet Module for the given AKD drive. The syntax will follow NAME:O where NAME is the name of the Generic Ethernet Module defined when it is first declared.
Axis_Internal	AKD_Axis	Tag	User Tag defining the name of Axis 1 of the AKD drive. This name is used in all other AOIs to be used with and correlated to this axis.

Structure

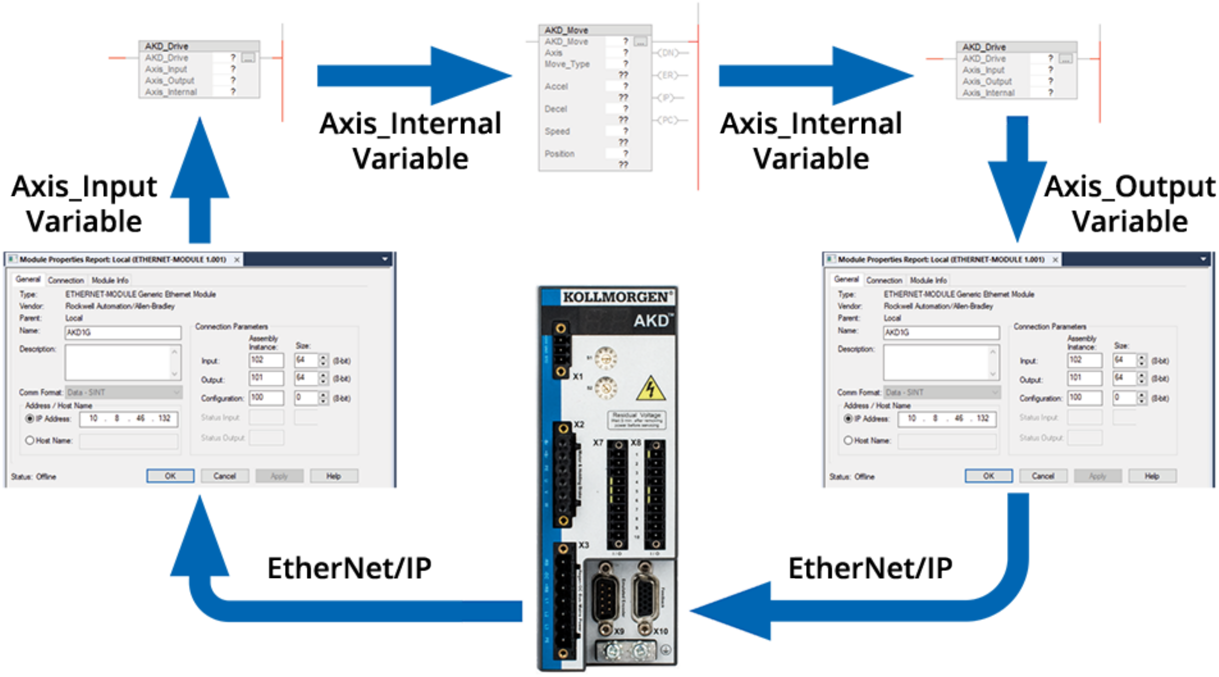
Mnemonic	Type	Description
.EnableIn	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.

Execution

Condition	Ladder Diagram Action
Prescan	• Unlatch the following bits of the Control Word: **Bit 0:** Load Command Data/Start Profile Move **Bit 1:** Execute command block of block chain **Bit 2:** Incremental (Select Absolute; 0 = Absolute) **Bit 3:** Direction in Velocity Mode (Select Reverse) **Bit 4:** Smooth Stop **Bit 5:** Hard Stop **Bit 6:** Registration Input Armed (Bit 6 is Reserved. There is a Tag in the axis structure for this.) **Bit 7:** Enable • Clear the entire Command Assembly Data Structure.
Logic	While the rung is True (.EnableIn = 1; True), the AOI outputs are updated with axis data on every PLC scan. RPI scan time may also affect how often the data is updated.

The example below shows how an AOI is codependent on the AKD_Drive AOI.

In a normal EtherNet/IP application, the Motion Task 0 data is transferred, from one variable to another, six times before the answer arrives back at the AKD_Move block. The data is transferred four times inside the PLC and two times outside the PLC.



All 64 Bytes of the Generic Ethernet Module Inputs are copied (0-63 for Axis 1) into the Axis_Internal structure for the axis. (Axis_Internal.Input).

1. Copy the **Input data** from the Generic Ethernet Module to the declared Axis_Internal name's **Input Bytes** (Axis_Internal.Input) which are part of the structure when the AKD_Drive AOI is declared. This is part of the structure when the AKD_Drive's Axis_Internal Tag is declared. The example below shows an axis declared as **AXIS_ONE**.

◢ AXIS_ONE	{...}	{...}		AKD_Axis
▷ AXIS_ONE.Control	{...}	{...}		AKD_Control
▷ AXIS_ONE.Status	{...}	{...}		AKD_Status
▷ AXIS_ONE.Input	{...}	{...}		AKD_Data
▷ AXIS_ONE.Output	{...}	{...}		AKD_Data
▷ AXIS_ONE.ResponseMsgType	0		Decimal	SINT
▷ AXIS_ONE.CommandTimeout	0		Decimal	INT
▷ AXIS_ONE.PositionFeedback	-5586956		Decimal	DINT
▷ AXIS_ONE.VelocityFeedback	-2405		Decimal	DINT
AXIS_ONE.IsAKD2G	1		Decimal	BOOL

2. Move from **Input** (Response Assembly data) to **Status** structure, which is part of the structure when the AKD_Drive Axis_Internal Tag is declared.

AXIS_ONE	{...}	{...}		AKD_Axis	Axis Data:
AXIS_ONE.Control	{...}	{...}		AKD_Control	Axis Data: Contr...
AXIS_ONE.Status	{...}	{...}		AKD_Status	Axis Data: Status...
AXIS_ONE.Status.Profile_In_Progress	0		Decimal	BOOL	Axis Data: Profile...
AXIS_ONE.Status.Block_In_Execution	0		Decimal	BOOL	Axis Data: Block ...
AXIS_ONE.Status.On_Target_Position	0		Decimal	BOOL	Axis Data: On Ta...
AXIS_ONE.Status.General_Fault	0		Decimal	BOOL	Axis Data: Gener...
AXIS_ONE.Status.Current_Direction	0		Decimal	BOOL	Axis Data: Curre...
AXIS_ONE.Status.Home_Level	1		Decimal	BOOL	Axis Data: Level ...
AXIS_ONE.Status.Reg_Level	0		Decimal	BOOL	Axis Data: Level ...
AXIS_ONE.Status.Enable	1		Decimal	BOOL	Axis Data: State ...
AXIS_ONE.Status.Fault_Input_Fault	0		Decimal	BOOL	Axis Data: Fault I...
AXIS_ONE.Status.Fwd_Limit	0		Decimal	BOOL	Axis Data: Forwa...
AXIS_ONE.Status.Rev_Limit	0		Decimal	BOOL	Axis Data: Rever...
AXIS_ONE.Status.Positive_Limit	0		Decimal	BOOL	Axis Data: Positi...
AXIS_ONE.Status.Negative_Limit	0		Decimal	BOOL	Axis Data: Negat...
AXIS_ONE.Status.FE_Fault	0		Decimal	BOOL	Axis Data: Follow...
AXIS_ONE.Status.Block_Fault	0		Decimal	BOOL	Axis Data: Block ...
AXIS_ONE.Status.Load_Complete	0		Decimal	BOOL	Axis Data: Comm...

3. Copy Byte 3 **Axis_Internal.Input.Data** to the **Axis.ResponseMsgType** Controller Tag.

AXIS_ONE	{...}	{...}		AKD_Axis
AXIS_ONE.Control	{...}	{...}		AKD_Control
AXIS_ONE.Status	{...}	{...}		AKD_Status
AXIS_ONE.Input	{...}	{...}		AKD_Data
AXIS_ONE.Output	{...}	{...}		AKD_Data
AXIS_ONE.ResponseMsgType	0		Decimal	SINT
AXIS_ONE.CommandTimeout	0		Decimal	INT
AXIS_ONE.PositionFeedback	-5586956		Decimal	DINT
AXIS_ONE.VelocityFeedback	-2405		Decimal	DINT
AXIS_ONE.IsAKD2G	1		Decimal	BOOL

4. Copy Position and Velocity feedback to **AxisName.PositionFeedback** and **AxisName.VelocityFeedback**.

AXIS_ONE	{...}	{...}		AKD_Axis
AXIS_ONE.Control	{...}	{...}		AKD_Control
AXIS_ONE.Status	{...}	{...}		AKD_Status
AXIS_ONE.Input	{...}	{...}		AKD_Data
AXIS_ONE.Output	{...}	{...}		AKD_Data
AXIS_ONE.ResponseMsgType	0		Decimal	SINT
AXIS_ONE.CommandTimeout	0		Decimal	INT
AXIS_ONE.PositionFeedback	-5586956		Decimal	DINT
AXIS_ONE.VelocityFeedback	-2405		Decimal	DINT
AXIS_ONE.IsAKD2G	1		Decimal	BOOL

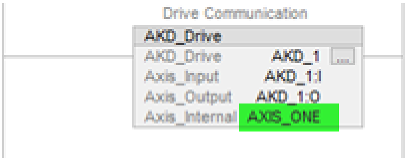
5. Copy the internal **Control Word Byte** (bits 0-7) for each axis to the **Axis_Internal.Output.Data[]x** which is represented by Axis_Internal.Output.Data[0].x.
6. Copy the **Axis_Internal.Output data** in the structure for each axis to the **Axis_Output.Data** (Generic Ethernet Module) for Axis 1 (Bytes 0-63). This passes the internal values on the scan as set by the other AOIs out to the Ethernet Module for that axis.

Changes to Axis Status Bits

All axis status bits are updated from the drive. See Status Word 1 and Status Word 2 for more information.

Example of Usage/Programming Guidelines

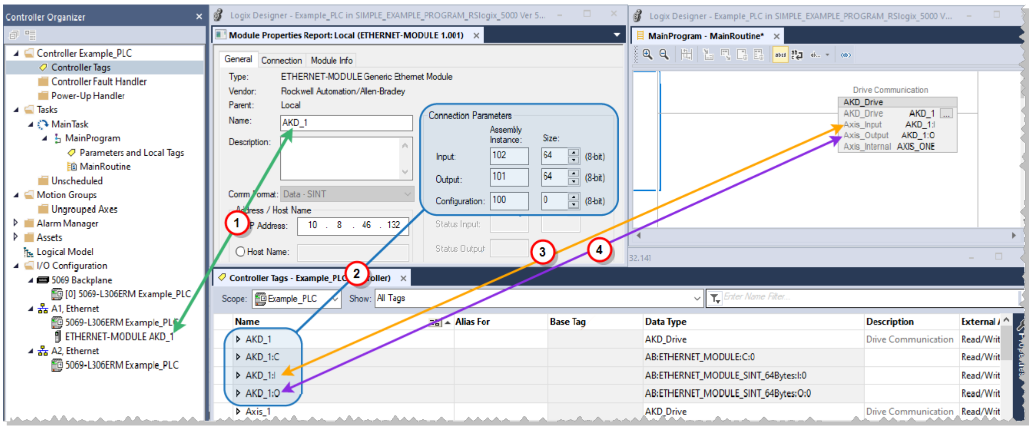
The screen capture below, taken from the Sample project, demonstrates usage. The AKD_Drive Add-On Instruction (AKD_Drive Communications) is always Enabled in the Ladder program and is executed on every scan. All other AOIs for a declared axis are codependent on the AKD_Drive AOI. Therefore, the AKD_Drive should be scanned first in the Ladder before the other AOIs for that axis.



The screen capture below shows:

- the AKD axis' Generic Ethernet Module
- the AKD_Drive Add-On Instruction is in rung 0 of the routine.
- the Controller Tags and structures which were declared when the Generic Ethernet Module was added and configured . (See arrow 2.)
- the instance Tag name that was used for the AKD_Drive Input of the AKD_Drive AOI when the AOI was added to the Ladder.

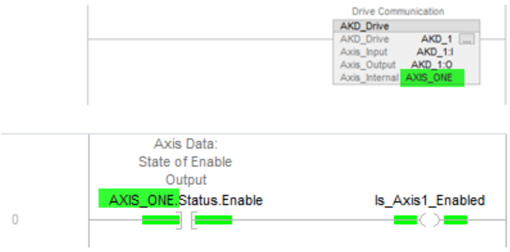
The name of the Generic Ethernet Module is up to the programmer. Since this is a Sample project, AKD_ 1 was used. (See arrow 1.)

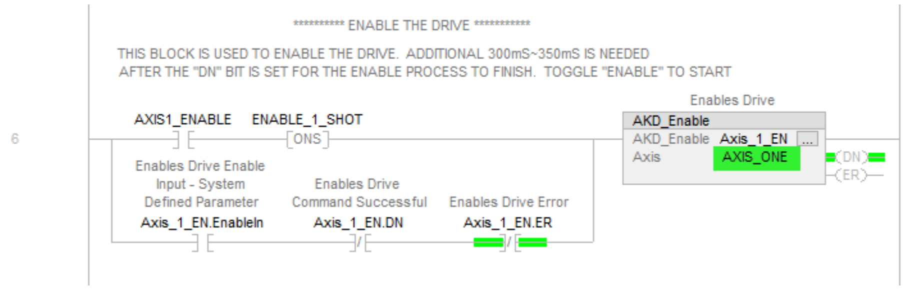


A few important notes about the screen capture above:

- When the Generic Ethernet Module is added under Ethernet and then configured by declaring a Name and the Connection Parameters, the Tag structure in the Controller database is created. (See arrow 2.)
- In the AKD_Drive AOI, the Axis_Input and Axis_Output entries point to the Name.I and Name:0 Tags to correlate the AKD_Drive AOI with the Generic Ethernet Module to be associated with the Axis_Internal Tag name. In this way, the AKD_Drive correlates and acts as a liaison between the AOI instances which are declared later with the same Axis name as the AKD_Drive Axis_Internal Tag name and the communications of the Generic Ethernet Module (The AKD with the respective IP Address). (See arrow 3 and arrow 4.)

Example from the Ladder:





Troubleshooting

- None

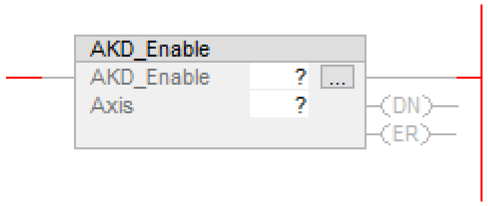
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	03/08/2011
v2.0	• Fix COP to Axis_Internal.PositionFeedback. Copying 16 Bytes worth of data; only 4 needed. • Fix COP to Axis_Internal.VelocityFeedback. Copying 16 Bytes worth of data; only 4 needed. • AOI library v5.0 revision	03/05/2015
v3.0	• Updated to sync with new AKD_Data datatype in the Axis structure. • AOI library v6.0 revision	07/16/2021

7.10 AKD_Enable



Description

The AKD_Enable AOI sets the Enable bit (bit 7) of the Control Word for the given axis.

Compatibility

The AKD_Drive AOI is compatible with both AKD1G and AKD2G drives.

Command Source and Operation Mode Requirements

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The AKD_Enable AOI sets the Enable bit (bit 7) of the Control Word for the given axis. The Done bit (.DN) is set when the axis' Enable state has been acknowledged in the Status Word (bit 7 of Status Word 1). The Enable bit of the Control Word, on rising edge, turns on the Fieldbus Enable.

NOTE

The AKD_Enable instruction may take multiple scans to execute since it requires the transmission of the message and for the drive to process and execute the request. An estimate is in the realm of 300-350 ms.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_ Enable	AKD_ Enable	Tag	The Enable Input bit indicates the instruction is Enabled.
Axis	AKD_ Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

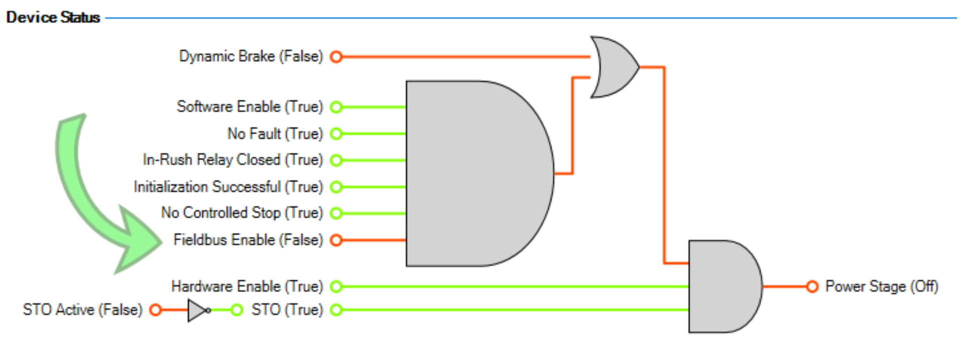
Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns on if the AOI execution completes successfully. Enable State is acknowledged via the Axis.Status.Enable (Status Word 1 bit 7).
.ER	Output	BOOL	The .ER bit is set if the Enable ON in Status Word 1 (bit 7) is not confirmed as True in 2000 ms.

Execution

Condition	Ladder Diagram Action
Prescan	Reset the Command Timeout timer and set the preset to the Command Timeout for 2000 ms.
.EnableIn False	Resets the Command Timeout timer.
Logic	• If there is no General Fault in Status Word 1 (bit 3), latch the Enable bit (bit 7 in Control Word). • When Status Word 1 confirms the Enable State (bit 7), set the Done bit (.DN). • For as long as the axis status does not confirm the Enable took place, start a timer based on the Command Timeout. • If there is a General Fault present (bit 3 in Status Word 1) during execution or if the Enable State (bit 7 in Status Word 1) is not confirmed after 2000 ms, the Error bit (.ER) is set.

The AKD_Enable Add-On Instruction is correlated to the Fieldbus Enable as shown below in the Enable/Disable screen of WorkBench.

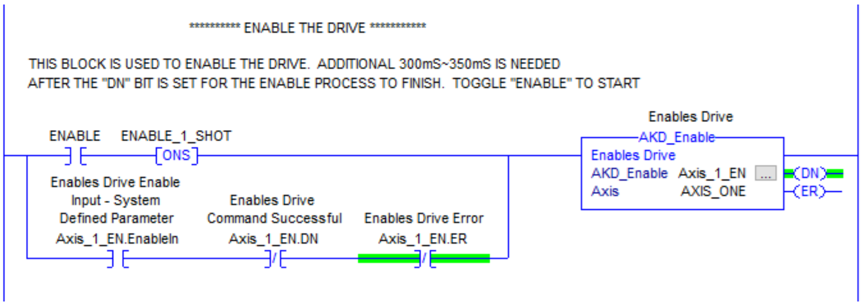


Changes to the Axis Status Bits

On success, the Enable State (bit 7 in Status Word 1) turns ON.

Example of Usage/Programming Guidelines

As shown in the Sample project sceen capture below, the best practice for the AKD_Enable AOI is to use a conditional N.O. Contact as a trigger (ENABLE in the example) and then use a One Shot (ONS) to trigger the AKD_Enable AOI. A parallel branch is implemented around both the N.O. Contact and One Shot to seal-in the .EnableIn of the AOI until execution either completes (.DN; Done) or fails (.ER; Error).



Troubleshooting

There are two conditions for the .ER bit to be set for the AKD_Enable AOI:

1. General Fault (bit 3 in Status Word 1) is present during execution
2. Enable Off (bit 7 in Status Word 1) was not confirmed within 2000 ms.
   Potential reasons:
   - Communications issue
   - Drive is not ready to be enabled, even if not faulted (i.e. HW Enable or Controlled Stop is OFF).

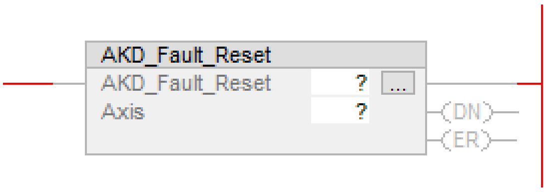
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
v2.0	AOI Library v5.0 revision	10/29/2014
v3.0	• Updated to sync with new AKD_Data data type in the Axis structure. • AOI library v6.0 revision	07/16/2021

7.11 AKD_Fault_Reset



Description

The AKD_Fault_Reset AOI clears and resets faults then unlatches the Enable bit (bit 7 in Control Word) for the given axis.

Compatibility

The AKD_Drive AOI is compatible with both AKD1G and AKD2G drives.

Command Source and Operation Mode Requirements

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The AKD_Fault_Reset AOI first attempts to reset faults by writing a 1 to Instance 113 (DRV.CLRFAULTS) if the given axis is an AKD1G drive or to Instance 5022 (AXIS#.CLRFAULTS) for an AKD2G drive.

AKD1G Parameter Listing:

Instance	Parameter	Data Size	Data Type
113	DRV.CLRFAULTS	Command	None

AKD2G:

Parameter	ID	Data Type	Access	Units
AXIS#.CLRFAULTS (See AKD2G EtherNet/IP Objects List)	5022	Unsigned8	Read/Write	

On success (i.e. the fault condition was removed and the command to reset faults succeeded), the AOI execution unlatches the Enable bit (bit 7 in Control Word) for the given axis.

NOTE

The AKD_Reset_Fault AOI may take multiple scans to execute since it requires transmission of the message and time for the drive to execute the command. The Done bit (.DN) is not set until all faults have been cleared.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_ Fault_ Reset	AKD_ Fault_ Reset	Tag	Tag name for the given instance of the AKD_Fault_Reset AOI in the Ladder.
Axis	AKD_ Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	The .DN bit turns on if the AOI execution completes successfully (write to the Clear Fault parameter was successful). AKD1G (DRV.CLRFAULTS) or AKD2G (AXIS#.CLRFAULTS)
.ER	Output	BOOL	The .ER bit is set if the attempt to clear faults by writing a 1 to Instance Number 113 on an AKD1G drive or Parameter Number 5022 on an AKD2G drive fails.

Execution

Condition	Ladder Diagram Action
Prescan	Set the Step Number to 0.
.EnableIn False	Set the Step Number to 0.

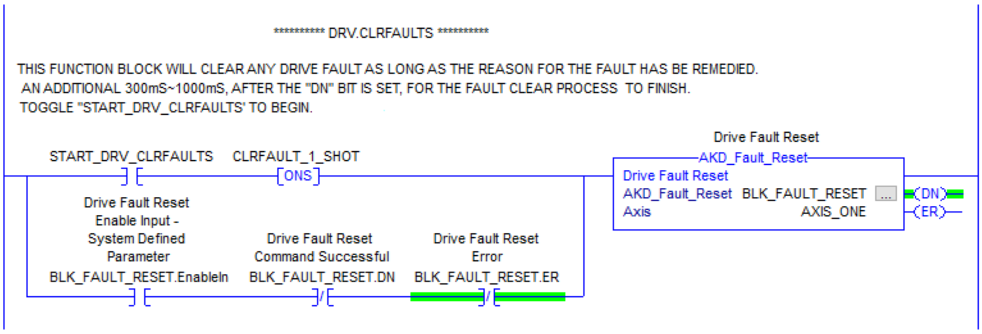
Condition	Ladder Diagram Action
Logic	**Step Number 0:** Unlatch the .DN bits of both internal AKD_Set_Parameter AOIs, as well as, the .DN bit of the AKD_Fault_Reset AOI. **Step 1 (AKD1G):** • Attempt to reset faults by writing a 1 to Instance 113 (DRV.CLRFAULTS). • If successful, set the Done bit (.DN) of the AKD_Fault_Reset AOI and unlatch the Enable bit (bit 7 of the Control Word). • Set Step Number to 2. **Step 1 (AKD2G):** • Attempt to reset faults by writing a 1 to Instance 5022 (AXIS#.CLRFAULTS) for the given axis. • If successful, set the Done bit (.DN) of the AKD_Fault_Reset AOI and unlatch the Enable bit (bit 7 of the Control Word) for the given axis. • Set Step Number to 2. **Step 2:** Success on Write. **Error:** • If the internal AKD_Set_Parameter attempt to write fails, then latch the AKD_Fault_Reset AOI's Error (.ER) output bit. • Set the Step Number to -1.

Changes to Axis Status Bits

General Fault (bit 3 in Status Word 1) of the given axis is cleared on fault reset.

Example of Usage/Programming Guidelines

As shown in the Sample project, the best practice for the AKD_Fault_Reset AOI is to use a conditional N.O. Contact as a trigger (START_DRV_CLRFAULTS in the example) followed by a One Shot (ONS) to trigger the AKD_Fault_Reset AOI. A parallel branch is implemented around the N.O. Contact and One Shot to seal-in the .EnableIn of the AOI until execution completes (.DN; Done) or fails (.ER; Error).



Troubleshooting

- The .ER bit is set if the attempt to clear faults by writing a 1 to Instance Number 113 on an AKD1G drive or Parameter Number 5022 on an AKD2G drive fails.

Step Summary

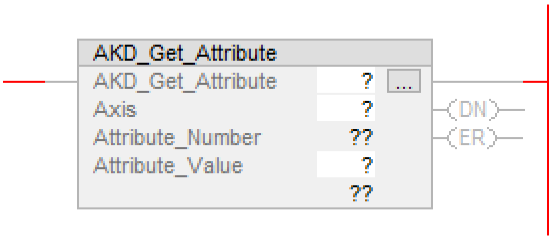
Step Number	Operation/Result
0	Unlatch the .DN bits of both internal AKD_Set_Parameter AOIs and the .DN bit of the AKD_Fault_Reset AOI.
1	Set Parameter 113 to a value of 1 (AKD1G - DRV.CLRFAULTS). Set Parameter 5022 for the given axis to 1 (AKD2G - AXIS#.CLRFAULTS)
2	Success on Write
-1	Error/Failure to Write

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
v4.0	• Added logic to disable drive (reset Enable bit in Control Word) when reset. • AOI library v5.0 revision	03/01/2016

v5.0	• Added logic based on whether AKD_Drive or AKD2G_Drive AOI is being used for the given axis.
	  ◦ If AKD1G drive, then reset fault Instance 113 (DRV.CLRFAULTS)
	  ◦ If AKD2G, then reset fault Instance 5022 (AXIS#.CLRFAULTS).
	• This version was updated to sync with the new data type AKD_Data used in the AKD_Axis data type.
	• AOI library v6.0 revision
		06/09/2021

7.12 AKD_Get_Attribute



Description

Use the AKD_Get_Attribute instruction to query a Position Controller attribute from an axis.

Compatibility

The AKD_Get_Attribute AOI is compatible with AKD1G and AKD2G drives.

Command Source and Operation Mode Requirements

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

Use the AKD_Get_Attribute instruction to query a Position Controller attribute from an axis.

NOTE

The Position Controller Attributes are a special set of drive parameters. See AKD1G EtherNet/IP Objects and Attributes or AKD2G EtherNet/IP Objects and Attributes for a list of available attributes and their Attribute ID.

In order to get an updated Attribute_Value, the AKD_Get_Attribute AOI must be disabled then enabled again. However, it is not recommended to attempt to use the AKD_Get_Attribute AOI where constant updates are needed in the application. Consider using Dynamic Mapping in this case. See Example of AKD1G Dynamic Mapping With Studio 5000 or Example of AKD2G Dynamic Mapping With Studio 5000.

Keep in mind this method is different from AKD_Get_Parameter which accesses the drive parameters (parameter listing) and does not access the Position Controller Attributes.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Get_Attribute	AKD_Get_Attribute	Tag	Tag name for the given instance of the AOI in the Ladder.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Attribute_Number	INT	Constant	See AKD1G EtherNet/IP Objects and Attributes or AKD2G EtherNet/IP Objects and Attributes.
Attribute_Value	DINT	Tag	Output Tag (Read Only)

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the AKD_Get_Attribute AOI execution completes successfully.
.ER	Output	BOOL	The .ER bit is set if there is a Response Type #14 (Error) or Command Timeout. See Response Type 0x14 - Command/Response Error for more information.

Behavior/Operation

The AOI automates the Get Attribute method internally.

Get Attribute

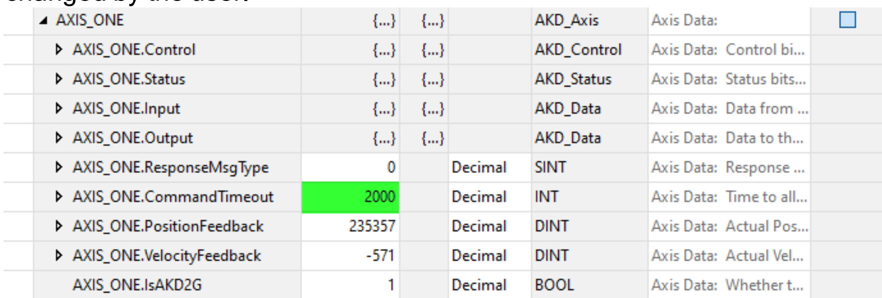
The Get Attribute method operates differently from other Command Types as it does not make use of the Command Type field or require the Load/Start to be set.

To read an attribute of the Position Controller in each cycle, set the Attribute To Get to the desired Attribute Number. The data will be returned in each Response Assembly Parameter Data and the Attribute To Get will be mirrored. See the table below for the respective Byte numbers.

	Where Desired Attribute # is Set	Response Assembly Parameter Data	Where Attribute To Get is Mirrored
AKD1G	Byte 32	Bytes 24-31	Byte 32
AKD2G (Axis 1)	Byte 28	Bytes 24-27	Byte 28
AKD2G (Axis 2)	Byte 92	Bytes 88-91	Byte 92

Execution

<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>Unlatch the One Shot and reset the Command Timeout. Set the Timeout Preset to match the Axis Command Timeout setting.

Example: AXIS_ONE
In the Controller Tags under the AXIS_ONE structure is the AXIS_ONE.CommandTimeout. The Sample project sets this to 2000 ms, but it can be changed by the user.

<table>
<tr><td>AXIS_ONE</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Axis</td><td>Axis Data:</td></tr>
<tr><td>AXIS_ONE.Control</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Control</td><td>Axis Data: Control bi...</td></tr>
<tr><td>AXIS_ONE.Status</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Status</td><td>Axis Data: Status bits...</td></tr>
<tr><td>AXIS_ONE.Input</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data from ...</td></tr>
<tr><td>AXIS_ONE.Output</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data to th...</td></tr>
<tr><td>AXIS_ONE.ResponseMsgType</td><td>0</td><td></td><td>Decimal</td><td>SINT</td><td>Axis Data: Response ...</td></tr>
<tr><td>AXIS_ONE.CommandTimeout</td><td>2000</td><td></td><td>Decimal</td><td>INT</td><td>Axis Data: Time to all...</td></tr>
<tr><td>AXIS_ONE.PositionFeedback</td><td>235357</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Pos...</td></tr>
<tr><td>AXIS_ONE.VelocityFeedback</td><td>-571</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Vel...</td></tr>
<tr><td>AXIS_ONE.IsAKD2G</td><td>1</td><td></td><td>Decimal</td><td>BOOL</td><td>Axis Data: Whether t...</td></tr>
</table></td></tr>
<tr><td>.EnableIn False</td><td>Unlatch the One Shot and timeout timer.</td></tr>
<tr><td>Instruction Execution</td><td>1. On Enable of the AOI, reset the .ER (Error) and .DN (Done) bits.
2. Set Byte 32 (AKD1G) or Byte 28 (AKD2G Axis 1) or Byte 92 (AKD2G Axis 2) of the Command Assembly to the Attribute Number.
3. Check if the Response Type is Error Code #14. If so, latch the AOI .ER bit and set the ErrorSource to -1.
4. While the command is not .DN (Done), if the timer preset is not 0 then start the timer. On Timeout (.DN; Done) latch the .ER (Error) bit and set the ErrorSource to -2.
5. Check for the Attribute Number to appear in the Response Assembly then copy the Attribute_Value from the Parameter Data Bytes of the Response Assembly to the AOIs Attribute_Value field and latch the AOIs .DN (Done) bit.
6. If the .ER (Error) bit is set, then set Byte 32 (AKD1G) or Byte 28 (AKD2G Axis 1) or Byte 92 (AKD2G Axis 2) in the Command Assembly to 0.</td></tr>
</table>

Changes to Axis Status Bits

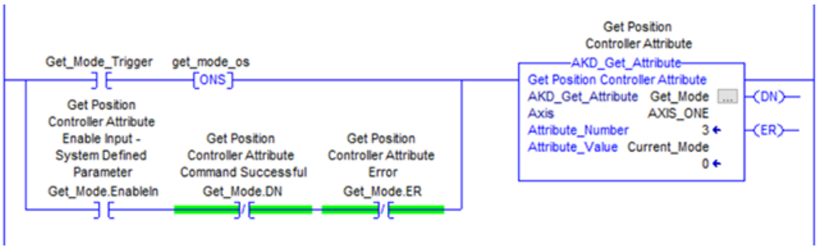
- None

Example of Usage/Programming Guidelines

The following example demonstrates setting the Attribute Mode (Attribute ID 3) value using the AKD_Get_Attribute AOI. For a list of Attribute IDs and their descriptions, see Position Controller Object 0x25 for AKD1G or see Position Controller Object 0x66 for AKD2G.

In the rung shown below, there is a N.O. Contact and One Shot to trigger the request to get the attribute. A parallel branch seals-in the AOI's .EnableIn until it returns a .DN (Done) or .ER (Error). In this example, the AKD_Get_Attribute is given an instance name (Tag) and the Attribute ID 3 (Mode). The Attribute_Value will be a Tag to hold the returned value from the drive.The returned Value can be either 0 (Position mode; default) ,1 (Velocity Mode), or 2 (Torque Mode) depending on the DRV.OPMODE (AKD1G) or AXIS#.OPMODE (AKD2G) the axis is in at the time the AKD_Get_Attribute request is executed.

Also note that there are other methods to access the same operation. For example, it is possible to access the DRV.OPMODE (AKD1G) or AXIS#.OPMODE (AKD2G) directly using the AKD_Get_Parameter AOI and the ID for those parameters instead of reading the attribute. See AKD1G Parameter Listing for AKD1G or AKD2G EtherNet/IP Objects List for AKD2G.



Troubleshooting

1. If there is an error code from the Response Type, the .ER bit is set.
2. If the Command Timeout times out indicating the command was not confirmed successful in the allotted time, the .ER bit is set.

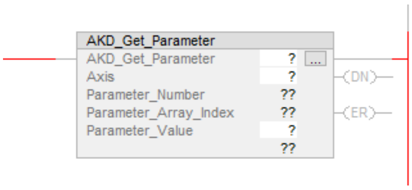
Step Summary

Step Number	Operation/Result
-1	Error from Response Type
-2	Command Timeout

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
	Fix COP to Attribute_Actual. Copying 16 Bytes worth of data, only 4 needed;	
v2.1	• Don't error is Command_Timeout preset is 0 • AOI library v5.0 revision	10/29/2014
v3.0	Initial Beta to accomodate AKD2G EtherNet/IP	06/01/2021
v3.1	• Changed internal logic to accommodate the differences in the Attribute To Get Byte assignment between AKD1G and AKD2G. • The update also synchronizes the AOI with the new AKD_Data datatype which is part of the Axis data type now. • AOI library v6.0 revision	07/27/2021

7.13 AKD_Get_Parameter



Description

The AKD_Get_Parameter AOI is used to query an axis parameter. Use Dynamic Mapping instead of the AKD_Get_Parameter AOI in cases where constant updates are needed in the application.

Compatibility

The AKD_Drive AOI is compatible with both AKD1G and AKD2G drives.

Command Source and Operation Mode Requirements

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The AKD_Get_Parameter AOI is used to query an axis parameter. See AKD1G Parameter Listing or AKD2G EtherNet/IP Objects List for the drive parameter list and their corresponding parameter numbers. The AKD_Get_Parameter AOI must be disabled and enabled again in order to get an updated Parameter_Value. However, it is not recommended to attempt use the AKD_Get_Parameter AOI where constant updates are needed in the application. Consider using Dynamic Mapping in this case. See Example of AKD1G Dynamic Mapping With Studio 5000 or Example of AKD2G Dynamic Mapping With Studio 5000 for more information.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Get_ Parameter	AKD_Get_ Parameter	Tag	Tag name for the given instance of the AOI in the Ladder.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_ Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Parameter_ Number	INT	Constant	See AKD1G Parameter Listing or AKD2G EtherNet/IP Objects List for the Instance ID to enter into this field.

Operand	Type	Format	Description
Parameter_Array_Index	DINT	Constants	AKD1G Always 0 for AKD1G Parameters. AKD2G Always 0 for non-array parameters Equal to the Array Index of Array Type parameters.
Parameter_Value	DINT	Tag	Output Tag (Read Only). Holds the returned value.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the AOI execution completes successfully where the Load Complete bit (Status Word 2, bit 7) is True (.DN = 1; True) and the data read from the Response Assembly is passed to the AOI's Parameter Value Tag.
.ER	Output	BOOL	The .ER bit is set if there is a Response Type #14 (Error) or Command Timeout.

Execution

The AKD_Get_Parameter AOI automates and uses the Command Type 0x1F - Read or Write Parameter Value method internally.

Learn more about the AKD1G Command Type 0x1F - Read or Write Parameter Value

This command type is used to configure or read any parameter in the drive. See AKD1G Parameter Listing 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.

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

Learn more about the AKD2G Command Type 0x1F - Read or Write Parameter Value

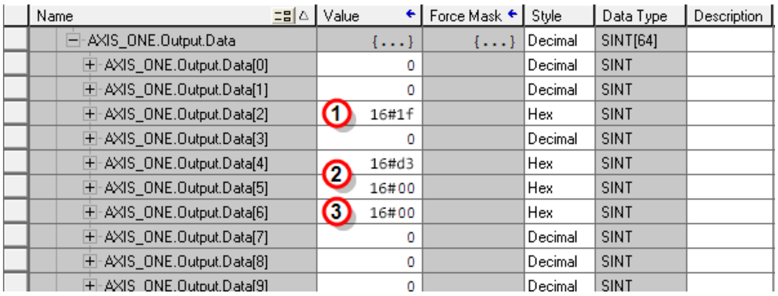
This command type is used to configure or read any parameter in the drive. See AKD2G EtherNet/IP Objects List for a listing of parameter indexes, data types, and scaling.

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.

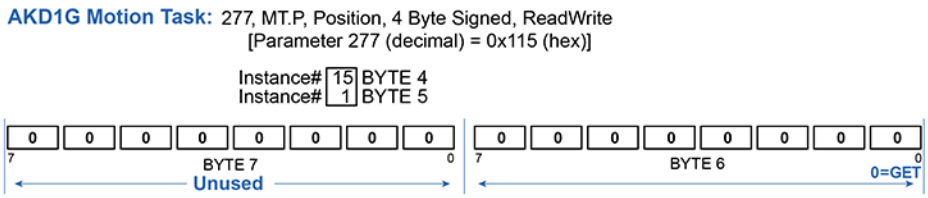
AKD1G Example (Internal Logic and Execution of AKD_Get_Parameter AOI)



Name	Value	Force Mask	Style	Data Type	Description
AXIS_ONE.Output.Data	{...}	{...}	Decimal	SINT[64]	
AXIS_ONE.Output.Data[0]	0		Decimal	SINT	
AXIS_ONE.Output.Data[1]	0		Decimal	SINT	
AXIS_ONE.Output.Data[2]	1 16#1f		Hex	SINT	
AXIS_ONE.Output.Data[3]	0		Decimal	SINT	
AXIS_ONE.Output.Data[4]	16#d3		Hex	SINT	
AXIS_ONE.Output.Data[5]	2 16#00		Hex	SINT	
AXIS_ONE.Output.Data[6]	3 16#00		Hex	SINT	
AXIS_ONE.Output.Data[7]	0		Decimal	SINT	
AXIS_ONE.Output.Data[8]	0		Decimal	SINT	
AXIS_ONE.Output.Data[9]	0		Decimal	SINT	

1. Byte 2 is set to Command Type 0x1F- Read or Write Parameter Value.
2. Bytes 4-5 are the Instance Number (16#d3 or 211 in this example).
3. For AKD1G Byte 6 is set to Read (0) and Byte 7 is unused.

*AKD1G: Index is 0 for all parameters and for non-array type parameters in the AKD2G drive.
*AKD2G only: Index is non-zero for array type parameters.

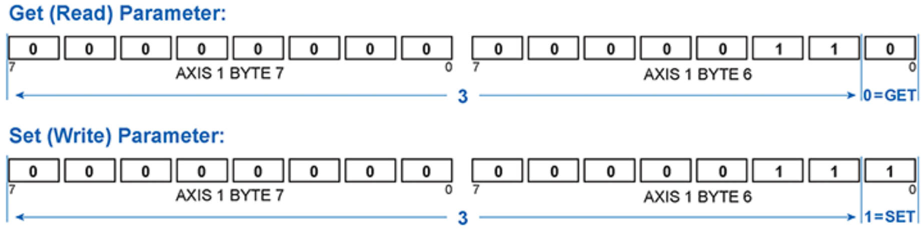


AKD2G Example (Internal Logic and Execution of AKD_Get_Parameter AOI)

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



AXIS_ONE.Output	{...}	{...}		AB:ETHERNET_MODU...	Axis Data: Data to th...
AXIS_ONE.Output.Data	{...}	{...}	Decimal	SINT[64]	Axis Data: Data to th...
AXIS_ONE.Output.Data[0]	0		Decimal	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[1]	0		Decimal	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[2]	16#1f		Hex	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[3]	0		Decimal	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[4]	16#a3		Hex	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[5]	16#18		Hex	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[6]	2#0000_0110		Binary	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[7]	2#0000_0000		Binary	SINT	Axis Data: Data to th...
AXIS_ONE.Output.Data[8]	0		Decimal	SINT	Axis Data: Data to th...

1. # = 0, GET
2. Command Type Byte 2 is set to Command Type - 0x1F- Read or Write Parameter Value.
3. Instance Number Bytes 4-5 are the Instance Number (16#18a3 or 6307 in this example).
4. Parameter Array Index For AKD2G axis 1 Byte 6, bit 0 is set to Read (0) Byte 6, bits 1-7 and Byte 7 bits 0-7 are used to hold the parameter array index

The data read will appear in the Parameter/Attribute Data bytes of the Response Assembly Data Structure.
AKD1G: Bytes 24-31;
AKD2G: Bytes 24-27 for Axis 1 and Bytes 88-91 for Axis 2.
See the Response Assembly Data Structure for more information.

Execution

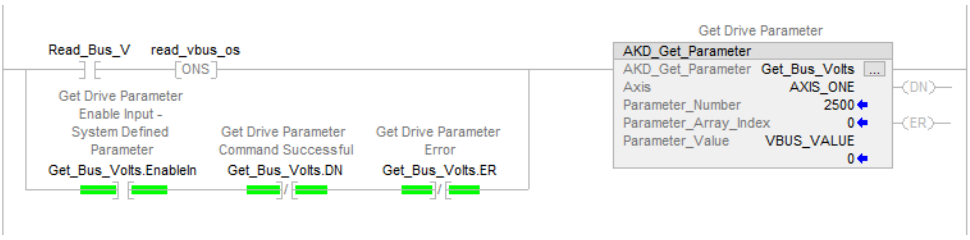
<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>Unlatch the loadstart One Shot.
Reset the timer.
Set the timer preset to 2000 ms.
Set the first internal Command_Bytes to zero.</td></tr>
<tr><td>.EnableIn False</td><td>If not Enabled, reset the One Shot and timeout timer. Clear the first eight internal Command_Bytes.</td></tr>
<tr><td>Instruction Execution</td><td>On Enable, set the Step Number to zero and clear the .DN and .ER bits.
Set Command Type to 16#1F and set Command Byte to 16#1F. (See Command Type 0x1F - Read or Write Parameter Value.)
Set Command_Bytes 4 and 5 of the Axis_Internal.Output.Data to the parameter number.
For AKD1G set Byte 6 of the Axis_Internal.Output.Data to Read (0).
For AKD2G set Byte 6 bit 0 of the Axis_Internal.Output.Data to Read (0) and set Byte 6 bits 1-7 and also Byte 7, bits 0-7 to the Parameter Array Index Number.
If the Response Assembly indicates the Load Complete bit (bit 7 Status Word 2) is High (Complete) then unlatch the Load/Start bit (bit 0) in the Control Word of the Command Assembly.
If the Step Number is 0 and the Load Complete is not .DN (Done) then latch the Load/Start bit (0) of the Control Word and set the Step Number to 1 (Next step).
If the Command bit is set (Step 1) then check for Success or errors. If the Response Message is Type 14 then there is an error. Set the AOI .ER bit and set the Step Number to -1.
If the Command bit is set (Step 1), start the 2000 ms Timeout timer. If the timer times out (.DN) then the command was not successful in the given time. Latch the AOI .ER bit and set the Step Number to -2.
If the Command bit is set (Step 1) and the Load Complete bit in the Response Assembly goes High (Success) then set the Step Number to 2.
If the Step Number is 2 (Success of load), copy the received data in the Response Assembly to the Parameter Value of the AOI and latch the AOIs .DN (Done) bit and set the Step Number to 3.
If there is an error, unlatch the Load/Start bit (bit 0) in the Control Word.
Copy the Control Word (Byte 0 of Axis Internal command) to the internal Command_ Bytes. Next, copy the eight internal Bytes of command data (Command_Bytes) to the Axis' output data.</td></tr>
</table>

Changes to Axis Status Bits

- None

Example of Usage/Programming Guidelines

A N.O. Contact and One Shot is used to trigger the request before the EnableIn is sealed-in using the parallel branch. The AKD_Get_Parameter stays Enabled until either .DN (Done) success or .ER (Error) failed. The AKD_Get_Parameter field is a unique Tag name for that instance of the AOI. The Axis field is the desired axis correlated to the Axis_Internal Tag name declared in the AKD_Drive AOI or Axis_Internal or Axis2_Internal for the AKD2G_Drive AOI at the time it was configured. The Parameter Number field is the Parameter Instance as noted in AKD1G Parameter Listing or AKD2G EtherNet/IP Objects List. The Parameter_Array_Index is always 0 for AKD1G parameters and in the AKD2G 0 for non-array type parameters or equal to the index number of an array-type parameter. The Parameter_Value is a Tag name to hold the value returned on success of the AKD_Get_Parameter AOI.



Troubleshooting

The Error (.ER) output bit of the AOI is set under one of the following conditions:

1. Error from the Response Type. (Communications error.)
2. Command Timeout. The command was not confirmed successful in the allotted time.

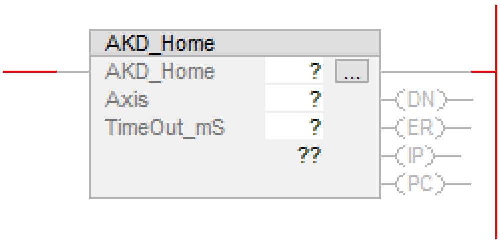
Step Summary

Step Number	Operation/Result
0	Enabled
1	Load/Start bit of the Control Word is set
2	Command successfully loaded (Load Complete); copy received data to AOI parameter value
3	AOI is declared Done (.DN bit is set)
-1	Error from Response Type
-2	Command Timeout

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
v2.0	AOI library v5.0 revision	10/29/2014
v3.0	• Eliminated axis number bits which could be a source of confusion and deleted the local Tag Command_Axis_Number which is now unused. • Added logic to accomodate AKD2G method including parameter array index. • Sync with Axis new data type AKD_Data • AOI library v6.0 revision	06/22/2021

7.14 AKD_Home



Description

The AKD_Home instruction starts the Home Move using the currently configured Homing Mode.

Compatibility

The AKD_Disable AOI is compatible with both AKD1G and AKD2G drives.

Command Source and Operation Mode Requirements

AKD1G

Required DRV.CMDSOURCE = Service

Required DRV.OPMODE = Position

AKD2G

Required AXIS#.CMDSOURCE = Fieldbus

Required AXIS#.OPMODE = Position

The AKD_Home instruction starts the Home Move using the currently configured Homing Mode. Note the instruction execution may require multiple scans to execute due to the message transmission time and the time required for the drive to perform the Home routine (some routines take longer to execute than others).

NOTE

TimeOut_mS (in milliseconds) is intended for cases where the Home routine may run for an extended period of time (i.e., while searching for the Home Input, but it is never triggered). The value entered also depends on the Home Mode as that effects how long the homing routine may take to complete. *Entering 0 msec does not disable the timeout timer*; instead the .ER (Error) bit will be set almost immediately. The timeout timer and .ER bit do not abort the move when using the .ER bit to unlatch the AKD_Home AOI. Once the Home Move has already started it will be necessary for the programmer to use the .ER bit in their program to handle the error (i.e., Disable, Smooth Stop, Hard Stop, etc.).

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Home	AKD_Home	Tag	Tag name for the given instance of the AOI in the Ladder.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Timeout_mS	DINT	Generally a Constant value, but Tag is possible	Set to 5000 ms in the Sample project. May be altered by the programmer as needed in the application.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the internal AKD_Set_Attribute AOI write to Attribute 101 (Home Move) is .DN (Success). Homing will commence.
.ER	Output	BOOL	The .ER (Error) bit is set if the: • Axis is faulted • Axis is not enabled • Time Out timer times out. • Internal set attribute AOI fails to write to Attribute 101. (See AKD1G EtherNet/IP Objects and Attributes.) • Axis Command Source or Op Mode are not Service Position (AKD1G) or Fieldbus Position (AKD2G).
.IP	Output	BOOL	While the Profile_In_Progress bit = 1, the Home Move is in progress and the .IP bit remains True for as long as the Profile_In_Progress bit is High (1).
.PC	Output	BOOL	When the Profile_In_Progress bit = 0 and Home_Level = 1 in Status Word 1, Homing is complete and the .PC bit is set on the output of the AKD_Home AOI.

Execution

The AKD_Home AOI makes use of Position Controller Object - Attribute ID 101 (Home Move) and automates the request to Home.

The Position Controller Object uses the same Attribute ID for AKD1G and AKD2G. However, the Position Controller Object for AKD1G is class 0x25 and AKD2G is class 0x66. See AKD1G EtherNet/IP Objects and Attributes or AKD2G EtherNet/IP Objects and Attributes.

AKD1G

When the AKD_Home AOI is executed successfully, writing a 1 to the controller attribute 101 (Home Move) changes the DRV.CMDSOURCE to Service (0) but does not change the DRV.OPMODE to Position Mode. The AKD_Home AOI will Error (.ER) if it is not in Position.

For example, if the drive is in Electronic Gearing and Position Mode, the DRV.CMDSOURCE changes to Service and the Home routine executes.

If the drive is not in Position Mode, e.g. DRV.OPMODE (Fieldbus Velocity). then the DRV.CMDSOURCE changes to Service but the DRV.OPMODE stays in Velocity Mode and the AKD_Home AOI errors (response error; failure to execute).

AKD2G

The AKD_Home instruction execution is different on the AKD2G drive in that the AKD_Home instruction does not change the AXIS#.CMDSOURCE. The axis' command source (AXIS#.CMDSOURCE) must be Fieldbus and the axis' OpMode (AXIS#.OPMODE) must be in Position Mode prior to executing the AKD_Home AOI or the AKD_Home AOI will error (.ER).

The following must be satisfied prior to commanding a Home Move:

- Faults are cleared
- Axis is enabled
- Axis is in Position Mode
- Axis command source is Service for AKD1G and Fieldbus for AKD2G.
- Smooth Stops and Hard Stops are cleared in Status Word 1.
- Position Limits (Postive SW Limit and Neg SW Limit) are cleared in Status Word 2.

Execution

<table>
<tr><th>Condition</th><th colspan="2">Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td colspan="2">Unlatch the StartONS One Shot and set Step Number to zero. Set the internal Tag Attribute Number of the internal Set_Attribute AOI to 101 (Home Move) and set the internal Tag value to 1.</td></tr>
<tr><td>.EnableIn False</td><td colspan="2">Unlatch the StartONS One Shot and set the Step Number to 0 then unlatch the .IP bit for the AOI.</td></tr>
<tr><td rowspan="4">Instruction Execution</td><td>On .EnableIn:</td><td>When the AKD_Home AOI is Enabled, One Shot and reset:
the TimeOut_mS timer
the Set_Attribute Done bit of the internal AOI
all of the AKD_Home AOI output and status bits (.DN, .ER, .PC).
set the Step Number to 1 and set the TimeOut timer preset to whatever the TimeOut_mS value is set for by the user. (The Sample project sets it to 5000 ms.)</td></tr>
<tr><td>General Fault:</td><td>The drive is not enabled, or the timer times out, latch the AKD_Home AOI's .ER bit and set the Step Number to -2.</td></tr>
<tr><td>If Step Number 1:</td><td>Then using the internal AKD_Set_Attribute AOI set the Attribute_ Number entry to 101 (Home Move) and the Attribute_Value entry's Value to 1 to command the Home Move. On success, the AKD_Set_ Attribute AOI .DN (Done) bit will be set. Next, latch the AKD_Home AOIs .DN (Done) bit and set the Step Number to 2. If the AKD_Set_ Attribute AOI errors, latch the AKD_Home AOI's .ER (Error) bit and set the Step Number to -1.</td></tr>
<tr><td>If Step Number 2:</td><td>The controller attribute was set successfully and the Profile_In_Progress bit from the Response Assembly Status Word 1 is ON, then turn on the AKD_Home AOI's .IP bit and start the TimeOut timer. The Profile_In_Progress bit is bit 0 (known as In Motion in AKD2G drives) from Status Word 1.
The controller attribute was set successfully) and the Profile_In_Progress bit from Status Word 1 is OFF and the Home_Level bit of the Status Word is ON then latch the AKD_Home AOI's PC (Process Complete) bit indicating Home complete.</td></tr>
</table>

Changes to Axis Status Bits

- Home_Level (bit 5 in Status Word 1) becomes True (1) when Homing is complete.
- Profile_In_Progress (In Motion) bit 0 in Status Word 1 isTrue (1) while Homing Move is in progress and False (0) when not in progress or complete.

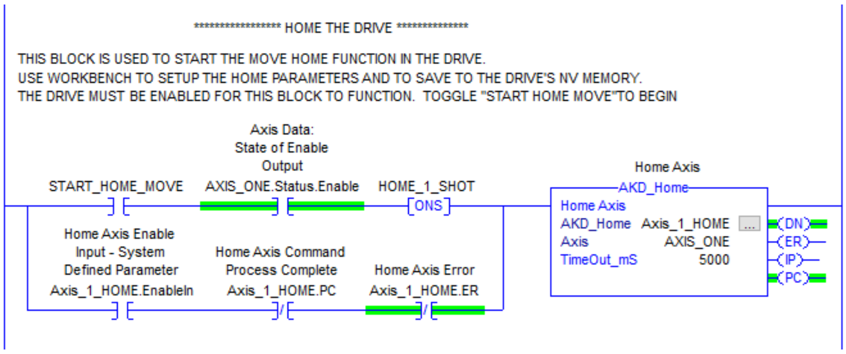
Example of Usage/Programming Guidelines

In the Sample project a N.O. Contact is used to trigger the request to Home the axis (i.e. Start_Home_Move). The trigger will One Shot the .EnableIn of the AKD_Home AOI and the parallel branch acts as a seal-in to keep the AKD_Home AOI Enabled until:

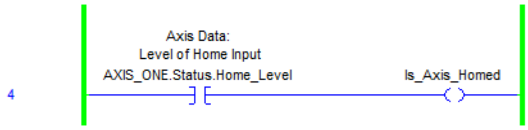
1. .PC (Process Complete) is set indicating execution is complete; Home Found; Home Done or
2. .ER (Error) execution failed.

NOTE

A N.O. Contact for the Axis(name).Status.Enable state is used as an interlock to prevent triggering of the AKD_Home AOI in the event the drive is not enabled. The Done (.DN) bit indicates the command successfully initiated. The .IP bit indicates execution is In Process.



Note the Sample project demonstrates how to add the Homed Status (Level) to the Ladder for use in the Ladder code.



Troubleshooting

Possible reasons the .ER bit was set:

1. Axis is faulted
2. Axis is not Enabled
3. Axis is not faulted but not ready for motion (i.e. Hardware disabled, Controlled Stopped, etc.).
4. Request to Set Attribute 101 to 1 failed (typically a communications issue).
5. Request to Set Attribute 101 to 1 took longer than the allotted time.
6. DRV.CMDSOURCE not set to Service for AKD1G or AXIS#.CMDSOURCE is not set to Fieldbus for AKD2G.
7. DRV.OPMODE is not in Position Mode for AKD1G or AXIS#.OPMODE is not in Position Mode for AKD2G.

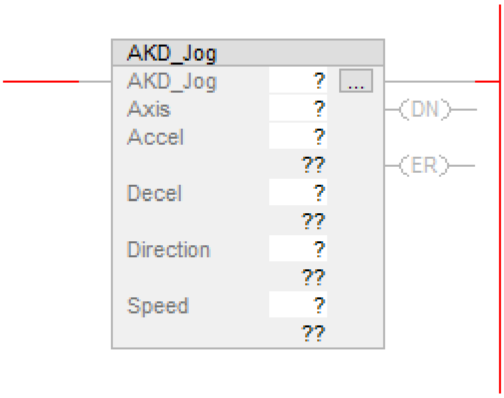
Step Summary

Step Number	Operation/Result
0	Enabled. Reset TimeoutTimer Unlatch .DN bit of internal Set_Attribute AOI. Unlatch .DN, .ER, and .PC output bits of AKD_Home AOI. Set Step Number to 1. Set TimeoutTimer preset to the TimeOut_mS entry of the AOI.
1	Set Attribute_Number to 101 (Home Move) and Value to 1. If internal Set_Attribute .DN (Success) latch the AKD_Home AOI .DN bit and set the step number to 2. If internal Set_Attribute ER (error) latch the AKD_Home AOI's ER bit and set the step number to -1.
2	Set Attribute was successful. Set .DN of AOI. While executing, the In Process bit (.IP) turns on for as long as the Profile_In_Progress status bit is ON. When no longer In Process and Home status bit from Status Word is True, set Process Complete (.PC) bit.
-1	Set Attribute failed. Unsuccessful. Set error (.ER) bit.
-2	General Fault, Drive not enabled, or timer timed out.

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
v2.1	.IP Output bit should be turned OFF when Enable Input is False.	
v4.0	• Fixed race from SetAttribute.DN bit. Added a timeout timer. • AOI library v5.0 revision.	03/01/2016
v5.0	• Updated to synchronize with the new AKD_Data Data Type which the Axis structure uses. • AOI library v6.0 revision	07/20/2021

7.15 AKD_Jog



Description

Use the AKD_Jog AOI to move the axis at a constant speed.

Compatibility

The AKD_Disable AOI is compatible with both AKD1G and AKD2G drives.

Command Source and Operation Mode Requirements

AKD1G

Required DRV.CMDSOURCE = Service

Required DRV.OPMODE = Position or Velocity

AKD2G

Required AXIS#.CMDSOURCE = Fieldbus

Required AXIS#.OPMODE = Position or Velocity

The AKD_Jog AOI execution is triggered using a rising edge, which means that even if the rung goes False the Jog motion will continue until either the axis is Disabled, a Smooth Stop or Hard Stop (shutdown) is commanded, the Hardware or Software Limit is reached, or the axis is faulted.

The In Motion bit (bit 0 in Status Word 1; also known as Profile_In_Progress in AKD1G) can be monitored to test if motion is In Progress.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Jog	AKD_ Jog	Tag	Tag name for the given instance of the AOI in the Ladder.
Axis	AKD_ Axis	Tag	Tag for which the Axis is declared. Must match the Axis_ Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Accel	DINT	Generally a Constant value but can be a Tag (variable)	Acceleration for Jog in EtherNet/IP units.
Decel	DINT	Generally a Constant value but can be a Tag (variable)	Deceleration for Jog in EtherNet/IP units.
Direction	BOOL	Generally a Constant value but can be a Tag (variable)	Direction bit for Jog. 0 = Reverse 1 = Forward
Speed	DINT	Generally a Constant value but can be a Tag (variable)	Velocity for Jog in EtherNet/IP units.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the AOI execution (Command To Jog) is successful. The Load_Complete (bit 7 in Status Word 2) indicates success to initiate jogging.
.ER	Output	BOOL	The .ER bit is set if there is a Response Type #14 (Error) or command timeout.

AKD1G Jog Execution

NOTE

The drive handles the Jog differently depending on if the DRV.OPMODE is set to Velocity Mode or Position Mode.

Velocity Mode

In Velocity Mode the drive uses the Command Type 0x07 - Jog Move. On success, the AKD_Jog AOI writes the Target Velocity to the setpoint VL.CMDU. Homing is not necessary in Velocity Mode.

In Velocity Mode the Accel and Decel values of the AKD_Jog AOI are written to EIP.ACC and EIP.DEC.

Note the Speed value in the AKD_Jog AOI will accept a signed value.

NOTE

The best practice is to use the Direction bit for direction and enter a positive value for the Speed in the AKD_Jog AOI field entry. For example, if Direction is set to 0 (Reverse) and Speed is set to -X where X is the Target Velocity, the effect is the Direction of the Jog will be positive. For this reason, using a positive Target Velocity for either direction and using the Direction bit is more consistent.

Position Mode

In Position Mode, values such as Target Velocity, Accel, Decel, etc. are loaded into Motion Task 0 (See Motion Task) and Motion Task 0 is setup and executed as a Repeating Motion Task. The drive must be Homed before using the AKD_Jog AOI in Position Mode. While in Position Mode, unlike in Velocity Mode, VL.CMDU is not used but the VL.CMD is at the Target Velocity of the Running Motion Task as a result of the trajectory of the position loop. Note the Speed value in the AKD_Jog AOI will accept a signed value, but in Position Mode the Target Velocity in the Motion Task is Motion Task Target Velocity = ABS(Speed) and the sign of the Speed entry is unused. The Direction bit will be used in the Position of Motion Task 0 and will set the Jog to positive or negative. See Command Type 0x06 - Position Move in the AKD EtherNet/IP Communication manual for more information on Motion Task 0 in AKD1G drives.

NOTE

The best practice is to use the Direction bit for direction and enter a positive value for the Speed in the AKD_Jog block. It is worth noting that the Speed entry will accept a signed value.

In Position Mode, even if the velocity is a negative value in the AKD_Jog AOI, only the magnitude is used (i.e. Motion Task 0 Target Velocity = ABS[Speed]) and the direction bit sets the direction.

Example of Motion Task 0 upon execution of the AKD_Jog AOI:

The Position value is set by the firmware using the formula below and the Direction bit in the AOI sets the sign of the Position.

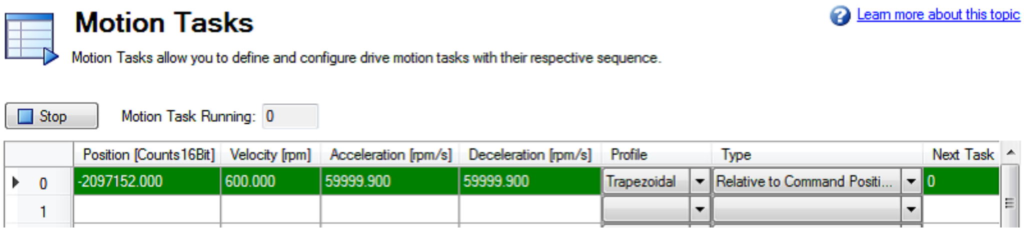
$$65536\,(counts/revs)\,x\,32 = rev\,2,097,152\,counts$$

NOTE

This assumes WorkBench Position Units are set to match the default EtherNet/IP units. If different, the value will be the equivalent of 32 revs and whichever position units are being used (i.e. mm, inch, deg, etc.).

- Velocity is set to the ABS(Speed) value.
- Accel and Decel are set to the entries in the AOI.
- Profile is set to Trapezoidal.
  (See Motion Profiles in the AKD EtherNet/IP Communication Manual manual for more information.)
- Type is Relative to Command Position.
  (See Motion task relative to command position (PL.CMD) in the AKD EtherNet/IP Communication manual for more information.)

Next Task is set to 0 (repeats itself) with no start condition (or dwell delay) and blend into acceleration is set.



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

- Faults are cleared (General Fault bit 3 in Status Word 1 is 0).
- Drive is Enabled (Enable State bit 7 in Status Word 1 is 1).
- Drive is in Position Mode and Command Source is Service.
- Smooth Stop and Hard Stop are cleared (not active) and 0 in Control Word.
- Position Limits are cleared (0) in Status Word 2 (bits 1, 2, 3, and 4).
- Homed status is True (1) in bit 5 in Status Word 1.

Note in either Velocity Mode or Position Mode once the execution of the AKD_Jog is acknowledged (Successful), even if the AKD_Jog .EnableIn becomes False, jogging will continue until either a Smooth Stop is executed, the drive faults, the drive is disabled, etc.

Stop Jogging

Since jogging will commence and continue, even if the AKD_Jog AOI .EnableIn becomes False in either Velocity Mode or Position Mode, use the AKD_Smooth_Stop AOI to stop jogging.

AKD2G Jog Execution

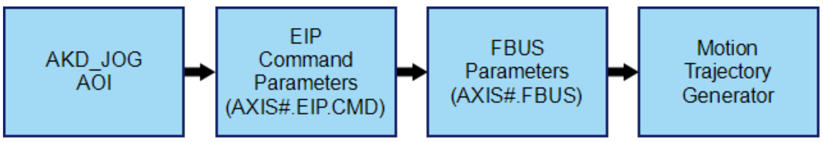
Overview

The AKD2G Command Type 0x07 - Jog Move differs in execution from the AKD1G in the following ways:

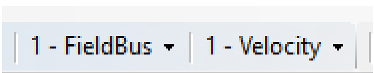
- The Command Source for the AKD2G must be Fieldbus, not Service as it is for the AKD1G.
- In Velocity Operation Mode, the Velocity command does not use the AXIS#.VL.CMDU in the AKD2G.
- In Position Operation Mode, the Jog command does not utilize the Motion Task table for the AKD2G but instead passes the EtherNet/IP values using the AXIS#.FBUS parameters to an internal move.

Regardless of AXIS#.OPMODE, (Velocity Mode or Position Mode) the flow of data and control for the AKD_Jog AOI to the AKD2G drive is as follows:

- The data and control over EtherNet/IP from the AKD_Jog AOI can be viewed in EtherNet/IP units in the EtherNet/IP command parameters (AXIS#.EIP Parameters) and in WorkBench units in the Fieldbus parameters (FBUS Parameters) with the final destination to the motion trajectory generator.
- The AXIS#.EIP.CMD and AXIS#.FBUS units will be displayed with the same values and same units as WorkBench units.
- Raw data from the EtherNet/IP Controller in EtherNet/IP scaling can be viewed in WorkBench using the keyword AXIS#.EIP.FIXEDCMDDATA. Alternatively, the raw data can be viewed under the Command Mapping tab at Communication → EtherNet/IP → Axis# → Command Mapping → Fixed Mapping.
- Note the key difference in execution from the AKD1G is the AKD2G Jog in Position Mode does not utilize a Motion Task in the Motion Task table, but instead uses the FBUS data and control internally.



Velocity Mode



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

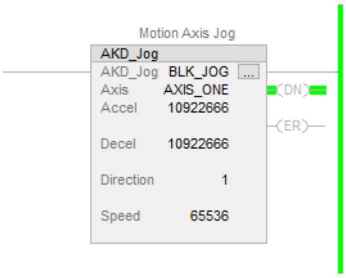
AXIS#.EIP.CMD.V → AXIS#.FBUS.VL.CMD

AXIS#.EIP.CMD.P → 0.

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#.FBUS.VL.CMD.

Jog Forward

Example Result: AKD_Jog (Forward) setup (AOI field entries)



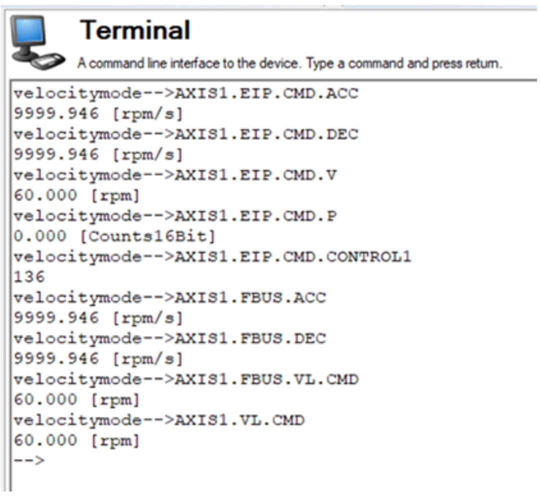
Result

NOTE

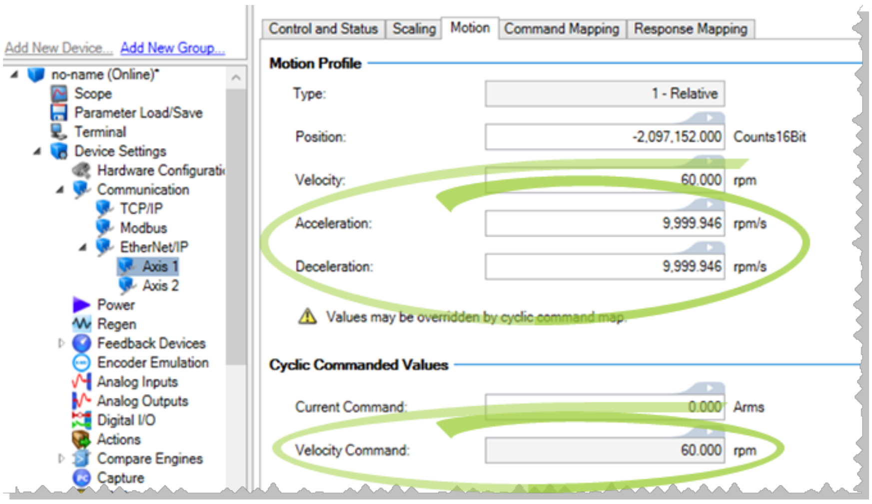
When in Velocity Mode, AXIS#.FBUS.V won't be utilized since it is only used in Position Mode.

From AXIS#.EIP.CMD.V to AXIS#.FBUS.VL.CMD is the Fieldbus command and is updated in Velocity Mode.

Note the Direction (bit) is used internally. In this example, AXIS1.EIP.CMD.V is a positive value and AXIS1.VL.CMD is a positive value. Also note AXIS#.VL.CMDU is not used with the Jog command.

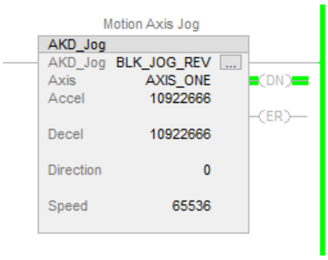


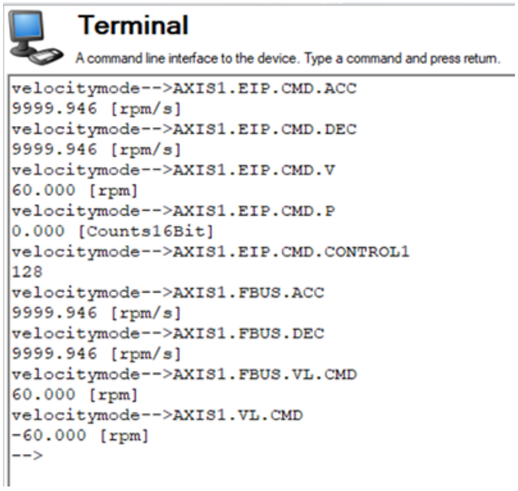
The AXIS#.FBUS Parameters can also be viewed under the Motion tab.



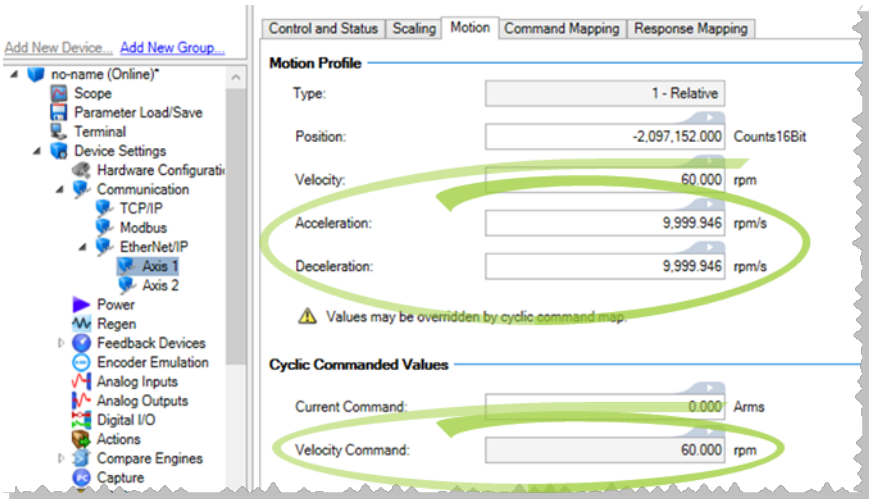
Jog Reverse

Example Result: AKD_Jog (Reverse) setup (AOI field entries)





The FBUS parameters can also be viewed under the Motion tab.



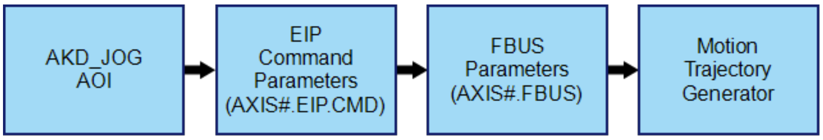
Note the Direction (bit) is used internally in the motion trajectory generator where in this example AXIS1.EIP.CMD.V is a positive value, but AXIS1.VL.CMD is negative. Also note AXIS#.VL.CMDU is not used with the Jog command.

Note the Speed value in the AKD_Jog AOI will accept a signed value.

NOTE

The best practice is to use the Direction bit for direction and to enter a positive value for the Speed in the AKD_Jog block. For example, if the Direction is set to 0 (Reverse) and the Speed is set to -X where X is the Target Velocity, the effect is the direction of the Jog will be positive. For this reason, using a positive Target Velocity for either direction and using the Direction bit is more consistent.

Position Mode



In Position operation mode the AXIS#.EIP Parameters 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.

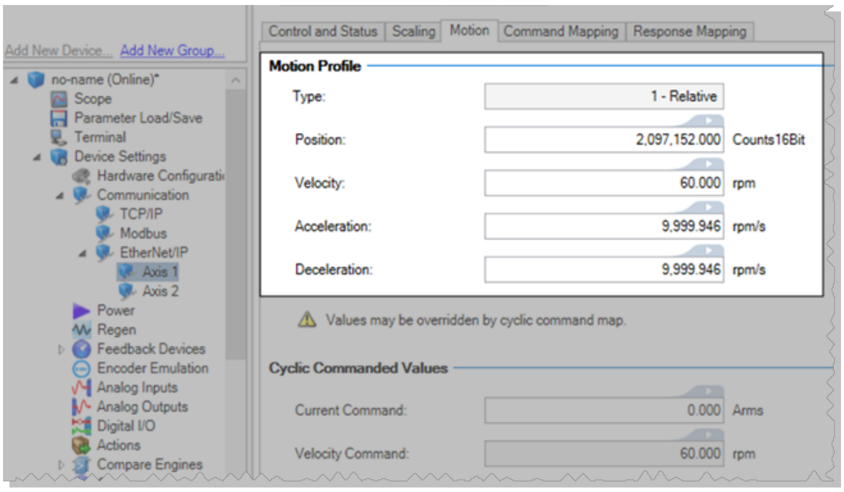
Jog Forward

The Jog Forward can be seen in the Terminal where the FBUS parameters are shown in WorkBench units and the EtherNet/IP CMD parameters. A key takeaway is there is no "P", Position passed from EtherNet/IP (AXIS#.EIP.CMD.P = 0) when executing the Jog Command Type, but the command is processed by the drive to set the AXIS#.FBUS.P to 2097152 counts in the forward direction and -2097152 counts in the reverse direction (WorkBench units were setup for 16 bit counts which is the same as EtherNet/IP units).

The Position value is set by the firmware using the formula below and the Direction bit in the AOI sets the sign of the Position.
The value of 2097152 is derived from 65536 EtherNet/IP counts/rev*32 revs.

$$65536\,(counts/revs)\,x\,32 = rev\,2,097,152\,counts$$

This is sufficiently long to avoid Motion Tasks errors when executing. Note these are the same values used in the AKD1G, but in that case Motion Task 0 (MT#0) was used to perform the Jog in Position Mode and in the AKD2G the AXIS#.FBUS Parameters are utilized and passed to the motion trajectory generator.



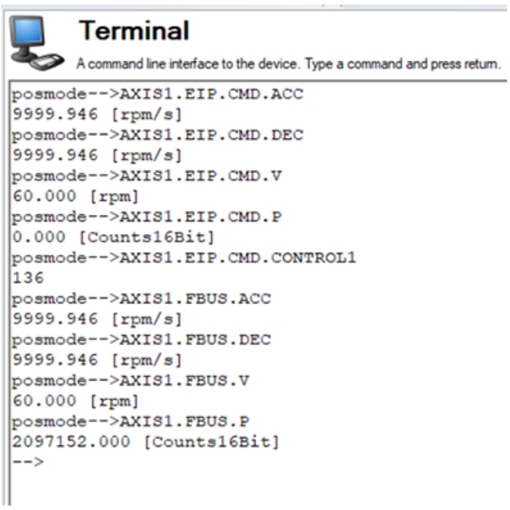
Jog Reverse

The Jog Reverse can be seen in the Terminal where the FBUS parameters are shown in WorkBench units and the EtherNet/IPCMD parameters. A key takeaway is there is no "P", position passed from EtherNet/IP (AXIS#.EIP.CMD.P = 0) when executing the Jog Command Type, but the command is processed by the drive to set the AXIS#.FBUS.P to +2097152 in the forward direction and -2097152 counts in the reverse direction (WorkBench units were setup for 16 bit counts which is the same as EtherNet/IP units).

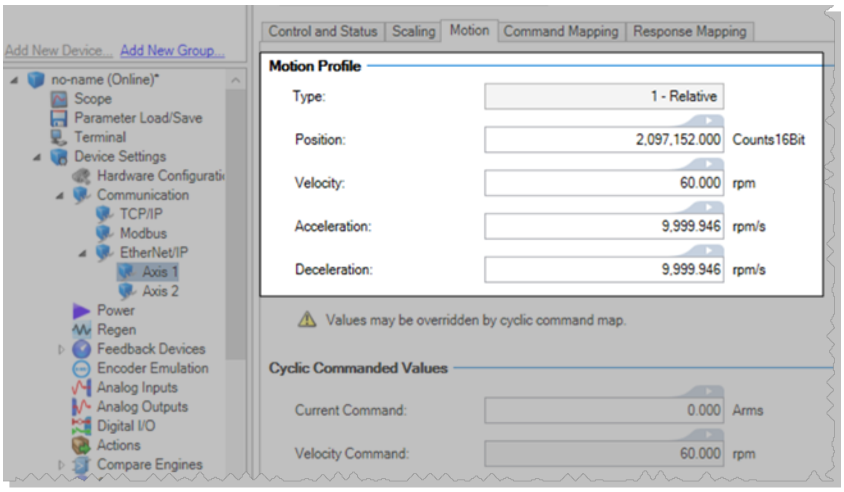
The value of 2097152 is derived from 65536 EtherNet/IP counts/rev*32 revolutions.

$$65536\,(counts/revs)\,x\,32 = rev\,2,097,152\,counts$$

This is sufficiently long to avoid Motion Task errors when executing. Note these are the same values used in the AKD1G, but in that case Motion Task 0 (MT#0) was used to perform the Jog in Position Mode and in the AKD2G the AXIS#.FBUS parameters are utilized and passed to the motion trajectory generator.



The FBUS parameters can also be viewed under the Motion tab.



Note the Speed value in the AKD_Jog AOI will accept a signed value.

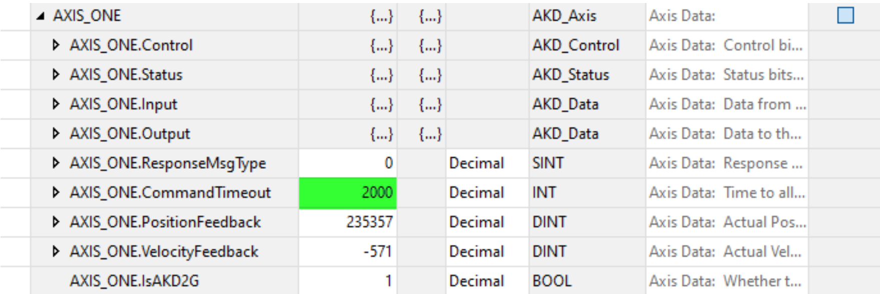
NOTE

The best practices is to use the Direction bit for direction and enter a positive value for the Speed in the AKD_Jog block. For example, if the Direction is set to 0 (Reverse) and the Speed is set to -X where X is the Target Velocity, the effect is the direction of the Jog will be positive. For this reason, using a positive Target Velocity for either direction and using the Direction bit is more consistent.

Execution

<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>

Unlatch the OS_Start_Sequence One Shot and reset the Command_Timeout timer.
Load CommandTimeout preset from the Axis.CommandTimeout and set the first 8 internal Command_Bytes 0-7 to values of zero.

Example: AXIS_ONE
In the Controller Tags under the AXIS_ONE structure is the AXIS_ONE.CommandTimeout. The Sample project sets this at 2000 ms (but can be changed by the user).

<table>
<tr><td>AXIS_ONE</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Axis</td><td>Axis Data:</td></tr>
<tr><td>AXIS_ONE.Control</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Control</td><td>Axis Data: Control bi...</td></tr>
<tr><td>AXIS_ONE.Status</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Status</td><td>Axis Data: Status bits...</td></tr>
<tr><td>AXIS_ONE.Input</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data from ...</td></tr>
<tr><td>AXIS_ONE.Output</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data to th...</td></tr>
<tr><td>AXIS_ONE.ResponseMsgType</td><td>0</td><td></td><td>Decimal</td><td>SINT</td><td>Axis Data: Response ...</td></tr>
<tr><td>AXIS_ONE.CommandTimeout</td><td>2000</td><td></td><td>Decimal</td><td>INT</td><td>Axis Data: Time to all...</td></tr>
<tr><td>AXIS_ONE.PositionFeedback</td><td>235357</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Pos...</td></tr>
<tr><td>AXIS_ONE.VelocityFeedback</td><td>-571</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Vel...</td></tr>
<tr><td>AXIS_ONE.IsAKD2G</td><td>1</td><td></td><td>Decimal</td><td>BOOL</td><td>Axis Data: Whether t...</td></tr>
</table>
</td></tr>
<tr><td>.EnableIn False</td><td>Unlatch the OS_Start_Sequence, reset Command_Timeout timer, and set the first 8 internal Command_Bytes 0-7 to values of zero.</td></tr>
</table>

Condition	Ladder Diagram Action
Instruction Execution	1. On .EnableIn, unlatch the .ER (Error) and .DN (Done) bits and set Step Number to 0. • If there is a fault, latch the .ER (Error) bit and set the Step Number to -5. • If the drive is not enabled, latch the .ER (Error) bit and set the Step Number to -6. 2. Use Direction (bit 3) to set in Control Word. 0 = Negative; 1 = Positive 3. Move 16#07 (Command Type 0x07 - Jog Move) into the Command Type Byte of the Command Assembly. 4. Write Speed, Accel, and Decel to the respective Bytes in the Command Assembly. **Load Complete:** If the Load Complete status (bit 7 in Status Word 2) is confirmed, then unlatch/clear the Load/Start bit (bit 0 in Control Word) in the Command Assembly. **Step Number 0:** • If the Step Number is 0 and the load is not complete, latch the Start Profile (Load/Start; bit 0 of the Control Word) and set the Step Number to 1. **Step Number 1:** • If the Step Number is 1 and there is an error (Response Type 0x14 - Command/Response Error), latch the .ER bit High and move the response data into the ErrorData registers for troubleshooting. Set the Step Number to -1. • If the Step Number is 1, and the Timeout preset is non-zero, then start the CommandTimeout timer. If the timer times out, set the .ER (Error) bit High and set the Step Number to -2. • If the Step Number is 1 and the Load Complete (bit 7 in Status Word 2) is True then set the .DN (Done) bit High and set the Step Number to 2. 5. If there is an error, unlatch the Load/Start bit (bit 0 in Control Word). 6. Copy the data from the Control Word and put it into the tempprary Command_Bytes and then take the Command_Bytes and put them into the actual Command Assembly (Output Data).

Changes to Axis Status Bits

- Current Direction (bit 4 in Status Word 1) indicates a 0 = Reverse or 1 = Forward
- The Profile_In_ Progress bit (bit 0 in Status Word 1; known as In Motion on AKD2G drives) indicates a 0 = No Motion or 1 = Motion In Progress.

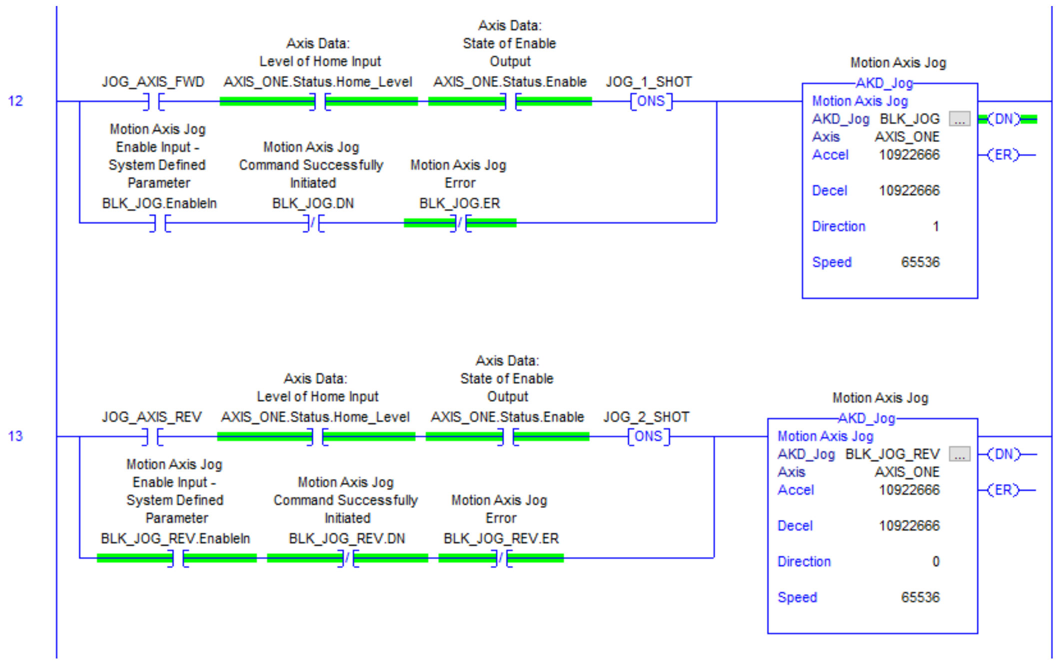
Example Of Usage/Programming Guidelines

As noted previously, the best practice is to use the Direction bit for direction and put a positive value for the Speed in the AKD_Jog AOI. The Speed entry will accept a signed value. For example, if the Direction is set to 0 (Reverse) and the Speed is set to -X where X is the Target Velocity, the effect is the direction of the Jog will be positive. For this reason, using a positive Target Velocity for either direction and using the Direction bit is more consistent.

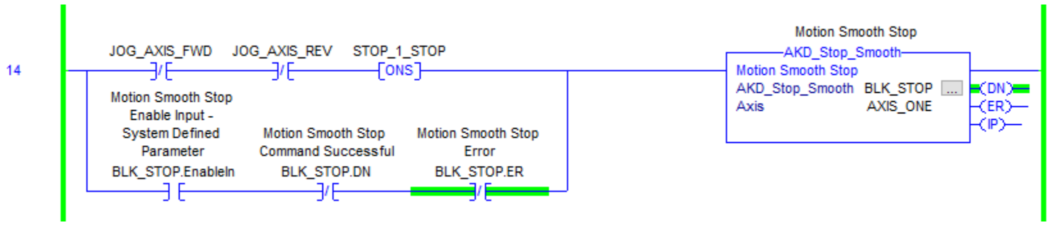
In Position Mode, even if the Velocity is a negative value in the AKD_Jog AOI only the magnitude is used (i.e. MT.V = ABS (Speed) and the Direction bit sets the direction.

From the Sample project, three rungs are used for the purpose of jogging. These three rungs work in either Velocity Mode or Position Mode. Velocity Mode does not require the axis to be Homed prior to execution, but Position Mode requires homing prior to execution. The Sample project assumes the drive is operating in Service Position Mode for an AKD1G axis or Fieldbus Position Mode for an AKD2G) axis.

In following example, rung 12 is set for Jog Forward and rung 13 is set for Jog Reverse. In each rung there are interlocks to check that the axis has been Homed and that it is Enabled. A N.O. Contact is used to trigger the AOI to execute (i.e. JOG_AXIS_FWD and JOG_AXIS_REV) the Jog command. The trigger uses a One Shot and then the parallel branches latches the AKD_Jog AOIs until the .DN (Done) bit turns on acknowledging Success to start execution or .ER (Error) when the request failed.



In the Sample project, rung 14 is set up where the two triggers (i.e. JOG_AXIS_FWD and JOG_AXIS_REV) on the falling edge will fire a One Shot and trigger the AKD_Stop_Smooth AOI. The parallel branch is used to seal-in the .EnableIn until either the .DN (Done) bit is set indicating success or the .ER (Error) is set indicating it failed. Success occurs when both the Smooth Stop bit is set and the Profile_In_Progress bit turns OFF. For more details see the AKD_Stop_Smooth section of this manual.



It is important to note AKD_Jog is a transitional instruction, meaning when the .EnableIn goes from False to True and the Jog command begins to execute, the Jog execution will continue even if the AKD_Jog AOI's .EnableIn goes False.

Troubleshooting

The AKD_Jog AOI can .ER (Error) due to the following :

1. The drive is faulted at the time the AKD_Jog AOI is triggered.
2. The drive is not enabled at the time the AKD_Jog AOI is triggered.
3. The Load/Start command was set, but the Response Message type was an error.
4. The Load/Start command was set, but Timeout timer expired.
5. DRV.CMDSOURCE (AKD1G) is not Service or AXIS#.CMDSOURCE (AKD2G) is not Fieldbus.
6. DRV.OPMODE (AKD1G) or AXIS#.OPMODE (AKD2G) is set to Torque (not Velocity or Position).
7. The axis is in Position Operation Mode, but the axis is not Homed.

AKD1G
(See Faults & Warnings F135 n135)

Id	Description
135	Warning: Homing is needed.

AKD2G
(See Faults & Warnings W6002)

Id	Description
6002	Warning: Homing is needed.

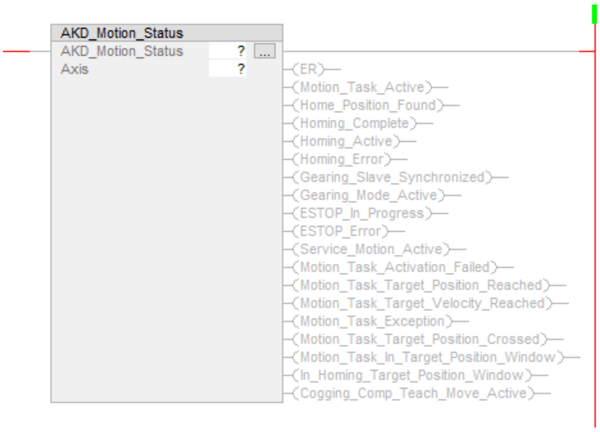
Step Summary

Step Number	Operation/Result
0	On .EnableIn reset the .DN and .ER bits of the AKD_Jog AOI.
1	Load/Start command bit is set in Control Word.
2	Load Complete from Status Word; set .DN (Done) bit.
-1	Load/Start command was set, but the Response Message type was an error. Set .ER (Error) bit.
-2	Step was Step Number 1 but command timed out. Set .ER (Error) bit.
-5	General Fault. Set .ER (Error) bit.
-6	Drive is disabled. Set .ER (Error) bit.

Revision History

Revision Number	Description/Notes	Date of Revision
v2.1	Initial release	02/23/2011
v2.3	• Removed unused .IP output • AOI library v5.0 revision	10/29/2014
v3.0	• Deleted unused AKD_Set_Decel, AKD_Set_Accel, and AKD_Set_Velocity in parameter declarations. Eliminated axis number bits which could be a source of confusion. • Deleted Command_Axis_Number and Response_Axis_Number which are unused now. • Update includes the AOI is synchronized with the new AKD_Data data type which the Axis structure uses. • AOI library v6.0 revision	06/02/2021

7.16 AKD_Motion_Status



Description

The AKD_Motion_Status AOI is intended for troubleshooting purposes and also makes the DRV.MOTIONSTAT data (bits) from the Communications and Controller Tags visible in the Ladder for monitoring.

Compatibility

The AKD_Motion_Status AOI is only compatible with the AKD1G drive. For AKD2G drives use the AKD2G_Motion_Status AOI.

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

The AKD_Motion_Status AOI is intended for troubleshooting purposes and also makes the DRV.MOTIONSTAT data (bits) from the Communications and Controller Tags visible in the Ladder for monitoring. When the AKD_Motion_Status AOIs .EnableIn is True, the status bits for the given axis are copied from the Response Assembly Data Structure (Bytes 16-19) to the outputs of the AKD_Motion_Status AOI.

ⓘIMPORTANT

It is important to check your drive's firmware revision and also AKD WorkBench Help for the description of DRV.MOTIONSTAT and the definition of each bit as they may be subject to change with firmware revision. The output names of the AOI are based on firmware 1-17-0-0. The implementation of the AOI should be an unconditional .EnableIn so the values are updated with every scan. See Example of Usage/Programming Practices for more details.

If the axis entered in the Axis field of the AOI is not an AKD1G (e.g., attempting to use an AKD2G drive axis) then the .ER (Error) bit on the output of the AKD_Motion_Status AOI will be set and the status outputs will all clear (turn OFF).

Operands

Operand	Type	Format	Description
AKD_ Motion_ Status	AKD_ Motion_ Status	Tag	Tag name for the given instance of the AOI in the Ladder.
Axis	AKD_Axis	Tag	Tag for Which Axis Declared; Must match the Axis_Internal Tag name of the AKD_Drive AOI for the given axis.

Structure

Mnemonic	Type	Format	Description	Read/Wwrite
.EnableIn	Input	BOOL	.The Enable Input bit indicates the instruction is Enabled.	Read Only
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.	Read Only
.ER	Output	BOOL	Turns ON if the axis is not an AKD1G drive.	Read Only
.Motion_Task_Active	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
.Home_Position_Found	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
.Homing_Complete	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
.Homing Active	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
.Homing_Error	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Gearing_Slave_ Synchronized	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Gearing_Mode_Active	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
ESTOP_In_Progress	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
ESTOP_Error	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Service_Motion_Active	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Motion_Task_ Activation_Failed	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Motion_Task_Target_ Position_Reached	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Motion_Task_Target_ Velocity_Reached	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only

Mnemonic	Type	Format	Description	Read/Wwrite
Motion_Task_Exception	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Motion_Task_Target_ Position_Crossed	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Motion_Task_In_ Target_Position_ Window	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
In_Homing_Target_ Position_Window	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only
Cogging_Comp_Teach_ Move_Active	Output	BOOL	See WorkBench Help and DRV.MOTIONSTAT for details.	Read Only

Execution

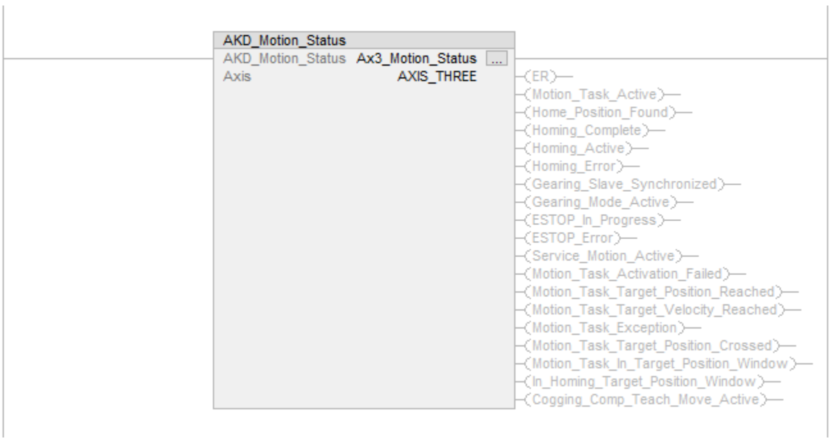
Condition	Ladder Diagram Action
Prescan	None
.EnableIn False	Output data of the AKD_Motion_Status AOI is not updated and retains the last value.
Instruction Execution	Output data of the AKD_Motion_Status AOI is updated on every scan.

Changes to Axis Status Bits

- None

Example of Usage/Programming Practices

In the example below, the AKD_Motion_Status AOI is added to a rung where the AOI's .EnableIn is tied unconditionally to the left rail. The AKD_Motion_Status entry is then given a unique Tag name for that instance and the Axis name is entered based on the Axis_Internal name of the AKD_Drive Add-On Instruction for the desired axis.



Troubleshooting

- The .ER (Error) bit turns ON if the Axis field entry of the AKD_Motion_Status AOI is not identified as an AKD1G drive.

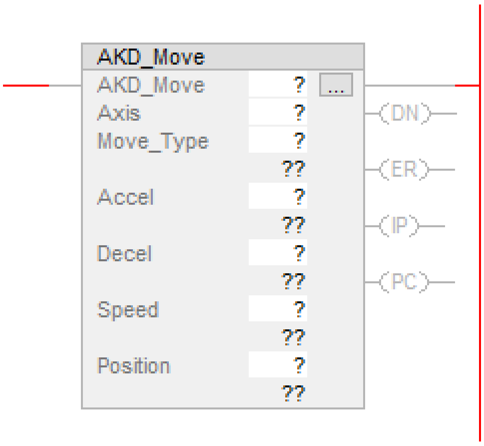
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	Initial release	08/14/2015
v2.0	• DRV.MOTIONSTAT upper status bits changed after FW 1.13. • Changed AOI output names to reflect changes and conform to FW 1.15. • Reordered Target Position Crossed and In Target Position Window on the outputs so they are displayed in the same order as DRV.MOTIONSTAT in WorkBench Help.	05/17/2017
v3.0	Added logic to flag an error if user attempts to use this AOI with an axis declared under an AKD2G_Drive AOI. This update includes synchronization with the new AKD_Data data type which is used by the Axis structure.	06/21/2021
v3.1	• Changed internal logic to eliminate unnecessary warnings when compiling the project. • AOI library v5.0 revision	09/20/2021

7.17 AKD_Move



Description

Use the AKD_Move instruction to move an axis by a specified Relative Distance or to a specified Absolute Position.

Compatibility

The AKD_Move AOI is compatible with both AKD1G and AKD2G drives.

AKD1G

DRV.CMDSOURCE = Service

DRV.OPMODE = Position

AKD2G

AXIS#.CMDSOURCE = Fieldbus

AXIS#.OPMODE = Position

Use the AKD_Move instruction to move an axis by a specified Relative Distance or to a specified Absolute Position. Note the Position entry of the AKD_Move will be the distance for a Relative Move type or Target Position for an Absolute Move. This selection is based on the Move_Type entry where 0 = Absolute and 1 = Relative.

In order to successfully execute this instruction the following conditions are required:

- The axis must be Enabled.
- The axis must be Homed.
- DRV.CMDSOURCE (AKD1G) must be Service and AXIS#.CMDSOURCE (AKD2G) must be Fieldbus.
- DRV.OPMODE (AKD1G) and AXIS#.OPMODE (AKD2G) must be Position.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Move	AKD_Move	Tag	Tag name for the given instance of the AOI in the Ladder.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Move_Type	SINT	Generally a Constant value but can be a Tag (variable)	Sets the Move Type for the move where: 0 = Absolute and 1 = Incremental (Known as Relative in the Control Word.)
Accel	DINT	Generally a Constant value but can be a Tag (variable)	Acceleration for the move in EtherNet/IP units
Decel	DINT	Generally a Constant value but can be a Tag (variable)	Deceleration for the move in EtherNet/IP units
Speed	DINT	Generally a Constant value but can be a Tag (variable)	Velocity for the move in EtherNet/IP units
Position	DINT	Generally a Constant value but can be a Tag (variable)	Distance for a Relative Move type and Target Position for an Absolute Move type. Entered in EtherNet/IP units.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	.The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the AOI execution (command to move) is successful. The Load_Complete from Status Word 2 (bit 7) indicates Success to initiate the Move.
.ER	Output	BOOL	The .ER bit is set when the AKD_Move .EnableIn is True and: • The axis is faulted • The axis is not Enabled • There is a Response Type 0x14 - Command/Response Error. This typically occurs when the Command Source and Op Mode are incorrect. • The Command Timeout timer times out. • The Profile_In_Progress is False and the On Target Position bit is False as a result of not being in the Target Position window. See the Troubleshooting section in this topic for more information.
.IP	Output	BOOL	The .IP output bit turns ON when the command to initiate the Move is successful and the Profile_In_Progress (bit 0 in Status Word 1; also known as In Motion for AKD2G drives) is ON (True). When the Profile_In_Progress (bit 0 in Status Word 1) transitions from ON to OFF indicating the trajectory is finished, the .IP bit turns OFF. If the Move is cancelled, the .IP output will turn OFF.
.PC	Output	BOOL	The .PC output bit is set when the Profile_In_Progress (bit 0 in Status Word 1) is False indicating the trajectory is finished and the On Target Position (bit 2 in Status Word 1) sourced from bit 11 of DRV.MOTIONSTAT) for (AKD1G) or AXIS#.MOTIONSTAT for AKD2G) is ON (True) indicating the axis is in the In Position window set by MT.TPOSWND (AKD1G) or AXIS#.SETTLE.P (AKD2G). The In Position window setting is application-dependent but too large of a value may result in the .PC bit turning on too early and too small of a value may result in the .PC bit flickering or never turning on at all.

AKD1G Execution

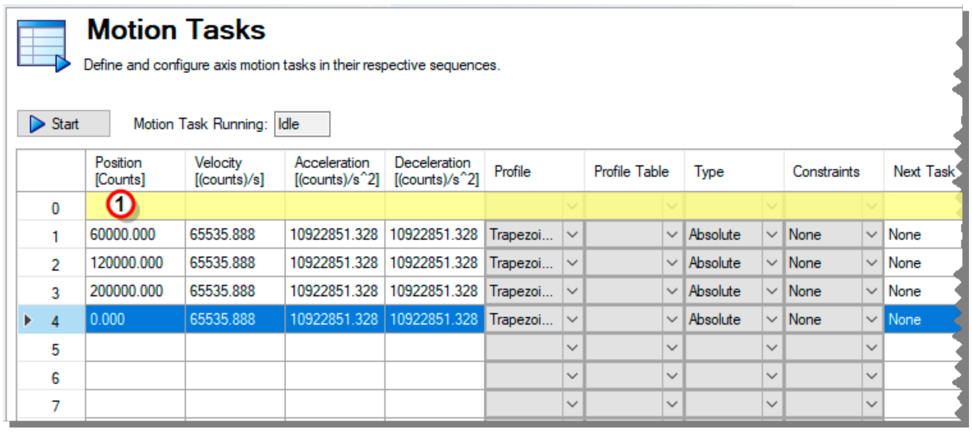
The AKD_Move AOI uses Command Type 0x06 - Position Move intermally.

Before triggering the AKD_Move AOI, the axis must meet the following conditions:

- The axis is not faulted.
- The axis is Enabled.
- The axis is in Service command source and Position Op Mode.
- Smooth Stop and Hard Stop are not active.
- Position Limits (both SW and HW) are not active.
- Axis is Homed.

Once all conditions have been met and the AKD_Move AOI is triggered:

- The Target Position, Velocity, Accel, Decel are loaded into the Command Assembly Bytes and the Incremental (Relative bit in the Control Word) is set or reset depending on the Move_Type prior to setting the Load/Start bit to initiate the Move. While the Position Move is executing, the In Process (.IP) bit turns ON and the Process Complete (.PC) bit will turn ON when the Target Position is reached. Use a Smooth Stop or Hard Stop to interrupt the AKD_Move.
- Motion Task 0 is used to load and execute any AKD_Move AOI instances and therefore should be reserved for the AKD_Move or AKD_Jog AOIs when also using preset Motion Tasks.

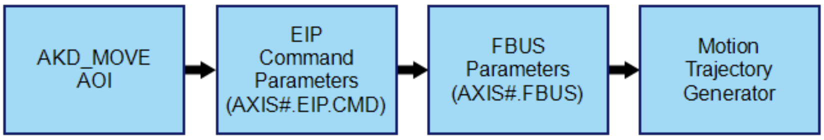


1. Motion Task 0 is reserved for AKD_Move or AKD_Jog AOI instructions.

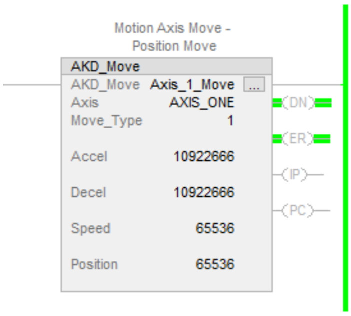
AKD2G Execution

The AKD2G Move Command Type differs in execution from the AKD1G in the following ways:

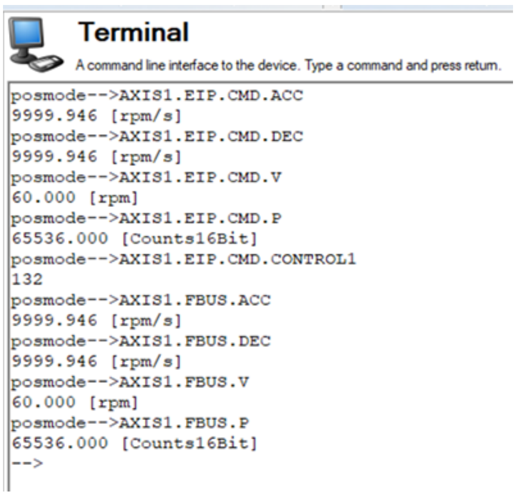
- The AKD2G axis' Command Source must be Fieldbus. AKD1G axis' are set to Service.
- The Move command does not utilize the Motion Task table in the AKD2G. Instead, it utilizes AXIS#.FBUS Parameters to pass the EtherNet/IP values and command to the Motion Trajectory Generator.



Example AKD_Move Setup

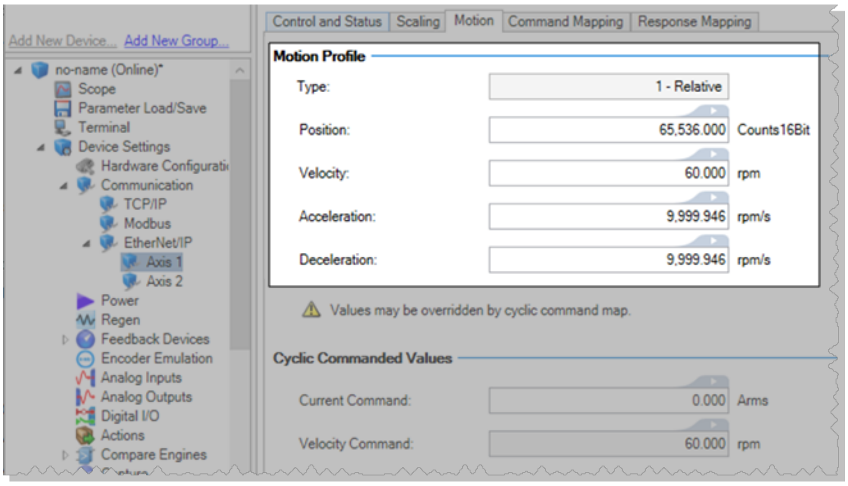


WorkBench Units for the given axis will affect the values displayed:




```
posmode-->AXIS1.EIP.CMD.ACC
9999.946 [rpm/s]
posmode-->AXIS1.EIP.CMD.DEC
9999.946 [rpm/s]
posmode-->AXIS1.EIP.CMD.V
60.000 [rpm]
posmode-->AXIS1.EIP.CMD.P
65536.000 [Counts16Bit]
posmode-->AXIS1.EIP.CMD.CONTROL1
132
posmode-->AXIS1.FBUS.ACC
9999.946 [rpm/s]
posmode-->AXIS1.FBUS.DEC
9999.946 [rpm/s]
posmode-->AXIS1.FBUS.V
60.000 [rpm]
posmode-->AXIS1.FBUS.P
65536.000 [Counts16Bit]
-->
```

The FBUS parameters may also be viewed under the Motion Tab.

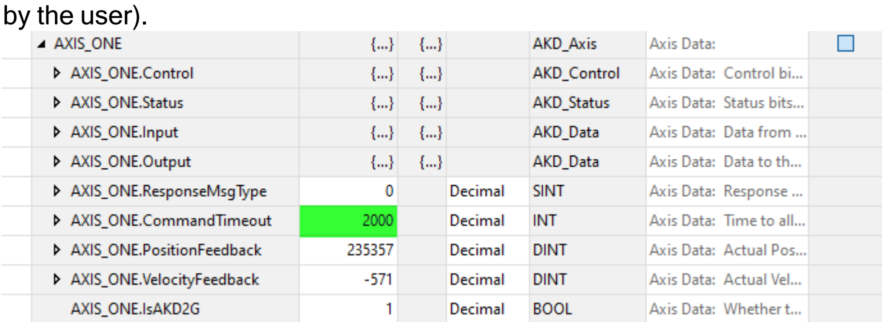


Execution

<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>

1. Unlatch the Start Sequence One Shot and reset the Command Timeout timer.
2. Initialize the Command Timeout preset into the Command Timeout timer preset value (*.PRE).
3. Initialize the internal Command_Bytes to 0.

Example: AXIS_ONE
In the Controller Tags under the AXIS_ONE structure is the AXIS_ONE.CommandTimeout. The Sample project sets this to 2000 ms (but it can be changed by the user).



<table>
<tr><td>AXIS_ONE</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Axis</td><td>Axis Data:</td></tr>
<tr><td>AXIS_ONE.Control</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Control</td><td>Axis Data: Control bi...</td></tr>
<tr><td>AXIS_ONE.Status</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Status</td><td>Axis Data: Status bits...</td></tr>
<tr><td>AXIS_ONE.Input</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data from ...</td></tr>
<tr><td>AXIS_ONE.Output</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data to th...</td></tr>
<tr><td>AXIS_ONE.ResponseMsgType</td><td>0</td><td></td><td>Decimal</td><td>SINT</td><td>Axis Data: Response ...</td></tr>
<tr><td>AXIS_ONE.CommandTimeout</td><td>2000</td><td></td><td>Decimal</td><td>INT</td><td>Axis Data: Time to all...</td></tr>
<tr><td>AXIS_ONE.PositionFeedback</td><td>235357</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Pos...</td></tr>
<tr><td>AXIS_ONE.VelocityFeedback</td><td>-571</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Vel...</td></tr>
<tr><td>AXIS_ONE.IsAKD2G</td><td>1</td><td></td><td>Decimal</td><td>BOOL</td><td>Axis Data: Whether t...</td></tr>
</table>

</td></tr>
<tr><td>.EnableIn False</td><td>

1. Unlatch the Start Sequence One Shot and reset the Command Timeout timer.
2. Clear the internal Command_Bytes.
3. If the profile is not In Progress (Status Word 1) then unlatch the .IP bit for the AOI.

</td></tr>
</table>

Condition	Ladder Diagram Action
Instruction Execution	1. On .EnableIn of the AOI, reset the .PC, .DN, and .ER bits of the AKD_Move AOI. 2. Set the Step Number to 0. If there is a General Fault (bit 3 in Status Word 1 or the drive is disabled, set the Error (.ER) bit and set the Step Number to -5. 3. Set Command Type (Byte 2 of the Command Assembly Data Structure) to Command Type 0x06 - Position Move. 4. Use the Move Type to set bit 2 of the Control Word (0 = Absolute; 1 = Incremental [Known as Relative in the Control Word.]) 5. Populate the Command Assembly Data Structure with the Move data for Position, Velocity, Acceleration, and Deceleration. **Step Number 0:** If at Step 0 and the Load Complete (bit 7 in Status Word 2) status is False, latch the Load/Start bit of the Control Word (Start Profile) and set the Step Number to 1. **Load Complete:** If the Load Complete status is True, unlatch the Load/Start bit (bit 0 in Control Word). **Step Number 1:** • If the Step Number is 1 (profile started) and the Response Assembly receives an Error Type 14 then set the Error (.ER) bit, copy the Response Assembly error data, and set the Step Number to -1. • If the Step Number is 1 (profile started) and the Command Timer times out, set the Error (.ER) bit and set the Step Number to -2. • If the Step Number is 1 (profile started) and the Load Complete from the Response Assembly is True then set the Done (.DN) bit and set the Step Number to 2. **Step Number 2:** • If the Step Number is 2 and the Profile_In_Progress is True (bit 0 in Status Word 1; also called In Motion) from the Response Assembly then set the In Process (.IP) bit. If the the Profile_In_Progress is False and not on the Target Position then set the .ER (Error) bit. • If the Step Number is 2 and the profile is not In Progress (bit 0 in Status Word 1; also called In Motion) and the On Target Position bit is set in the Response Assembly (bit 2 in Status Word 1), set the Process Complete (.PC) bit. • If the Step Number is 2 and the profile is not In Progress (bit 0 Status Word 1; also called In Motion) and the On Target Position bit is set in the response assembly (bit 2 in Status Word 1), set the Process Complete (.PC) bit. **Error (.ER):** If there is an Error (.ER bit is set) then unlatch the Load/Start bit 0 of the Control Word. 6. Copy the axis Control Word data to the internal Command_Bytes. 7. Copy the internal Command_Bytes 0-24 to the output data.

Example of Usage/Programming Guidelines

The AKD_Move AOI is intended for point-to-point moves.

- AKD1G uses Motion Task 0.
- AKD2G uses the internal move using AXIS#.FBUS Parameters.

The AKD_Move AOI is not capable of:

1. Continuously making updates to the trajectory on the fly.
2. Doing S-Curve Acceleration over the EtherNet/IP Communications. The only way to use S-Curve profiling in the AKD-P is to setup the S-Curve data table using WorkBench.
3. Making blended moves or registration moves over EtherNet/IP: It is possible to start a Motion Task using the block method. See the following article for details covering starting a chain of predefined Motion Tasks setup in WorkBench to make blended moves. It also covers using the registration application to start the initial move that indexes the material and looks for the registration input to trigger a registration move in the Motion Task.

   For more details see the application article: Can you do blended moves or registration over EtherNet/IP with the AKD?

   Also, refer to the AKD_Start_MotionTask, AKD_Set_Motion_Task , or AKD2G_Set_Motion_Task Add-On Instructions sections within this manual.

NOTE

The AKD_Move is a transitional instruction which starts execution on .EnableIn. While it is possible to start another AKD_Move AOI or turn OFF the .EnableIn of the given AKD_Move AOI and re-trigger it and update the Motion Task without stopping, best practices indicate the AKD_Move AOI was intended for point-to-point moves and not on the fly changes. It is up to the user to break the seal-in logic in the case the Move is aborted or stopped using the Smooth Stop, Hard Stop, etc.

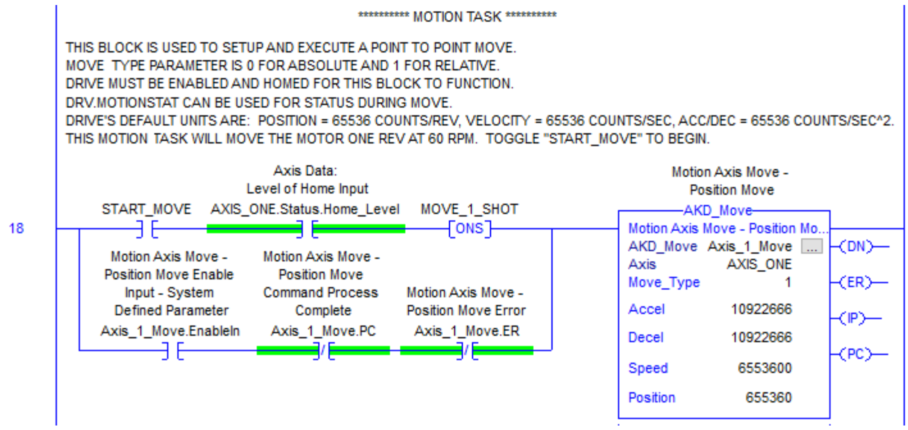
The AKD_Move AOI only supports Relative To Command Position Move (Incremental = 1; known as Relative in the Control Word) and Move Type Absolute (0).

From the Sample project:

A N.O. Contact (Start_Move) is used as a trigger to command the AKD_Move AOI to execute. There is an interlock which is the given axis' Home_Level from bit 5 in Status Word 1 of the Response Assembly to check that the axis has been Homed prior to allowing the AKD_Move AOI to execute. *The AOI will error without the axis being Homed first.* The trigger is sent as a One Shot and the parallel branch seals-in the .EnabledIin until either the .PC bit is set (Success and Move Complete) or .ER (Error, Execution Failed).

The sample Ladder rung shows:

- The AKD_Move entry is the name of the instance of the AKD_Move AOI
- The Axis entry is the name of the desired axis which is used as the Tag name for Axis_Internal for the AKD_Drive AOI (AKD1G) or for an AKD2G drive's Axis_Internal or Axis2_Internal entry.
- Move_Type is application-dependent and is entered as either 0 = Absolute or 1 = Relative.
- Accel and Decel are entered in EtherNet/IP counts/s^2.
- Speed is entered in EtherNet/IP counts/s.
- Position is entered in EtherNet/IP counts.
  See the section on Scaling for AKD1G or Scaling for AKD2G for more details on these units.



Troubleshooting

The AKD_Move AOI will Error (.ER) if any of the following conditions are true:

1. The axis is not in Position Operation Mode.
2. The command source is not Service (AKD1G) or Fieldbus (AKD2G)
3. The axis is faulted (General Fault, bit 3 in Status Word 1 is ON or 1).
4. The axis is not Enabled or ready to move (i.e. Software Enable or Hardware Enable disabled, STO is off, etc.).
5. Axis is not Homed (bit 5 in Status Word 1 is OFF or 0).
6. The Smooth Stop bit of the Control Word is ON and the Axis.CommandTimeout in the Controller Tags is set to a non-zero value. The .ER bit will be set based on Command Timeout in this case. If the Axis.CommandTimeout is zero the AKD_Move will not make the Move and the .EnableIn bit will stay latched until otherwise unlatched.
7. The Hard Stop bit of the Control Word is ON and the Axis.CommandTimeout in the Controller Tags is set to a non-zero value. The .ER bit will be set based on Command Timeout in this case. If the Axis.CommandTimeout is zero then the AKD_Move will not make the Move, but the .EnableIn bit will stay latched until otherwise unlatched.
8. There is a Response Error.
9. There is a Command Timeout.
10. The Profile_In_Progress bit is False and the On Target Position is False.

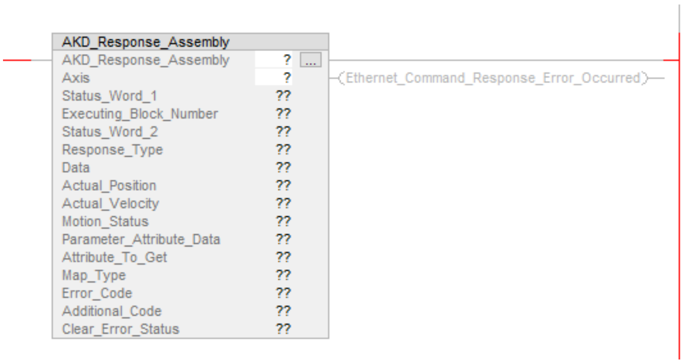
Step Summary

Step Number	Operation/Result
0	On .EnableIn reset the .PC, .DN and .ER bits of the AKD_Move AOI.
1	Load/Start command bit is set in Control Word.
2	Load Complete from Status Word 2; set Done (.DN) bit. If profile is In Process then set .IP bit. If profile is not In Process and On Target Position is True, set the .PC bit.
-1	Step was Step 1 but Response Message Type was an error. Set Error (.ER) bit.
-2	Step was Step 1 but command timed out. Set Error (.ER) bit.
-5	General Fault or Axis Not Enabled. Set Error (.ER) bit.

Revision History

Revision Number	Description/Notes	Date of Revision
v2.0	Initial release	02/12/2011
v3.1	• Doesn't error if command timeout preset is 0 • Added error code to AKD_Move AOI. When motion was stopped with Smooth Motion (i.e. AKD_Stop_Smooth), the AKD_Move AOI would hang up.	02/16/2015
v3.2	• Wouldn't restart if other reason it was stopped. • AOI v5.0 revision	06/23/2017
v4.0	• Eliminated axis number bits which could be a source of confusion. • Deleted Command_Axis_Number and Response_Axis_Number since they are now unused. • Deleted unused AKD_Set_Position, AKD_Set_Velocity, AKD_Set_Decel and AKD_Set_Accel in parameter declarations which were not being used. • Initial Beta to accommodate AKD2G EtherNet/IP	06/02/2021
v4.1	• Changed the copy of internal Command_Bytes length from 33 to 24 to prevent overlap into the AKD2G's dynamic map. • This update also synchronizes the AOI with the new AKD_Data data type which is used by the Axis structure. • AOI v6.0 revision	07/27/2021

7.18 AKD_Response_Assembly



Description

The AKD_Response_Assembly AOI can be used in the Ladder program to allow the static mapping of the given axis' Response Assembly to be monitored in the Ladder during runtime. This can also be useful for troubleshooting.

Compatibility

The AKD_Response_Assembly is compatible with both AKD1G and AKD2G.

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The AKD_Response_Assembly AOI can be used in the Ladder program to allow the static mapping of the given axis' Response Assembly Data Structure to be monitored in the Ladder during runtime. This can also be useful for troubleshooting. Without the AOI, the user must access the Bytes directly in the Controller Tags for the given axis. The AKD_Response_Assembly AOI copies the values on each scan and populates the values in the declared Tags used in that instance of the AKD_Response_Assembly AOI. *The values are all Read-Only and the Tag values cannot be changed.*

If Axis Is Associated With	Displayed Map_Type	
AKD1G drive	0, 1, or 2 depending on configuration and usage	AKD1G EtherNet/IP Response Assembly Data Structure Mappings
AKD2G drive	Always 0.	AKD2G EtherNet/IP Response Assembly Data Structure Mappings

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_ Response_ Assembly	AKD_ Response_ Assembly	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_ Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

The following fields are not entered by the user and are populated automatically with Read-Only data once the Operands above are entered.

TIP

See the AKD1G EtherNet/IP Response Assembly Data Structure or AKD2G Response Assembly Data Structure for more information.

Mnemonic	Type	Format	Description
Status_Word_1	Output	SINT	Displays value and monitors Byte 0 (AKD1G) or Byte 0 (AKD2G Axis 1) or Byte 64 (AKD2G Axis 2) of the Response Assembly; displayed in the AOI as a binary value.
Executing_Block_ Number	Output	SINT	Displays value and monitors Byte 1 (AKD1G) or Byte 1 (AKD2G Axis 1) or Byte 65 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a decimal value.
Status_Word_2	Output	SINT	Displays value and monitors Byte 2 (AKD1G) or Byte 2 (AKD2G Axis 1) or Byte 66 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a binary value.
Response Type	Output	SINT	Displays value and monitors Byte 3 (AKD1G) or Byte 3 (AKD2G Axis 1) or Byte 67 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a hex value.
Data	Output	DINT	Displays value and monitors Byte 4-7 (AKD1G) or Byte 4-7 (AKD2G Axis 1) or Byte 68-71 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a decimal value.
Actual_Position	Output	DINT	Displays value and monitors Bytes 8-11 (AKD1G) or Bytes 8-11 (AKD2G Axis 1) or Byte 72-75 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a decimal value.
Velocity	Output	DINT	Displays value and monitors Bytes 12-15 (AKD1G) or Bytes 12-15 (AKD2G Axis 1) or Byte 76-79 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a decimal value.
Motion_Status	Output	DINT	Displays value and monitors Bytes 16-19 (AKD1G) or Bytes 16-19 (AKD2G Axis 1) or Bytes 80-83 (AKD2G Axis 2) of Response Assembly. Displayed in the AOI as a binary value.

Mnemonic	Type	Format	Description
Parameter_ Attribute_Data	Output	DINT	Displays value and monitors Bytes 24-27 (AKD1G) or Bytes 24-27 (AKD2G Axis 1) or Bytes 88-91 (AKD2G Axis 2) of Response Assembly; displayed in the AOI as a decimal value.
Attribute_To_Get	Output	SINT	Display the value and monitor Byte 32 AKD1G or Byte 28 (AKD2G Axis 1) or Byte 92 (AKD2G Axis 2) of Response Assembly. Displayed in the AOI as a hex value.
Map_Type	Output	SINT	Displays value and monitors Byte 33 (AKD1G) of the Response Assembly. If AKD2G then the Map_Type value will always be 0. Displayed in the AOI as a decimal value.
Error_Code	Output	SINT	Displays value and monitors Byte 4 (AKD1G) or Byte 4 (AKD2G Axis 1) or Byte 68 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a hex value.
Additional_Code	Output	SINT	Display the value and monitor Byte 5 (AKD1G) or Byte 5 (AKD2G Axis 1) or Byte 69 (AKD2G Axis 2) of the Response Assembly. Displayed in the AOI as a hex value.
Clear_Error_ Status	Input	BOOL	Clears the captured Error_Code and Additional_Code on the rising edge. Requires user programming to utilize the Reset.
Ethernet_ Command_ Response_Error_ Occurred	Output	BOOL	Indicates whether a response error to a command is present or not. The output is latched on .ER (Error) and reset when the user entry Clear_Error_Status transitions from 0 to 1.

Execution

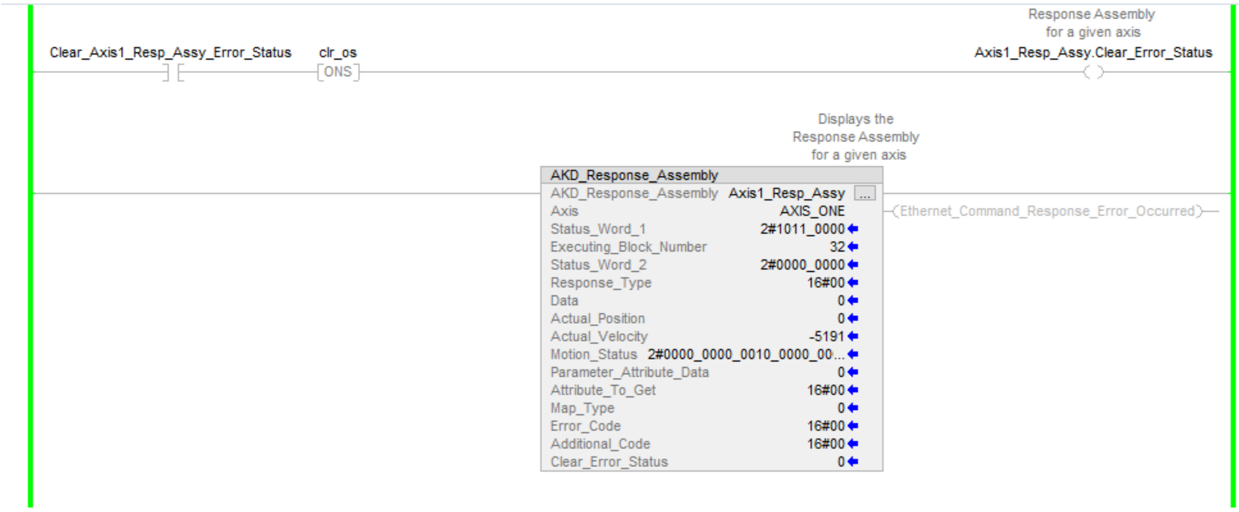
Condition	Ladder Diagram Action
Prescan	Not applicable.
.EnableIn False	Not applicable. While rung is False data in the AOI fields (outputs) are not updated and are frozen in their last state at the time when the rung went False.
Instruction Execution	• While the rung is True, the data in the AOI fields (outputs) are updated with axis data on every PLC scan. • RPI scan time may affect how often the data is updated. • If the Response Type in Byte 3 of the AKD1G Response Assembly Data Structure or the AKD2G Response Assembly Data Structure is a 16#14 and the error code is not zero then the Error_Code and Additional_Code are displayed (the values are captured on the rising edge of the error condition). When the Clear_Error_Status is transitioned from 0 to 1 by the user programming the Error_Code, Additional_Code, and output Ethernet_ Command_Response_Error_Occurred are all reset.

Changes to Axis Status Bits

- None

Example of Usage/Programming Guidelines

The AKD_Response_Assembly AOI is always Enabled in the Ladder program and is executed on every scan. The Axis field entry must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis. All other fields (outputs) in the AOI will populate automatically and it is not necessary to declare Tags for them. To clear the Error Status the Clear_Error_Status attribute of the instance of AKD_Response_Assembly is toggled (shown as a N.O. Contact as the trigger and a One Shot to a coil assigned to the attribute).



Troubleshooting

- None

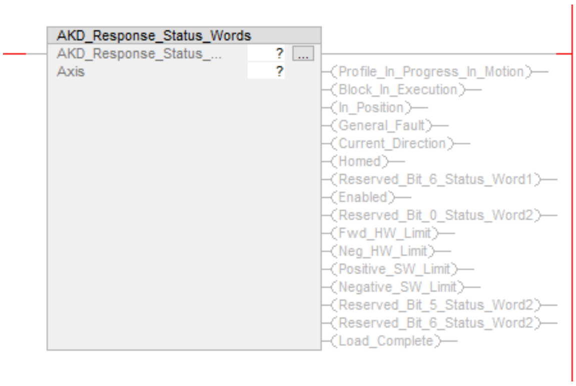
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	Initial release	08/13/2015
v2.0	• Changed conditions on Ethernet_Command_Response_Error_Present so only when type 16#14 is present will the status be reported (i.e. not for Response Type 16#5 Torque for example). • Added ability to clear the error status (Error_Code and Additional_Code). This will also clear the output bit Ethernet_Command_Response_Error_Occurred. • AOI library v5.0 revision	01/30/2019
v3.0	• Added logic based on whether AKD_Drive or AKD2G_Drive AOI is being used for the given axis. If AKD then Map Type in Byte 33 is shown; if AKD2G then Map Type is always displayed as zero. • This update also synchronizes the AOI with the new AKD_Data data type which is used by the Axis structure.	06/02/2021
v3.1	• Added code to account for the different Byte location for Attribute to Get in the AKD1G vs. AKD2G Response Assemblies. • AOI library v6.0 revision	07/21/2021

7.19 AKD_Response_Status_Words



Description

The intent of the AKD_Response_Status_Words AOI is to allow the status bits of the given axis' Response Assembly's Status Word 1 and Status Word 2 to be viewed in the Ladder for the purpose of montoring and troubleshooting.

Compatibility

The AKD_Response_Status_Words AOI is compatible with both AKD1G and AKD2G.

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

The intent of the AKD_Response_Status_Words AOI is to bring the status bits of the given axis' Response Assembly's Status Word 1 and Status Word 2 out to the Ladder for the purpose of montoring and troubleshooting. Without the AOI the user must access the Bytes (and bits) directly in the Controller Tags for the given axis. The AKD_Response_Status_Words AOI copies the values of the Status Words for the given axis each scan and populates the output values of the AOI accordingly.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_ Response_ Status_ Words	AKD_ Response_ Status_ Words	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_ Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

The following fields are populated automatically with Read Only data once the Operands above are entered.

Mnemonic	Type	Format	Description
Profile_In_Progress_In_Motion	Output	BOOL	Bit 0 in Status Word 1
Block_In_Execution	Output	BOOL	Bit 1 in Status Word 1
In_Position	Output	BOOL	Bit 2 in Status Word 1
General_Fault	Output	BOOL	Bit 3 in Status Word 1
Current_Direction	Output	BOOL	Bit 4 in Status Word 1
Homed	Output	BOOL	Bit 5 in Status Word 1
Reserved_Bit_6_Status_Word1	Output	BOOL	Bit 6 in Status Word 1
Enable	Output	BOOL	Bit 7 in Status Word 1
Reserved_Bit_0_Status_Word2	Output	BOOL	Bit 0 in Status Word 2
Fwd_HW_Limit	Output	BOOL	Bit 1 in Status Word 2
Neg_HW_Limit	Output	BOOL	Bit 2 in Status Word 2
Positive_SW_Limit	Output	BOOL	Bit 3 in Status Word 2
Negative_SW_Limit	Output	BOOL	Bit 4 in Status Word 2
Reserved_Bit_5_Status_Word2	Output	BOOL	Bit 5 in Status Word 2
Reserved_Bit_6_Status_Word2	Output	BOOL	Bit 6 in Status Word 2
Load_Complete	Output	BOOL	Bit 7 in Status Word 2

Execution

Condition	Ladder Diagram Action
Prescan	Not applicable.
.EnableIn False	Not applicable. While rung is False data in the AOI fields (outputs) are not updated and are frozen in their last state at the time when the rung went False.
Instruction Execution	While the rung is True, the data in the AOI fields (outputs) are updated with Axis data on every PLC scan. RPI scan time may affect how often the data is updated.

Changes to Axis Status Bits

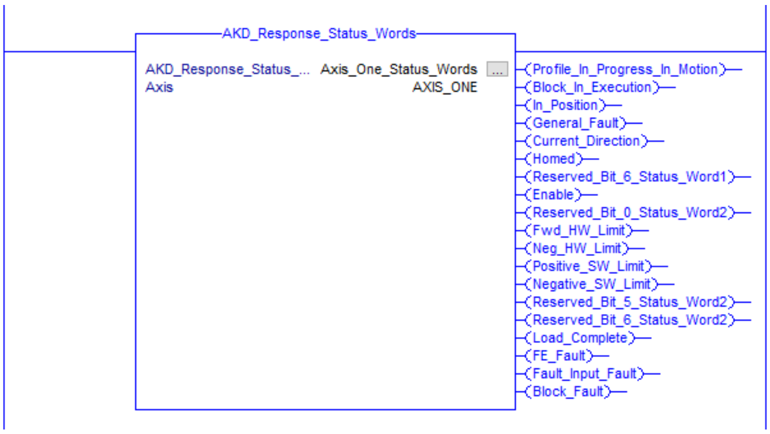
- None

Example of Usage/Programming Guidelines

The AKD_Response_Status_Words AOI is always Enabled in the Ladder program and is executed on every scan.

The AKD_Response_Status_Words field must have a declared Tag for that instance and the Axis must be the same Tag for the given axis as the Axis_Internal entry in the AKD_Drive for AKD1G or Axis_ Internal or Axis2_Internal in the AKD2G_Drive Add-On Instruction for that axis.

All other fields (outputs) in the AOI will populate automatically and it is not necessary to declare Tags for these outputs.



Troubleshooting

- None

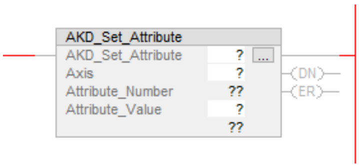
Step Summary

- None

Revision History

Revision Number	Description/Notes	Date of Revision
v1.0	Initial release	08/13/2015
v2.0	AOI library v5.0 revision	01/30/2018
v3.0	• This update asynchronizes the AOI with the new AKD_Data data type which is used by the Axis structure. • AOI library v6.0 revision	07/16/2021

7.20 AKD_Set_Attribute



Description

Use the AKD_Set_Attribute AOI to set a value to a Position Controller Attribute for an axis.

Compatibility

The AKD_Set_Attribute AOI is compatible with both the AKD1G and AKD2G drives.

Use the AKD_Set_Attribute AOI to set a value to a Position Controller Attribute for an axis. (See AKD1G EtherNet/IP Objects and Attributes or AKD2G EtherNet/IP Objects and Attributes)

Note that the Position Controller Object List of attributes is limited. For a method to access a wider range of parameters consider using the AKD_Set_Parameter AOI to write to a parameter. (See AKD1G Parameter Listing or AKD2G EtherNet/IP Objects List)

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

AKD2G

Required AXIS#.CMDSOURCE = ANY/ALL

Required AXIS#.OPMODE = ANY/ALL

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Set_Attribute	AKD_Set_Attribute	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Attribute_Number	INT	Constant	Constant for Attribute_Number to be set. (See AKD1G EtherNet/IP Objects and Attributes or AKD2G EtherNet/IP Objects and Attributes)
Attribute_Value	DINT	Tag or Constant	Value to be written to the given Attribute_Number.

Structure

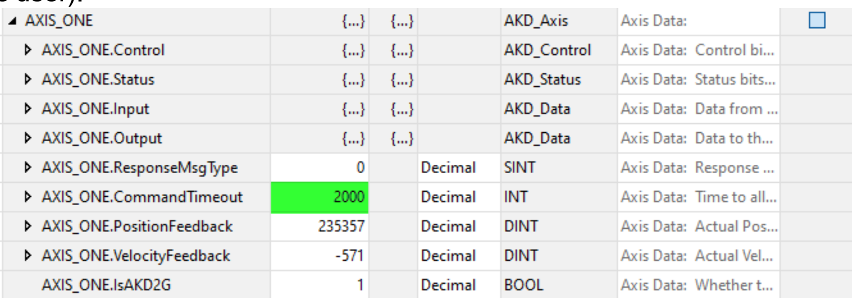
The following fields are populated automatically with Read Only data once the Operands above are entered.

Mnemonic	Type	Description
.EnableIn	BOOL	.EnableIn indicates the Add-On Instruction is enabled and is ON as long as the rung remains True.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	BOOL	The .DN (Done) output bit is based on the Load Complete (bit 7 in Status Word 2) turning ON (True) indicating a successful write to the attribute.
.ER	BOOL	The .ER (Error) bit indicates the attempt to set the value failed. See Troubleshooting for possible reasons.

Execution

<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>

Reset Load/Start One Shot and Command Timeout timer.
Set the timeout timer preset and zero out Bytes 0-7 of the internal Command_ Bytes.

Example: AXIS_ONE
In the Controller Tags under the AXIS_ONE structure is the AXIS_ ONE.CommandTimeout. The Sample project sets this at 2000 ms (can be changed by the user).

<table>
<tr><td>AXIS_ONE</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Axis</td><td>Axis Data:</td></tr>
<tr><td>AXIS_ONE.Control</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Control</td><td>Axis Data: Control bi...</td></tr>
<tr><td>AXIS_ONE.Status</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Status</td><td>Axis Data: Status bits...</td></tr>
<tr><td>AXIS_ONE.Input</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data from ...</td></tr>
<tr><td>AXIS_ONE.Output</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data to th...</td></tr>
<tr><td>AXIS_ONE.ResponseMsgType</td><td>0</td><td></td><td>Decimal</td><td>SINT</td><td>Axis Data: Response ...</td></tr>
<tr><td>AXIS_ONE.CommandTimeout</td><td>2000</td><td></td><td>Decimal</td><td>INT</td><td>Axis Data: Time to all...</td></tr>
<tr><td>AXIS_ONE.PositionFeedback</td><td>235357</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Pos...</td></tr>
<tr><td>AXIS_ONE.VelocityFeedback</td><td>-571</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Vel...</td></tr>
<tr><td>AXIS_ONE.IsAKD2G</td><td>1</td><td></td><td>Decimal</td><td>BOOL</td><td>Axis Data: Whether t...</td></tr>
</table>
</td></tr>
<tr><td>Enable In False</td><td>

Reset the Load/Start One Shot and Command Timeout timer.
Zero out Bytes 0-7 of the internal Command_Bytes.

</td></tr>
</table>

Condition	Ladder Diagram Action
Logic	1. On .EnableIn, set Step Number to 0 and unlatch the .DN and .ER bits of the AKD_Set_Attribute AOI. 2. Set Byte 2 of the internal Command_Bytes to Command Type 0x1B - Set Attribute of Position Controller Object. 3. Set the internal Command_Bytes 4 and 5 to the Attribute_Number. 4. Copy the Attribute_Value to the axis structure's output data (Bytes 24-27: 4 Bytes of data). 5. On Load Complete from the Status Word 2 of the Response Assembly, unlatch the Load/Start bit of theControl Word. If the Step Number is: **Step Number 0:** If the Load Complete is not True, latch the Load/Start bit of the Control Word and set the Step Number to 1. **Step Number 1:** • If the Response Type is 16#14 (Response Type 0x14 - Command/Response Error), there is an EtherNet/IP error. Set the Error (.ER) bit and set the Step Number to -1. • If the Command Timeout timer times out, set the error (.ER) bit and set the Step Number to -2. • If the Load Complete bit is set in Status Word 2 of the Response Assembly, then declare the operation Successful and set the Done (.DN) bit and set the Step Number to 2. **Error (.ER):** • If the Error (.ER) bit is set for any reason, unlatch the Load/Start bit of the Control Word. • Copy the Control Word data to internal Command_Byte 0, then copy Command_Bytes 0-7 to the output data of the axis structure's output data.

Changes to Axis Status Bits

- None

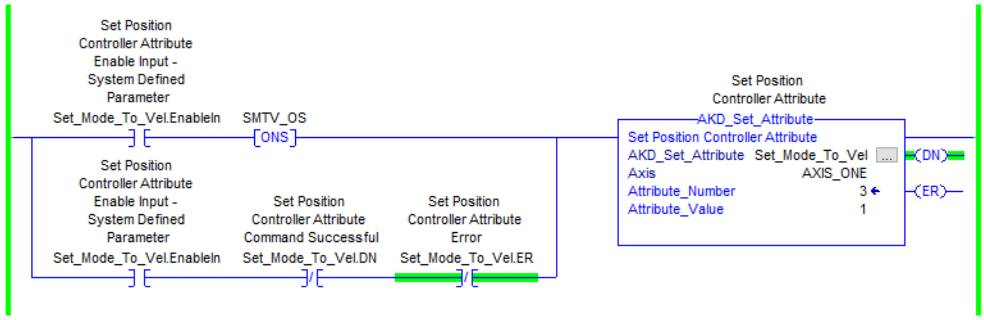
Example of Usage/Programming Guidelines

A N.O. Contact is used as a trigger to command the AKD_Set_Attribute AOI. A One Shot is used on the .EnableIn and a parallel branch is used as a seal-in to keep the AOI Enabled until either .DN (Done; Success) or 2) .ER (Error; Failure).

In this example the AKD_Set_Attribute entry is a unique Tag name for this instance and the Axis entry is for the desired axis (the same name as Axis_Internal in the given axis' AKD_Drive AOI or Axis_Internal or Axis2_Internal of the AKD2G_Drive AOI.

The Attribute_Number is found in AKD1G EtherNet/IP Objects and Attributes where Attribute ID 3 = Mode (DRV.OPMODE - AKD1G and AXIS#.OPMODE - AKD2G) and Attribute_Value is 1 (Velocity). See AKD2G EtherNet/IP Objects List for AKD2G.

The following example sets the Mode to Velocity as indicated in the description of Mode in either AKD1G EtherNet/IP Objects and Attributes or AKD2G EtherNet/IP Objects and Attributes. Note the following is for the AKD1G: The Attribute IDs are the same for the AKD2G; however, the Position Controller Object is 0x25 for AKD1G and 0x66 for the AKD2G.



Troubleshooting

Possible reasons for error condition:

- More than one AOI is Enabled or executing at the same time.
- The attempt to set the attribute failed (i.e. communications error or value out of range, etc.) See Response Type 0x14 - Command/Response Error.
- The attempt to set the attribute took too long and the command timed out.

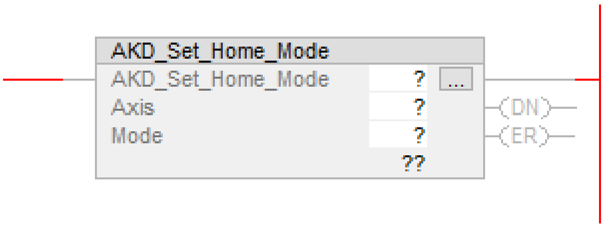
Step Summary

Step Number	Operation/Result
0	On Enable set Step Number to 0 and clear .DN and .ER bits.
1	Load/Start bit of the Control Word latched.
2	Load Complete. Success. .DN bit set.
-1	Load/Start command was set, but Response Message Type was an error. Set Error (.ER) bit.
-2	Step was Step 1, but command timed out. Set error (.ER) bit.

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
	Doesn't error if Command_timeout preset is 0	10/29/2014
v2.1	• Fixed COP to Parameter_Value. • Copying 16 Bytes worth of data but only 4 needed • AOI library v5.0 revision	03/05/2015
v3.0	• Eliminated axis number bits which could cause confusion. • Deleted Command_Axis_Number since it is now unused. • This update also synchronizes the AOI with the new data type AKD_Data which the Axis structure uses. • AOI library v6.0 revision	07/20/2021

7.21 AKD_Set_Home_Mode



Description

Use the AKD_Set_Home_Mode AOI to set or change the Homing Mode used by the axis when the AKD_Home command is called. The instruction may take multiple scans to execute due to communications transmission time and command execution time.

Compatibility

The AKD_Set_Home_Mode AOI is compatible with both AKD1G and AKD2G EtherNet/IP drives.

Required Command Source and Operation Mode

The AKD_Set_Home_Mode AOI changes the value of the following parameters regardless of the current DRV.CMDSOURCE or DRV.OPMODE (AKD1G) or AXIS#.CMDSOURCE or AXIS#.OPMODE (AKD2G):

AKD1G

HOME.MODE

AKD2G

AXIS#.HOME.MODE

Once the AKD_Set_Home AOI has selected a Home Mode, the AKD_Home AOI is used to start the Home Move. The Axis must be in Service Position Mode (AKD1G) or Fieldbus Position (AKD2G) to execute the AKD_Home (Home Move).

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Set_Home_Mode	AKD_Set_Home_Mode	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Mode	SINT	Tag or Constant	Value is the Home Mode Number to be set. See WorkBench Help for HOME.MODE (AKD1G) or AXIS#.HOME.MODE (AKD2G) for charts detailing the Home Mode Number and Homing Type description.

Structure

Mnemonic	Type	Description
.EnableIn	BOOL	.EnableIn indicates the Add-On Instruction is enabled and is ON as long as the rung remains True.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	BOOL	The Done (.DN) bit indicates the Home Mode was successfully set by the internal AKD_Set_Attribute AOI write to Attribute 100 (Home Mode).
.ER	BOOL	The Error (.ER) bit indicates the attempt to set the Home Mode failed when the internal AKD_Set_Attribute attempted to write to Attribute100 (Home Mode). See Troubleshooting for possible reasons.

Execution

The AKD_Set_Home_Mode AOI utilizes the Set Controller Attribute method internally to set the Home Mode using either HOME.MODE (AKD1G) or AXIS#.HOME.MODE (AKD2G) over EtherNet/IP.

This AOI sets the Position Controller Object using Command Type 0x1B - Set Attribute of Position Controller Object.
Home Mode has an Attribute ID of 100. (See Position Controller Object 0x25 (AKD1G) or Position Controller Object 0x66 (AKD2G) for more information.)

See WorkBench Help for values to set.

AKD1G = HOME.MODE

AKD2G = AXIS#.HOME.MODE

Execution

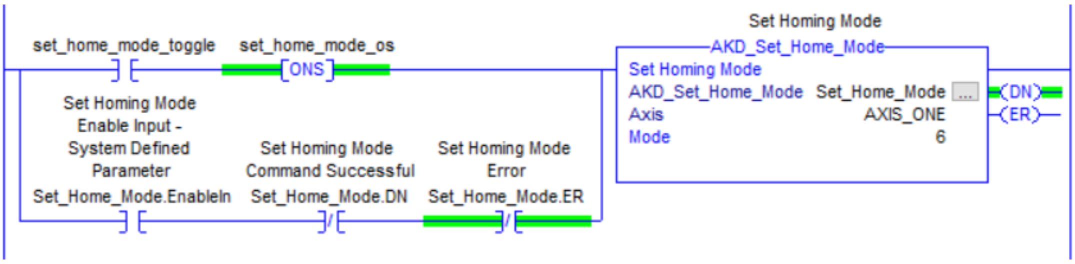
Condition	Ladder Diagram Action
Prescan	1. Set the Step Number to 0. 2. Set the Attribute Number of the internal AKD_Set_Attribute AOI to 100 (Home Mode)
.EnableIn False	Set the Step Number to 0.
Instruction Execution	**Step Number 0:** If the Step Number is 0, reset the Done (.DN) and Error (.ER) bits of the AKD_Set_Home_Mode AOI and then set the Step Number to 1. **Step Number 1:** If the Step Number is 1, execute the internal AKD_Set_Attribute AOI and write the Mode value to the Home Mode attribute 100. • If successful, set the Done (.DN) bit and set the Step Number to 2. • If unsuccessful, set the Error (.ER) bit and set the Step Number to -1.

Changes to Axis Status Bits

- None

Example of Usage/Programming Guidelines

In this example, a N.O. Contact set_home_mode_toggle is used to trigger the request to set the Homing Mode. A One Shot is used on the AKD_Set_Home_Mode's .EnableIn. A parallel branch is used to seal-in the .EnableIn of the AOI until either Done (.DN; Success) or Error (.ER; Failure). The AKD_Set_Home_Mode field is given a Tag name for that instance and the Axis entry is the same as AKD_Drive's Axis_Internal Tag name for the desired axis (AKD1G) or the AKD2G_Drive AOI's Axis_Internal (Axis1) or Axis2_Internal (Axis2) Tag name for the desired axis. The Mode entry is given a Constant value of the desired HOME.MODE (AKD1G). This can also be a Tag and variable.



Troubleshooting

Possible reasons the .ER bit was set:

- More than one AOI is Enabled or executing at the same time.
- The attempt to set the Home Mode (attribute) failed (i.e. communications error or value out of range, etc.)
- The Attribute_Number setting occurs on the prescan. It may be required to stop the program (Program Mode) and then restart (Run Mode) after adding the AKD_Set_Home_Mode to your Ladder.

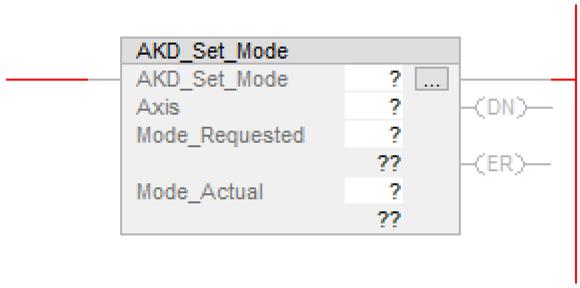
Step Summary

Step Number	Operation/Result
0	Set Step Number to 1 and clear Done (.EN) and Error (.ER) bits.
1	Set attribute value of Attribute 100 (Home Mode)
-1	Set attribute block failed. Set Error (.ER) bit.

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
v2.0	AOI library v5.0 revision	10/29/2014
v3.0	• Eliminated axis number bits which could cause confusion. • Deleted Command_Axis_Number since it is now unused. This update also synchronizes the AOI with the new data type AKD_Data which the Axis structure uses. • AOI library v6.0 revision	07/20/2021

7.22 AKD_Set_Mode



Description

The AKD_Set_Mode AOI sets either the DRV.OPMODE (AKD1G) or AXIS#.OPMODE (AKD2G) to Position, Velocity, or Torque.

Compatibility

The AKD_Set_Mode AOI is compatible with both AKD1G and AKD2G EtherNet/IP drives.

Required Command Source and Operation Mode

AKD1G

DRV.CMDSOURCE = Any/All

DRV.OPMODE = Any/All

AKD2G

AXIS#.CMDSOURCE = Any/All

AXIS#.OPMODE = Any/All

The AKD_Set_Mode AOI sets either the DRV.OPMODE (AKD1G) or AXIS#.OPMODE (AKD2G) to Position, Velocity, or Torque.

Use the AKD_Set_Mode AOI to set the Operation Mode for the axis. Note the instruction may take multiple scans to execute due to communications transmission time and command execution time. The Done (.DN) bit is not set until after the Mode is set successfully.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Set_Mode	AKD_Set_Mode	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Mode_Requested	SINT	Tag or Constant	0 = Position 1 = Velocity 2 = Torque Note these values do not match the AKD display or the axis Op Mode number in WorkBench, but they are required in order to set the Op Mode over EtherNet/IP.
Mode_Actual	SINT	Tag	On successful execution of the AKD_Set_Mode AOI, the actual Mode is read back from the drive and displayed in the user defined Tag in the Mode_Actual field of the AOI.

Structure

Mnemonic	Type	Description
.EnableIn	BOOL	.EnableIn indicates the Add-On Instruction is Enabled and is ON as long as the rung remains True.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	BOOL	The .DN bit indicates the Set Mode was successfully set by the internal AKD_Set_Attribute write to Attribute 3 (Mode) and that it was also successfully read back using the internal AKD_Get_Attribute AOI.
.ER	BOOL	The .ER bit indicates the attempt to set and/or get Mode using the internal AKD_Set_Attribute AOI and AKD_Get_Attribute AOIs failed. See Troubleshooting for possible reasons.

Execution

The AKD_Set_Mode AOI uses the AKD_Set_Attribute method internally to set the Position Controller's Attribute ID 3 (Mode). Note the Position Controller Object class for AKD1G is 0x25 and the class for AKD2G is 0x66.

Position Controller Object 0x25 - AKD1G

Attribute ID (Decimal Value)	Name	Access Rule	Type	Description
3	Mode	Get/Set	USINT	Operating mode. 0 = Position Mode (default), 1 = Velocity Mode, 2 = Torque Mode.

Position Controller Object 0x66 - AKD2G

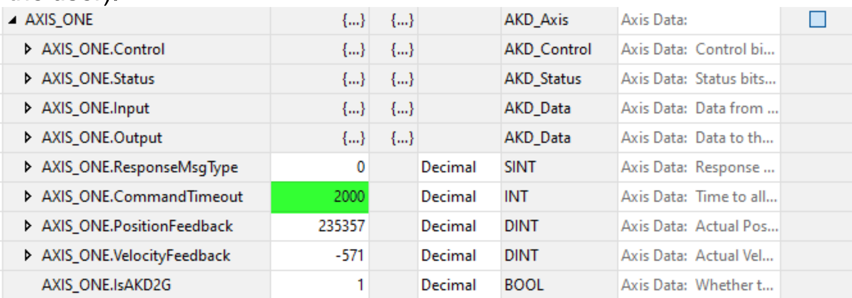
Attribute ID (Decimal Value)	Name	Access Rule	Type	Description
3	Mode	Get/Set	USINT	Operating mode. 0 = Position Mode (default), 1 = Velocity Mode, 2 = Torque Mode. Values get converted to AXIS#.OPMODE.

Note the values sent using this method over EtherNet/IP use values dictated by the EtherNet/IP model for the Position Controller Object and do not follow the same convention as the AKD1G or AKD2G drive in WorkBench or on the display of the AKD1G drive.

WorkBench Values		
AKD1G	**AKD2G**	**Mode**
DRV.OPMODE	AXIS#.OPMODE	0 = Torque
DRV.OPMODE	AXIS#.OPMODE	1 = Velocity
DRV.OPMODE	AXIS#.OPMODE	2 = Position

Execution

<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>
1. Reset One Shot and Command Timeout timer.
2. Set the preset of the Command Timeout timer and set the Step Number to 0.

Example: AXIS_ONE
In the Controller Tags under the AXIS_ONE structure is the AXIS_ ONE.CommandTimeout. The Sample project sets this at 2000 ms (but it can be changed by the user).

<table>
<tr><td>AXIS_ONE</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Axis</td><td>Axis Data:</td></tr>
<tr><td>AXIS_ONE.Control</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Control</td><td>Axis Data: Control bi...</td></tr>
<tr><td>AXIS_ONE.Status</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Status</td><td>Axis Data: Status bits...</td></tr>
<tr><td>AXIS_ONE.Input</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data from ...</td></tr>
<tr><td>AXIS_ONE.Output</td><td>{...}</td><td>{...}</td><td></td><td>AKD_Data</td><td>Axis Data: Data to th...</td></tr>
<tr><td>AXIS_ONE.ResponseMsgType</td><td>0</td><td></td><td>Decimal</td><td>SINT</td><td>Axis Data: Response ...</td></tr>
<tr><td>AXIS_ONE.CommandTimeout</td><td>2000</td><td></td><td>Decimal</td><td>INT</td><td>Axis Data: Time to all...</td></tr>
<tr><td>AXIS_ONE.PositionFeedback</td><td>235357</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Pos...</td></tr>
<tr><td>AXIS_ONE.VelocityFeedback</td><td>-571</td><td></td><td>Decimal</td><td>DINT</td><td>Axis Data: Actual Vel...</td></tr>
<tr><td>AXIS_ONE.IsAKD2G</td><td>1</td><td></td><td>Decimal</td><td>BOOL</td><td>Axis Data: Whether t...</td></tr>
</table>
</td></tr>
<tr><td>.EnableIn False</td><td>
1. Reset One Shot and Command Timeout timer.
2. Set the Step Number to 0.
</td></tr>
<tr><td>Instruction Execution</td><td>
On Enable: Reset the .DN bits of the internal AKD_Set_Attribute and AKD_Get_ Attribute (Set_Mode and Get_Mode respectively). Then reset the .DN and .ER bits of the AKD_Set_Mode AOI and set the Step Number to 1.

Step Number 1: Set the value of Attribute 3 (Mode) to the value from the Mode_ Requested entry. If successful, set the Done (.DN) bit and set the Step Number to 2. If unsuccessful then set the Error (.ER) bit and set the Step Number to -1.

Step Number2:
• Mode Change/Set Attribute was successful. Next, Read/Get the Mode. If the Read/Get is successful, check that the Mode_Request = Mode_Actual and set the Step Number to 3. If the Read/Get is unsuccessful, set the Error (.ER) bit and set the Error Source to -2.
• If the Mode/Change was successful but the Step Number does not move on (i.e. step 3) in the allotted time and the Command Timeout timer expires, set theError (.ER) bit and set the Error Source to -3.

Step Number 3:
• Both the Set and Get of the Mode succeeded, therefore declare the AOI done and set the Done (.DN) bit.

Error (.ER) bit is Set:
• If for any reason the internal AKD_Set_Attribute or AKD_ Get_Attribute failed and the Error (.ER) bit is Set, then set the Step Number to -1. See Troubleshooting for more information.
</td></tr>
</table>

Changes to Axis Status Bits

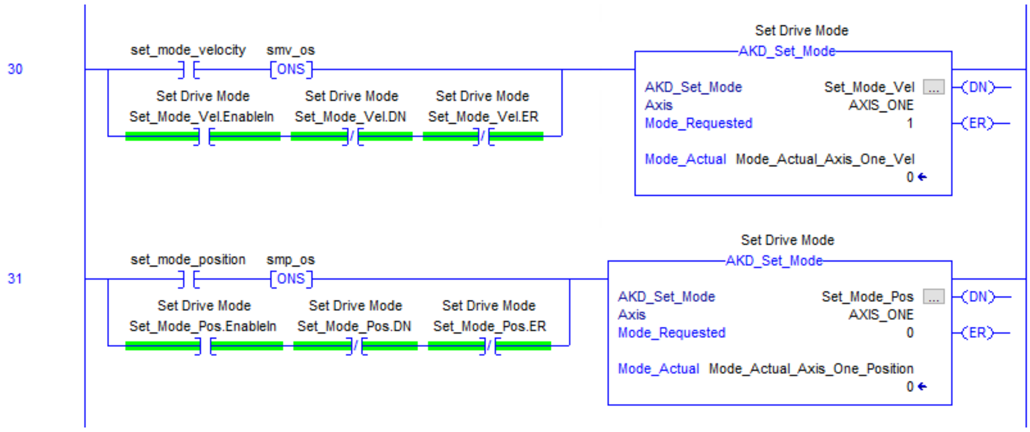
- None

Example Usage/Best Programming Practice:

The following example demonstrates switching between two operation modes (Velocity and Position) using two instances of the AKD_Set_Mode AOI.

The first rung uses a Set_Mode_Velocity N.O. Contact as a toggle to trigger the request to Change/Set the operation mode to Velocity. A One Shot is used and the parallel branch seals in execution of the AOI using the .EnableIn (N.O.), the .DN/Done (N.C.), and the .ER/error (N.C.) such that the Add-On Instruction will remain enabled until either .DN (Success) or .ER (Error/Fail). The Mode_Requested value is 1 per the Operands table.

The second rung follows the same method, but the AOI gets another unique instance name and the Mode_Requested is per the Operands table.The toggle and One Shot get new Tag names and the parallel branch gets the .EnableIn, .DN, and .ER associated with that instance of AKD_Set_Mode AOI.



NOTE

If the same Tag is used for Mode_Actual in both AOI instances above, the result observed is the value is always zero. In order to see the change when triggered, use a unique Tag for Mode_Actual for each block. Alternatively, the Actual Mode could be queried after the AKD_Set_Mode is Done (.DN) using an AKD_Get_Attribute AOI.

Troubleshooting

Possible reasons for error condition:

- More than one AOI is enabled/executing at the same time.
- Attempt to Set the attribute failed (i.e. communications error or value out of range, etc.)
- Attempt to Get the attribute failed (i.e. communications error).
- Attempt to Get and verify/compare requested vs. actual timed out

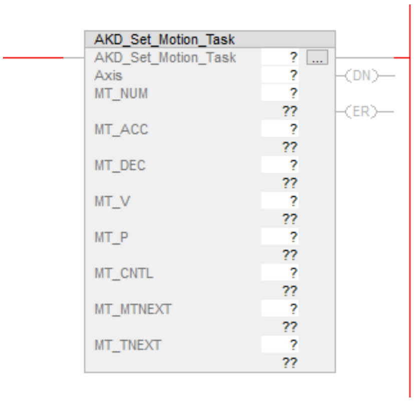
Step Summary

Step Number	Operation/Result
0	Set Step Number to 1 and clear .DN and .ER bits of the internal AOIs and the AKD_Set_ Mode AOI.
1	Set Attribute value of Attribute 3 (Mode).
2	Set mode was successful. Get mode.
3	Get mode was successful and Mode_Requested is equal to Mode_Actual.
Error Source	**Operation/Result**
-1	Set Attribute AOI failed. Set Error (.ER) bit.
-2	Get Attribute AOI failed.
-3	Attempt to Get and verify/compare Mode_Requested vs. Mode_Actual timed out

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
v2.2	Now able to recover from errors when they are enabled a 2nd time. Previously, the error would not clear.	03/05/2015
v4.0	• Removed race condition with GET_MODE and SET_MODE Done (.DN) bit (These are names for internal AKD_Get_Attribute and AKD_Set_Attribute AOIs) • AOI library v5.0 revision	04/28/2016
v5.0	• Updated to synchronize with new AKD_Data data type which the Axis structure uses. • AOI library v6.0 revision	07/16/2021

7.23 AKD_Set_Motion_Task



Description

The AKD_Set_Motion_Task AOI allows the programmer to create or modify Motion Tasks in the AKD from the PLC.

Compatibility

The AKD_Set_Motion_Task AOI can only be used with the AKD EtherNet/IP drive. Use the AKD2G_Set_Motion_Task AOI with AKD2G EtherNet/IP drives.

Required Command Source and Operation Mode

AKD1G

Required DRV.CMDSOURCE = ANY/ALL

Required DRV.OPMODE = ANY/ALL

NOTE

The AKD_Set_Motion_Task AOI can execute in any DRV.CMDSOURCE or DRV.OPMODE and the Motion Task will only be created or changed if the MT parameters are valid. The AKD_Set_Motion_Task AOI does not intiate a move. In order to start a Motion Task in the Motion Task table, use the AKD_Start_MotionTask AOI. Before starting the Motion Task, the AKD drive must be in Service Position Mode and the drive must also be Homed. See MT Parameters and Commands for a list of MT Parameters.

The AKD_Set_Motion_Task AOI allows the programmer to create or modify Motion Tasks in the AKD from the PLC. The operands includes important MT (Motion Tasking) group keyword members such as MT.P, MT.ACC, etc. The MT.NUM entry serves as a pointer to select which row of the Motion Tasking table the values will be written to. The AKD_Set_Motion_Task is different than the AKD_Move AOI in that:

1. The AKD_Set_Motion_Task AOI, when successfully executed, creates the Motion Task (any non-zero MT.NUM) but does not start a Motion Task (i.e. MT.MOVE).The AKD_Move AOI on execution always writes or overwrites the values in Motion Task 0 and then starts Motion Task 0. It is recommended to reserve Motion Task 0 for the AKD_Move or AKD_Jog (in Position Operation Mode) and use Motion Tasks 1 or higher with the AKD_Set_Motion_Task AOI.
2. The AKD_Set_Motion_Task is intended to be used with the AKD_Start_MotionTask AOI. As stated above, the AKD_Set_Motion_Task allows the Motion Task(s) to be created while the AKD_Start_MotionTask AOI starts the specified Motion Task. See AKD_Start_MotionTask for more details.
3. The AKD_Set_Motion_Task AOI sets each Motion Task parameter (i.e. MT.NUM , MT.P, MT.V, etc.) one by one, requiring multiple scans and thus a longer execution time. The AKD_Set_Motion_Task AOI creates the Motion Tasks prior to execution by the AKD_Start_MotionTask AOI(s).
4. The AKD_Set_Motion_Task AOI provides a way to set MT.CNTL, a bit-wise control word which allows additional move types and advanced motion selection such as blended mves, etc. in addition

to specifying a Relative or Absolute Move. The AKD_Move method only makes use of Relative or Absolute type moves with no other options.

The AKD_Set_Motion_Task AOI makes use of the Write Parameter method internally and uses many iterations to achieve all of the writes required to create the Motion Task.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Set_Motion_Task	AKD_Set_Motion_Task	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI for the given axis.
MT_NUM	DINT	Constant or Tag	Constant or Tag for MT.NUM.
MT_ACC	DINT	Constant or Tag	Constant or Tag for MT.ACC.
MT_DEC	DINT	Constant or Tag	Constant or Tag for MT.DEC.
MT_V	DINT	Constant or Tag	Constant or Tag for MT.V.
MT_P	DINT	Constant or Tag	Constant or Tag for MT.P.
MT_CNTL	DINT	Constant or Tag	Constant or Tag for MT.CNTL. See WorkBench Help for the bitwise descriptions of the MT.CNTL Control Word parameter.
MT_MTNEXT	DINT	Constant or Tag	Constant or Tag for MT.MTNEXT which sets the following Motion Task to execute in a chain of moves. This is displayed as Next Task in WorkBench.
MT_TNEXT	DINT	Constant or Tag	Constant or Tag for MT.TNEXT which sets the time delay before the following Motion Task executes after the previous Motion Task that calls it is finished. This is displayed as Dwell Time in WorkBench.

Structure

The following fields are automatically populated with Read Only data once the Operands above are entered.

Mnemonic	Type	Description
.EnableIn	BOOL	The Enable Input bit indicates the instruction is Enabled.
Enable Out	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	BOOL	The .DN (Done) bit indicates the Motion Task was successfully Set.
.ER	BOOL	The .ER (Error) bit indicates the attempt to set the Motion Task has failed or the axis was identified as an AKD2G drive.. See Troubleshooting: Summary of Reasons for Errors/Failure below for possible reasons.

Execution

Condition	Ladder Diagram Action
Prescan	Unlatch the First_Scan_Bit and set the Step Number to 0.
.EnableIn False	Unlatch the First_Scan_Bit when not enabled.

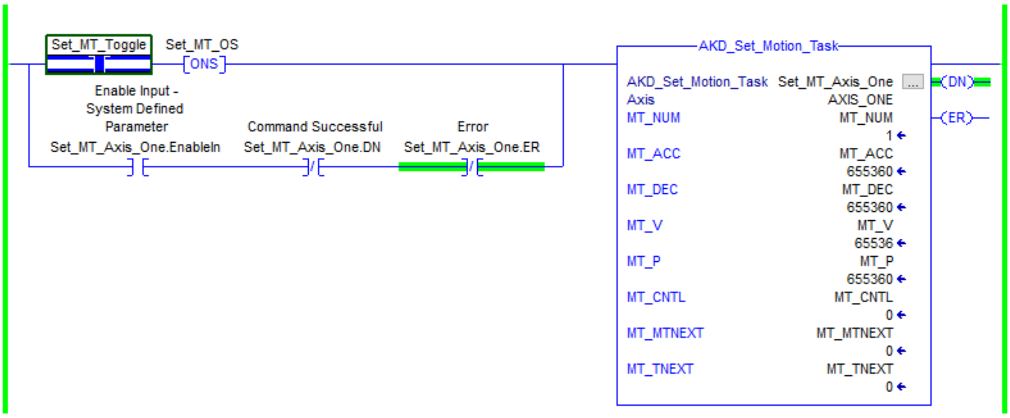
Condition	Ladder Diagram Action
Instruction Execution	On first scan: • Unlatch the .DN and .ER bits. • Unlatch the Load/Start bit of the Control Word. • Reset the Command_Timeout timer. • Set the Command_Timeout preset to 5000 ms. • Set the Step Number to 1. • Latch the First_Scan_Bit so this happens only once during execution. If the Step Number is: **Step Number 1:** Move a 16#1F (Command Type 0x1F - Read or Write Parameter Value) into the Command_Bytes (Byte 2 of the axis output data). Set the Step Number to 2. **Step Number 2:** Set then set bit 6 of Byte 2 to a 1. Set the Step Number to 3. **Step Number 3:** Set Byte 6 to a 1 (Set Parameter) and then set the Step Number to 4. **Step Number 4:** Set the Instance Number to 275 (MT.NUM) and move this number into Command_Bytes 4 and 5 (of the axis output data) per the Write Parameter method. Set the Step Number to 5. **Step Number 5:** Copy the MT.NUM value (input parameter of the AOI) to the data of Bytes 24-27 of the axis output data per the Write Parameter method. Set the Step Number to 6. **Step Number 6:** Unlatch the Load/Start bit of the Control Word and set the Step Number to 7. **Step Number 7:** If the Step Number is 7 and the Load Complete of the axis Status Word is cleared, then the AOI is ready for a new command. Latch the Load/Start bit of the Control Word and set the Step Number to 8. **Step Number 8:** If the Step Number is 8 and the Load Complete bit of the Status word is High it signifies the command completed successfully. Set the Step Number to 9. **Rungs 9 to 48:** Repeat of the method for MT.NUM. The order of writing is MT.ACC, MT.DEC, MT.V, MT.P, MT.CNTRL, MT.TNEXT, MT.MTNEXT, and MT.SET. However, on MT.SET on rung 48, the Done (.DN) bit of the AOI is set. **Response Type 16#14:** If the Load/Start bit of the Control Word is ON and the Load Complete bit (bit 7 of Status Word 2) is OFF/not complete and the Response Type is 16#14 (Response Type 0x14 - Command/Response Error) then there was an Error. On Error, unlatch the Load/Start bit of the Control Word and latch the .ER (Error) bit of the AOI. Set the Step Number to -1. **Command Timeout:** If the Load/Start bit of the Control Word is ON and the Load Complete bit (bit 7 of Status Word 2) is OFF, start the Command Timeout timer (Start timing once the command is given and wait for the Load Complete to occur). If the Load Complete never happens and the timer times out, unlatch the Load/Start bit of the Control Word and set the Error (.ER) bit of the AOI. Set the Step Number to -2. If the axis is not an AKD1G (i.e. AKD2G) axis, then set the .ER bit of the AOI and set the Step Number to -3.

Changes to Axis Status bits

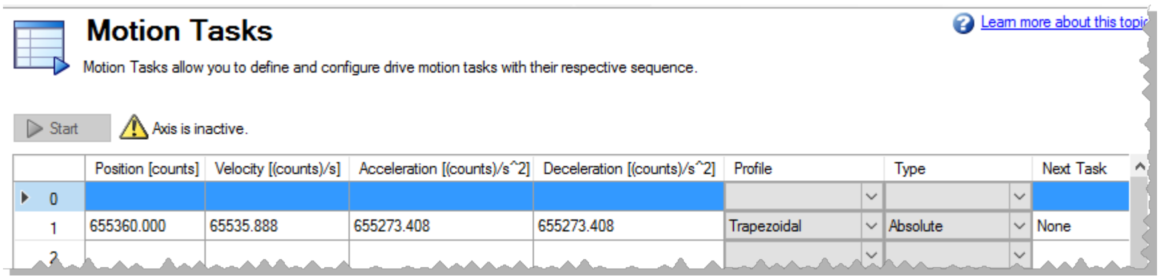
- None

Example of Usage/Programming Guidelines

From the Sample logic below, a normally open contact (N.O.) Set_MT_Toggle is used to trigger the AOI. The trigger is One Shot to the AOI .EnableIn. The parallel branch will seal-in on the .EnableIn and will open if either the command is successful (.DN) or failed (.ER). This example shows where each entry (i.e., MT_NUM, MT_ACC, etc.) is a Tag name so the entry's value can be changed. It is also possible to input constant values into the AOI instead of Tags for these attributes.



Upon success, check the Motion Task screen in WorkBench to verify the AOI was created.



Troubleshooting: Summary of Reasons for Errors/Failure

- The AKD_Set_Motion_Task AOI requires firmware 1-13-9-000 or newer. See the Revision History at the end of this section for more information.
- Conflict from triggering the AOI while another AOI is executing.
- The AOI Command Timer timed out before the Load Complete in Status Word 2 occurred.
- Response Type is #14 indicating an error occurred on attempt to write a parameter value. This is often due to a value being out of range for the given parameter write.
  See Response Type 0x14 - Command/Response Error for more information.

  If any of the MT parameters are deemed invalid (i.e., out of range, etc.), WorkBench will report a 159: Warning: Failed to set motion task parameters in the AKD1G. This typically occurs when attempting to write a 0 to the Move Profile Target Velocity (MT.V).
  See Fault and Warning Messages for more information.

Fault (“F”) Warning (“n”)	Message/Warning	Cause	Remedy	Drive Response to Fault
n159	Failed to set motion task parameters	Invalid motion task parameters assignment. This warning can appear upon an MT.SET command.	Activation of any new motion or using of DRV.CLRFAULTS will clear the warning. Check motion task settings and parameters.	None
n160*	Motion task activation failed.	Activation of the motion task failed due to incompatible parameters, or motion task does not exist. This warning can appear upon an MT.MOVE command.	Activation of any new motion or using of DRV.CLRFAULTS will clear the warning. Check motion task settings and parameters to make sure that the values entered will produce a valid motion task.	None

*This warning occurs with an attempt to start a non-existing or invalid Motion Task (i.e. AKD_Start_MotionTask AOI).

Step Summary

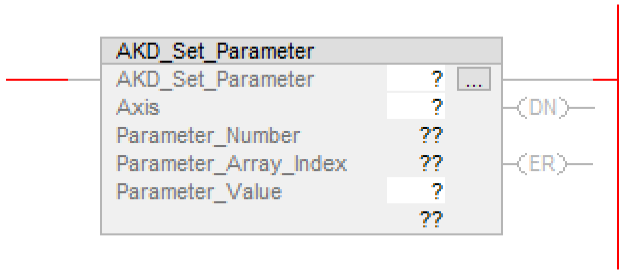
Step Number	Operation/Result
0	On First_Scan unlatch the .DN and .ER bits, unlatch the Load/Start bit of the Control Word, reset the Command_Timeout timer, set the Command Timeout preset to 5000 ms and set the Step Number to 1. Latch the First_Scan_Bit so this happens only once during execution.
1	If the Step Number is 1, then move a 16#1F (Command Type 0x1F - Read or Write Parameter Value into the Command_Bytes of Byte 2) and set the Step Number to 2.
2	If the Step Number is 2, then set then set bit 6 to a 1. Set the Step Number to 3
3	If the Step Number is 3, then set Byte 6 to a 1 (Set parameter) and then set the Step Number to 4.
4	If the Step Number is 4, then set the Instance number to 275 (MT.NUM) and move this number into Command_Bytes 4 and 5 per the Write Parameter method. Set the Step Number to 5.
5	If the Step Number is 5, then copy the MT.NUM value (input parameter of the AOI) to the data of Bytes 24-27per the Write Parameter method. Set the Step Number to 6.
6	If the Step Number is 6, unlatch the Load/Start bit of the Control Word and set the Step Number to 7.
7	If the Step Number is 7 and the Load Complete of axis Status Word 2 is cleared, then it is ready for a new command. Latch the Load/Start bit of the Control Word and set the Step Number to 8.
8	If the Step Number is 8 and the Load Complete bit of Status Word 2 is High it signifies the command completed successfully. Set the Step Number to 9.
9-48	Rungs 9 to 48 are a repeat of the method for MT.NUM for all other MT commands. However, MT.SET on rung 48, the Done (.DN) bit of the AOI is set.

Error Step Number	Operation/Result
-1	Rung 49. If the Load/Start bit of the Control Word is ON and the Load Complete bit of the Status Word 2 is OFF/Not Complete and the Response Type is a 16#14 (Command Type 0x1F - Read or Write Parameter Value) then there was an error. On Error, unlatch the Load/Start bit of the Control Word and latch the Error (.ER) bit of the AOI. Set the Step Number to -1.
-2	Rung 50. If the Load/Start bit of the Control Word is ON and the Load Complete bit of Status Word 2 is OFF then start the Command Timeout timer once the command is given and wait for the Load Complete to occur. If the Load Complete never happens and the timer times out, unlatch the Load/Start bit of the Control Word and set the Error (.ER) bit of the AOI. Set the Step Number to -2.
-3	Rung 51. If the axis is identified as an AKD2G axis, latch the .ER bit and set the Step Number to -3.

Revision History

<table>
<tr><th>Revision Number</th><th>Description/Notes</th><th>Revision Date</th></tr>
<tr><td></td><td>Initial release</td><td>05/06/2016</td></tr>
<tr><td>v1.0</td><td>Used 64 bit instances of MT.ACC, MT.DEC, MT.V, and MT.P.</td><td>02/27/2017</td></tr>
<tr><td>v2.0</td><td>
Changes were made the AOI to utilize the 32 bit versions (Instances) of MT.ACC, MT.DEC, MT.V, and MT.P (Introduced in firmware version 01-13-09-000 on 06/18/2015). This allows a negative MT.P to be set correctly. See additional notes below.
<table>
<tr><th></th><th>New</th><th></th><th>Old</th></tr>
<tr><td>MT.NUM</td><td>275</td><td>1 byte</td><td>275</td></tr>
<tr><td>MT.ACC</td><td>265</td><td>4 byte</td><td>264</td></tr>
<tr><td>MT.DEC</td><td>270</td><td>4 byte</td><td>269</td></tr>
<tr><td>MT.V</td><td>1084</td><td>4 byte</td><td>284</td></tr>
<tr><td>MT.P</td><td>277</td><td>4 byte</td><td>276</td></tr>
<tr><td>MT.CNTL</td><td>267</td><td>4 byte</td><td>267</td></tr>
<tr><td>MT.MTNEXT</td><td>274</td><td>1 byte</td><td>274</td></tr>
<tr><td>MT.TNEXT</td><td>279</td><td>2 Bytes</td><td>279</td></tr>
<tr><td>MT.SET</td><td>278</td><td>1 byte</td><td>278</td></tr>
</table>
<table>
<tr><th>Id</th><th>Parameter</th><th>Type</th><th>Size</th></tr>
<tr><td>275</td><td>MT.NUM</td><td>Integer</td><td>1 Byte</td></tr>
<tr><td>265</td><td>MT.ACC (32 bit version)</td><td>Acceleration</td><td>4 Byte</td></tr>
<tr><td>270</td><td>MT.DEC (32 bit version)</td><td>Acceleration</td><td>4 Byte</td></tr>
<tr><td>1084</td><td>MT.V (32 bit version)</td><td>Velocity</td><td>4 Byte</td></tr>
<tr><td>277</td><td>MT.P (32 bit version)</td><td>Position</td><td>4 Byte Signed</td></tr>
<tr><td>267</td><td>MT.CNTL</td><td>Integer</td><td>4 Byte</td></tr>
<tr><td>274</td><td>MT.MTNEXT</td><td>Integer</td><td>1 Byte</td></tr>
<tr><td>279</td><td>MT.TNEXT</td><td>Integer</td><td>2 Byte</td></tr>
</table>
MT.SET
<table>
<tr><th>Fieldbus</th><th>Instance</th><th>Data Size</th><th>Data Type</th></tr>
<tr><td>EtherNet/IP</td><td>278</td><td>Command</td><td>None</td></tr>
</table>
AOI library v5.0 revision
</td><td>07/10/2019</td></tr>
<tr><td>v3.0</td><td>
Added logic to set .ER bit on output of AOI if the axis is AKD2G.
Update includes synchronization with new data type AKD_Data which the Axis structure uses.
AOI library v6.0 revision
</td><td>06/21/2021</td></tr>
</table>

7.24 AKD_Set_Parameter



Description

Use the AKD_Set_Parameter AOI to modify a drive parameter or execute a drive command on an axis. The time required to execute is highly dependent on the particular parameter, PLC scan time, and EtherNet/IP communcation scan time.

Compatibility

The AKD_Set_Parameter AOI is compatible with both the AKD1G and AKD2G EtherNet/IP drives.

NOTE

The AKD_Set_Parameter is different than the AKD_Set_Attribute AOI.

- AKD_Set_Attribute applies only to the Attribute listing for the Position Controller Object.
- AKD_Set_Parameter provides access to writing to a paramter listing in the AKD1G Parameter Listing or AKD2G EtherNet/IP Objects List .

AKD1G and AKD2G have different parameter listings and IDs (known as Instance Numbers on AKD1G drives).

See AKD1G EtherNet/IP Objects and Attributes or AKD2G EtherNet/IP Objects and Attributes.
See AKD1G Parameter Listing or AKD2G EtherNet/IP Objects List.

The AKD_Set_Parameter AOI adds the Parameter_Array_Index field entry to the AOI to accommodate array type parameters which exist in the AKD2G drive only.
For all AKD1G parameters and non-array type AKD2G parameters the Parameter_Array_Index field entry will always be 0.

Required Command Source and Operation Mode

AKD1G

DRV.CMDSOURCE = Any/All

DRV.OPMODE = Any/All

AKD2G

AXIS#.CMDSOURCE = Any/All

AXIS#.OPMODE = Any/All

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Set_Parameter	AKD_Set_Parameter	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
Parameter_Number	INT	Constant	See the AKD1G Parameter Listing or AKD2G EtherNet/IP Objects List for the ID (Decimal) to enter into this field. Note AKD1G and AKD2G have different parameter listings and IDs.
Parameter_Array_Index	DINT	Constant	AKD1G Always 0 for AKD1G Parameters. AKD2G Always 0 for non-array parameters Equal to the Array Index of Array Type parameters.
Parameter_Value	DINT	Tag or Constant value	The value to be written to the drive or axis parameter.

Structure

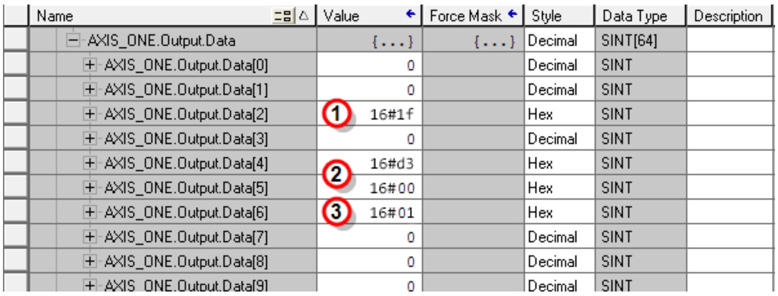
Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON if the AOI execution completes successfully when the Load Complete bit in Status Word 2 is True (1).
.ER	Output	BOOL	The .ER bit is set if there is a Response Type #14 (Error). See Response Type 0x14 - Command/Response Error.

Execution

AKD1G

The AKD_Set_Parameter AOI uses the Write Parameter method internally.

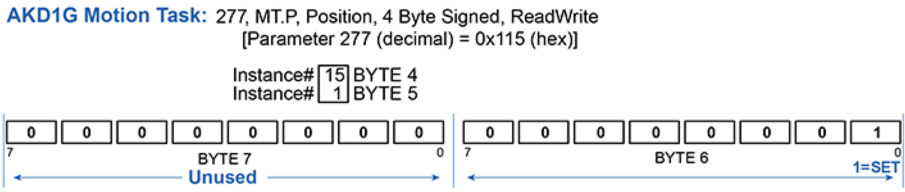
1. Byte 2 of the axis' Output.Data is set to 1F for the Command Type 0x1F - Read or Write Parameter Value.
2. Bytes 4-5 of the axis' Output.Data is the Instance Number (Example: HOME.V, 211 decimal, 16#00d3 hex)
3. Set Byte 6 of the axis' Output.Data to a 1 to Write.
   (Read/Get = 0; Write/Set = 1)



Name	Value	Force Mask	Style	Data Type	Description
AXIS_ONE.Output.Data	{...}	{...}	Decimal	SINT[64]	
AXIS_ONE.Output.Data[0]	0		Decimal	SINT	
AXIS_ONE.Output.Data[1]	0		Decimal	SINT	
AXIS_ONE.Output.Data[2]	1 16#1f		Hex	SINT	
AXIS_ONE.Output.Data[3]	0		Decimal	SINT	
AXIS_ONE.Output.Data[4]	2 16#d3		Hex	SINT	
AXIS_ONE.Output.Data[5]	16#00		Hex	SINT	
AXIS_ONE.Output.Data[6]	3 16#01		Hex	SINT	
AXIS_ONE.Output.Data[7]	0		Decimal	SINT	
AXIS_ONE.Output.Data[8]	0		Decimal	SINT	
AXIS_ONE.Output.Data[9]	0		Decimal	SINT	

*Index is 0 for all AKD1G and non-array type AKD2G parameters.

*Index is non-zero for array-type parameters (AKD2G only).

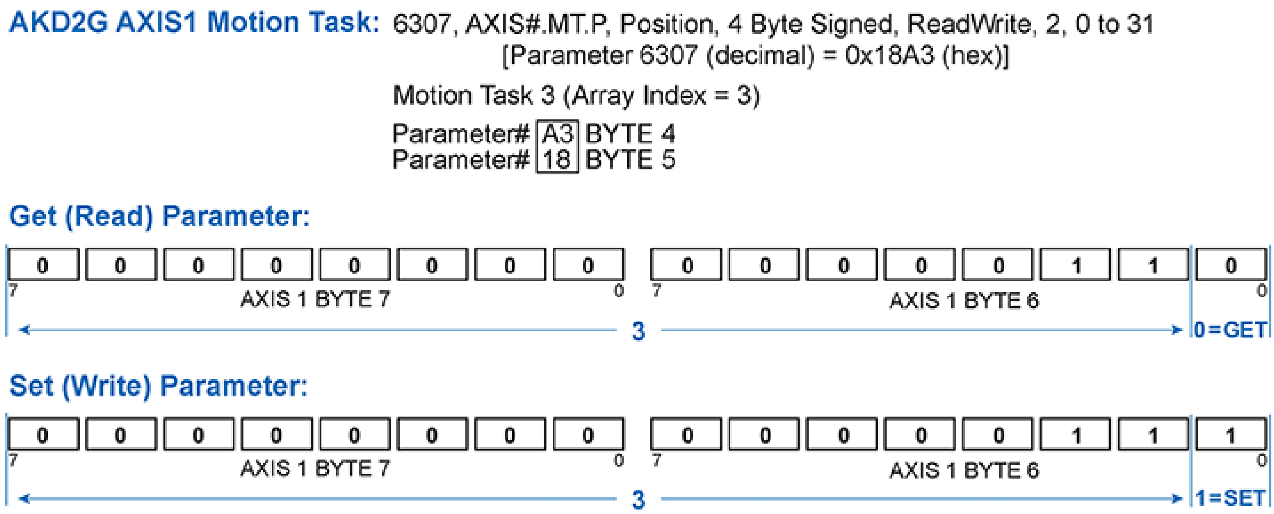


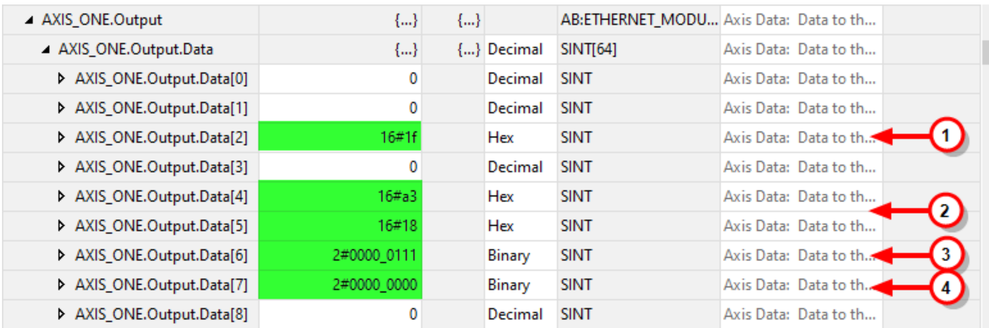
4. Move the Parameter Value of the AOI into Bytes 24-27 of the axis' Output.Data.
5. Latch the Load/Start bit (bit 0 in Control Word).
6. On success (Load Complete in bit 7 of Status Word 2 is True) unlatch the Load/Start bit.

AKD2G

The AKD_Set_Parameter AOI uses the Write Parameter method internally.

1. Byte 2 of the axis' Output.Data is set to Command Type 0x1F - Read or Write Parameter Value.
2. Bytes 4-5 of the axis' Output.Data is the Instance Number.
3. Set Byte 6, bit 0 of the axis' Output.Data to a 1 to Write. (Read/Get = 0; Write/Set = 1)
4. Set Byte 6, bits 1-7 and Byte 7, bits 0-7 of the axes.





AXIS_ONE.Output	{...}	{...}		AB:ETHERNET_MODU...	Axis Data: Data to th...	
AXIS_ONE.Output.Data	{...}	{...}	Decimal	SINT[64]	Axis Data: Data to th...	
AXIS_ONE.Output.Data[0]	0		Decimal	SINT	Axis Data: Data to th...	
AXIS_ONE.Output.Data[1]	0		Decimal	SINT	Axis Data: Data to th...	
AXIS_ONE.Output.Data[2]	16#1f		Hex	SINT	Axis Data: Data to th...	1
AXIS_ONE.Output.Data[3]	0		Decimal	SINT	Axis Data: Data to th...	
AXIS_ONE.Output.Data[4]	16#a3		Hex	SINT	Axis Data: Data to th...	
AXIS_ONE.Output.Data[5]	16#18		Hex	SINT	Axis Data: Data to th...	2
AXIS_ONE.Output.Data[6]	2#0000_0111		Binary	SINT	Axis Data: Data to th...	3
AXIS_ONE.Output.Data[7]	2#0000_0000		Binary	SINT	Axis Data: Data to th...	4
AXIS_ONE.Output.Data[8]	0		Decimal	SINT	Axis Data: Data to th...	

5. Move the Parameter_Value of the AOI into Bytes 24-27 of the axis' Output.Data.
6. Latch the Load/Start bit (bit 0 in Control Word).
7. On success (Load Complete, bit 7 in Status Word 2 is True) unlatch the Load/Start bit (bit 0 in Control Word).

Condition	Ladder Diagram Action
Prescan	Reset OS_Loadstart and clear Bytes 0-8 of the Command_Bytes (AOI internal).
.EnableIn False	Reset OS_Loadstart and clear Bytes 0-8 of the Command_Bytes (AOI internal).
Instruction Execution	1. On .EnableIn, One Shot and set the Step Number to 0 and reset the .DN and .ER bits. 2. Set the Command Type in Byte 2 of the axis' internal Command_Bytes to 16#1F (Write Parameter). (See Command Type 0x1F - Read or Write Parameter Value.) 3. Set the parameter (Instance) number in Bytes 4 and 5 of the axis' internal Command_Bytes. 4. AKD1G: Set Byte 6 to 1 (Write). 5. AKD2G: Set Byte 6, bit 0 to 1 (Write) AKD2G: Populate Byte 6, bits 1-7 and Byte 7, bits 0-7 with the binary equivalent of the parameter index array which is 0 for non-array parameters and non-zero for array parameters. 6. Move the data to Write to the axis' internal Command_Bytes 24-27 (4 Bytes). AKD1G: The method allows for 8 Bytes, but the Parameter_Value for the AOI is limited to a DINT. The best practice is to use the 32 bit (4 Byte) instance for the equivalent 64 Byte parameter. AKD2G: No parameters in the EtherNet/IP Parameter List have data type larger than 32 bits (4 Bytes). **Step 0:** • If the Step Number is 0 and Load Complete from Status Word 2 is False, latch the Load/Start bit of the Control Word and set the Step Number to 1. • When the Load Complete in Status Word 2 is True, unlatch the Load bit of the Control Word. **Step 1:** • If the Step Number is 1 and the Response Type is a 14 (Error), then set the Error (.ER) bit and set the Step Number to -1. • If the Step Number is 1 and the Load Complete is True (Success of Write) then set the .DN (Done) bit and set the Step Number to 2. **Step 2:** AOI execution was successful. **Error (.ER):** If the Error (.ER) bit is set, then unlatch the Load/Start bit of the Control Word. 7. Take the internal Control Word and Command_Byte settings of the AOI and send them out to the drive to be executed.

Changes to the Axis Status Bits

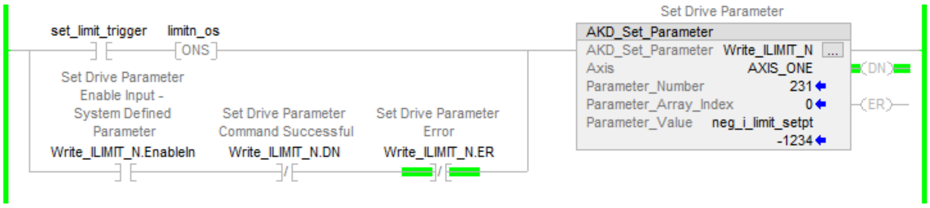
- None

Example of Usage/Programming Guidelines

AKD1G

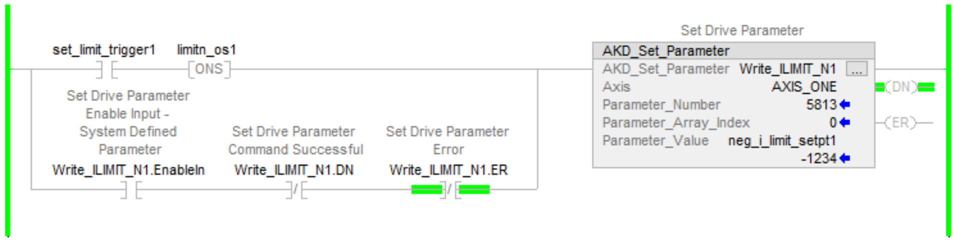
The example below demonstrates writing to the IL.LIMITN (Negative Current Limit). The keyword and its Instance Number can be found in AKD1G Parameter Listing.

A Normally Open (N.O.) contact called set_limit_trigger is used to toggle the Write command and a One Shot in tandem with a parallel seal-in branch of the AOI's .EnableIn, Done (.DN), and Error (.ER) conditions. This is used to seal-in the .EnableIn and execution of the AKD_Set_Parameter AOI until either Success (.DN) or failed (Error; .ER). Note the .EnableIn contact is N.O. and the .DN and .ER are Normally Closed (N.C.).



AKD2G

The example below demonstrates writing to AXIS#.IL.LIMITN (Negative Current Limit). The parameter and its ID can be found in AKD2G EtherNet/IP Objects List.



Troubleshooting: Summary of Reasons for Errors/Failure

- Another AOI is executing.
- The parameter is out of range.
  In the example above, you cannot write a positive value to the parameter. Any attempt to do so will result in the .ER output bit of the AOI turning ON (i.e. Response Error). See WorkBench Help or in the WorkBench Terminal type **DRV.HELP <keyword>** and select **Enter** to see the range of values.

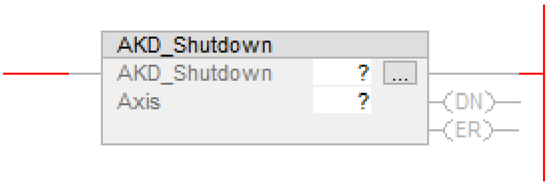
Step Summary

Step Number	Operation/Result
0	On .EnableIn set the Step Number to 0 and reset the .DN and .ER bits. Set the Command Byte to 16#1F and load parameter number into Bytes 4 and 5. If AKD1G, set Byte 6 to 1 (Write); If AKD2G, set Byte 6, bit 0 to 1 (Write) and set Byte 6, bits 1-7 and Byte 7, bits 0-7 to the parameter array index. Set Bytes 24-27 to the Parameter Value. When the status bit Load Complete is False, latch the Load/Start bit of the Control Word and set the Step Number to 1.
1	Check for response errors and on Load Complete from the Status Word, set the Step Number to 2. Set the .DN bit.
2	AOI execution was successful.
-1	Step Number was 1 but the Response Type from the Response Assembly was a 14 (Error).

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial release	02/23/2011
v2.1	• Removed CommandTimeout timer. This instruction will now never timeout (some instructions can take a long time) • AOI library v5.0 revision	10/29/2014
v3.0	• Eliminated axis number bits and deleted Command_Axis_Number since it is now unused. • Added necessary code to accommodate AKD2G index array parameter types. Update also synchronizes the AOI with the new data type AKD_Data that the Axis structure uses. • AOI library v6.0 revision	07/16/2021

7.25 AKD_Shutdown



Description

The AKD_Shutdown instruction executes the Hard Stop (bit 5 in Control Word).

Compatibility

The AKD_Shutdown is compatible with both AKD1G and AKD2G EtherNet/IP Drives.

Required Command Source

AKD1G

DRV.CMDSOURCE = Any/All

DRV.OPMODE = Any/All

AKD2G

AXIS#.CMDSOURCE = Any/All

AXIS#.OPMODE = Any/All

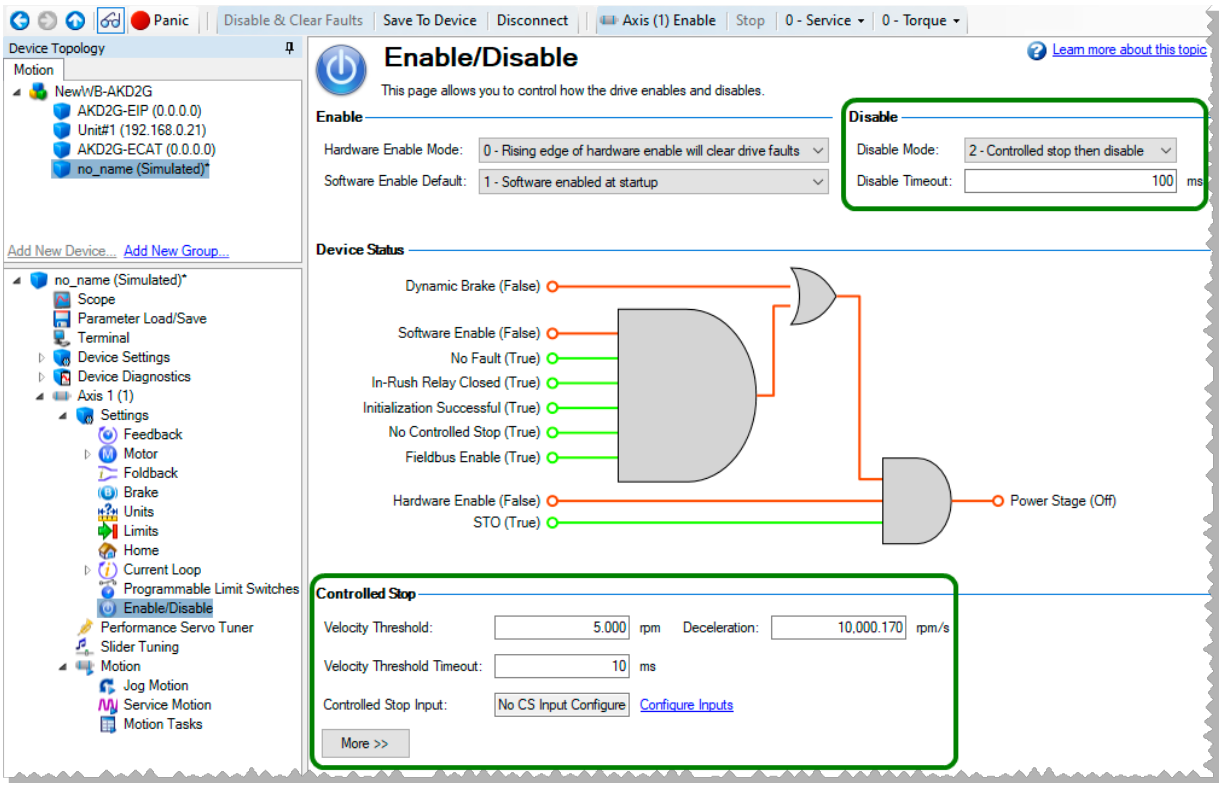
The AKD_Shutdown instruction executes the Hard Stop (bit 5 in Control Word).

AKD1G Hard Stop

DRV.DISMODE (AKD1G) will affect if the Controlled Stop is used or not.

If the DRV.DISMODE is set to **0 - Disable axis immediately**, setting the Hard Stop bit in Control Word over EtherNet/IP or by the AKD_Shutdown AOI will result in the Software Enable and Fieldbus Enable being disabled and the motor will coast to rest. Setting the DRV.DISMODE to **2 - Controlled stop then disable** and setting the Hard Stop bit causes the Axis to follow the controlled stop decel rate and then disable. In other words, the Hard Stop will invoke a Software Disable and the Disable Mode will determine the mode of action.

The Disable Mode and Controlled Stop settings in WorkBench are found on the Enable/Disable screen:



AKD2G Hard Stop

On the AKD2G drive the Hard Stop will always execute the Controlled Stop using the Controlled Stop settings as found on the Enable/Disable screen for the given axis in WorkBench.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_ Shutdown	AKD_ Shutdown	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_ Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON when both the Enable Status bit and Profile_In_ Progress bit turn OFF.
.ER	Output	BOOL	The .ER bit is set if there is a Command Timeout (axis stays Enabled for the duration of the Command Timeout time).

Execution

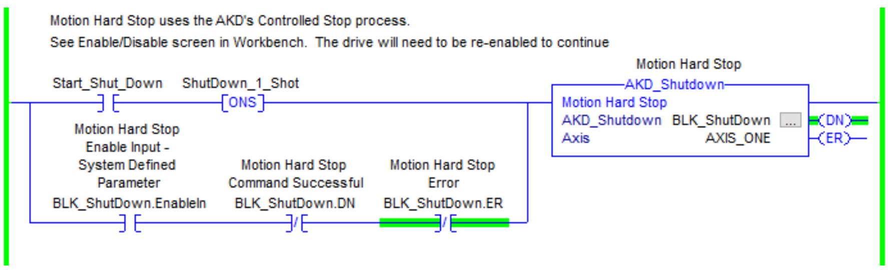
Condition	Ladder Diagram Action
Prescan	Reset the Command_Timeout timer and Set Timeout Preset to 2000 ms.
.EnableIn False	Set Step Number to 0 and reset Command_Timeout timer.
Instruction Execution	**On Enable:** If Step Number is 0 then reset .DN (Done) and .ER (Error) bits of the AKD_Shutdown AOI and latch the Hard Stop (bit 5 in Control Word). **AKD1G:** Hard Stop behavior is dependent on DRV.DISMODE. **Step Number 1 (Hard Stop):** If the Step Number is1 (Hard Stop was commanded) then for as long as the Axis is Enabled run the Command_Timeout timer. **Axis Disabled:** If the Axis is disabled (bit 7 in Status Word 1) it may indicate that either the Controlled Stop completed or the Axis was disabled while the Controlled Stop was executing and the Profile_In_Progress (bit 0 in Status Word 1) is False. 1. Set the AKD_Shutdown AOI's .DN bit. 2. Unlatch the Hard Stop (bit 5 in Control Word). 3. Unlatch the Enable (bit 7 in Control Word). **Step Number 1 (Timed out):** If Step Number is 1, but the timer times out, set the AKD_Shutdown AOI's .ER bit, unlatch the Enable (bit 7 in Control Word), and unlatch the Hard Stop (bit 5 in the Control Word). Set the Step Number to -2.

Changes to Axis Status bits:

- Profile_In_Progress (bit 0 in Control Word) changes to 0 (False) once the Hard Stop operation is complete.
- Enable State (bit 7 in Status Word 1) changes to 0 (False) once the Hard Stop operation is complete.

Example of Usage/Programming Guidelines

A normally open (N.O.) contact called Start_Shut_Down is used to trigger the AKD_Shutdown request and a One Shot is used on the AOI's .EnableIn. A parallel seal-in branch is used to seal-in the .EnableIn until 1) The .DN bit is set; success or 2) the .ER bit is set; error. As indicated in the Sample project's rung comments, see the Enable/Disable screen in WorkBench in regards to the DRV.DISMODE (AKD1G) or AXIS#.DISMODE (AKD2G) and the Controlled Stop (if used).



Troubleshooting: Summary of Reasons for Errors/Failure

- If the Controlled Stop deceleration rate is Low, so that the Hard Stop takes longer than 2 seconds (Command Timeout), then the Hard Stop bit is reset and the AOI disabled per the example logic above.
- An observation is that the ramp will continue down and likely will eventually generate an emergency timeout which is dependent on the setup on the Enable/Disable screen in WorkBench and the drive.

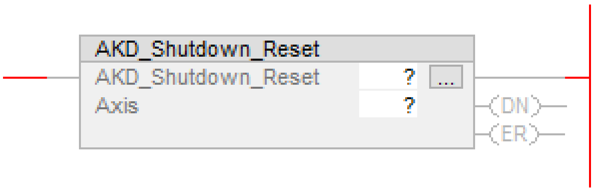
Step Summary

Step Number	Operation/Result
0	Clear .DN and .ER bits. Latch Hard Stop bit in Control Word (bit 5). Set Step Number to 1.
1	While axis is Enabled, run Command_Timeout timer (2000 ms or 2 seconds). If axis is Disabled or Controlled Stop, the disable finishes and the Profile_In_Progress bit is OFF, set the .DN bit of the AOI and clear the Hard Stop bit (bit 5 in Control Word) and clear the Enable bit (bit 7 in the Control Word).
-2	Step was Step 1, but command timed out. Set Error (.ER) bit of AOI and unlatch the .Enable and Hard Stop bits in the Control Word.

Revision History

Revision Number	Description/Notes	Date of Revsion
	Initial version	02/23/2011
v4.0	• There was a race with the Axis.Control.Hardstop bit. It was removed from the .EnableIn False and did the reset in the logic • AOI library v5.0 revision	03/01/2016
v5.0	• Updated to synchronize AOI with new AKD_Data data type which the Axis structure uses. • AOI library v6.0 revision	07/16/2021

7.26 AKD_Shutdown_Reset



Description

Use the AKD_Shutdown_Reset to reset the axis after executing an AKD_Shutdown (Hard Stop).

Compatibility

The AKD_Shutdown_Reset is compatible with both AKD1G and AKD2G EtherNet/IP drives.

Required Command Source and Operation Mode

AKD1G

DRV.CMDSOURCE = Any/All

DRV.OPMODE = Any/All

AKD2G

AXIS#.CMDSOURCE = Any/All

AXIS#.OPMODE = Any/All

Use the AKD_Shutdown_Reset to reset the axis after executing an AKD_Shutdown (Hard Stop).

This AOI executes internally:

- the AKD_Disable (reset the Enable bit in Control Word)
- an AKD_Fault_Reset (reset all faults)

The AKD_Shutdown_Reset instruction may take multiple scans to complete execution due to communication scan time and the time required for the axis to process the command.

The .DN bit is set only after the Disable and Fault Reset are successful and the General Fault bit (bit 3 in Status Word 1) is cleared (No Faults).

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Shutdown_Reset	AKD_Shutdown_Reset	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

Mnemonic	Type	Format	Description
.EnableIn	Input	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	Output	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	Output	BOOL	Turns ON when both the Disable and Fault Reset are successful and the General Fault Status bit (bit 3 in Status Word 1) is False (OFF).
.ER	Output	BOOL	Set the .ER bit if the internal AKD_Disable or internal AKD_Fault_Reset AOIs fail to execute (Error).

Execution

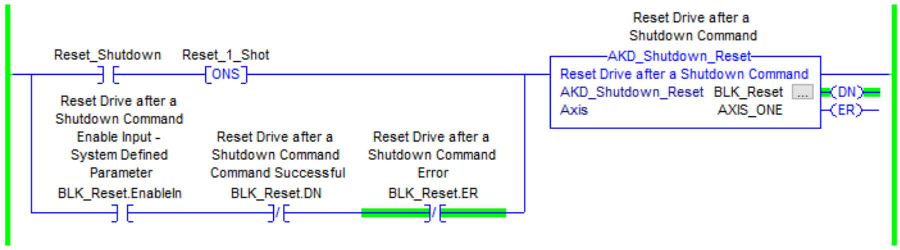
Condition	Ladder Diagram Action	
Prescan	Unlatch the Start Sequence One Shot, Command Timeout timer, Disable Timer, set the Command Timeout timer preset to 10000 ms, set the Disable Timer preset to 100, and set the Step Number to 0.	
.EnableIn False	Unlatch the Start Sequence One Shot, the Command Timeout timer, the Disable Timer and set the Step Number to 0.	
Instruction Execution	**AOI Enabled:**	One Shot to unlatch the .ER and .DN bits on the AKD_Shutdown_Reset AOI and set the Step Number to 1.
	Step Number 1:	Disable the axis using the internal AKD_Disable AOI. Once the .DN (Done) bit of the internal AKD_Disable AOI indicates it was successfully disabled, set the Step Number to 2.
	Step Number 2:	Start the Disable Timer (100 ms). After the timer is .DN (Done), set the Step Number to 3.
	Step Number 3:	Reset the axis Fault using the internal AKD_Fault_Reset AOI. When the internal AKD_Fault_Reset is .DN (Done; Success), set the Step Number to 5.
	Step Number 5:	Wait for the General Fault (bit 3) to clear Status Word 1, then set the .DN (Done) bit of AKD_Shutdown_Reset, and set the Step Number to 10.
	Command Timeout timer:	If the Command Timeout timer (10000 ms; 10 seconds) times out indicating the General Fault bit in Status Word 1 never turned OFF or if the internal AKD_Disable or AKD_Fault_Reset AOIs Error (Fail on Execution), then set the AKD_Shutdown_Reset AOI .ER bit. Note the AKD_Disable Block occurs in Step 1 and the AKD_Fault_Reset occurs in Step 3.

Changes to Axis Status bits

- General_Fault status (bit 3 in Status Word 1) is OFF indicating "No general fault is present."
- Enable_Status status (bit 7 in Status Word 1) is OFF indicating "Axis is disabled."

Example of Usage/Programming Guidelines

A normally open (N.O.) contact called Reset_Shutdown is used as a trigger to execute the AKD_ Shutdown_Reset AOI. A One Shot sends the request to the AOI's .EnableIn and a parallel branch is used as a seal-in until 1) the .DN bit is set; success or 2) the .ER bit is set; failure.



Troubleshooting: Summary of Reasons for Errors/Failure

- Failure to disable drive
- Failure to reset faults
- CommandTimeout timer timed out before both the drive is disabled and before faults are reset.

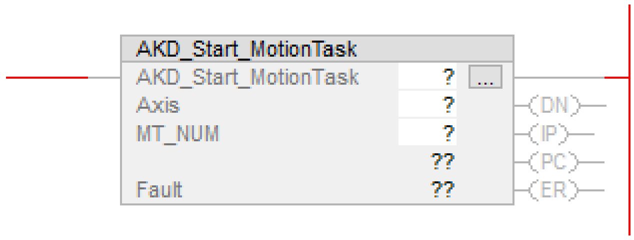
Step Summary

Step Number	Operation/Result
.EnableIn	Clear .ER and .DN bits of the AKD_Shutdown_Reset AOI and set Step Number to 1.
1	Execute the internal AKD_Disable AOI in order to clear Enable (bit 7 in Control Word). Set Step Number to 2.
2	Start Disable Timer (100 ms). After timer is Done (.DN), set Step Number to 3.
3	Execute the internal AKD_Fault_Reset AOI in order to clear faults. On Done (.DN). Set Step Number to 5.
5	Start 10000 ms (10 seconds) Command Timeout timer and wait for the fault to clear. This is indicated by the General Fault bit (bit 3 in Status Word 1). If the fault is cleared, then set the .DN bit of the AOI and set the Step Number to 10.
.ER	If the CommandTimeout timer (10000 ms; 10 seconds) times out, the command failed to set the .ER bit of the AKD_Shutdown_Reset AOI. This can happen when the Step Number: • is greater than or equal to 1 and the AKD_Disable AOI failed • is greater than or equal to 3 and the AKD_Fault_Reset AOI failed

Revision History

Revision Number	Description/Notes	Date of Revision
	Initial version	02/23/2011
v2.1	• Set disable timeout to 100 ms • AOI library v5.0 revision	05/27/2015
v3.0	• Updated to synchronize AOI with new AKD_Data data type which the Axis structure uses. • AOI library v6.0 revision	07/16/2021

7.27 AKD_Start_MotionTask



Description

The AKD_Start_MotionTask AOI initiates a predefined Motion Task for a given axis in the Motion Task Table by specifying the Motion Task Number in the AOI's MT_NUM field entry.

Compatibility

The AKD_Start_MotionTask AOI is compatible with both AKD1G and AKD2G EtherNet/IP drives.

Required Command Source and Operation Mode

AKD1G

DRV.CMDSOURCE = Service

DRV.OPMODE = Position

AKD2G

AXIS#.CMDSOURCE = Fieldbus

AXIS#.OPMODE = Position

The AKD_Start_MotionTask AOI uses the Block Method to run a stored Motion Task Sequence. The AKD_Start_MotionTask Add-On Instruction does not create or set a Motion Task; it only initiates a pre-defined Motion Task. The Block Number, which provides a way to point to a Motion Task Number and run it, can be found in Byte 1 (AKD1G or AKD2G Axis 1) or Byte 65 (AKD2GAxis 2) of the Command Assembly. The Motion Task can either be created in WorkBench or by using the AKD_Set_Motion_Task AOI (AKD1G) or AKD2G_Set_Motion_Task AOI (AKD2G) to create or modify Motion Tasks from the PLC.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Start_ MotionTask	AKD_Start_ MotionTask	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_ Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.
MT_Num	SINT	Tag or Constant	Tag or Constant that sets the Block Number (Index) of the Motion Task to be executed in the Motion Task table.

Structure

The following fields are automatically populated with Read Only data once the Operands above are entered.

Mnemonic	Type	Description
Fault	DINT	Read-only that provides a fault code in the event of a fault within the AOI during execution. These are flags with negative values.
.EnableIn	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	BOOL	Set when the Block Number reported by Byte 1 (AKD1G or Axis 1 of AKD2G) or Byte 65 (Axis 2 AKD2G) of the Response Assembly's Executing Block Number equals the Block Number in the equivalent Bytes of the Command Assembly (set by MT_Num in the AKD_Start_MotionTask AOI).
.IP	BOOL	Turns ON while the Motion Task is executing and the Profile_In_Progress (bit 0 in Status Word 1) is ON. Known as the In Motion bit in AKD2G.
.PC	BOOL	The .PC Output bit is set when two conditions are satisfied: • The Profile_In_Progress (bit 0 in Status Word 1) is False indicating that the trajectory is finished. • The On Target Position (bit 2 in Status Word 1) is ON (True indicating the axis is in the In Position Window set by MT.TPOSWND (AKD1G) or AXIS#.SETTLE.P (AKD2G). The In Position Window setting is application-dependent. Be aware of the following: • a value too large may result in the .PC bit turning on too early • a value too small may result in the .PC bit flickering or never turning on at all. The On Target position bit is sourced from bit 11 of the DRV.MOTIONSTAT (AKD1G) or AXIS#.MOTIONSTAT (AKD2G).
.ER	BOOL	Any Error (Fault) unlatches the .DN, .IP, and.ER bits of the AKD_Start_MotionTask AOI output and also unlatches the Start Block bit (bit 1 in Control Word). Step Number is also set to -1 for all errors. The Fault (Read Only) will capture the reason for the fault and also display it in the Fault field of the AKD_Start_Motiontask AOI. • Fault -1: Axis is not Enabled. • Fault -2: Axis is not Homed. • Fault -3: If the Step Number is 4 and the executing Motion Task number is never confirmed by the Response Assembly in the 500 ms time. • Fault -4: If the Smooth Stop or Hard Stop bit is executed. This is typically set by the AKD_Smooth_Stop or AKD_Shutdown AOI while the AKD_Start_MotionTask AOI is executing. • Fault -5: If the Response Type from the Response Assembly is 16#14. (See Response Type 0x14 - Command/Response Error).

Execution

The AKD_Start_MotionTask Add-On Instruction uses the Block Method, which is detailed in the EtherNet/IPCommunications manual, under Running a Stored Motion Task Sequence.

See the AKD EtherNet/IP Communication Manual or AKD2G EtherNet/IP Communications Manual for more information.

NOTE

Running a Stored Motion Task Sequence is an automated process used internally in the AKD_Start_MotionTask AOI.

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.

The Motion Task can be created either in WorkBench or by using the AKD_Set_Motion_Task AOI (AKD1G) or AKD2G_Set_Motion_Task AOI (AKD2G) AOI to create or modify tasks from the PLC.

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.

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 Axis 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.

NOTE

The axis must be Enabled, Homed, and in Position Operation Mode prior to enabling the AKD_Start_MotionTask AOI. For AKD1G DRV.CMDSOURCE must be set to Service; for AKD2G AXIS#.CMDSOURCE must be set to Fieldbus.

Prior to transitioning the Start Block (bit 1 in Control Word):

- the Homed bit (bit 5 in Status Word 1) must be True (ON, 1)
- the Enabled State bit (bit 7 in Status Word 1) must be True (ON, 1)

1. Set the Block # to the value of MT_NUM.
   AKD1G or AKD2G Axis 1: Byte 1
   AKD2G Axis 2: Byte 65

   NOTE

   - The AKD_Start_MotionTask AOI sets the Block Number to the MT_NUM field entry of the AOI and triggers the Start Block (bit 1 in Control Word).
   - AKD1G: Motion Task 0 should be reserved for AKD_Jog or AKD_Move execution. Best practices is MT_NUM should not be set to zero in the AKD_Start_MotionTask AOI.
   - AKD2G: Does not use Motion Tasks for AKD_Jog and AKD_Move instructions; an internal Fieldbus Move is used instead.
   - For more information on the Start Block, see Control Word.

2. To start the Motion Task selected by the Block #, transition the Start Block (bit 1 of the axis' Control Word) from 0 to 1 (a rising edge).
   AKD1G or AKD2G axis 1: Byte 0
   AKD2G axis 2: Byte 64
   For more information on Block # see the Command Assembly Data Structure in the AKD EtherNet/IP Communication Manual or AKD2G EtherNet/IP Communications Manual.

Once the AKD_Start_MotionTask is In Process (.IP), the index of the Motion Task which is currently being executed can be monitored in the Response Assembly (Executing Block Number; Byte 1 for either AKD1G or AKD2G axis 1 or AKD2G Byte 65 axis 2). For more information on the Executing Block # see the Response Assembly Data Structure.

Response Assembly Data Structure	Executing Block #
AKD1G or AKD2G Axis 1	Byte 1
AKD2G Axis 2	Byte 65

Condition Ladder Diagram Action

<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>None</td></tr>
<tr><td>.EnableIn False</td><td>Unlatch the First_Scan_Bit and set the Step Number to 0.</td></tr>
<tr><td>Instruction Execution</td><td>

On first scan of execution of the AKD_Start_MotionTask AOI the First_Scan_ Bit is False.

Clear the .DN, .IP, .PC, and .ER bits of the AKD_Start_MotionTask AOI.
Clear the Load/Start bit of the Control Word.
Reset Timers 1, 2, and 3.
Set Timer_1 preset to 500 ms, Timer_2 preset to 100 ms, and Timer_3 preset to 50 ms.
Set the Step Number to 1 and latch the First Scan bit.

NOTE
Note the AKD_Start_MotionTask AOI differs from most EtherNet/IP methods in the AKD in that it uses Bit 1, Start Block instead of the usual Bit 0 Load/Start to execute.
Step Number 1:

Check to see if the Start Block (bit 1 of the Control Word) is High. It takes one communication cycle to reset the bit. Start Timer_2 (100 ms)
When Timer_2 is Enabled, unlatch the Load/Start bit (bit 0) and Start Block (bit 1) of the Control Word.
When Timer_2 is Done, or the Start Block (bit 1 of the Control Word) turns off or is OFF, set the Step Number to 2.

Step Number 2: If the Step Number is 2 and the Load/Complete bit in Status Word 2 is OFF, move the MT_NUM input value of the AKD_Start_MotionTask AOI into the Start Block (Byte 1 for AKD or AKD2G Axis 1 or Byte 65 for AKD2G Axis 2) of the Command Assembly. Set the Step Number to 3.
Step Number 3:

Latch the Start Block (bit 1 of the Control Word).
Set the Step Number to 4.

</td></tr>
</table>

Condition	Ladder Diagram Action	

	Step Number 4:	• Start Timer_1 (500 ms).
		• Check to see if the Response Assembly Data Structure Executing Block # returns confirmation that the requested Motion Task or Block Number (MT_NUM) is executing.
		    • If MT_NUM is executing, then set the .DN bit of the AOI and unlatch the On Target Position in the Response Assembly. Set the Step Number to 5.
		    • If the Response Assembly never confirms the Executing Motion Task Number in the 500 ms time window, later in the logic there is error checking for this.

	Step Number 5:	• Monitor the Profile_In_Progress (bit 0 in Status Word 1) in the Response Assembly Data Structure. • While Profile_In_Progress is in process, latch the .IP bit. This latch should begin at the start of the move. • Set the Step Number to 6.
	Step Number 6:	• If the Step Number is 6 and the Profile_In_Progress bit are OFF indicating the trajectory is .DN (Done) and the On Target Position bit is ON then the move is complete. **NOTE** Note this is sourced from bit 11 of the DRV.MOTIONSTAT (AKD1G) or AXIS#.MOTIONSTAT (AKD2G) where the trajectory is Done and the PL.FB is in the In Position Window defined by MT.TPOSWND (AKD1G) or AXIS#.SETTLE.P (AKD2G). Unlatch the Start Block (bit 1 of the Control Word) and start Timer_3 (50 ms). After 50 ms set the Step Number to 7. Per the rung comments, this small time delay was added to allow enough time to reset the Start Block bit, otherwise an execution conflict may occur if a Start_MotionTask is triggered too soon.
	Step Number 7:	• Unlatch the .IP (In Process) bit and latch the .PC (Process Complete) bit.
	Error Checking:	• Any Error (Fault) unlatches the .DN and .IP bits of the AKD_Start_MotionTask AOI output and also unlatches the Start Block (bit 1 in Control Word). It also latches the .ER bit of the AKD_Start_MotionTask AOI. The Step Number is set to -1 for all errors. • Fault -1: Drive isn't Enabled. • Fault -2: Drive isn't Homed. • Fault -3: If the Step Number is 4 and the Executing Motion Task Number is never confirmed by the Response Assembly in the 500 ms time. • Error -4: If the Smooth Stop or Hard Stop bit is executed. This is typically set by the AKD_Smooth_Stop or AKD_ Shutdown AOI while the AKD_Start_MotionTask AOI is

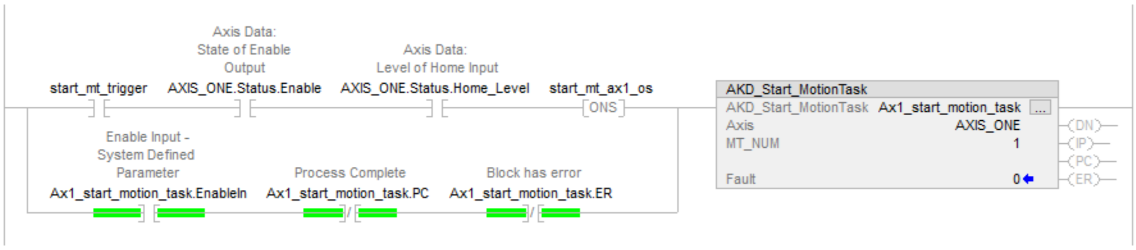
Condition	Ladder Diagram Action
	executing. • Fault -5: If the Response Type from the Response Assembly is 16#14 Response Type 0x14 - Command/Response Error.

Changes to Axis Status bits

Bit Name	State	Meaning
Profile_In_Progress	On	While executing, the Profile_In_Progress bit (bit 0 in Status Word 1) will turn ON. When the motion is complete, it will turn OFF. (Bit 0 in Status Word 1 is also known as In Motion for AKD2G.)
Block In Execution	On/Off	While executing, Block In Execution bit (bit 1 in Status Word 1) will turn ON. When the motion is complete, it will turn OFF.
In Position	On	If the Block In Execution (bit 1 in Status Word 1) is turns OFF (motion complete) and the In Position bit (bit 2 in Status Word 1) turns ON, then the AKD_Start_MotionTask AOI execution was successful, .PC (Process Complete).

Example of Usage/Programming Guidelines

A normally open (N.O.) contact called start_mt_trigger is used to trigger the request. Interlocks are used from the Axis status to ensure the AOI cannot be triggered unless the Axis is Enabled and Homed. On trigger, a One Shot triggers the .EnableIn of the AKD_Start_MotionTask. A parallel branch is used to seal-in the .EnableIn for as long as the AOI executes and either 1) the .PC bit is set, success or 2) the .ER bit is set, Error. Note the one N.O. Contact and the two N.C. contacts in the parallel branch. The MT_NUM field entry can be a Constant value as shown in the example or a Tag name used to make the value variable in the PLC program. The Fault field is Read-Only and no entry is required.



Troubleshooting: Summary of Reasons for Errors/Failure

- If the drive is not enabled (Enabled State, bit 7 in Status Word 1 is not True = 1) then the AKD_Start_MotionTask AOI's Fault field will report a -1.
- If the drive is not Homed (Homed status, bit 5 in Status Word 1 is not True = 1) then the AKD_Start_MotionTaskAOI's Fault field will report a -2.
- If the Step Number is 4 and the Executing Motion Task Number is never confirmed by the Response Assembly in the 500 ms time window, set the Fault Number to -3.
- If the Smooth Stop or Hard Stop bit is executed then set the Fault Number to -4. This is typically set by the AKD_Smooth_Stop or AKD_Shutdown AOI while the AKD_Start_MotionTask AOI is executing.
- Response Type is 16#14 Response Type 0x14 - Command/Response Error. This can happen when:
  - The axis is in the wrong Operation Mode. Prior to executing the AKD_Start_MotionTask AOI DRV.OPMODE (AKD1G) or AXIS#.OPMODE (AKD2G) must be set to Position.
  - The axis is set to the wrong Command Source. Prior to executing the AKD_Start_MotionTask AOI DRV.CMDSOURCE (AKD1G) must be set to Service and AKD2G must be set to Fieldbus.

The AOI's Fault field entry will report a -5 (Response Type #14) in these cases.

- Invalid Motion Task parameters for the given Motion Task or Following Motion Task. On attempt to start the following warnings may appear:

AKD1G Warnings Table

- Conflict by triggering the AOI while another AOI is in execution.
- Attempt to trigger an invalid or non-defined/non-existing Motion Task.

Fault ("F") Warning ("n")	Message/Warning	Cause	Remedy	Drive Response to Fault
n160	Motion task activation failed.	Activation of the motion task failed due to incompatible parameters, or motion task does not exist. This warning can appear upon an MT.MOVE command.	Activation of any new motion or using of DRV.CLRFAULTS will clear the warning. Check motion task settings and parameters to make sure that the values entered will produce a valid motion task.	None

AKD2G Warnings Table

Warning	Description	Cause	Remedy
W6009	Following motion failed, check motion parameters.	Activation of the motion task failed due to incompatible parameters or motion task does not exist.	Activation of any new motion or use of AXIS#.CLRFAULTS clears the warning. Check following motion task settings and parameters to make sure the values entered will produce a valid motion task.
W6014	Motion task activation failed.	Activation of the motion task failed due to incompatible parameters or motion task does not exist. This warning can appear upon an AXIS#.MT.MOVE command.	Activation of any new motion or use of AXIS#.CLRFAULTS clears the warning. Check motion task settings and parameters to make sure that the values entered will produce a valid motion task.

Step Summary

Step Number	Operation/Result
.EnableIn	On first scan of execution of the AKD_Start_MotionTask AOI First_Scan_Bit is False. Clear the .DN, .IP, .PC, and .ER bits of the AKD_Start_MotionTask AOI. Clear the Load/Start bit of the Control Word. Reset Timers 1, 2, and 3. Set Timer_1 preset to 500 ms, Timer_2 preset to 100 ms, and Timer_3 preset to 50 ms. Set the Step Number to 1.
1	If the Step Number is 1, check to see if the Start Block bit of the Control Word is High. It takes one communication cycle to reset it so start Timer_2 (100 ms). When Timer_2 is Enabled, unlatch the Load/Start bit and Start Block bit of the Control Word. When the Start Block bit of the Control Word is OFF and Timer_2 is Done, set the Step Number to 2.
2	If the Load/Complete in Status Word 2 is OFF then move the MT_NUM input value of the AOI into the Start Block byte1 of the axis structure Command Assembly. Set Step Number to 3.
3	Latch the Start Block (bit 1 of the Control Word) in the axis structure Command Assembly. Set the Step Number to 4.
4	Start Timer_1 (500 ms). Check to see if the Response Assembly returns confirmation that the requested Motion Task/Block Number (MT_NUM) is executing; if it is, then set the .DN bit of the AOI and unlatch the On Target Position in the Response Assembly. Set the Step Number to 5. Note if the Response Assembly never confirms the executing Motion Task Number in the 500 ms time window, later in the logic there is error checking for this.
5	While the Profile_In_Progress bit is on (assume the move is In Process [.IP]), latch the .IP bit. This latch should begin at the start of the move. Set the Step Number to 6.
6	If the Step Number is 6 and Profile_In_Progress bit is OFF indicating the trajectory is done and the On Target Position bit is ON then move is complete (note this is based on bit 11 of the DRV.MOTIONSTAT (AKD1G) or AXIS#.MOTIONSTAT (AKD2G) where the trajectory is Done and the PL.FB is in the In Position Window defined by MT.TPOSWND (AKD1G) or AXIS#.SETTLEP (AKD2G). Unlatch the Start_Block bit and start Timer_3 (50 ms). After 50 ms set the Function_Step Number to 7. Per the rung comments, this small time delay was added so there is enough time to reset the bit at the drive, otherwise the AOI will have problems if another Start_MotionTask is triggered too soon.
7	Unlatch the .IP (In Process) bit and latch the .PC (Process Complete) bit.

Fault Codes / Values

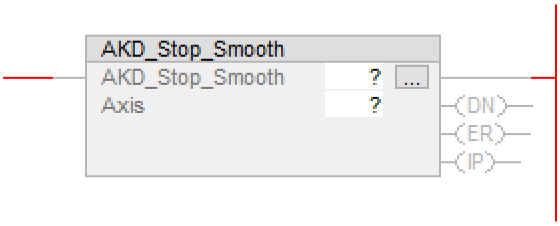
Any Error (Fault) unlatches the .DN, .IP, and .ER bits of the AOI output and also unlatches the Start Block bit (bit 1 in Control Word). Step Number is also set to -1 for all errors.

Error #	Operation/Result
-1	Axis isn't Enabled. Set Fault value to -1.
-2	Axis isn't Homed. Set Fault value to -2.
-3	If the Step Number is 4 and the Executing Motion Task Number is never confirmed by the Response Assembly in the 500 ms time, set the Fault value to -3.
-4	If the Smooth Stop or Hard Stop bit is executed then set the Fault value to -4. This is typically set by the AKD_Smooth_Stop or AKD_Shutdown AOI while the AKD_Start_MotionTask AOI is executing.
-5	If the Response Type from the Response Assembly is 16#14 (Response Type 0x14 - Command/Response Error) then set the Fault Number to -5.

Revision History

Revision Number	Description/Notes	Date of Revision
v2.1	Initial Version	05/05/2016
v2.2	• Changed .IP status to be based on Profile_In_Progress bit instead of "not on target position" • Added N.C. contact for No Profile_In_Progress status in addition to the existing On Target Position status to unlatch the Start_ Block bit of the Control Word. • This fixed the issue of the AOI running successfully once but then being immediately Done and .PC bit complete before ever executing the subsequent move. • AOI library v5.0 revision	07/19/2019
v3.0	• Updated to synchronize AOI with new AKD_Data data type which the Axis structure uses. • AOI library v6.0 revision	07/16/2021

7.28 AKD_Stop_Smooth



Description

Use the AKD_Stop_Smooth instruction to end any controlled motion currently in progress for the given axis and to decelerate to a stop without disabling the axis.

Compatibility

The AKD_Stop_Smooth AOI is compatible with both the AKD1G and AKD2G EtherNet/IP drives.

Required Command Source and Operation Mode

AKD1G

DRV.CMDSOURCE = Any/All

DRV.OPMODE = Any/All

AKD2G

AXIS#.CMDSOURCE = Any/All

AXIS#.OPMODE = Any/All

The AKD_Stop_Smooth AOI calls the DRV.STOP (AKD1G) or AXIS#.STOP (AKD2G) by setting the Smooth Stop (bit 4 in Control Word). See WorkBench Help for more information on these commands.

Use the AKD_Stop_Smooth instruction to end any controlled motion currently in progress for the given axis and to decelerate to a stop without disabling the axis.

Use the AOI to:

- Stop a specific motion process such as jogging or moving.
- Stop the axis completely.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_Stop_Smooth	AKD_Stop_Smooth	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_Internal Tag name of the AKD2G_Drive AOI for the given axis.

Structure

The following fields are populated automatically with Read Only data once the Operands above are entered.

Mnemonic	Data Type	Description
.EnableIn	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	BOOL	The Done bit indicates the Smooth Stop command has completed and the motion has stopped. This occurs when the Profile_In_Progress bit (Also known as In Motion in AKD2G); bit 0 in Status Word 1) turns OFF.
.ER	BOOL	There is no internal logic of AKD_Stop_Smooth that sets the .ER output bit.
.IP	BOOL	The In Process bit turns ON during execution of the AKD_Stop_Smooth AOI and while the Profile_In_Progress bit in Status Word 1 is ON. When the motion has stopped, indicated by the Profile_In_Progress bit turning OFF, the .IP bit turns OFF.

Execution

This is the same behavior as a DRV.STOP (AKD1G) or AXIS#.STOP (AKD2G).

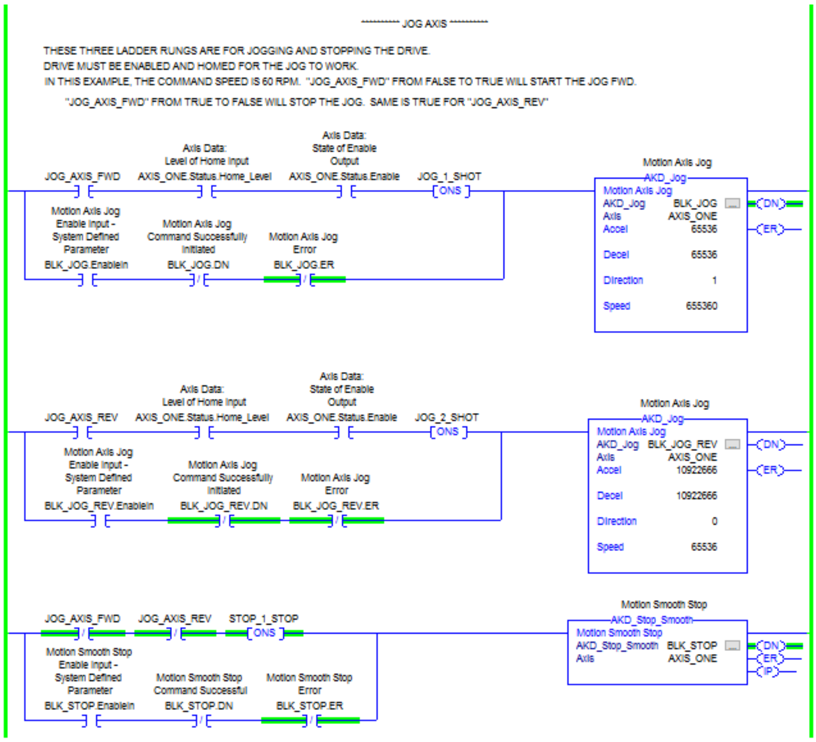
Condition	Ladder Diagram Action
Prescan	Set the Step Number to 0.
.EnableIn False	Set the Step Number to 0.
Instruction Execution	**Step Number 0:** If the AKD_Stop_Smooth AOI is Enabled and the Step Number is 0, unlatch the .IP (In Process) bit and the .DN (Done) bit and set the Step Number to 1. **Step Number 1:** If the Step Number is 1, latch the Smooth Stop bit (bit 4 in Control Word) and set the Step Number to 2. **Step Number 2:** • If the Step Number is 2 and a Profile Move (bit 0 in Status Word 1) is In Progress turn ON the AKD_Stop_Smooth AOI’s .IP output. • If the Step Number is 2 and a Profile Move is not In Progress (stop complete), unlatch the Smooth Stop (bit 4 in Control Word) and turn ON the AKD_Stop_Smooth AOI’s .DN bit.

Changes to Axis Status bits

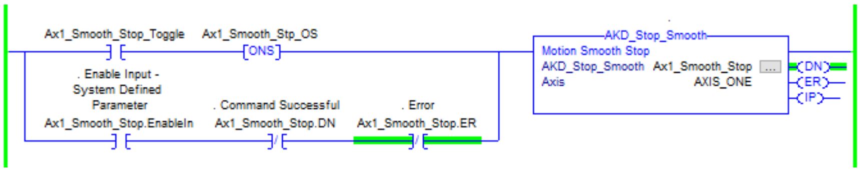
- Profile_In_Progress (bit 0) is ON while the stop is executing and turns OFF when the stop is complete.

Example of Usage/Programming Guidelines

The Sample project makes use of AKD_Stop_Smooth to stop Jogging. Note the toggle contacts (N.O) Jog_Axis_Fwd and Jog_Axis_Rev are used to trigger the AKD_Jog AOI (one for forward and one for reverse). When one of the two toggles goes from True to False (stop Jogging) the falling edge condition and One Shot triggers the AKD_Stop_Smooth AOI.



The AKD_Stop_Smooth AOI can also simply be used to trigger a Smooth Stop (using DRV.STOP or AXIS#.STOP) through the EtherNet/IP Control Word on any conditions determined by the programmer:



Troubleshooting: Summary of Reasons for Errors/Failure

- None

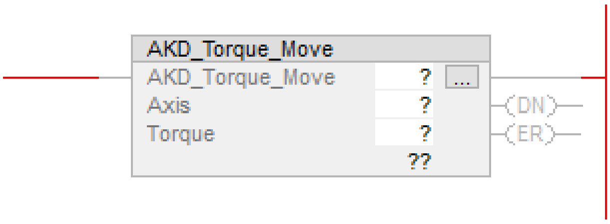
Step Summary

Step Number	Operation/Result
.EnableIn (Step 0)	• Unlatch .IP and .DN bits of AKD_Smooth_Stop AOI. • Set the Step Number to 1.
1	• Latch the Smooth Stop (bit 4 in Control Word). • Set Step Number to 2.
2	• If the Profile_In_Progress bit from Status Word of Response Assembly is ON then turn ON .IP bit of the AKD_Smooth_Stop AOI. • If the Profile_In_Progress bit is OFF, unlatch the Smooth Stop bit in the Control Word and set the .DN bit of the AKD_Smooth_Stop AOI to ON. (Smooth Stop complete)

Revision History

Revision Number	Description/Notes	Date of Revision
	Original version 02/23/2011	02/23/2011
v2.3	• Added an in progress In Process (.IP) bit to show when the motor is stopping. • Removed the command timer to allow unlimited time for the motor to stop.	
v4.0	• There was a race with the Control Word Smooth Stop bit. Removed the logic in .EnableIn False and did the reset as part of the normal logic. • AOI library v5.0 revision	03/1/2016
v5.0	• Updated to synchronize AOI with new AKD_Data data type which the Axis structure uses. • AOI library v6.0 revision	07/16/2021

7.29 AKD_Torque_Move



Description

Use the AKD_Torque_Move instruction to move an axis at a Constant Torque.

Compatibility

The AKD_Torque_Move is compatible with both the AKD1G and AKD2G EtherNet/IP drives.

Required Command Source and Operation Mode

AKD1G

DRV.CMDSOURCE = Fieldbus

DRV.OPMODE = Torque

AKD2G

AXIS#.CMDSOURCE = Fieldbus

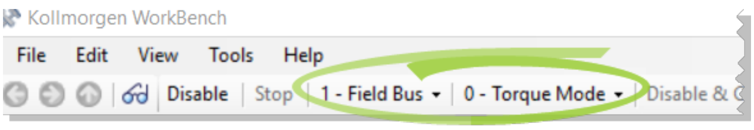
AXIS#.OPMODE = Torque

Use the AKD_Torque_Move instruction to move an axis at a Constant Torque. The instruction may take multiple scans to execute due to the time required for transmission of the message (communications) and time for the axis to execute the command. The Done (.DN) bit is not set until the Load Complete bit confirms the request for a Torque Move was successful. This AOI utilizes Command Type 0x05 - Torque and the Load/Start bit in the Control Word internally.

The AKD_Torque_Move AOI does not change the axis Command Source or axis Operation Mode on execution. The AKD_Torque_Move AOI sets the Command Type to Command Type 0x05 - Torque and uses the Setpoint to set the IL.CMD (AKD1G) or AXIS#.FBUS.IL.CMD (AKD2G) which shows the target torque from EtherNet/IP in axis units. The final destination will be AXIS#.IL.CMD.

Note the Torque Move does not use the IL.CMDU (AKD1G) or AXIS#.IL.CMDU (AKD2G). The value used in the AOI for the Torque Setpoint is in milliamps where XXXX over EtherNet/IP equals X.XXX in the AKD inWorkBench Units. For example, a torque value of 1 in the AOI is 0.001 or 1mA, 1000 is 1.000 or 1000mA or 1A, etc. in WorkBench.

The AKD_Torque_Move will fail (.ER; Error) if the axis Command Source is not Fieldbus and the Operation Mode is not set to Torque prior to triggering and executing the AOI.



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 or use the AKD_Fault_Reset AOI.)
- Axis is enabled (Set Enable bit in the Control Word). Set Enable bit 7 in Control Word and the Status Word 1 bit 7 Enable State indicates the Enable status using the AKD_Enable AOI.
- 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.)

While a Torque Move is active the Profile In Progress (bit 0 inStatus Word 1) will turn ON and the In Position bit (bit 2 in Status Word 1) will be cleared.

AKD1G

NOTE

The Profile_In_Progress bit (bit 0 in Status Word 1) was not set during a Torque Move prior to firmware 1-17-03-000 (AKD1G).

With DRV.OPMODE set to 0 (Torque), the axis will set the Profile In Progress bit (bit 0 in Status Word 1) as long as it has not detected zero velocity. Detection of zero velocity can be configured using the parameters CS.TO and CS.VTHRESH.

Note the Controlled Stop settings are used to detect zero velocity for the Profile In Progress bit). See the Torque Moves and Control Word sections in the AKD1G EtherNet/IP Communication Manual for more information.

The Direction Status bit (Bit 4, Current Direction) in Status Word 1 will reflect the actual direction of motion.

AKD2G

NOTE

In AKD2G the Profile_In_Progress bit is known as In Motion. This bit was not set during a Torque Move prior to firmware 02-07-02-000 (AKD2G).

Detection of zero velocity can be configured using the AXIS#.ZERO commands for setting the conditions for zero speed.

Stopping The Torque Move

To stop a Torque Move, use the AKD_Stop_Smooth AOI where the Smooth Stop will utilize the DRV.STOP (AKD1G) or AXIS#.STOP (AKD2G). Note the IL.CMD (AKD1G) and AXIS#.IL.CMD (AKD2G) will be set to 0 on Smooth Stop. For AKD2G the AXIS#.IL.CMD will be set via the AXIS#.FBUS.IL.CMD over EtherNet/IP.

The AKD_Torque_Move AOI performs the following actions internally:

1. Command Type is set to 16#05 (Command Type 0x05 - Torque).
2. Torque Setpoint value is set in Bytes 4-7 of the Command Assembly
3. Load/Start bit (bit 0 in Control Word) is set to make the Current Torque Setpoint effective.

AKD1G

If a Hard Stop (via the AKD_Shutdown AOI) is used and the DRV.DISMODE is set for Controlled Stop then Disable, note the following in WorkBench Help.

When configuring the controlled stop feature, please note the following:

- If the HW limit switch is active and any of the other CS activated, the only difference will be that in this case the DRV.DISTO will limit the time before disabling the drive.
- If the value of DRV.OPMODE is torque mode, the drive will execute the controlled stop by switching internally to velocity mode. Therefore, it is recommended to tune the velocity loop properly, even though the drive might only be used in torque mode.
- Set DRV.DISTO to an appropriate value that will allow the motor to decelerate from any velocity to 0 with DRV.DEC . This value must also allow the motor to afterwards remain within VL.FB for CS.TO consecutively within 0 ± CS.VTHRESH.
- CS.DEC: Deceleration ramp that is used for the controlled stop.

The drive issues a fault F703 in case that the DRV.DISTO counter expires during a controlled stop procedure. For more information see Controlled Stop in WorkBench Help.

AKD2G

The Hard Stop (AKD_Shutdown) always uses a Controlled Stop in the AKD2G.

Other Methods

Other methods include setting the Torque Setpoint in the AKD_Torque_Move AOI to zero and retriggering or using the AKD_Disable AOI to disable the axis. Which method is used is application-dependent and up to the programmer.

Monitoring Actual Torque (Current)

There are two primary methods:

Method 1: Response Type 0x05 - Actual Torque

This I/O Response Assembly is used to return the Actual Torque (current) of the motor in milliamps. Data will be received in the Data field, Bytes 4-7 for AKD1G or AKD2G Axis 1; Bytes 68-71 for AKD2G Axis 2. Set Response Type = 0x05 (Byte 3 for AKD1G or AKD2G Axis 1; Byte 67 for AKD2G Axis 2) in the Command Assembly to read this value.

Method 2: Preferred

AKD1G

IL.CMD and/or IL.FB can be dynamically mapped. See Example of AKD1G Dynamic Mapping With Studio 5000 for more information.

AKD2G

AXIS#.IL.CMD and/or AXIS#.IL.FB can be dynamically mapped. See Example of AKD2G Dynamic Mapping With Studio 5000 for more information.

Operands

These entries are required by the user.

Operand	Type	Format	Description
AKD_ Torque_ Move	AKD_ Torque_ Move	Tag	Tag name for this instance of the AOI.
Axis	AKD_Axis	Tag	Tag for which the Axis is declared. Must match the Axis_Internal Tag name of the AKD_Drive AOI or Axis_Internal or Axis2_ Internal Tag name of the AKD2G_Drive AOI for the given axis.
Torque	DINT	Constant or Tag	Constant or Tag Setpoint where the value is in milliamps for the Torque Move command.

Structure

The following fields are populated automatically with Read Only data once the Operands above are entered.

Mnemonic	Type	Description
.EnableIn	BOOL	The Enable Input bit indicates the instruction is Enabled.
.EnableOut	BOOL	The Enable Output bit is the output of the Enable Input (.EnableIn) bit.
.DN	BOOL	The Done (.DN) bit indicates the Load Complete status bit has confirmed the request to execute the Torque Move.
.ER	BOOL	The Error (.ER) bit is set if the Response Type is Response Type 0x14 - Command/Response Error or the Command Timeout timer expires.

Execution

<table>
<tr><th>Condition</th><th>Ladder Diagram Action</th></tr>
<tr><td>Prescan</td><td>Unlatch OS_LoadStart bit.
Reset Command_Timeout timer.
Move Command Timeout value into Command_Timeout timer preset (PRE).
Clear the first 8 Bytes of the internal Command Bytes array (Bytes 0-7).</td></tr>
<tr><td>.EnableIn False</td><td>Unlatch the OS_LoadStart bit
Reset the Command_Timeout timer
Clear the internal array Command_Bytes (first 8 Bytes; Bytes 0-7).</td></tr>
<tr><td>Instruction Execution</td><td>On .EnableIn
One Shot to set the Step Number to 0.
Unlatch (Clear) the .DN and .ER bits of the AOI.
Set Command_Bytes in the Command Assembly to 16#05 (Command Type 0x05 - Torque).
Move the Torque Setpoint value set in the AOI into the Command Assembly Data Bytes.
Bytes 4-7 (AKD1G or AKD2G Axis 1) or Bytes 68-71 Axis 2 for AKD2G).
Step Number 0: If the Step Number is 0 and the Load/Complete is cleared then latch the Load/Start bit of the Control Word (Byte 0) and set the Step Number to 1.
Load/Complete Confirmed: If the Load/Complete is confirmed in Status Word 2 then unlatch the Load/Start bit in the Control Word (Byte 0).
Step Number 1:
If Step Number is 1 (Load/Start bit is set to send command) and the Response Type 0x14 - Command/Response Error in the Response Assembly indicates the Request/Command failed. Declare an error and latch the Error (.ER) bit of the AOI and set the Step Number to -1.
If the Step Number is 1 and the Command_Timeout.PRE (Preset) is non-zero AND the timeout timer expires (due to the request not being confirmed in the allotted time), latch the .ER bit of the AOI and set the Step Number to -2.
If the Step Number is 1 and the Load Complete (bit 7 in Status Word 2) is True (Success) then set the Done (.DN) bit of the AKD_Torque_Move AOI and set the Step Number to 2 (success of execution).
<ol start="4">
On any error, unlatch the Load/Start bit.
Move the Control Word values into the Command_Bytes array and copy the entire internal Command_Bytes array (Bytes 0 through 7) into the Command Assembly (output data).</td></tr>
</table>

Changes to Axis Status bit

NOTE

AKD1G

When making a Torque Move, the In Motion (Profile_In_Progress) bit never turned ON prior to firmware 1-17-3-0. This is necessary for using the AKD_Stop_Smooth AOI. The conditions for turning the bit ON/OFF in this mode is based on sensing zero velocity and the setup of CS.TO and CS.VTHRESH parameters.

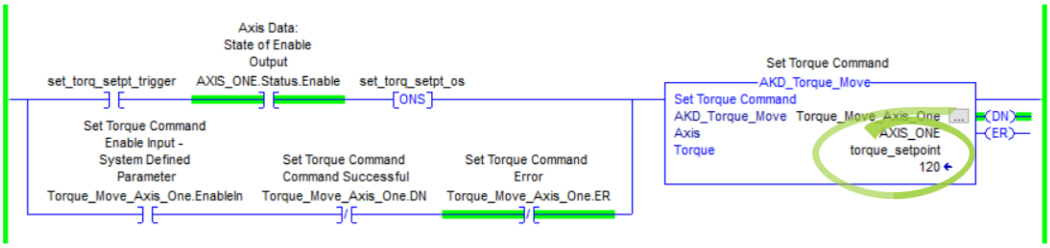
AKD2G

When making a Torque Move, the In Motion (Profile_In_Progress) bit never turned ON in the firmware prior to the bug fix in firmware 02-07-02-000. This is necessary for using the AKD_Stop_Smooth AOI. The conditions for turning the bit ON/OFF in this mode is based on sensing zero velocity and the setup of AXIS#.ZERO and AXIS#.ZEROV parameters.

Example of Usage/Programming Guidelines

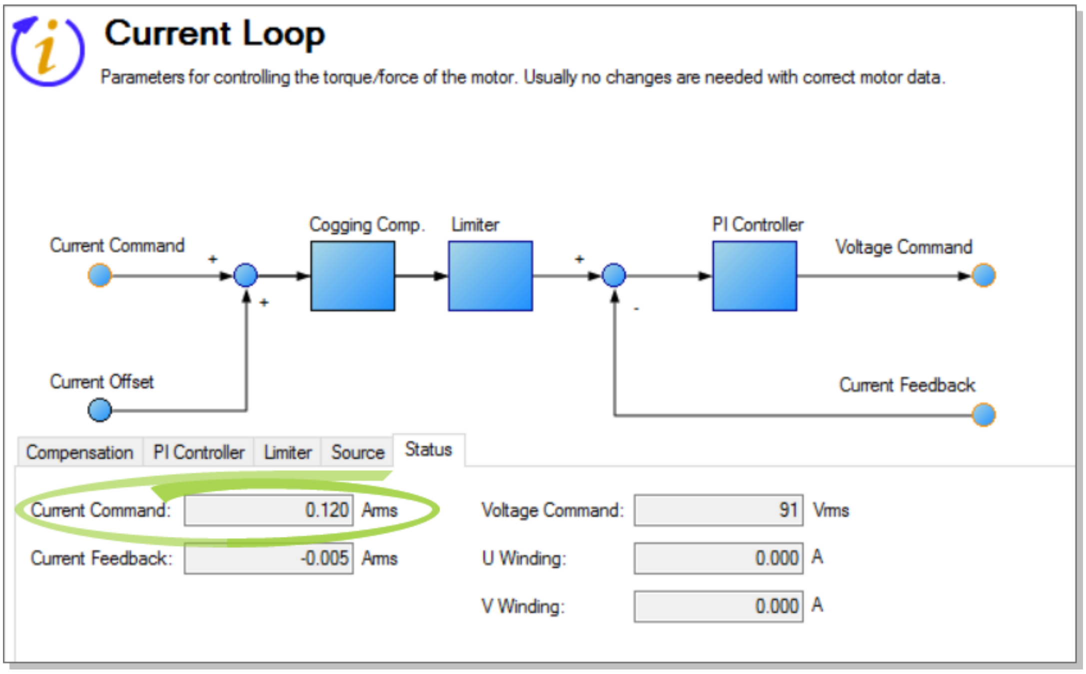
In the example shown, a N.O. Contact with the Tag set_torq_setpt_trigger serves as the trigger (Toggle) for commanding the AKD_Torque_Move. The logic shows a N.O. Contact for the Enable Status of Status Word 1 to be True (Enabled) serving as an interlock prior to triggering a One Shot to the .EnableIn of the AKD_Torque_Move AOI. A parallel branch provides a seal-in based on when the AKD_Torque_Move AOI is enabled (.EnableIn) and two N.C. Contacts are used to seal-in the Enable until either 1) the Torque Move Execution is successful (Done; .DN) or 2) an error occurred and execution failed (Error; .ER).

Set the AKD_Torque_Move with the desired Setpoint value prior to the trigger. In this example, a Tag was entered into the AKD_Torque_Move's Torque field entry so the value may be varied in the program. Note a constant value can also be used.

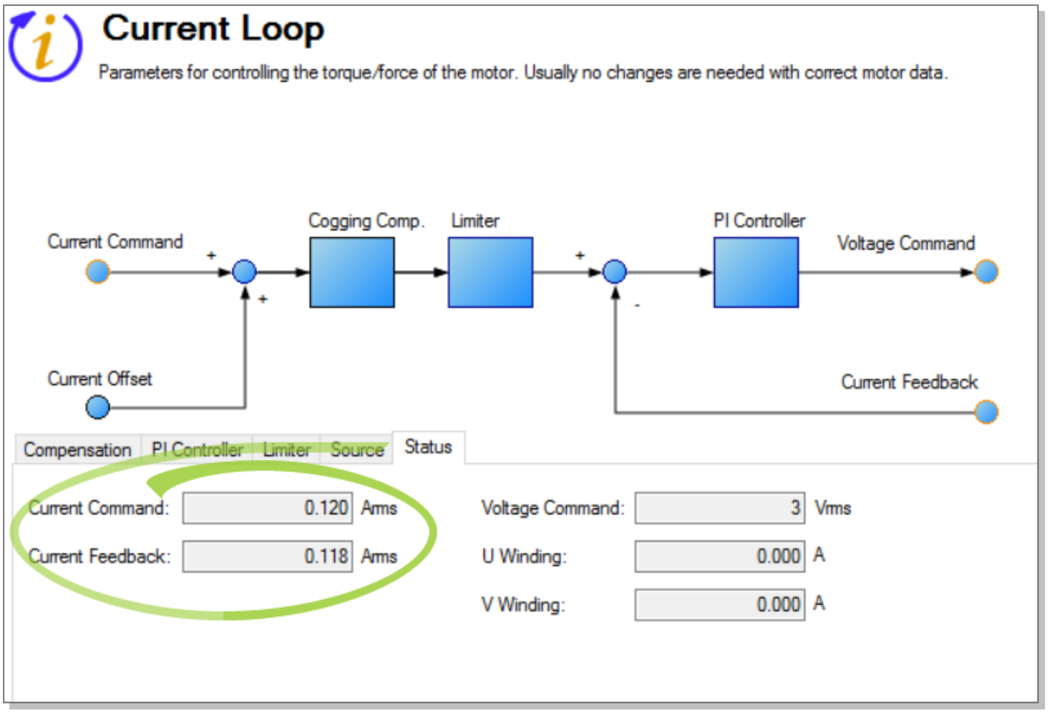


The current can be monitored in WorkBench or over EtherNet/IP.

WorkBench Current Loop Screen - No Load



WorkBench Current Loop Screen - Loaded



To change the Torque Setpoint, the new Setpoint must be entered and the AKD_Torque_Move AOI retriggered in order to update the Setpoint Command.

Current Limits

It is the programmer’s responsibility to limit the current command used with the AKD_Torque_Move AOI.

Note IL.LIMITN and IL.LIMITP (AKD1G) or AXIS#.IL.LIMITN and AXIS#.IL.LIMITP (AKD2G) can be set using WorkBench or over EtherNet/IP.

Speed Limits

The AKD Torque Drive Op Mode is a pure torque controller (current loop regulator). There is no line speed override mechanism where the drive switches to Velocity Mode under an unloaded condition.

AKD1G

However, there is a parameter called IL.VLIMIT that sets a velocity limit in Torque Mode. The method reduces the current, if necessary, to keep the VL.FB at or below the IL.VLIMIT.

Be aware that if the value is set too low then it will affect the Current Loop by reducing the current to maintain the velocity limit and there will not be enough headroom for the Current Loop to regulate about the Torque Setpoint. For example, attempting to set the IL.VLIMIT to 1 rpm or less.

AKD2G

The AKD2G drive does not have an equivalent parameter to the AKD1G drive's IL.VLIMIT parameter. Therefore, the monitoring and control of any speed override is the programmer’s responsibility.

Torque or Force Applications Requiring Accuracy

- Data sampling will be affected by the RPI scan setting and potentially the PLC scan time as well.
- When wanting to convert amps to torque or force mathematically: There is no torque or torque percent parameter in the AKD; only amps.
- If using AKM motors, for example, the Kt constant (i.e. N*m/Arms) is +/-10% from motor to motor even if the motors have identical part numbers. This is a variance intrinsic in the motor manufacturing, etc.
- Note losses that are present in the power transmission of every system such as belt compliance, backlash of teeth, friction, etc. that will affect the accuracy of your data since it is based only on motor load (amps).
- The only way to truly determine or measure the torque or force applied at the load with accuracy is via torque or force transducers and to use data acquisition hardware/software to record the transducers’ information.

Troubleshooting: Summary of Reasons for Errors/Failure

- Axis is faulted.
- Axis is not Enabled (axis inactive)
- DRV.CMDSOURCE (AKD1G) or AXIS#.CMDSOURCE (AKD2G) is not set to Fieldbus
- DRV.OPMODE (AKD1G) or AXIS#.OPMODE (AKD2G) is not set to Torque.
- Response Type is 16#14 Response Type 0x14 - Command/Response Error (Error; i.e. data out of range such as the Torque Setpoint exceeds the peak of the motor or drive or current limits.)

Step Summary

Step Number	Operation/Result
On .EnableIn	One Shot and set Step Number to 0 and unlatch .DN and .ER bits
Enabled	• Set Command_Bytes 2 equal to 16#5 (Torque Move). Move the Torque Setpoint value set in the AOI into the Command Assembly Data Bytes. The destination is ultimately Bytes 4-7 (AKD1G or Axis 1 for AKD2G) or Bytes 68-71 (Axis 2 for AKD2G). • On Load Complete unlatch Load/Start bit. On any error unlatch the Load/Start bit. • Move the Control Word values into the Command_Bytes array and also copy the entire internal Command_Bytes array Bytes 0 through 7 into the Command Assembly (output data).
0	Once Load Complete bit is clear (ready for a new command), latch theLoad/Start bit to send a new command. Set the Step Number to 1.
1	Check for a Response Type 16#14 or Command Timeout errors. If Load Complete, then Success and set the .DN bit of the AKD_Torque_Move AOI. Set the Step Number to 2.
2	If Step Number is 2, the AOI execution was successful.
-1	On Step 1 (Send new command) Response Message Type came back as 16#14 (Error).
-2	On Step 1 (Send new command) Command Timeout timer timed out.

Revision History

Revision Number	Description/Notes	Date of Revision
	Original version	02/23/2011
v2.1	• Fixed COP to Attribute_Actual. Copying 16 Bytes worth of data; only 4 needed. • AOI library v5.0 revision	03/05/2015
v3.0	• Updated to synchronize AKD_Torque_Move AOI with new AKD_Data data type which the Axis structure uses. • AOI library v6.0 revision	06/02/2021

8 AKD1G Appendices

8.1 AKD1G EtherNet/IP Objects and Attributes

The following table provides supported attributes.

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 WorkBench Online Help.
101	Home Move	Set	BOOL	Initiate a home move.

8.2 AKD1G Parameter Listing

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 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 (ID)	Parameter	Hex	Data Type	Data Size	Access
1	AIN.CUTOFF	1	Float	4 Byte	Read/Write
2	AIN.DEADBAND	2	Float	2 Byte	Read/Write
3	AIN.ISCALE	3	Float	4 Byte	Read/Write
4	AIN.OFFSET	4	Float	2 Byte Signed	Read/Write
5	AIN.PSCALE	5	Position	8 Byte Signed	Read/Write
6	AIN.PSCALE (32 bit version)	6	Position	4 Byte Signed	Read/Write
7	AIN.VALUE	7	Float	2 Byte	Read Only
8	AIN.VSCALE	8	Velocity	8 Byte	Read/Write
10	AOUT.ISCALE	A	Float	4 Byte	Read/Write
11	AOUT.MODE	B	Integer	2 Byte	Read/Write
12	AOUT.OFFSET	C	Float	2 Byte Signed	Read/Write
13	AOUT.PSCALE	D	Position	8 Byte Signed	Read/Write
14	AOUT.PSCALE (32 bit version)	E	Position	4 Byte Signed	Read/Write
15	AOUT.VALUE	F	Float	8 Byte Signed	Read Only
16	AOUT.VALUE (32 bit version)	10	Float	4 Byte Signed	Read Only
17	AOUT.VALUEU	11	Float	8 Byte Signed	Read/Write
18	AOUT.VALUEU (32 bit version)	12	Float	4 Byte Signed	Read/Write
19	AOUT.VSCALE	13	Velocity	8 Byte	Read/Write
20	BODE.EXCITEGAP	14	Integer	1 Byte	Read/Write
21	BODE.FREQ	15	Float	4 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
22	BODE.IAMP	16	Float	4 Byte Signed	Read/Write
23	BODE.INJECTPOINT	17	Integer	1 Byte	Read/Write
24	BODE.MODE	18	Integer	1 Byte	Read/Write
25	BODE.MODETIMER	19	Integer	4 Byte	Read/Write
26	BODE.PRBDEPTH	1A	Integer	1 Byte	Read/Write
27	BODE.VAMP	1B	Velocity	8 Byte Signed	Read/Write
28	CAP0.EDGE	1C	Integer	1 Byte	Read/Write
29	CAP0.EN	1D	Integer	1 Byte	Read/Write
30	CAP0.EVENT	1E	Integer	1 Byte	Read/Write
31	CAP0.FILTER	1F	Integer	1 Byte	Read/Write
32	CAP0.MODE	20	Integer	1 Byte	Read/Write
33	CAP0.PLFB	21	Position	8 Byte Signed	Read Only
34	CAP0.PLFB (32 bit version)	22	Position	4 Byte Signed	Read Only
35	CAP0.PREEDGE	23	Integer	1 Byte	Read/Write
36	CAP0.PREFILTER	24	Integer	1 Byte	Read/Write
37	CAP0.PRESELECT	25	Integer	1 Byte	Read/Write
38	CAP0.STATE	26	Integer	1 Byte	Read Only
39	CAP0.T	27	Integer	4 Byte	Read Only
40	CAP0.TRIGGER	28	Integer	1 Byte	Read/Write
41	CAP1.EDGE	29	Integer	1 Byte	Read/Write
42	CAP1.EN	2A	Integer	1 Byte	Read/Write
43	CAP1.EVENT	2B	Integer	1 Byte	Read/Write
44	CAP1.FILTER	2C	Integer	1 Byte	Read/Write
45	CAP1.MODE	2D	Integer	1 Byte	Read/Write
46	CAP1.PLFB	2E	Position	8 Byte Signed	Read Only
47	CAP1.PLFB (32 bit version)	2F	Position	4 Byte Signed	Read Only
48	CAP1.PREEDGE	30	Integer	1 Byte	Read/Write
49	CAP1.PREFILTER	31	Integer	1 Byte	Read/Write
50	CAP1.PRESELECT	32	Integer	1 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
51	CAP1.STATE	33	Integer	1 Byte	Read Only
52	CAP1.T	34	Integer	4 Byte	Read Only
53	CAP1.TRIGGER	35	Integer	1 Byte	Read/Write
54	CS.DEC	36	Acceleration	8 Byte	Read/Write
55	CS.DEC (32 bit version)	37	Acceleration	4 Byte	Read/Write
56	CS.STATE	38	Integer	1 Byte	Read Only
57	CS.TO	39	Integer	4 Byte	Read/Write
58	CS.VTHRESH	3A	Velocity	8 Byte	Read/Write
59	DIN.ROTARY	3B	Integer	1 Byte	Read Only
61	DIN1.INV	3D	Integer	1 Byte	Read/Write
62	DIN1.MODE	3E	Integer	2 Byte	Read/Write
63	DIN1.PARAM	3F	Varies	8 Byte Signed	Read/Write
64	DIN1.PARAM (32 bit version)	40	Varies	4 Byte Signed	Read/Write
65	DIN1.STATE	41	Integer	1 Byte	Read Only
66	DIN2.INV	42	Integer	1 Byte	Read/Write
67	DIN2.MODE	43	Integer	2 Byte	Read/Write
68	DIN2.PARAM	44	Varies	8 Byte Signed	Read/Write
69	DIN2.PARAM (32 bit version)	45	Varies	4 Byte Signed	Read/Write
70	DIN2.STATE	46	Integer	1 Byte	Read Only
71	DIN3.INV	47	Integer	1 Byte	Read/Write
72	DIN3.MODE	48	Integer	2 Byte	Read/Write
73	DIN3.PARAM	49	Varies	8 Byte Signed	Read/Write
74	DIN3.PARAM (32 bit version)	4A	Varies	4 Byte Signed	Read/Write
75	DIN3.STATE	4B	Integer	1 Byte	Read Only
76	DIN4.INV	4C	Integer	1 Byte	Read/Write
77	DIN4.MODE	4D	Integer	2 Byte	Read/Write
78	DIN4.PARAM	4E	Varies	8 Byte Signed	Read/Write
79	DIN4.PARAM (32 bit version)	4F	Varies	4 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
80	DIN4.STATE	50	Integer	1 Byte	Read Only
81	DIN5.INV	51	Integer	1 Byte	Read/Write
82	DIN5.MODE	52	Integer	2 Byte	Read/Write
83	DIN5.PARAM	53	Varies	8 Byte Signed	Read/Write
84	DIN5.PARAM (32 bit version)	54	Varies	4 Byte Signed	Read/Write
85	DIN5.STATE	55	Integer	1 Byte	Read Only
86	DIN6.INV	56	Integer	1 Byte	Read/Write
87	DIN6.MODE	57	Integer	2 Byte	Read/Write
88	DIN6.PARAM	58	Varies	8 Byte Signed	Read/Write
89	DIN6.PARAM (32 bit version)	59	Varies	4 Byte Signed	Read/Write
90	DIN6.STATE	5A	Integer	1 Byte	Read Only
91	DIN7.INV	5B	Integer	1 Byte	Read/Write
92	DIN7.MODE	5C	Integer	2 Byte	Read/Write
93	DIN7.PARAM	5D	Varies	8 Byte Signed	Read/Write
94	DIN7.PARAM (32 bit version)	5E	Varies	4 Byte Signed	Read/Write
95	DIN7.STATE	5F	Integer	1 Byte	Read Only
96	DOUT.CTRL	60	Integer	1 Byte	Read/Write
97	DOUT.RELAYMODE	61	Integer	1 Byte	Read/Write
99	DOUT1.MODE	63	Integer	1 Byte	Read/Write
100	DOUT1.PARAM	64	Float	8 Byte Signed	Read/Write
101	DOUT1.PARAM (32 bit version)	65	Float	4 Byte Signed	Read/Write
102	DOUT1.STATE	66	Integer	1 Byte	Read Only
103	DOUT1.STATEU	67	Integer	1 Byte	Read/Write
104	DOUT2.MODE	68	Integer	1 Byte	Read/Write
105	DOUT2.PARAM	69	Float	8 Byte Signed	Read/Write
106	DOUT2.PARAM (32 bit version)	6A	Float	4 Byte Signed	Read/Write
107	DOUT2.STATE	6B	Integer	1 Byte	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
108	DOUT2.STATEU	6C	Integer	1 Byte	Read/Write
109	DRV.ACC	6D	Acceleration	8 Byte	Read/Write
110	DRV.ACC (32 bit version)	6E	Acceleration	4 Byte	Read/Write
111	DRV.ACTIVE	6F	Integer	1 Byte	Read Only
114	DRV.CMDSOURCE	72	Integer	1 Byte	Read/Write
115	DRV.DBILIMIT	73	Float	4 Byte	Read/Write
116	DRV.DEC	74	Acceleration	8 Byte	Read/Write
117	DRV.DEC (32 bit version)	75	Acceleration	4 Byte	Read/Write
118	DRV.DIR	76	Integer	1 Byte	Read/Write
120	DRV.DISMODE	78	Integer	1 Byte	Read/Write
121	DRV.DISSOURCES	79	Integer	2 Byte	Read Only
122	DRV.DISTO	7A	Integer	4 Byte	Read/Write
123	DRV.EMUEDIR	7B	Integer	1 Byte	Read/Write
124	DRV.EMUEMODE	7C	Integer	2 Byte	Read/Write
125	DRV.EMUEMTURN	7D	Integer	4 Byte	Read/Write
126	DRV.EMUERES	7E	Integer	4 Byte	Read/Write
127	DRV.EMUEZOFFSET	7F	Integer	2 Byte	Read/Write
129	DRV.ENDEFAULT	81	Integer	1 Byte	Read/Write
130	DRV.HANDWHEEL	82	Integer	4 Byte	Read Only
131	DRV.HWENMODE	83	Integer	1 Byte	Read/Write
132	DRV.ICONT	84	Float	4 Byte Signed	Read Only
133	DRV.IPEAK	85	Float	4 Byte Signed	Read Only
134	DRV.IZERO	86	Float	4 Byte	Read/Write
135	DRV.MOTIONSTAT	87	Integer	4 Byte	Read Only
136	DRV.OPMODE	88	Integer	1 Byte	Read/Write
139	DRV.TYPE	8B	Integer	1 Byte	Read Only
140	DRV.ZERO	8C	Integer	1 Byte	Read/Write
141	FB1.BISSBITS	8D	Integer	1 Byte	Read/Write
142	FB1.ENCRES	8E	Integer	4 Byte	Read Only
143	FB1.IDENTIFIED	8F	Integer	1 Byte	Read Only
144	FB1.INITSIGNED	90	Integer	1 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
145	FB1.MECHPOS	91	Integer	4 Byte	Read Only
146	PL.FBOFFSET	92	Position	8 Byte Signed	Read/Write
147	PL.FBOFFSET (32 bit version)	93	Position	4 Byte Signed	Read/Write
148	FB1.ORIGIN	94	Position	8 Byte	Read/Write
149	FB1.ORIGIN (32 bit version)	95	Position	4 Byte	Read/Write
150	FB1.PFIND	96	Integer	1 Byte	Read/Write
151	FB1.PFINDCMDU	97	Float	4 Byte	Read/Write
152	FB1.POLES	98	Integer	2 Byte	Read/Write
153	FB1.PSCALE	99	Integer	1 Byte	Read/Write
154	FB1.RESKTR	9A	Float	2 Byte	Read/Write
155	FB1.RESREFPHASE	9B	Float	4 Byte Signed	Read/Write
156	FB1.SELECT	9C	Integer	1 Byte Signed	Read/Write
157	FB1.TRACKINGCAL	9D	Integer	1 Byte	Read/Write
158	FBUS.PARAM01	9E	Integer	4 Byte	Read/Write
159	FBUS.PARAM02	9F	Integer	4 Byte	Read/Write
160	FBUS.PARAM03	A0	Integer	4 Byte	Read/Write
161	FBUS.PARAM04	A1	Integer	4 Byte	Read/Write
162	FBUS.PARAM05	A2	Integer	4 Byte	Read/Write
163	FBUS.PARAM06	A3	Integer	4 Byte	Read/Write
164	FBUS.PARAM07	A4	Integer	4 Byte	Read/Write
178	FBUS.PLLTHRESH	B2	Integer	2 Byte	Read/Write
179	FBUS.SAMPLEPERIOD	B3	Integer	1 Byte	Read/Write
180	FBUS.SYNCACT	B4	Integer	4 Byte	Read Only
181	FBUS.SYNCDIST	B5	Integer	4 Byte	Read/Write
182	FBUS.SYNCWND	B6	Integer	4 Byte	Read/Write
183	FBUS.TYPE	B7	Integer	1 Byte	Read Only
184	GEAR.ACCMAX	B8	Acceleration	8 Byte	Read/Write
185	GEAR.ACCMAX (32 bit version)	B9	Acceleration	4 Byte	Read/Write
186	GEAR.DECMAX	BA	Acceleration	8 Byte	Read/Write
187	GEAR.DECMAX (32 bit version)	BB	Acceleration	4 Byte	Read/Write
188	GEAR.IN	BC	Integer	2 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
189	GEAR.MODE	BD	Integer	2 Byte	Read/Write
191	GEAR.OUT	BF	Integer	2 Byte Signed	Read/Write
192	GEAR.VMAX	C0	Velocity	8 Byte	Read/Write
193	HOME.ACC	C1	Acceleration	8 Byte	Read/Write
194	HOME.ACC (32 bit version)	C2	Acceleration	4 Byte	Read/Write
195	HOME.AUTOMOVE	C3	Integer	1 Byte	Read/Write
196	HOME.DEC	C4	Acceleration	8 Byte	Read/Write
197	HOME.DEC (32 bit version)	C5	Acceleration	4 Byte	Read/Write
198	HOME.DIR	C6	Integer	2 Byte	Read/Write
199	HOME.DIST	C7	Position	8 Byte Signed	Read/Write
200	HOME.DIST (32 bit version)	C8	Position	4 Byte Signed	Read/Write
201	HOME.FEEDRATE	C9	Integer	2 Byte	Read/Write
202	HOME.IPEAK	CA	Float	4 Byte Signed	Read/Write
204	HOME.MODE	CC	Integer	2 Byte	Read/Write
206	HOME.P	CE	Position	8 Byte Signed	Read/Write
207	HOME.P (32 bit version)	CF	Position	4 Byte Signed	Read/Write
208	HOME.PERRTHRESH	D0	Position	8 Byte Signed	Read/Write
209	HOME.PERRTHRESH (32 bit version)	D1	Position	4 Byte Signed	Read/Write
211	HOME.V	D3	Velocity	8 Byte	Read/Write
212	HWLS.NEGSTATE	D4	Integer	1 Byte	Read Only
213	HWLS.POSSTATE	D5	Integer	1 Byte	Read Only
214	IL.BUSFF	D6	Float	4 Byte Signed	Read Only
215	IL.CMD	D7	Float	4 Byte Signed	Read Only
216	IL.CMDU	D8	Float	4 Byte Signed	Read/Write
217	IL.FB	D9	Float	4 Byte Signed	Read Only
218	IL.FF	DA	Float	4 Byte	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
219	IL.FOLDFTHRESH	DB	Float	4 Byte	Read Only
220	IL.FOLDFTHRESHU	DC	Float	4 Byte Signed	Read/Write
221	IL.FOLDWTHRESH	DD	Float	4 Byte Signed	Read/Write
222	IL.FRICTION	DE	Float	4 Byte	Read/Write
223	IL.IFOLD	DF	Float	4 Byte	Read Only
224	IL.IUFB	E0	Float	4 Byte Signed	Read Only
225	IL.IVFB	E1	Float	4 Byte Signed	Read Only
226	IL.KACCFF	E2	Float	4 Byte Signed	Read/Write
227	IL.KBUSFF	E3	Float	4 Byte	Read/Write
228	IL.KP	E4	Float	2 Byte	Read/Write
229	IL.KPDRATIO	E5	Float	4 Byte	Read Only
230	IL.KVFF	E6	Float	4 Byte Signed	Read/Write
231	IL.LIMITN	E7	Float	4 Byte Signed	Read/Write
232	IL.LIMITP	E8	Float	4 Byte Signed	Read/Write
233	IL.MFOLDD	E9	Float	4 Byte	Read Only
234	IL.MFOLDR	EA	Float	4 Byte	Read Only
235	IL.MFOLDT	EB	Float	4 Byte	Read Only
236	IL.MIFOLD	EC	Float	4 Byte	Read Only
237	IL.OFFSET	ED	Float	4 Byte Signed	Read/Write
238	IL.VCMD	EE	Float	2 Byte Signed	Read Only
239	IL.VUFB	EF	Integer	2 Byte Signed	Read Only
240	IL.VVFB	F0	Integer	2 Byte Signed	Read Only
241	MOTOR.AUTOSET	F1	Integer	1 Byte	Read/Write
242	MOTOR.BRAKE	F2	Integer	1 Byte	Read/Write
243	MOTOR.BRAKERLS	F3	Integer	1 Byte	Read/Write
244	MOTOR.CTF0	F4	Float	4 Byte	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
245	MOTOR.ICONT	F5	Float	4 Byte	Read Only
246	MOTOR.IDDATAVALID	F6	Integer	1 Byte	Read Only
247	MOTOR.INERTIA	F7	Float	4 Byte	Read Only
248	MOTOR.IPEAK	F8	Float	4 Byte	Read Only
249	MOTOR.KT	F9	Float	4 Byte	Read Only
250	MOTOR.LQLL	FA	Float	4 Byte	Read Only
251	MOTOR.PHASE	FB	Integer	2 Byte	Read/Write
252	MOTOR.PITCH	FC	Float	4 Byte	Read/Write
253	MOTOR.POLES	FD	Integer	2 Byte	Read Only
254	MOTOR.R	FE	Float	4 Byte	Read Only
255	MOTOR.RTYPE	FF	Integer	1 Byte	Read Only
256	MOTOR.TBRAKEAPP	100	Integer	2 Byte	Read/Write
257	MOTOR.TBRAKERLS	101	Integer	2 Byte	Read/Write
258	MOTOR.TEMP	102	Integer	4 Byte	Read Only
259	MOTOR.TEMPFAULT	103	Integer	4 Byte	Read Only
260	MOTOR.TEMPWARN	104	Integer	4 Byte	Read/Write
261	MOTOR.TYPE	105	Integer	1 Byte	Read/Write
262	MOTOR.VMAX	106	Integer	2 Byte	Read/Write
263	MOTOR.VOLTMAX	107	Integer	2 Byte	Read/Write
264	MT.ACC	108	Acceleration	8 Byte	Read/Write
265	MT.ACC (32 bit version)	109	Acceleration	4 Byte	Read/Write
266	MT.CLEAR	10A	Integer	2 Byte Signed	Write Only
267	MT.CNTL	10B	Integer	4 Byte	Read/Write
269	MT.DEC	10D	Acceleration	8 Byte	Read/Write
270	MT.DEC (32 bit version)	10E	Acceleration	4 Byte	Read/Write
271	MT.EMERGMT	10F	Integer	2 Byte Signed	Read/Write
273	MT.MOVE	111	None	2 Byte	Write Only
274	MT.MTNEXT	112	Integer	1 Byte	Read/Write
275	MT.NUM	113	Integer	1 Byte	Read/Write
276	MT.P	114	Position	8 Byte Signed	Read/Write
277	MT.P (32 bit version)	115	Position	4 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
279	MT.TNEXT	117	Integer	2 Byte	Read/Write
280	MT.TNUM	118	Integer	1 Byte	Read/Write
281	MT.TPOSWND	119	Position	8 Byte Signed	Read/Write
282	MT.TPOSWND (32 bit version)	11A	Position	4 Byte Signed	Read/Write
283	MT.TVELWND	11B	Velocity	8 Byte	Read/Write
284	MT.V	11C	Velocity	8 Byte	Read/Write
285	MT.VCMD	11D	Velocity	8 Byte Signed	Read Only
286	PL.CMD	11E	Position	8 Byte	Read Only
287	PL.CMD (32 bit version)	11F	Position	4 Byte	Read Only
288	PL.ERR	120	Position	8 Byte	Read Only
289	PL.ERR (32 bit version)	121	Position	4 Byte	Read Only
290	PL.ERRMODE	122	Integer	1 Byte	Read/Write
291	PL.ERRFTHRESH	123	Position	8 Byte	Read/Write
292	PL.ERRFTHRESH (32 bit version)	124	Position	4 Byte	Read/Write
293	PL.ERRWTHRESH	125	Position	8 Byte	Read/Write
294	PL.ERRWTHRESH (32 bit version)	126	Position	4 Byte	Read/Write
295	PL.FB	127	Position	8 Byte Signed	Read Only
296	PL.FB (32 bit version)	128	Position	4 Byte Signed	Read Only
297	PL.FBSOURCE	129	Integer	1 Byte	Read/Write
298	PL.INTINMAX	12A	Position	8 Byte	Read/Write
299	PL.INTINMAX (32 bit version)	12B	Position	4 Byte	Read/Write
300	PL.INTOUTMAX	12C	Position	8 Byte	Read/Write
301	PL.INTOUTMAX (32 bit version)	12D	Position	4 Byte	Read/Write
302	PL.KI	12E	Float	4 Byte	Read/Write
303	PL.KP	12F	Float	4 Byte	Read/Write
304	PL.MODP1	130	Position	8 Byte Signed	Read/Write
305	PL.MODP1 (32 bit version)	131	Position	4 Byte Signed	Read/Write
306	PL.MODP2	132	Position	8 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
307	PL.MODP2 (32 bit version)	133	Position	4 Byte Signed	Read/Write
308	PL.MODPDIR	134	Integer	1 Byte	Read/Write
309	PL.MODPEN	135	Integer	1 Byte	Read/Write
310	PLS.EN	136	Integer	2 Byte	Read/Write
311	PLS.MODE	137	Integer	2 Byte	Read/Write
312	PLS.P1	138	Position	8 Byte Signed	Read/Write
313	PLS.P1 (32 bit version)	139	Position	4 Byte Signed	Read/Write
314	PLS.P2	13A	Position	8 Byte Signed	Read/Write
315	PLS.P2 (32 bit version)	13B	Position	4 Byte Signed	Read/Write
316	PLS.P3	13C	Position	8 Byte Signed	Read/Write
317	PLS.P3 (32 bit version)	13D	Position	4 Byte Signed	Read/Write
318	PLS.P4	13E	Position	8 Byte Signed	Read/Write
319	PLS.P4 (32 bit version)	13F	Position	4 Byte Signed	Read/Write
320	PLS.P5	140	Position	8 Byte Signed	Read/Write
321	PLS.P5 (32 bit version)	141	Position	4 Byte Signed	Read/Write
322	PLS.P6	142	Position	8 Byte Signed	Read/Write
323	PLS.P6 (32 bit version)	143	Position	4 Byte Signed	Read/Write
324	PLS.P7	144	Position	8 Byte Signed	Read/Write
325	PLS.P7 (32 bit version)	145	Position	4 Byte Signed	Read/Write
326	PLS.P8	146	Position	8 Byte Signed	Read/Write
327	PLS.P8 (32 bit version)	147	Position	4 Byte Signed	Read/Write
328	PLS.RESET	148	Integer	2 Byte	Write Only
329	PLS.STATE	149	Integer	2 Byte	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
330	PLS.T1	14A	Integer	2 Byte	Read/Write
331	PLS.T2	14B	Integer	2 Byte	Read/Write
332	PLS.T3	14C	Integer	2 Byte	Read/Write
333	PLS.T4	14D	Integer	2 Byte	Read/Write
334	PLS.T5	14E	Integer	2 Byte	Read/Write
335	PLS.T6	14F	Integer	2 Byte	Read/Write
336	PLS.T7	150	Integer	2 Byte	Read/Write
337	PLS.T8	151	Integer	2 Byte	Read/Write
338	PLS.UNITS	152	Integer	1 Byte	Read/Write
339	PLS.WIDTH1	153	Position	8 Byte Signed	Read/Write
340	PLS.WIDTH1 (32 bit version)	154	Position	4 Byte Signed	Read/Write
341	PLS.WIDTH2	155	Position	8 Byte Signed	Read/Write
342	PLS.WIDTH2 (32 bit version)	156	Position	4 Byte Signed	Read/Write
343	PLS.WIDTH3	157	Position	8 Byte Signed	Read/Write
344	PLS.WIDTH3 (32 bit version)	158	Position	4 Byte Signed	Read/Write
345	PLS.WIDTH4	159	Position	8 Byte Signed	Read/Write
346	PLS.WIDTH4 (32 bit version)	15A	Position	4 Byte Signed	Read/Write
347	PLS.WIDTH5	15B	Position	8 Byte Signed	Read/Write
348	PLS.WIDTH5 (32 bit version)	15C	Position	4 Byte Signed	Read/Write
349	PLS.WIDTH6	15D	Position	8 Byte Signed	Read/Write
350	PLS.WIDTH6 (32 bit version)	15E	Position	4 Byte Signed	Read/Write
351	PLS.WIDTH7	15F	Position	8 Byte Signed	Read/Write
352	PLS.WIDTH7 (32 bit version)	160	Position	4 Byte Signed	Read/Write
353	PLS.WIDTH8	161	Position	8 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
354	PLS.WIDTH8 (32 bit version)	162	Position	4 Byte Signed	Read/Write
355	REC.ACTIVE	163	Integer	1 Byte	Read Only
356	REC.DONE	164	Integer	1 Byte	Read Only
357	REC.GAP	165	Integer	2 Byte	Read/Write
358	REC.NUMPOINTS	166	Integer	2 Byte	Read/Write
360	REC.STOPTYPE	168	Integer	1 Byte	Read/Write
362	REC.TRIGPOS	16A	Integer	1 Byte	Read/Write
364	REC.TRIGSLOPE	16C	Integer	1 Byte	Read/Write
365	REC.TRIGTYPE	16D	Integer	1 Byte	Read/Write
366	REC.TRIGVAL	16E	Varies	8 Byte Signed	Read/Write
367	REC.TRIGVAL (32 bit version)	16F	Varies	4 Byte Signed	Read/Write
368	REGEN.POWER	170	Integer	8 Byte	Read Only
369	REGEN.POWER (32 bit version)	171	Integer	4 Byte	Read Only
370	REGEN.REXT	172	Integer	2 Byte	Read/Write
371	REGEN.TEXT	173	Float	4 Byte	Read/Write
372	REGEN.TYPE	174	Integer	1 Byte Signed	Read/Write
373	REGEN.WATTEXT	175	Integer	2 Byte	Read/Write
374	SM.I1	176	Float	4 Byte Signed	Read/Write
375	SM.I2	177	Float	4 Byte Signed	Read/Write
376	SM.MODE	178	Integer	2 Byte	Read/Write
378	SM.T1	17A	Integer	2 Byte	Read/Write
379	SM.T2	17B	Integer	2 Byte	Read/Write
380	SM.V1	17C	Velocity	8 Byte Signed	Read/Write
381	SM.V2	17D	Velocity	8 Byte Signed	Read/Write
382	STO.STATE	17E	Integer	1 Byte	Read Only
383	SWLS.EN	17F	Integer	2 Byte	Read/Write
384	SWLS.LIMIT0	180	Position	8 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
385	SWLS.LIMIT0 (32 bit version)	181	Position	4 Byte Signed	Read/Write
386	SWLS.LIMIT1	182	Position	8 Byte Signed	Read/Write
387	SWLS.LIMIT1 (32 bit version)	183	Position	4 Byte Signed	Read/Write
388	SWLS.STATE	184	Integer	2 Byte	Read Only
389	UNIT.ACCLINEAR	185	Integer	1 Byte	Read/Write
390	UNIT.ACCROTARY	186	Integer	1 Byte	Read/Write
391	UNIT.PIN	187	Integer	4 Byte	Read/Write
392	UNIT.PLINEAR	188	Integer	1 Byte	Read/Write
393	UNIT.POUT	189	Integer	4 Byte	Read/Write
394	UNIT.PROTARY	18A	Integer	1 Byte	Read/Write
395	UNIT.VLINEAR	18B	Integer	1 Byte	Read/Write
396	UNIT.VROTARY	18C	Integer	1 Byte	Read/Write
398	VBUS.OVFTHRESH	18E	Integer	2 Byte	Read Only
399	VBUS.OVWTHRESH	18F	Integer	2 Byte	Read/Write
400	VBUS.RMSLIMIT	190	Integer	1 Byte	Read Only
401	VBUS.UVFTHRESH	191	Integer	2 Byte	Read/Write
402	VBUS.UVMODE	192	Integer	1 Byte	Read/Write
403	VBUS.UVWTHRESH	193	Integer	2 Byte	Read/Write
404	VBUS.VALUE	194	Float	4 Byte Signed	Read Only
405	VL.ARPF1	195	Float	4 Byte	Read/Write
406	VL.ARPF2	196	Float	4 Byte	Read/Write
407	VL.ARPF3	197	Float	4 Byte	Read/Write
408	VL.ARPF4	198	Float	4 Byte	Read/Write
409	VL.ARPQ1	199	Float	4 Byte	Read/Write
410	VL.ARPQ2	19A	Float	4 Byte	Read/Write
411	VL.ARPQ3	19B	Float	4 Byte	Read/Write
412	VL.ARPQ4	19C	Float	4 Byte	Read/Write
413	VL.ARTYPE1	19D	Integer	1 Byte	Read/Write
414	VL.ARTYPE2	19E	Integer	1 Byte	Read/Write
415	VL.ARTYPE3	19F	Integer	1 Byte	Read/Write
416	VL.ARTYPE4	1A0	Integer	1 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
417	VL.ARZF1	1A1	Float	4 Byte	Read/Write
418	VL.ARZF2	1A2	Float	4 Byte	Read/Write
419	VL.ARZF3	1A3	Float	4 Byte	Read/Write
420	VL.ARZF4	1A4	Float	4 Byte	Read/Write
421	VL.ARZQ1	1A5	Float	4 Byte	Read/Write
422	VL.ARZQ2	1A6	Float	4 Byte	Read/Write
423	VL.ARZQ3	1A7	Float	4 Byte	Read/Write
424	VL.ARZQ4	1A8	Float	4 Byte	Read/Write
425	VL.BUSFF	1A9	Velocity	8 Byte Signed	Read Only
426	VL.CMD	1AA	Velocity	8 Byte Signed	Read Only
427	VL.CMDU	1AB	Velocity	8 Byte Signed	Read/Write
428	VL.ERR	1AC	Velocity	8 Byte Signed	Read Only
429	VL.FB	1AD	Velocity	8 Byte Signed	Read Only
430	VL.FBFILTER	1AE	Velocity	8 Byte Signed	Read Only
431	VL.FBSOURCE	1AF	Integer	1 Byte	Read/Write
432	VL.FF	1B0	Velocity	8 Byte Signed	Read Only
433	VL.GENMODE	1B1	Integer	2 Byte	Read/Write
434	VL.KBUSFF	1B2	Float	4 Byte	Read/Write
435	VL.KI	1B3	Float	4 Byte	Read/Write
436	VL.KO	1B4	Float	4 Byte	Read/Write
437	VL.KP	1B5	Float	4 Byte	Read/Write
438	VL.KVFF	1B6	Float	4 Byte	Read/Write
439	VL.LIMITN	1B7	Velocity	8 Byte Signed	Read/Write
440	VL.LIMITP	1B8	Velocity	8 Byte	Read/Write
441	VL.LMJR	1B9	Float	4 Byte	Read/Write
442	VL.MODEL	1BA	Velocity	8 Byte Signed	Read Only
443	VL.OBSBW	1BB	Float	4 Byte	Read/Write
444	VL.OBSMODE	1BC	Integer	4 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
445	VL.THRESH	1BD	Velocity	8 Byte Signed	Read/Write
447	WS.DISTMAX	1BF	Position	8 Byte Signed	Read/Write
448	WS.DISTMAX (32 bit version)	1C0	Position	4 Byte Signed	Read/Write
449	WS.DISTMIN	1C1	Position	8 Byte Signed	Read/Write
450	WS.DISTMIN (32 bit version)	1C2	Position	4 Byte Signed	Read/Write
451	WS.IMAX	1C3	Float	4 Byte Signed	Read/Write
452	WS.MODE	1C4	Integer	1 Byte	Read/Write
453	WS.NUMLOOPS	1C5	Integer	1 Byte	Read/Write
454	WS.STATE	1C6	Integer	1 Byte	Read Only
455	WS.T	1C7	Integer	2 Byte	Read/Write
456	WS.TDELAY1	1C8	Integer	2 Byte	Read/Write
457	WS.TDELAY2	1C9	Integer	2 Byte	Read/Write
458	WS.TDELAY3	1CA	Integer	2 Byte	Read/Write
459	WS.VTHRESH	1CB	Velocity	8 Byte Signed	Read/Write
460	DIN1.FILTER	1CC	Integer	2 Byte	Read/Write
461	DIN2.FILTER	1CD	Integer	2 Byte	Read/Write
462	DIN3.FILTER	1CE	Integer	2 Byte	Read/Write
463	DIN4.FILTER	1CF	Integer	2 Byte	Read/Write
464	DIN5.FILTER	1D0	Integer	2 Byte	Read/Write
465	DIN6.FILTER	1D1	Integer	2 Byte	Read/Write
466	DIN7.FILTER	1D2	Integer	2 Byte	Read/Write
467	FB1.HALLSTATEU	1D3	Integer	1 Byte	Read Only
468	FB1.HALLSTATEV	1D4	Integer	1 Byte	Read Only
469	FB1.HALLSTATEW	1D5	Integer	1 Byte	Read Only
471	MODBUS.DIO	1D7	Integer	4 Byte	Read Only
472	MODBUS.DRV	1D8	Integer	4 Byte	Read/Write
473	MODBUS.DRVSTAT	1D9	Integer	4 Byte	Read Only
474	MODBUS.HOME	1DA	Integer	4 Byte	Read/Write
475	MODBUS.MOTOR	1DB	Integer	4 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
476	MODBUS.MT	1DC	Integer	2 Byte	Read/Write
477	MODBUS.SM	1DD	Integer	4 Byte	Read/Write
478	DRV.FAULT1	1DE	Integer	2 Byte	Read Only
479	DRV.FAULT2	1DF	Integer	2 Byte	Read Only
480	DRV.FAULT3	1E0	Integer	2 Byte	Read Only
481	DRV.FAULT4	1E1	Integer	2 Byte	Read Only
482	DRV.FAULT5	1E2	Integer	2 Byte	Read Only
483	DRV.FAULT6	1E3	Integer	2 Byte	Read Only
484	DRV.FAULT7	1E4	Integer	2 Byte	Read Only
485	DRV.FAULT8	1E5	Integer	2 Byte	Read Only
486	DRV.FAULT9	1E6	Integer	2 Byte	Read Only
487	DRV.FAULT10	1E7	Integer	2 Byte	Read Only
488	MODBUS.PIN	1E8	Integer	4 Byte	Read/Write
489	MODBUS.POUT	1E9	Integer	4 Byte	Read/Write
490	MODBUS.PSCALE	1EA	Integer	2 Byte	Read/Write
493	FB2.ENCRES	1ED	Integer	4 Byte	Read/Write
494	FB2.MODE	1EE	Integer	2 Byte	Read/Write
495	FB2.SOURCE	1EF	Integer	2 Byte	Read/Write
496	MOTOR.TBRAKETO	1F0	Integer	4 Byte Signed	Read/Write
497	MODBUS.MSGLOG	1F1	Integer	1 Byte	Read/Write
498	USER.INT	1F2	Integer	4 Byte Signed	Read/Write
499	USER.INT	1F3	Integer	4 Byte Signed	Read/Write
500	USER.INT	1F4	Integer	4 Byte Signed	Read/Write
501	USER.INT	1F5	Integer	4 Byte Signed	Read/Write
502	USER.INT	1F6	Integer	4 Byte Signed	Read/Write
503	USER.INT	1F7	Integer	4 Byte Signed	Read/Write
504	USER.INT	1F8	Integer	4 Byte Signed	Read/Write
505	USER.INT	1F9	Integer	4 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
506	USER.INT	1FA	Integer	4 Byte Signed	Read/Write
507	USER.INT	1FB	Integer	4 Byte Signed	Read/Write
508	USER.INT	1FC	Integer	4 Byte Signed	Read/Write
509	USER.INT	1FD	Integer	4 Byte Signed	Read/Write
510	USER.INT	1FE	Integer	4 Byte Signed	Read/Write
511	USER.INT	1FF	Integer	4 Byte Signed	Read/Write
512	USER.INT	200	Integer	4 Byte Signed	Read/Write
513	USER.INT	201	Integer	4 Byte Signed	Read/Write
514	USER.INT	202	Integer	4 Byte Signed	Read/Write
515	USER.INT	203	Integer	4 Byte Signed	Read/Write
516	USER.INT	204	Integer	4 Byte Signed	Read/Write
517	USER.INT	205	Integer	4 Byte Signed	Read/Write
518	USER.INT	206	Integer	4 Byte Signed	Read/Write
519	USER.INT	207	Integer	4 Byte Signed	Read/Write
520	USER.INT	208	Integer	4 Byte Signed	Read/Write
521	USER.INT	209	Integer	4 Byte Signed	Read/Write
522	DRV.NVCHECK	20A	Integer	8 Byte	Read Only
523	FB3.MODE	20B	Integer	2 Byte	Read/Write
524	FB3.P	20C	Integer	8 Byte Signed	Read Only
525	MODBUS.SCALING	20D	Integer	1 Byte	Read/Write
526	DRV.EMUEPULSEWIDTH	20E	Float	4 Byte	Read/Write
527	DRV.EMUECHECKSPEED	20F	Integer	1 Byte	Read/Write
593	IL.MI2T	251	Float	4 Byte	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
594	AIN.DEADBANDMODE	252	Integer	2 Byte	Read/Write
595	AIN.MODE	253	Integer	1 Byte	Read/Write
596	DIO10.DIR	254	Integer	1 Byte	Read/Write
597	DIO10.INV	255	Integer	1 Byte	Read/Write
598	DIO11.DIR	256	Integer	1 Byte	Read/Write
599	DIO11.INV	257	Integer	1 Byte	Read/Write
600	DIO9.DIR	258	Integer	1 Byte	Read/Write
601	DIO9.INV	259	Integer	1 Byte	Read/Write
602	FAULT130.ACTION	25A	Integer	1 Byte	Read/Write
603	FAULT131.ACTION	25B	Integer	1 Byte	Read/Write
604	FAULT132.ACTION	25C	Integer	1 Byte	Read/Write
605	FAULT134.ACTION	25D	Integer	1 Byte	Read/Write
606	FAULT702.ACTION	25E	Integer	1 Byte	Read/Write
607	IP.MODE	25F	Integer	2 Byte	Read/Write
608	LOAD.INERTIA	260	Float	4 Byte	Read/Write
609	MOTOR.KE	261	Float	4 Byte	Read Only
610	VBUS.HALFVOLT	262	Integer	1 Byte	Read/Write
611	FB2.DIR	263	Integer	1 Byte	Read/Write
612	DRV.HANDWHEELSRC	264	Integer	1 Byte	Read/Write
613	DRV.HWENDELAY	265	Integer	1 Byte	Read/Write
614	IL.KPLOOKUPINDEX	266	Integer	2 Byte	Read/Write
615	IL.KPLOOKUPVALUE	267	Float	4 Byte	Read/Write
616	FAULT451.ACTION	268	Integer	1 Byte	Read/Write
617	MOTOR.BRAKEIMM	269	Integer	1 Byte	Read/Write
631	AIO.ISCALE	277	Float	4 Byte	Read/Write
632	AIO.PSCALE	278	Position	8 Byte	Read/Write
633	AIO.PSCALE (32 bit version)	279	Position	4 Byte	Read/Write
634	AIO.VSCALE	27A	Velocity	8 Byte	Read/Write
635	AIO.VSCALE (32 bit version)	27B	Velocity	4 Byte	Read/Write
636	AOUT.CUTOFF	27C	Float	4 Byte	Read/Write
649	BODE.IFLIMIT	289	Float	4 Byte Signed	Read/Write
650	BODE.IFTHRESH	28A	Float	4 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
651	BODE.VFLIMIT	28B	Float	4 Byte Signed	Read/Write
652	BODE.VFTHRESH	28C	Velocity	8 Byte Signed	Read/Write
653	BODE.VFTHRESH (32 bit version)	28D	Velocity	4 Byte Signed	Read/Write
654	DIN10.STATE	28E	Integer	1 Byte	Read Only
655	DIN11.STATE	28F	Integer	1 Byte	Read Only
728	DIN9.STATE	2D8	Integer	1 Byte	Read Only
729	DOUT10.STATE	2D9	Integer	1 Byte	Read Only
730	DOUT10.STATEU	2DA	Integer	1 Byte	Read/Write
731	DOUT11.STATE	2DB	Integer	1 Byte	Read Only
732	DOUT11.STATEU	2DC	Integer	1 Byte	Read/Write
783	DOUT9.STATE	30F	Integer	1 Byte	Read Only
784	DOUT9.STATEU	310	Integer	1 Byte	Read/Write
791	DRV.SETUPREQBITS	317	Integer	4 Byte	Read Only
792	DRV.WARNING1	318	Integer	4 Byte	Read Only
793	DRV.WARNING2	319	Integer	4 Byte	Read Only
794	DRV.WARNING3	31A	Integer	4 Byte	Read Only
795	EIP.CONNECTED	31B	Integer	1 Byte	Read Only
796	EIP.POSUNIT	31C	Integer	4 Byte	Read/Write
797	EIP.PROFUNIT	31D	Integer	4 Byte	Read/Write
798	FAULT139.ACTION	31E	Integer	1 Byte	Read/Write
806	FB1.P	326	Position	8 Byte Signed	Read Only
807	FB1.P (32 bit version)	327	Position	4 Byte Signed	Read Only
808	FB1.PDIR	328	Integer	1 Byte	Read/Write
809	FB1.PIN	329	Integer	4 Byte	Read/Write
810	FB1.POFFSET	32A	Position	8 Byte Signed	Read/Write
811	FB1.POFFSET (32 bit version)	32B	Position	4 Byte Signed	Read/Write
812	FB1.POUT	32C	Integer	4 Byte	Read/Write
813	FB1.PUNIT	32D	Integer	4 Byte	Read/Write
814	FB1.USERBYTE	32E	Integer	1 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
815	FB1.USERDWORD	32F	Integer	4 Byte	Read/Write
816	FB1.USERWORD	330	Integer	2 Byte	Read/Write
817	FB2.P	331	Integer	8 Byte Signed	Read Only
818	FB2.P (32 bit version)	332	Integer	4 Byte Signed	Read Only
819	FB2.PIN	333	Integer	4 Byte	Read/Write
820	FB2.POFFSET	334	Position	8 Byte Signed	Read/Write
821	FB2.POFFSET (32 bit version)	335	Position	4 Byte Signed	Read/Write
822	FB2.POUT	336	Integer	4 Byte	Read/Write
823	FB2.PUNIT	337	Integer	4 Byte	Read/Write
824	FB3.P	338	Integer	8 Byte Signed	Read Only
825	FB3.P (32 bit version)	339	Integer	4 Byte Signed	Read Only
826	FB3.PDIR	33A	Integer	1 Byte	Read/Write
827	FB3.PIN	33B	Integer	4 Byte	Read/Write
828	FB3.POFFSET	33C	Position	8 Byte Signed	Read/Write
829	FB3.POFFSET (32 bit version)	33D	Position	4 Byte Signed	Read/Write
830	FB3.POUT	33E	Integer	4 Byte	Read/Write
831	FB3.PUNIT	33F	Integer	4 Byte	Read/Write
832	HOME.MAXDIST	340	Position	8 Byte Signed	Read/Write
833	HOME.MAXDIST (32 bit version)	341	Position	4 Byte Signed	Read/Write
834	IL.DIFOLD	342	Float	4 Byte	Read Only
835	IL.MI2TWTHRESH	343	Integer	1 Byte	Read/Write
836	IL.MIMODE	344	Integer	1 Byte	Read/Write
838	MOTOR.VOLTMIN	346	Integer	2 Byte	Read/Write
839	MOTOR.VOLTRATED	347	Integer	2 Byte	Read/Write
840	MOTOR.VRATED	348	Float	8 Byte Signed	Read/Write
841	MOTOR.VRATED (32 bit version)	349	Float	4 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
846	VL.FBUNFILTERED	34E	Velocity	8 Byte Signed	Read Only
847	VL.FBUNFILTERED (32 bit version)	34F	Velocity	4 Byte Signed	Read Only
849	WS.FREQ	351	Float	4 Byte	Read/Write
850	WS.TDELAY4	352	Integer	2 Byte	Read/Write
851	WS.CHECKT	353	Integer	2 Byte	Read/Write
852	WS.CHECKV	354	Velocity	8 Byte Signed	Read/Write
853	WS.CHECKV (32 bit version)	355	Velocity	4 Byte Signed	Read/Write
859	AOUT.VSCALE	35B	Velocity	8 Byte	Read/Write
860	AOUT.VSCALE (32 bit version)	35C	Velocity	4 Byte	Read/Write
861	WS.TSTANDSTILL	35D	Integer	2 Byte	Read/Write
862	WS.TIRAMP	35E	Integer	2 Byte	Read/Write
863	FB1.EXTENDEDMULTITURN	35F	Integer	1 Byte	Read/Write
865	MOTOR.IMTR	361	Integer	2 Byte	Read/Write
866	IL.FBSOURCE	362	Integer	1 Byte	Read/Write
867	MOTOR.IMID	363	Float	4 Byte	Read/Write
868	WS.CHECKMODE	364	Integer	1 Byte	Read/Write
869	REGEN.POWERFILTERED	365	Integer	8 Byte	Read Only
870	REGEN.POWERFILTERED (32 bit version)	366	Integer	4 Byte	Read Only
872	FBUS.PROTECTION	368	Integer	1 Byte	Read/Write
873	FBUS.BLOCKING	369	Integer	1 Byte	Read Only
874	FBUS.STATE	36A	Integer	1 Byte Signed	Read Only
875	TEMP.CONTROL	36B	Integer	2 Byte Signed	Read Only
876	TEMP.POWER	36C	Integer	2 Byte Signed	Read Only
877	TEMP.POWER	36D	Integer	2 Byte Signed	Read Only
878	TEMP.POWER	36E	Integer	2 Byte Signed	Read Only
879	MODBUS.ERRORMODE	36F	Integer	1 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
881	IL.CMDACC	371	Float	8 Byte Signed	Read Only
882	IL.CMDACC (32 bit version)	372	Float	4 Byte Signed	Read Only
883	DRV.DOWNLOADALLOWED	373	Integer	4 Byte	Read Only
884	CAP0.FBSOURCE	374	Integer	1 Byte	Read/Write
885	CAP1.FBSOURCE	375	Integer	1 Byte	Read/Write
886	FB1.INITPSAVED	376	Position	8 Byte Signed	Read Only
887	FB1.INITPSAVED (32 bit version)	377	Position	4 Byte Signed	Read Only
888	FB1.INITPWINDOW	378	Position	8 Byte	Read/Write
889	FB1.INITPWINDOW (32 bit version)	379	Position	4 Byte	Read/Write
890	FB1.INITPSTATUS	37A	Integer	1 Byte	Read Only
891	FB1.LASTIDENTIFIED	37B	Integer	1 Byte	Read/Write
892	DRV.MOTIONDISSOURCES	37C	Integer	2 Byte	Read Only
893	MOTOR.LDLL	37D	Float	4 Byte	Read Only
894	MOTOR.LISAT	37E	Float	4 Byte	Read/Write
895	MOTOR.IDMAX	37F	Float	4 Byte	Read/Write
896	MOTOR.PHSADVK1	380	Float	4 Byte Signed	Read/Write
897	MOTOR.PHSADVK2	381	Float	4 Byte Signed	Read/Write
901	MOTOR.TEMPC	385	Integer	2 Byte Signed	Read Only
902	VL.VFTHRESH	386	Velocity	8 Byte Signed	Read Only
903	VL.VFTHRESH (32 bit version)	387	Velocity	4 Byte Signed	Read Only
904	IL.PWMFREQ	388	Float	2 Byte	Read Only
905	IL.DEADBAND	389	Integer	2 Byte	Read Only
906	DRV.POWERBOARDID	38A	Integer	1 Byte	Read Only
907	DRV.EMUESTEPCMD	38B	Integer	4 Byte Signed	Read/Write
908	DRV.EMUESTEPMODE	38C	Integer	2 Byte	Read/Write
911	CMP0.MODE	38F	Integer	1 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
912	CMP0.SOURCE	390	Integer	2 Byte Signed	Read/Write
913	CMP1.SOURCE	391	Integer	2 Byte Signed	Read/Write
914	CMP1.MODE	392	Integer	1 Byte	Read/Write
915	CMP0.ARM	393	Integer	1 Byte	Read/Write
916	CMP1.ARM	394	Integer	1 Byte	Read/Write
917	CMP0.OUTMASK	395	Integer	4 Byte	Read/Write
918	CMP0.SETPOINT	396	Position	8 Byte Signed	Read/Write
919	CMP0.SETPOINT (32 bit version)	397	Position	4 Byte Signed	Read/Write
920	CMP0.STATE	398	Integer	1 Byte	Read Only
921	CMP0.WIDTH	399	Float	8 Byte	Read/Write
922	CMP0.WIDTH (32 bit version)	39A	Float	4 Byte	Read/Write
923	CMP0.WIDTHTYPE	39B	Integer	1 Byte	Read/Write
924	CMP1.OUTMASK	39C	Integer	4 Byte	Read/Write
925	CMP1.SETPOINT	39D	Position	8 Byte Signed	Read/Write
926	CMP1.SETPOINT (32 bit version)	39E	Position	4 Byte Signed	Read/Write
927	CMP1.STATE	39F	Integer	1 Byte	Read Only
928	CMP1.WIDTH	3A0	Float	8 Byte	Read/Write
929	CMP1.WIDTH (32 bit version)	3A1	Float	4 Byte	Read/Write
930	CMP1.WIDTHTYPE	3A2	Integer	1 Byte	Read/Write
931	CMP0.MODBOUND1	3A3	Position	8 Byte Signed	Read/Write
932	CMP0.MODBOUND1 (32 bit version)	3A4	Position	4 Byte Signed	Read/Write
933	CMP0.MODBOUND2	3A5	Position	8 Byte Signed	Read/Write
934	CMP0.MODBOUND2 (32 bit version)	3A6	Position	4 Byte Signed	Read/Write
935	CMP0.MODEN	3A7	Integer	1 Byte	Read/Write
936	CMP0.MODVALUE	3A8	Position	8 Byte Signed	Read Only
937	CMP0.MODVALUE (32 bit version)	3A9	Position	4 Byte Signed	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
938	CMP1.MODBOUND1	3AA	Position	8 Byte Signed	Read/Write
939	CMP1.MODBOUND1 (32 bit version)	3AB	Position	4 Byte Signed	Read/Write
940	CMP1.MODBOUND2	3AC	Position	8 Byte Signed	Read/Write
941	CMP1.MODBOUND2 (32 bit version)	3AD	Position	4 Byte Signed	Read/Write
942	CMP1.MODEN	3AE	Integer	1 Byte	Read/Write
943	CMP1.MODVALUE	3AF	Position	8 Byte Signed	Read Only
944	CMP1.MODVALUE (32 bit version)	3B0	Position	4 Byte Signed	Read Only
945	FB3.DIR	3B1	Integer	1 Byte	Read/Write
946	CMP0.ADVANCE	3B2	Float	4 Byte Signed	Read/Write
947	CMP1.ADVANCE	3B3	Float	4 Byte Signed	Read/Write
948	SFD.DIAGMODE	3B4	Integer	1 Byte	Read/Write
949	SFD.ADDR	3B5	Integer	4 Byte	Read/Write
951	SFD.WRITEENABLE	3B7	Integer	1 Byte	Read/Write
953	DOUT9.MODE	3B9	Integer	1 Byte	Read/Write
954	DOUT9.PARAM	3BA	Float	8 Byte Signed	Read/Write
955	DOUT9.PARAM (32 bit version)	3BB	Float	4 Byte Signed	Read/Write
956	DOUT10.MODE	3BC	Integer	1 Byte	Read/Write
957	DOUT10.PARAM	3BD	Float	8 Byte Signed	Read/Write
958	DOUT10.PARAM (32 bit version)	3BE	Float	4 Byte Signed	Read/Write
959	DOUT11.MODE	3BF	Integer	1 Byte	Read/Write
960	DOUT11.PARAM	3C0	Float	8 Byte Signed	Read/Write
961	DOUT11.PARAM (32 bit version)	3C1	Float	4 Byte Signed	Read/Write
962	CMP0.SOURCEVALUE	3C2	Float	8 Byte Signed	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
963	CMP0.SOURCEVALUE (32 bit version)	3C3	Float	4 Byte Signed	Read Only
964	CMP1.SOURCEVALUE	3C4	Float	8 Byte Signed	Read Only
965	CMP1.SOURCEVALUE (32 bit version)	3C5	Float	4 Byte Signed	Read Only
967	FAULT570.ACTION	3C7	Integer	1 Byte	Read/Write
968	DRV.FAULTDISPLAYMODE	3C8	Integer	1 Byte	Read/Write
969	FB1.MOTORPHASE	3C9	Integer	2 Byte	Read/Write
970	FB1.MOTORPOLES	3CA	Integer	2 Byte	Read/Write
971	FB2.MOTORPHASE	3CB	Integer	2 Byte	Read/Write
972	FB2.MOTORPOLES	3CC	Integer	2 Byte	Read/Write
973	FB3.MOTORPHASE	3CD	Integer	2 Byte	Read/Write
974	FB3.MOTORPOLES	3CE	Integer	2 Byte	Read/Write
982	PL.PDELAY	3D6	Integer	4 Byte	Read/Write
983	VL.FFDELAY	3D7	Integer	4 Byte	Read/Write
984	MOTOR.FIELDWEAKENING	3D8	Integer	1 Byte	Read/Write
1010	SM.ACC	3F2	Acceleration	8 Byte	Read/Write
1011	SM.ACC (32 bit version)	3F3	Acceleration	4 Byte	Read/Write
1012	SM.DEC	3F4	Acceleration	8 Byte	Read/Write
1013	SM.DEC (32 bit version)	3F5	Acceleration	4 Byte	Read/Write
1014	DRV.REBOOT	3F6	Integer	4 Byte	Write Only
1016	AIN.UVFTHRESH	3F8	Float	2 Byte Signed	Read/Write
1017	AIN.UVWTHRESH	3F9	Float	2 Byte Signed	Read/Write
1018	AIN.OVFTHRESH	3FA	Float	2 Byte Signed	Read/Write
1019	AIN.OVWTHRESH	3FB	Float	2 Byte Signed	Read/Write
1021	PL.FILTERTIME	3FD	Float	4 Byte	Read/Write
1022	VL.THRESHCUTOFF	3FE	Float	4 Byte	Read/Write
1026	PL.ERRFACTOR	402	Float	4 Byte	Read/Write
1027	PL.ERRTIME	403	Integer	2 Byte	Read/Write
1028	HOME.IPEAKACTIVE	404	Integer	1 Byte	Read/Write
1029	DRV.EMUESTEPCMDPIN	405	Integer	4 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
1030	DRV.EMUESTEPCMDPOUT	406	Integer	4 Byte	Read/Write
1031	HOME.TPOSWND	407	Position	8 Byte Signed	Read/Write
1032	HOME.TPOSWND (32 bit version)	408	Position	4 Byte Signed	Read/Write
1033	IL.VLIMIT	409	Velocity	8 Byte	Read/Write
1034	IL.VLIMIT (32 bit version)	40A	Velocity	4 Byte	Read/Write
1035	IL.KPSOURCE	40B	Float	1 Byte	Read/Write
1036	GEAR.SYNCWND	40C	Velocity	4 Byte	Read/Write
1040	COGCOMP.EN	410	Integer	1 Byte	Read/Write
1041	COGCOMP.CORRECTIONVALUE	411	Integer	4 Byte Signed	Read Only
1042	FBUS.SYNCACQUIREWND	412	Integer	4 Byte	Read/Write
1043	FBUS.SYNCLOCKWND	413	Integer	4 Byte	Read/Write
1045	COGCOMP.V	415	Velocity	8 Byte Signed	Read/Write
1046	COGCOMP.V (32 bit version)	416	Velocity	4 Byte Signed	Read/Write
1047	COGCOMP.RANGEHIGH	417	Position	8 Byte Signed	Read/Write
1048	COGCOMP.RANGEHIGH (32 bit version)	418	Position	4 Byte Signed	Read/Write
1049	COGCOMP.RANGELOW	419	Position	8 Byte Signed	Read/Write
1050	COGCOMP.RANGELOW (32 bit version)	41A	Position	4 Byte Signed	Read/Write
1054	ECAT.LEGACYREV	41E	Integer	1 Byte	Read/Write
1056	WS.FORCEOFF	420	Integer	1 Byte	Read/Write
1057	FAULT314.ACTION	421	Integer	1 Byte	Read/Write
1058	FB1.ENCSIGN	422	Integer	1 Byte	Read/Write
1069	PL.GEARIN	42D	Integer	2 Byte	Read/Write
1070	PL.GEAROUT	42E	Integer	2 Byte	Read/Write
1071	AIN.VSCALE	42F	Velocity	8 Byte	Read/Write
1072	AIN.VSCALE (32 bit version)	430	Velocity	4 Byte	Read/Write
1073	BODE.VAMP	431	Velocity	8 Byte Signed	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
1074	BODE.VAMP (32 bit version)	432	Velocity	4 Byte Signed	Read/Write
1075	CS.VTHRESH	433	Velocity	8 Byte	Read/Write
1076	CS.VTHRESH (32 bit version)	434	Velocity	4 Byte	Read/Write
1077	GEAR.VMAX	435	Velocity	8 Byte	Read/Write
1078	GEAR.VMAX (32 bit version)	436	Velocity	4 Byte	Read/Write
1079	HOME.V	437	Velocity	8 Byte	Read/Write
1080	HOME.V (32 bit version)	438	Velocity	4 Byte	Read/Write
1081	MT.TVELWND	439	Velocity	8 Byte	Read/Write
1082	MT.TVELWND (32 bit version)	43A	Velocity	4 Byte	Read/Write
1083	MT.V	43B	Velocity	8 Byte	Read/Write
1084	MT.V (32 bit version)	43C	Velocity	4 Byte	Read/Write
1085	MT.VCMD	43D	Velocity	8 Byte Signed	Read Only
1086	MT.VCMD (32 bit version)	43E	Velocity	4 Byte Signed	Read Only
1087	SM.V1	43F	Velocity	8 Byte Signed	Read/Write
1088	SM.V1 (32 bit version)	440	Velocity	4 Byte Signed	Read/Write
1089	SM.V2	441	Velocity	8 Byte Signed	Read/Write
1090	SM.V2 (32 bit version)	442	Velocity	4 Byte Signed	Read/Write
1091	VL.BUSFF	443	Velocity	8 Byte Signed	Read Only
1092	VL.BUSFF (32 bit version)	444	Velocity	4 Byte Signed	Read Only
1093	VL.CMD	445	Velocity	8 Byte Signed	Read Only
1094	VL.CMD (32 bit version)	446	Velocity	4 Byte Signed	Read Only
1095	VL.CMDU	447	Velocity	8 Byte Signed	Read/Write
1096	VL.CMDU (32 bit version)	448	Velocity	4 Byte Signed	Read/Write
1097	VL.ERR	449	Velocity	8 Byte Signed	Read Only

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
1098	VL.ERR (32 bit version)	44A	Velocity	4 Byte Signed	Read Only
1099	VL.FB	44B	Velocity	8 Byte Signed	Read Only
1100	VL.FB (32 bit version)	44C	Velocity	4 Byte Signed	Read Only
1101	VL.FBFILTER	44D	Velocity	8 Byte Signed	Read Only
1102	VL.FBFILTER (32 bit version)	44E	Velocity	4 Byte Signed	Read Only
1103	VL.FF	44F	Velocity	8 Byte Signed	Read Only
1104	VL.FF (32 bit version)	450	Velocity	4 Byte Signed	Read Only
1105	VL.LIMITN	451	Velocity	8 Byte Signed	Read/Write
1106	VL.LIMITN (32 bit version)	452	Velocity	4 Byte Signed	Read/Write
1107	VL.LIMITP	453	Velocity	8 Byte	Read/Write
1108	VL.LIMITP (32 bit version)	454	Velocity	4 Byte	Read/Write
1109	VL.MODEL	455	Velocity	8 Byte Signed	Read Only
1110	VL.MODEL (32 bit version)	456	Velocity	4 Byte Signed	Read Only
1111	VL.THRESH	457	Velocity	8 Byte Signed	Read/Write
1112	VL.THRESH (32 bit version)	458	Velocity	4 Byte Signed	Read/Write
1113	WS.VTHRESH	459	Velocity	8 Byte Signed	Read/Write
1114	WS.VTHRESH (32 bit version)	45A	Velocity	4 Byte Signed	Read/Write
1115	DRV.NVCHECK	45B	Integer	8 Byte	Read Only
1116	DRV.NVCHECK (32 bit version)	45C	Integer	4 Byte	Read Only
1123	PL.MOTIONLIMITEN	463	Integer	1 Byte	Read/Write
1124	FBUS.VLCMDFILTERTIME	464	Float	4 Byte	Read/Write
1125	MOTOR.RSOURCE	465	Integer	1 Byte	Read Only
1126	FAULT587.ACTION	466	Integer	1 Byte	Read/Write
1130	IL.MI2TFTHRESH	46A	Integer	1 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
1131	FB1.OFFSET	46B	Position	8 Byte Signed	Read/Write
1132	FB1.OFFSET (32 bit version)	46C	Position	4 Byte Signed	Read/Write
1133	FB2.OFFSET	46D	Position	8 Byte Signed	Read/Write
1134	FB2.OFFSET (32 bit version)	46E	Position	4 Byte Signed	Read/Write
1135	FB3.OFFSET	46F	Position	8 Byte Signed	Read/Write
1136	FB3.OFFSET (32 bit version)	470	Position	4 Byte Signed	Read/Write
1138	FB1.SIGNALAMPLITUDE	472	Float	4 Byte	Read Only
1139	ECAT.ENEMCYREQ	473	Integer	1 Byte	Read/Write
1140	CANOPEN.ADDMANUEMCYCODE	474	Integer	1 Byte	Read/Write
1144	FAULT107.ACTION	478	Integer	1 Byte	Read/Write
1145	FAULT108.ACTION	479	Integer	1 Byte	Read/Write
1147	COGCOMP.TEACHMODE	47B	Integer	1 Byte	Read/Write
1149	REGEN.TESTTO	47D	Integer	2 Byte	Read/Write
1150	REGEN.TESTVTHRESH	47E	Integer	1 Byte	Read/Write
1151	FB1.DIR	47F	Integer	1 Byte	Read/Write
1153	FAULT583.ACTION	481	Integer	1 Byte	Read/Write
1154	FB1.SFDCRCERRORCOUNT	482	Integer	1 Byte	Read Only
1155	MT.TPOSMINTIME	483	Integer	2 Byte	Read/Write
1156	VL.FBDISPLAY	484	Velocity	8 Byte Signed	Read Only
1157	VL.FBDISPLAY (32 bit version)	485	Velocity	4 Byte Signed	Read Only
1158	VL.FBDISPLAYCUTOFF	486	Float	4 Byte	Read/Write
1159	PWM0.DUTYCYCLE	487	Float	4 Byte	Read/Write
1160	PWM0.PERIOD	488	Float	4 Byte	Read/Write
1163	JOG.ACC	48B	Acceleration	8 Byte	Read/Write
1164	JOG.ACC (32 bit version)	48C	Acceleration	4 Byte	Read/Write
1165	JOG.DEC	48D	Acceleration	8 Byte	Read/Write
1166	JOG.DEC (32 bit version)	48E	Acceleration	4 Byte	Read/Write
1167	JOG.V	48F	Velocity	8 Byte	Read/Write

Instance (ID)	Parameter	Hex	Data Type	Data Size	Access
1168	JOG.V (32 bit version)	490	Velocity	4 Byte	Read/Write
8192	EipOperationMode	2000	Integer	1 Byte	Read/Write
8193	EipStatus3	2001	Integer	1 Byte	Read Only
8194	EipVBusValue	2002	Integer	2 Byte	Read Only
8195	EipDinStates	2003	Integer	1 Byte	Read Only
8196	EipDoutStates	2004	Integer	1 Byte	Read Only
8197	EipControlWord2	2005	Integer	1 Byte	Write Only

8.3 AKD1G Explicit Messaging using the MSG Instruction

In addition to the Add-On Instructions listed in this manual, most of the drive parameters can be read or set using explicit Messaging and the MSG instruction. The AKD1G Parameter Listing appendix provides a list of available parameters.

8.3.1 Read A Drive Parameter

To read a parameter, create a MSG instruction with the following settings:

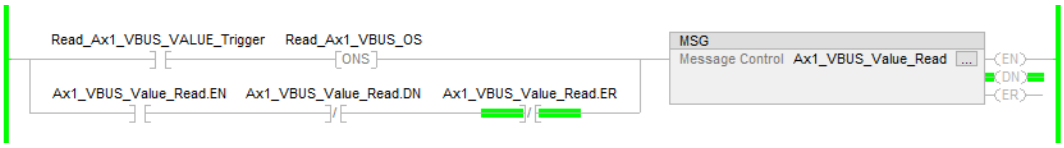
Field	Value
Message Type	CIP Generic
Service Type	Parameter Read or Get Attribute Single depending on the version of RSLogix 5000 or Studio 5000 and the MSG instruction.
Service Code	0xE (Hex)
Class	0xF (Hex)
Instance	Parameter Instance (ID) from AKD1G Parameter Listing
Attribute	1 (Always set to 1)
Destination	Create a Tag to hold the value.
Communication → Path	Name of the ETHERNET-MODULE for the AKD axis.Use the Browse button.

Example:

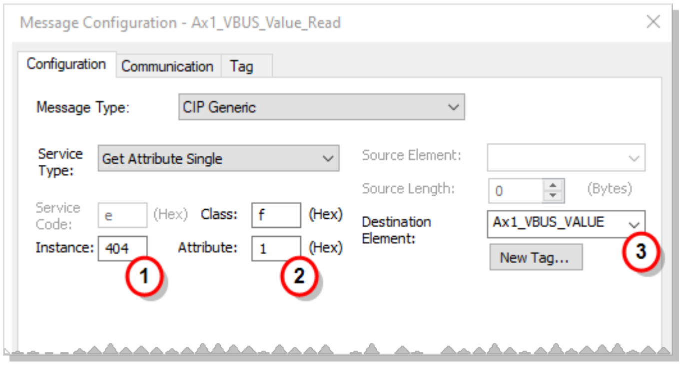
Read the VBUS.VALUE.

Instance (ID)	Parameter	Data Size	Data Type
404	VBUS.VALUE	4 Byte	Float

The Read_Ax1_VBUS_Value_Trigger N.O. Contact triggers a One Shot on the .EnableIn of the MSG instruction. A seal-in parallel branch uses the .EN, .DN, and .ER bits of the MSG instruction to keep the MSG instruction enabled until either .DN (Done; success) or .ER (Error; fail).

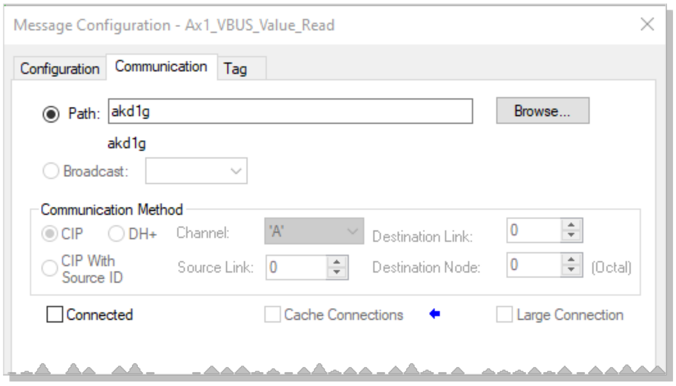


Set the **Instance** and **Attribute** fields to access the Axis along with any additional required settings.



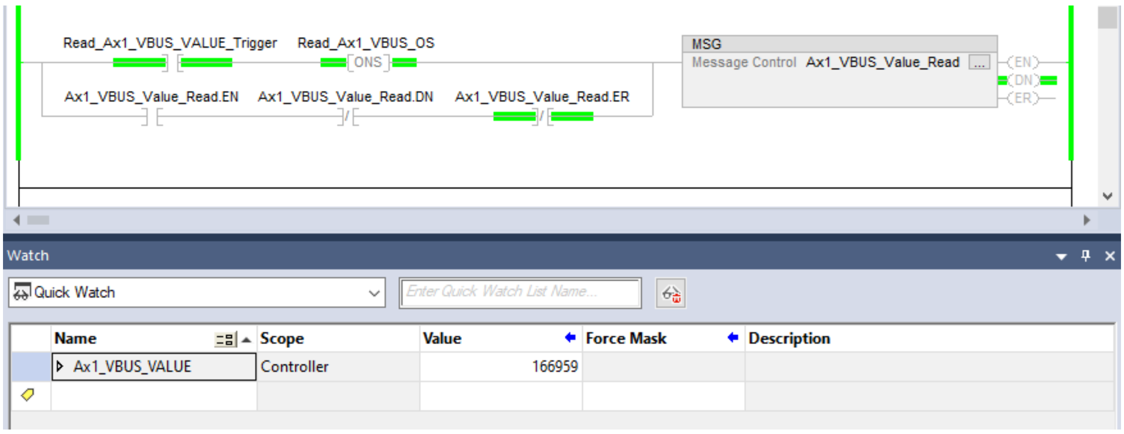
1. Parameter Number in decimal format.
2. Attribute (Always a value of 1 for AKD1G.)
3. Tag to hold the value read.
   Tag is declared with a data type matching the data type of the parameter to read.

Set the **Path** field to the Generic Ethernet Module for the corresponding drive.



Execution

On toggle of Read_Ax1_VBUS_Value_Trigger N.O. Contact the One Shot to the .EnableIn of the MSG instruction enables the instruction and a parallel branch using the .EnableIn, the .DN, and .ER bits seals in the MSG instruction until .DN (Done; success) or .ER (Error; fail).



The Quick Watch in Studio 5000 was used to display the value of the destination Tag of the MSG instruction.

8.3.2 Write A Drive Parameter

To write a parameter, create a MSG instruction with the following settings:

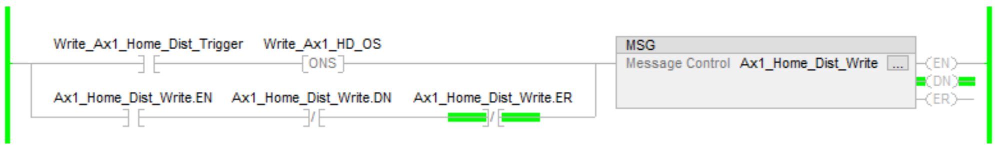
Field	Value
Message Type	CIP Generic
Service Type	Parameter Write
Service Code	0x10 (Hex)
Class	0xF (Hex)
Instance	Parameter Instance (ID) from AKD1G Parameter Listing
Attribute	1 (Always set to 1)
Source Element	Create a Tag to hold the value.
Source Length	Number of Bytes required; Same number of Bytes as the Parameter Instance Data Size from AKD1G Parameter Listing.
Communication → Path	Name of the ETHERNET-MODULE for the AKD axis.Use the Browse button.

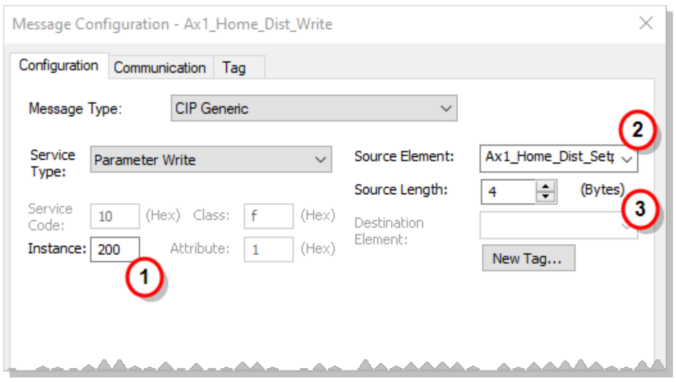
Example:

Write to USER.INT1.

Instance (ID)	Parameter	Data Size	Data Type
200	HOME.DIST	4 Byte Signed (32 bit version)	Position

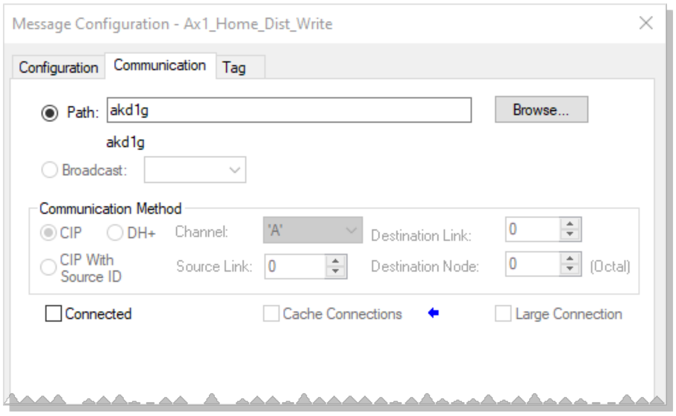
The Write_Ax1_Home_Dist_Trigger N.O. Contact triggers a One Shot on the .EnableIn of the MSG instruction. A seal-in parallel branch uses the .EN, .DN, and .ER bits of the MSG instrcution to keep the MSG instruction enabled until either .DN (Done; success) or .ER (Error; fail).





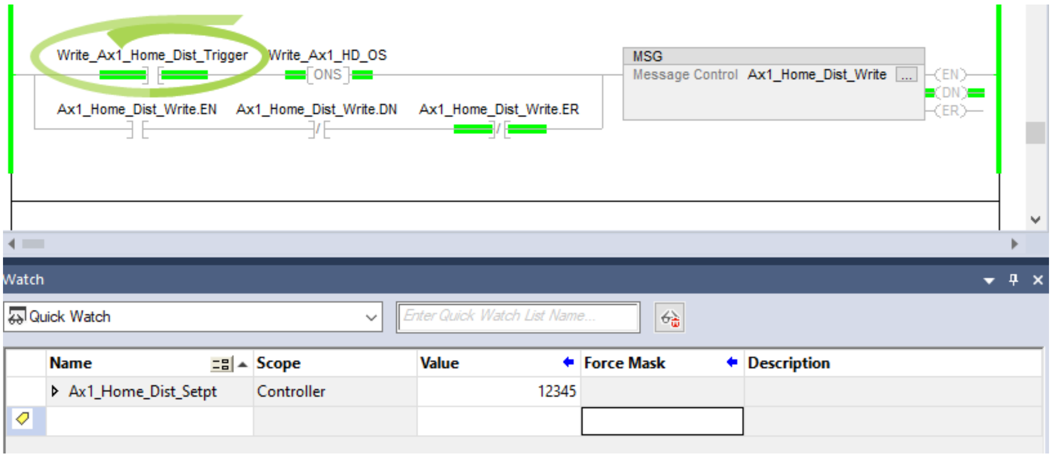
1. Parameter's Instance (ID) in decimal.
2. Source Element: Tag to source data to write. Tag is declared with a data type matching the data type of the AKD parameter.
3. Source Length: Matches the number of Bytes for the given AKD drive parameter.

Set the **Path** to the Generic Ethernet Module for the corresponding drive.



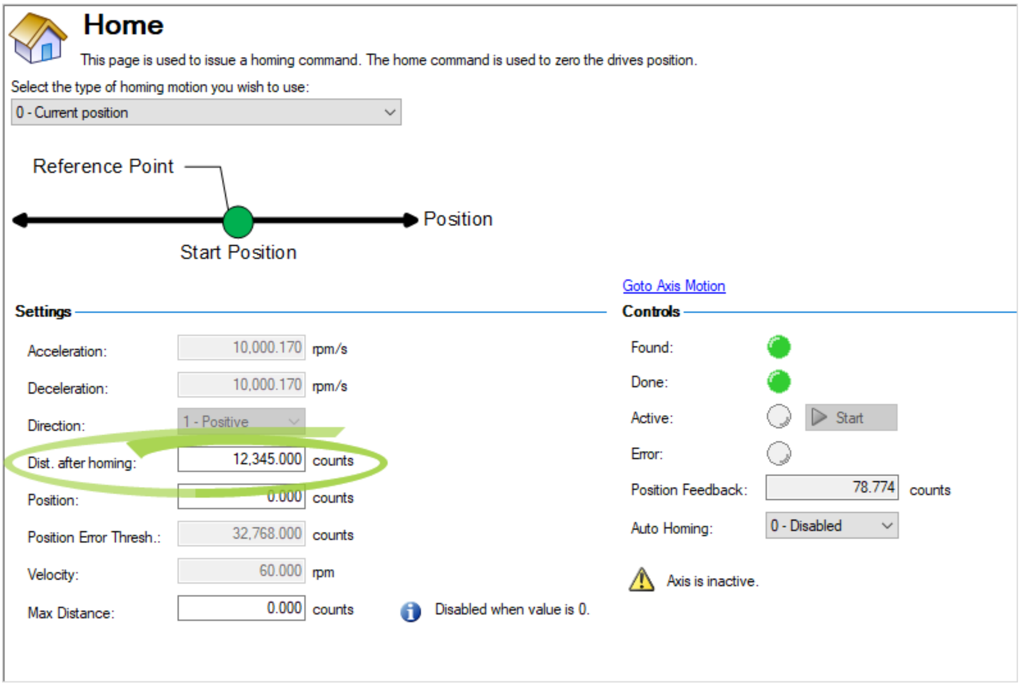
Execution

On toggle of Write_Ax1_Home_Dist_Trigger N.O. Contact the One Shot to the .EnableIn of the MSG instruction enables the instruction and a parallel branch using the .EnableIn, the .DN, and .ER bits seals in the MSG instruction until .DN (Done; success) or .ER (Error; fail).



The Quick Watch in Studio 5000 was used to set the value of the source Tag of the MSG instruction.

WorkBench displays the value written to HOME.DIST.


Home
This page is used to issue a homing command. The home command is used to zero the drives position.
Select the type of homing motion you wish to use:
0 - Current position
Reference Point
Position
Start Position
Goto Axis Motion
Settings
Acceleration: 10,000.170 rpm/s
Deceleration: 10,000.170 rpm/s
Direction: 1 - Positive
Dist. after homing: 12,345.000 counts
Position: 0.000 counts
Position Error Thresh.: 32,768.000 counts
Velocity: 60.000 rpm
Max Distance: 0.000 counts
Disabled when value is 0.
Controls
Found:
Done:
Active:
Start
Error:
Position Feedback: 78.774 counts
Auto Homing: 0 - Disabled
Axis is inactive.

8.3.3 Execute A Drive Command Parameter

Some drive parameters are commands which do not take a value, but execute a drive function such as the HOME.MOVE or DRV.CLRFAULTS. To execute a command, create a MSG instruction to write to the command. Altough the command parameter doesn't take a value, it is common practice to write a value of 1.

To execute a drive command, create a MSG instruction with the following settings:

Field	Value
Message Type	CIP Generic
Service Type	Parameter Write
Service Code	0x10 (Hex)
Class	0xF (Hex)
Instance	Parameter Instance (ID) from AKD1G Parameter Listing
Attribute	1 (Always set to 1)
Source Element	Create a Tag to hold the value. Any actual value may be used since it is ignored. Common practice is to write a value of 1.
Source Length	1 Byte
Communication → Path	Name of the ETHERNET-MODULE for the AKD axis.Use the Browse button.

8.3.4 Axis Parameter Data Size and PLC Tag Type

Axis Parameter Data Size	AB PLC Tag Type*
Command	SINT
1 Byte	SINT
1 Byte Signed	SINT
2 Byte	INT
2 Byte Signed	INT
4 Byte	DINT
4 Byte Signed	DINT
8 Byte**	LINT
8 Byte Signed**	LINT

** Tag Type REAL is not appicable with the AKD EtherNet/IP parameters.*
*** Note: There is no 8 Byte or 8 Byte Signed for AKD2G drives.*

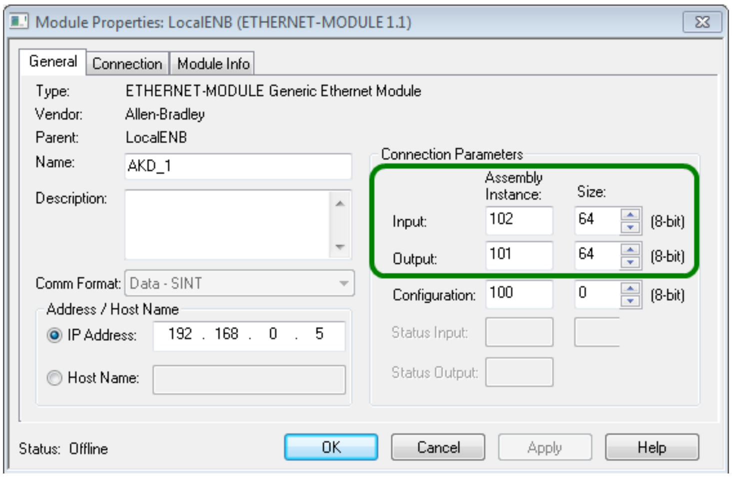
8.4 Example of AKD1G Dynamic Mapping With Studio 5000

8.4.1 AKD1G Ethernet IP: Diagnostics and Dynamic Mapping

How to read parameters for information (i.e. position feedback) and diagnostics beyond what the Sample project provides:

For users of CompactLogix and ControlLogix, cyclic data is supported with the AKD EtherNet/IP drive. The MicroLogix 1400 does not support cyclic data; parameter read/writes must be achieved explicitly via the MSG blocks. This application note is specific to cyclic data.

The Command Assembly and the Response Assembly are made up of Bytes 0 to 63 (64 Bytes total) with 64 Bytes for the input and 64 Bytes for the output of each assembly. This correlates to the PLC and drive via the AKD drive being setup as a Generic Ethernet Module in the PLC. See the AKD EtherNet/IP manual for more information on these assemblies.



Configuring the Generic Ethernet Module adds the AKD_1 to the Controller Tags.

8.4.2 Static vs. Dynamic Mapping

It is important to note there are portions of the assemblies which are static (pre-configured) and portions which are dynamic (flexible; not pre-configured).

- Bytes 0-35 are static (pre-configured). This mapping cannot be changed.
- Byte 33 is the Map Type. By default, the Map Type is 0. This means only the static portion of both the Command Assembly and Response Assembly are on the cyclic data. As noted in the table, if Byte 33 in the Command Assembly is set to a value of 2 it enables the data exchange for the parameters which are dynamically configured.
- Parameters dynamically mapped to the Command Assembly and Response Assembly in the AKD drive will not be written to or read from unless the Map Type is 2: Dynamic.

Command Assembly: Static Bytes 0-32

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)

*Least significant byte first for all data fields

Response Assembly: Static Bytes 0-32

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.

* Least significant byte first for all data fields

8.4.3 Map Type and Dynamic Command Assembly

The Map Type in the Command Assembly determines what Bytes and mapping are used.

The default is 0: Static Map and only Bytes 0-35 are used.

If the PLC sets the Map Type (Byte 33) to 2 in the Command Assembly, then Bytes 36-63 (which were dynamically configured in the Command Assembly) are sent to the Command Assembly. The data sent to the drive will be included on the cyclic data updated on every RPI scan.

Byte	Data	Comment
33	Map Type	0. Static Map (only bytes 0 to 35 are sent) 1. Custom Map 1 2. Dynamic Map (bytes 36-63 are dynamically configurable)
34-35	Reserved	
36-63	Command Dynamic Map	See EIP.CMDMAP or using WorkBench GUI.

8.4.4 Map Type and Dynamic Response Assembly

Like the Command Assembly, the Response Assembly has 64 Bytes from 0-63.

- Bytes 0-35 make up the Response Assembly's static mapping.
- Byte 33: Map Type reflects the current Map Type as set by Byte 33 in the Command Assembly.
  - If Byte 33 is 0 then only the static Bytes 0-35 are read.
  - If Byte 33 reads a 2 (as in the case the PLC sets Byte 33 to a value of 2) then Bytes 36-63 when dynamically mapped will read the drive's parameter data on every RPI scan.

Map Type and Dynamic Response Assembly Bytes

Byte	Data	Comment
33	Map Type	0: Static Map (only bytes 0 to 35 are sent) 1: Custom Map 1 2: Dynamic Map (bytes 36-63 are dynamically configurable)
34-35	Reserved	
36-63	Response Dynamic Map	See EIP.RSPMAP or using WorkBench GUI.

8.4.5 Using Static Mapping

There is already important diagnostic information built into the default static map of the Response Assembly without additional dynamic mapping. This includes:

- Status Word 1
- Status Word 2
- Actual Position
- Actual Velocity
- Motion Status (equivalent to AKD parameter DRV.MOTIONSTAT)

Byte	Data	Comment
0	Status Word 1	Various status bits
2	Status Word 2	Various status bits
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.

* Least significant byte first for all data fields

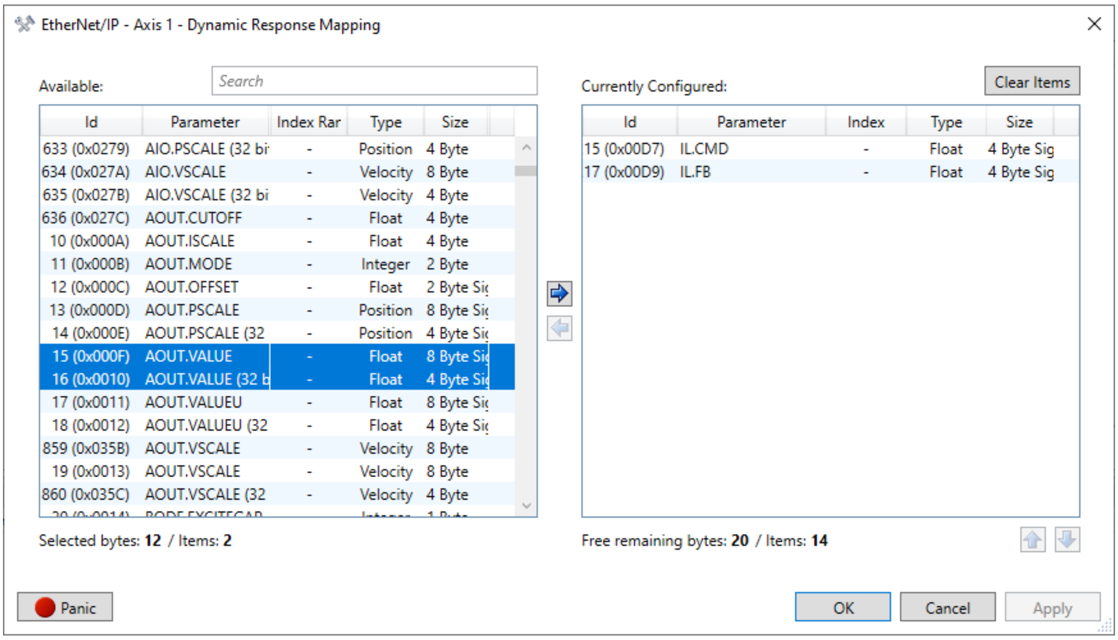
If other parameters and diagnostics are desired beyond what is contained in the static assembly then these drive parameters can be mapped using one of two methods in WorkBench:

- Use the terminal in WorkBench with the EIP.CMDMAP and EIP.RSPMAP keywords. (See AKD EtherNet/IP Communications manual.)
- Use the WorkBench GUI under **device_name (Online)** → **Settings** → **Communication** → **EtherNet/IP** → **Axis**.

Note the Parameter Listings in the manuals will sometimes list a Data Type of Float, but in the WorkBench GUI Dynamic Mapping List will show as Integer.

Floating Point Values Over EtherNet/IP:

To determine the 32-bit ID, add 1 to the ID of the 8 Byte version. Example: AIN.PSCALE, 8 Byte Instance 5. Since 5 + 1 = 6 then the AIN.PSCALE (32 bit version) will be ID6. The WorkBench Dynamic Response Mapping List provides an example of this.



Three Methods To Look Up IDs:

1. Refer to the AKD1G Parameter Listing.
2. In the WorkBench Terminal, run the command EIP.OBJECTLIST.
3. In the WorkBenchGUI navigate to the Dynamic Response Mapping tab as shown and click the Configure button.

Key takeaways:

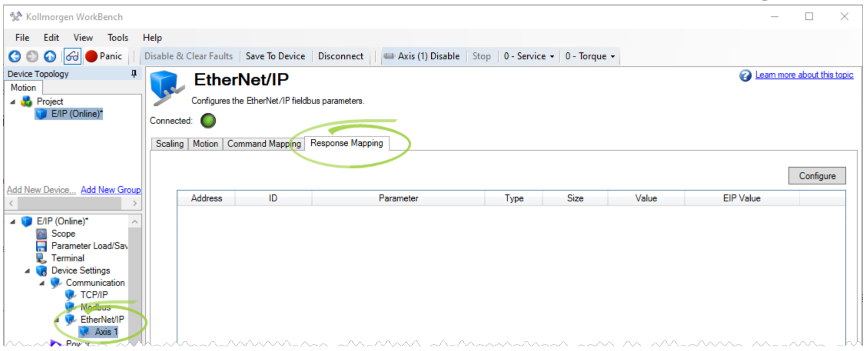
- Use the latest version of WorkBench and AKD firmware on new applications.
- Every Read/Write should be tested by the programmer for data validity between the PLC and the AKD.

8.4.6 Example of using Dynamic Mapping AKD1G

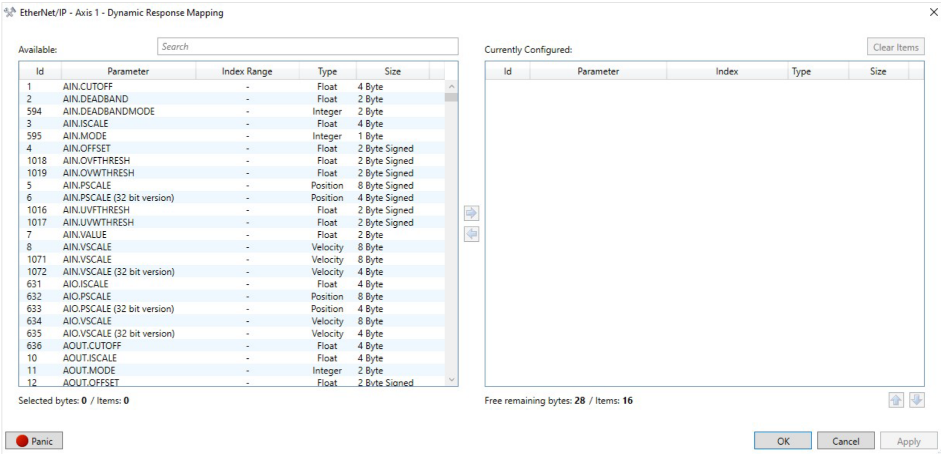
Using WorkBench to Dynamically Map EtherNet/IP Parameters

This example demonstrates dynamically mapping the IL.CMD and IL.FB using the WorkBench GUI to perform the Response Mapping of commands for the axis. Command Mapping follows the same procedure.

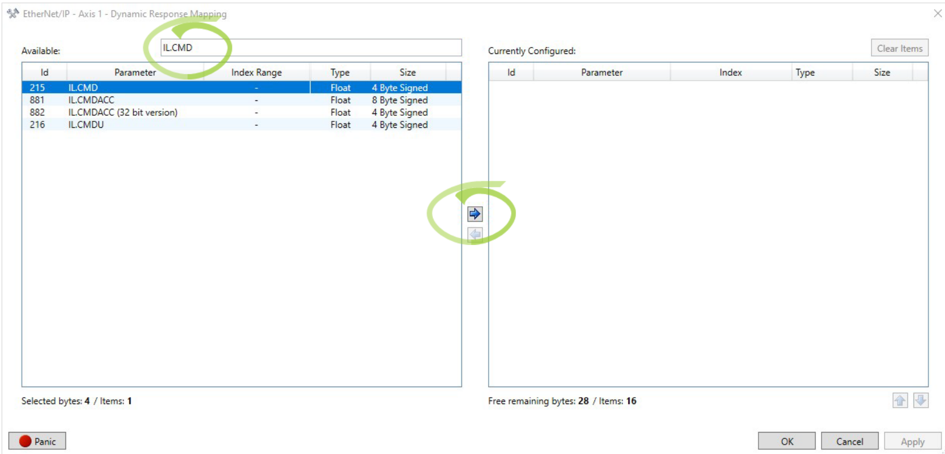
1. Using the WorkBench GUI, navigate to **device_name (Online)** → **Device Settings** → **Communication** → **EtherNet/IP** → **Axis** → **Response Mapping** tab.



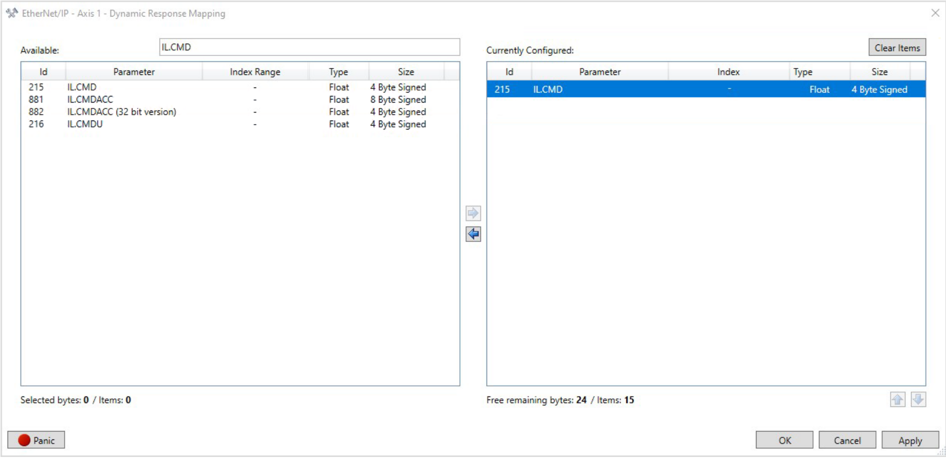
2. Select the **Configure** button to open the Dynamic Response Mapping screen.
The Available list on the left shows all available Parameters (IDs). The Currently Configured list on the right is empty by default.



3. Use the Search field to find the desired parameter.
In this example the Current Loop command (IL.CMD) and feedback (IL.FB) are dynamically mapped.



4. Select the desired parameter(s) in the Available list and use the **right arrow button** at the center of the screen to move the parameter into the **Currently Configured** list.



5. Click **Apply** and then **OK**. Repeat these steps to dynamically map IL.FB.
   The Response Mapping tab now shows the addesses (in Byes), ID, Parameter Name, Type, Size, WorkBench Value and Units, and EIP Value.

See Using WorkBench to Dynamically Map EtherNet/IP Parameters for instructions on performing Response Mapping.

Monitoring Current over EtherNet/IP using the AKD1G EtherNet/IP Drive

It is not recommended to use an AOI to read the drive's current with continuous updates.

In order to read the drive's current with continuous updates, it is not recommended to use an AOI for this purpose.

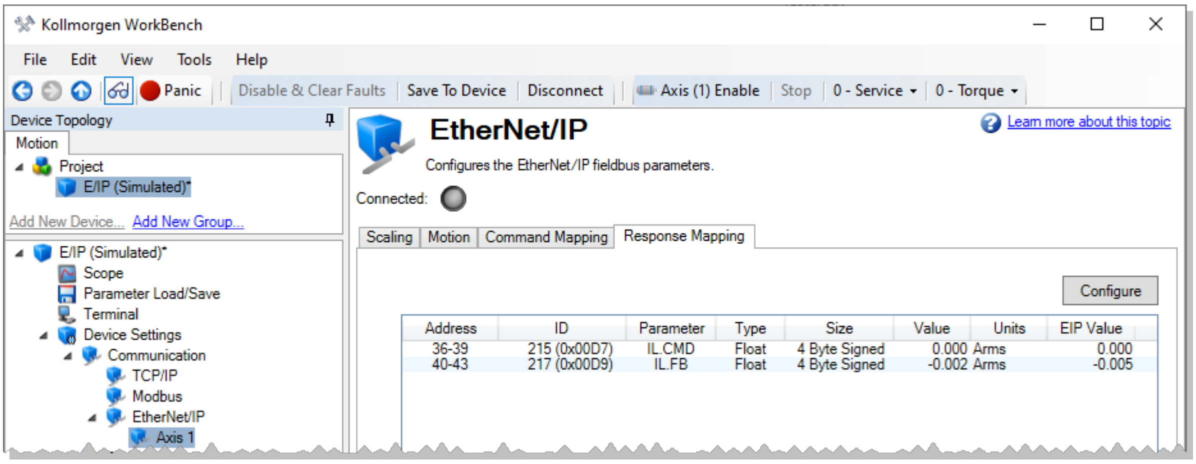
NOTE

When monitoring or sampling current values, the fastest rate for an RPI scan for each Generic Ethernet Module is 20 msec.

- Only one AOI per axis can be executing at a time. All AOIs use the same static Bytes 0-35 from the Command Assembly and Response Assembly, effecting the EtherNet/IP communcations speed.
- A Torque Parameter does not exist in the AKD drive so only milliamps or amps can be queried.
- To monitor current in the drive, check either IL.FB (Feedback) or IL.CMD (Command).

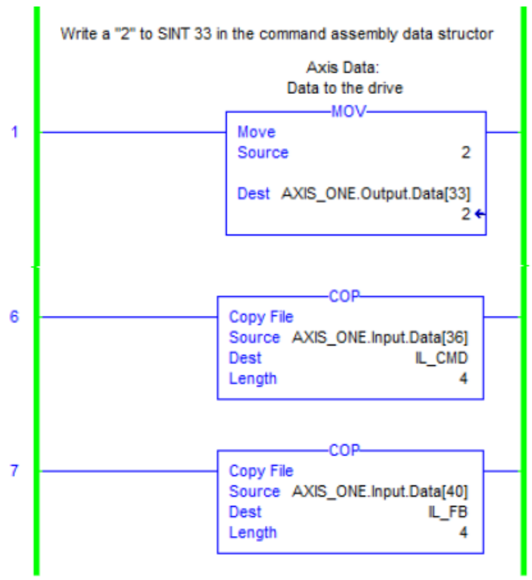
The best method for monitoring current is via Dynamic Mapping (cyclic data). To do this:

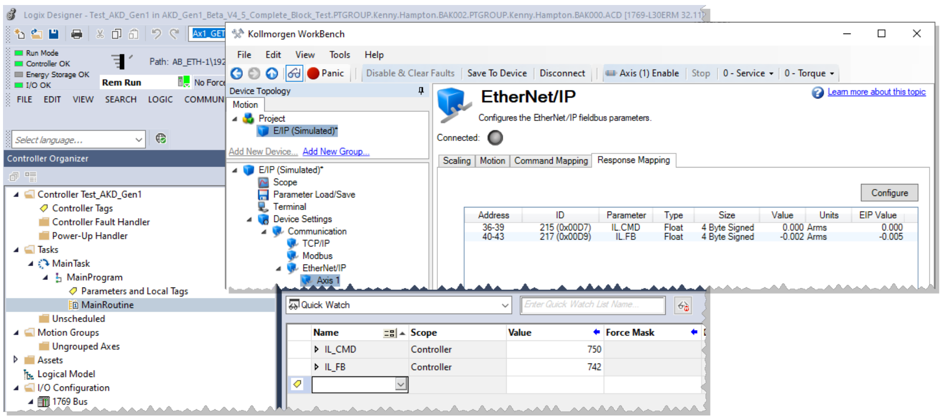
1. Lookup the Instance (ID) for the given parameter in AKD1G Parameter Listing.
2. To read the parameters, map the desired parameters in the Response Assembly Data Structure under **Settings** → **Communication** → **EtherNet/IP** → **Response Mapping** tab.



NOTE

Mapping does not put the parameters on the poll. To enable polling, it is required to move a value of 2 into Byte 33 of the Command Assembly Data Structure. The rung below shows an example of this along with data copied into Tags in the Ladder using the COP instruction. The data is 4 Byte signed, or a DINT data type which is what was declared in the IL_CMD and IL_FB Tags as in the PLC Controller's Tags. For a list of data sizes and tag types see AKD1G Explicit Messaging using the MSG Instruction.


Write a "2" to SINT 33 in the command assembly data structor
Axis Data:
Data to the drive
1
MOV
Move
Source 2
Dest AXIS_ONE.Output.Data[33]
2
6
COP
Copy File
Source AXIS_ONE.Input.Data[36]
Dest IL_CMD
Length 4
7
COP
Copy File
Source AXIS_ONE.Input.Data[40]
Dest IL_FB
Length 4


Kollmorgen WorkBench
EtherNet/IP
Configures the EtherNet/IP fieldbus parameters.
Controller Organizer
Quick Watch

9 AKD2G Appendices

9.1 AKD2G EtherNet/IP Objects and Attributes

The following table provides supported attributes.

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.

9.2 AKD2G 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. A value displayed in WorkBench as 1.001 will be transmitted through EtherNet/IP as 1001.

ID	Hex	Name	Data Type	Access
2000	0x7d0	DRV.DIS	Unsigned8	Read/Write
2001	0x7d1	DRV.BLINKDISPLAY	Unsigned8	Read/Write
2002	0x7d2	DRV.CLRFAULTS	Unsigned8	Read/Write
2003	0x7d3	DRV.TYPE	Unsigned8	Read Only
2004	0x7d4	DRV.RSTVAR	Unsigned8	Read/Write
2005	0x7d5	DRV.NVSAVE	Unsigned8	Read/Write
2006	0x7d6	DRV.STOP	Unsigned8	Read/Write
2007	0x7d7	DRV.NVLOAD	Unsigned8	Read/Write
2008	0x7d8	DRV.RUNTIME	Unsigned32	Read Only
2009	0x7d9	DRV.NVCHECK	Unsigned32	Read Only
2010	0x7da	DRV.CUSTOMIDENTIFIER	String	Read/Write
2011	0x7db	DRV.NAME	String	Read/Write
2012	0x7dc	DRV.NVVER	String	Read/Write
2013	0x7dd	DRV.REBOOT	Unsigned32	Read/Write
2014	0x7de	DRV.DOWNLOADALLOWED	Unsigned8	Read Only
2015	0x7df	DRV.ADDUSER	String	Read/Write
2016	0x7e0	DRV.DELUSER	String	Read/Write
2017	0x7e1	DRV.LOGOUT	Unsigned8	Read/Write
2018	0x7e2	DRV.SETUSERPWD	String	Read/Write
2019	0x7e3	DRV.TEMPFTHRESH	Unsigned8	Read Only
2020	0x7e4	DRV.TEMPWTHRESH	Unsigned8	Read Only
2021	0x7e5	DRV.TEMP	Signed32	Read Only
2200	0x898	IP.DEFAULTINTERFACE	Unsigned8	Read/Write
2201	0x899	IP.MODE	Unsigned16	Read/Write
2202	0x89a	IP.ADDRESS	Unsigned32	Read/Write
2203	0x89b	IP.GATEWAY	Unsigned32	Read/Write
2204	0x89c	IP.PROTOCOL	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
2205	0x89d	IP.RESET	Unsigned8	Read/Write
2206	0x89e	IP.SUBNET	Unsigned32	Read/Write
2300	0x8fc	REC.ACTIVE	Unsigned8	Read Only
2301	0x8fd	REC.DONE	Unsigned8	Read Only
2302	0x8fe	REC.GAP	Unsigned16	Read/Write
2303	0x8ff	REC.NUMPOINTS	Unsigned16	Read/Write
2304	0x900	REC.OFF	Unsigned8	Read/Write
2305	0x901	REC.STOPTYPE	Unsigned8	Read/Write
2306	0x902	REC.TRIG	Unsigned8	Read/Write
2307	0x903	REC.TRIGPOS	Unsigned8	Read/Write
2308	0x904	REC.RETRIEVESIZE	Unsigned16	Read/Write
2309	0x905	REC.TRIGSLOPE	Unsigned8	Read/Write
2310	0x906	REC.TRIGTYPE	Unsigned8	Read/Write
2312	0x908	REC.CH1	String	Read/Write
2313	0x909	REC.CH2	String	Read/Write
2314	0x90a	REC.CH3	String	Read/Write
2315	0x90b	REC.CH4	String	Read/Write
2316	0x90c	REC.CH5	String	Read/Write
2317	0x90d	REC.CH6	String	Read/Write
2318	0x90e	REC.RETRIEVEFRMT	Unsigned8	Read/Write
2319	0x90f	REC.TRIGMASK	Unsigned32	Read/Write
2320	0x910	REC.TRIGPARAM	String	Read/Write
2400	0x960	REGEN.POWER	Unsigned32	Read Only
2401	0x961	REGEN.REXT	Unsigned16	Read/Write
2402	0x962	REGEN.TEXT	Unsigned32	Read/Write
2403	0x963	REGEN.TYPE	Signed8	Read/Write
2404	0x964	REGEN.WATTEXT	Unsigned16	Read/Write
2405	0x965	REGEN.POWERFILTERED	Unsigned32	Read Only
2406	0x966	REGEN.TEST	Unsigned8	Read/Write
2407	0x967	REGEN.TESTSTATUS	Unsigned8	Read Only
2408	0x968	REGEN.TESTTO	Unsigned16	Read/Write
2409	0x969	REGEN.TESTVTHRESH	Unsigned8	Read/Write
2500	0x9c4	VBUS.VALUE	Signed32	Read Only
2501	0x9c5	VBUS.CAP	Unsigned32	Read Only

ID	Hex	Name	Data Type	Access
2502	0x9c6	VBUS.DCOPERATION	Unsigned8	Read/Write
2503	0x9c7	VBUS.ICAP	Signed32	Read Only
2504	0x9c8	VBUS.ICAPLIMIT	Signed32	Read Only
2505	0x9c9	VBUS.INRUSHOFF	Unsigned16	Read Only
2506	0x9ca	VBUS.INRUSHON	Unsigned16	Read Only
2507	0x9cb	VBUS.OVFTHRESH	Unsigned16	Read Only
2508	0x9cc	VBUS.OVWTHRESH	Unsigned16	Read/Write
2509	0x9cd	VBUS.THREEPHASE	Unsigned8	Read/Write
2510	0x9ce	VBUS.UVFTHRESH	Unsigned16	Read/Write
2511	0x9cf	VBUS.UVWTHRESH	Unsigned16	Read/Write
2512	0x9d0	VBUS.UVMODE	Unsigned8	Read/Write
2513	0x9d1	VBUS.ACNOMINAL	Unsigned16	Read/Write
2514	0x9d2	VBUS.DCNOMINAL	Unsigned16	Read/Write
2600	0xa28	FBUS.TYPE	Unsigned8	Read Only
2700	0xa8c	DIN.STATES	Unsigned32	Read Only
2701	0xa8d	DIO.STATES	Unsigned32	Read Only
2702	0xa8e	DOUT.STATES	Unsigned32	Read Only
2703	0xa8f	FBUS.DOUT.STATES	Unsigned32	Read/Write
2800	0xaf0	MW.USERBUFFER	Signed32[32]	Read/Write
2801	0xaf1	MW.MODEL1.STATE	Unsigned8	Read/Write
2802	0xaf2	MW.MODEL2.STATE	Unsigned8	Read/Write
2900	0xb54	HW.FANSPEED1	Signed32	Read Only
3000	0xbb8	GANTRY.PL.ERR	Signed32	Read Only
3001	0xbb9	GANTRY.PL.ERRFTHRESH	Signed32	Read/Write
3002	0xbba	GANTRY.PL.ERRWTHRESH	Signed32	Read/Write
3003	0xbbb	GANTRY.STATE	Unsigned8	Read Only
3004	0xbbc	GANTRY.HOME.REQUIRED	Unsigned8	Read/Write
3005	0xbbd	GANTRY.PL.ERRTTHRESH	Unsigned16	Read/Write
3100	0xc1c	BRAKE1.AXIS	Unsigned8	Read/Write
3101	0xc1d	BRAKE2.AXIS	Unsigned8	Read/Write
3105	0xc21	BRAKE1.STATE	Unsigned8	Read Only
3106	0xc22	BRAKE2.STATE	Unsigned8	Read Only
3200	0xc80	USER.INT1	Signed16	Read/Write
3201	0xc81	USER.INT2	Signed16	Read/Write

ID	Hex	Name	Data Type	Access
3202	0xc82	USER.INT3	Signed16	Read/Write
3203	0xc83	USER.INT4	Signed16	Read/Write
3204	0xc84	USER.INT5	Signed16	Read/Write
3205	0xc85	USER.INT6	Signed16	Read/Write
3206	0xc86	USER.INT7	Signed16	Read/Write
3207	0xc87	USER.INT8	Signed16	Read/Write
3208	0xc88	USER.INT9	Signed16	Read/Write
3209	0xc89	USER.INT10	Signed16	Read/Write
3302	0xce6	EIP.SAMPLEPERIOD	Unsigned32	Read Only
3400	0xd48	SD.LOGEN	Unsigned8	Read/Write
3401	0xd49	SD.STATUS	Unsigned8	Read Only
3500	0xdac	LOG.SOURCE	Unsigned32	Read/Write
3600	0xe10	X22.MODE	Unsigned8	Read/Write
3601	0xe11	X23.MODE	Unsigned8	Read/Write
3700	0xe74	MODBUS.CLRERRORS	Unsigned8	Read/Write
3702	0xe76	MODBUS.EN	Unsigned8	Read/Write
3703	0xe77	MODBUS.ENDIAN	Unsigned8	Read/Write
3704	0xe78	MODBUS.ERRORCOUNT	Unsigned16	Read Only
3705	0xe79	MODBUS.ERRORMODE	Unsigned8	Read/Write
3706	0xe7a	MODBUS.KEEPALIVE	Unsigned8	Read/Write
3707	0xe7b	MODBUS.MSGLOG	Unsigned8	Read/Write
3708	0xe7c	MODBUS.RSTMAP	Unsigned8	Read/Write
3709	0xe7d	MODBUS.WATCHDOG	Unsigned16	Read/Write
5000	0x1388	AXIS#.NAME	String	Read/Write
5001	0x1389	AXIS#.ZEROT	Unsigned32	Read/Write
5002	0x138a	AXIS#.ZEROV	Signed32	Read/Write
5003	0x138b	AXIS#.ACTIVE	Unsigned8	Read Only
5004	0x138c	AXIS#.CMDSOURCE	Unsigned8	Read/Write
5005	0x138d	AXIS#.DBILIMIT	Unsigned32	Read Only
5006	0x138e	AXIS#.DIR	Unsigned8	Read/Write
5007	0x138f	AXIS#.DIS	Unsigned8	Read/Write
5008	0x1390	AXIS#.DISMODE	Unsigned8	Read/Write
5009	0x1391	AXIS#.DISSOURCES	Unsigned16	Read Only
5010	0x1392	AXIS#.DISTO	Unsigned32	Read/Write

ID	Hex	Name	Data Type	Access
5011	0x1393	AXIS#.EN	Unsigned8	Read/Write
5012	0x1394	AXIS#.ENDEFAULT	Unsigned8	Read/Write
5013	0x1395	AXIS#.ICONT	Signed32	Read Only
5014	0x1396	AXIS#.IPEAK	Signed32	Read Only
5015	0x1397	AXIS#.OPMODE	Unsigned8	Read/Write
5016	0x1398	AXIS#.STOP	Unsigned8	Read/Write
5017	0x1399	AXIS#.SETUPREQBITS	Unsigned32	Read Only
5018	0x139a	AXIS#.WARNING1	Unsigned16	Read Only
5019	0x139b	AXIS#.WARNING2	Unsigned16	Read Only
5020	0x139c	AXIS#.WARNING3	Unsigned16	Read Only
5021	0x139d	AXIS#.MOTIONDISSOURCES	Unsigned16	Read Only
5022	0x139e	AXIS#.CLRFAULTS	Unsigned8	Read/Write
5023	0x139f	AXIS#.DISSOURCESMASK	Unsigned16	Read Only
5024	0x13a0	AXIS#.FAULTED	Unsigned8	Read Only
5025	0x13a1	AXIS#.MOTIONSTAT	Unsigned32	Read Only
5026	0x13a2	AXIS#.TEMP	Signed32	Read Only
5027	0x13a3	AXIS#.TEMPFTHRESH	Signed32	Read Only
5028	0x13a4	AXIS#.TEMPWTHRESH	Signed32	Read Only
5029	0x13a5	AXIS#.UTFTHRESH	Signed32	Read Only
5030	0x13a6	AXIS#.UTWTHRESH	Signed32	Read Only
5031	0x13a7	AXIS#.ZEROACC	Unsigned32	Read/Write
5032	0x13a8	AXIS#.ZEROREACHED	Unsigned8	Read Only
5033	0x13a9	AXIS#.MOTIONCONTROL	Unsigned16	Read/Write
5034	0x13aa	AXIS#.CLRWARNINGS	Unsigned8	Read/Write
5035	0x13ab	AXIS#.DBILIMITACTUAL	Unsigned32	Read Only
5036	0x13ac	AXIS#.MOTIONSTAT.HOMEFOUND	Unsigned8	Read Only
5100	0x13ec	AXIS#.FBUS.ACC	Unsigned32	Read/Write
5101	0x13ed	AXIS#.FBUS.DEC	Unsigned32	Read/Write
5102	0x13ee	AXIS#.FBUS.PROTECTION	Unsigned8	Read/Write
5103	0x13ef	AXIS#.FBUS.BLOCKING	Unsigned8	Read Only
5104	0x13f0	AXIS#.FBUS.IL.CMD	Signed32	Read/Write
5105	0x13f1	AXIS#.FBUS.V	Signed32	Read/Write
5106	0x13f2	AXIS#.FBUS.P	Signed32	Read/Write
5300	0x14b4	AXIS#.CS.DEC	Unsigned32	Read/Write

ID	Hex	Name	Data Type	Access
5301	0x14b5	AXIS#.CS.STATE	Unsigned8	Read Only
5400	0x1518	AXIS#.GUI.PARAM01	Signed32	Read/Write
5401	0x1519	AXIS#.GUI.PARAM02	Signed32	Read/Write
5402	0x151a	AXIS#.GUI.PARAM03	Signed32	Read/Write
5403	0x151b	AXIS#.GUI.PARAM04	Signed32	Read/Write
5404	0x151c	AXIS#.GUI.PARAM05	Signed32	Read/Write
5405	0x151d	AXIS#.GUI.PARAM06	Signed32	Read/Write
5406	0x151e	AXIS#.GUI.PARAM07	Signed32	Read/Write
5407	0x151f	AXIS#.GUI.PARAM08	Signed32	Read/Write
5408	0x1520	AXIS#.GUI.PARAM09	Signed32	Read/Write
5409	0x1521	AXIS#.GUI.PARAM10	Signed32	Read/Write
5500	0x157c	AXIS#.GEAR.ACC	Unsigned32	Read/Write
5501	0x157d	AXIS#.GEAR.DEC	Unsigned32	Read/Write
5502	0x157e	AXIS#.GEAR.IN	Unsigned16	Read/Write
5503	0x157f	AXIS#.GEAR.MOVE	Unsigned8	Read/Write
5504	0x1580	AXIS#.GEAR.OUT	Signed16	Read/Write
5505	0x1581	AXIS#.GEAR.FBSOURCE	Unsigned8	Read/Write
5506	0x1582	AXIS#.GEAR.STATE	Signed16	Read Only
5507	0x1583	AXIS#.GEAR.AUTOSTART	Unsigned8	Read/Write
5600	0x15e0	AXIS#.HWLS.NEGSTATE	Unsigned8	Read Only
5601	0x15e1	AXIS#.HWLS.POSSTATE	Unsigned8	Read Only
5602	0x15e2	AXIS#.HWLS.NEGSOURCE	Unsigned8	Read/Write
5603	0x15e3	AXIS#.HWLS.POSSOURCE	Unsigned8	Read/Write
5700	0x1644	AXIS#.LOAD.INERTIA	Unsigned32	Read/Write
5800	0x16a8	AXIS#.FBUS.IL.FF	Signed32	Read/Write
5801	0x16a9	AXIS#.IL.CMD	Signed32	Read Only
5802	0x16aa	AXIS#.IL.CMDU	Signed16	Read/Write
5803	0x16ab	AXIS#.IL.FB	Signed32	Read Only
5804	0x16ac	AXIS#.IL.FF	Signed32	Read Only
5805	0x16ad	AXIS#.IL.FOLDFTHRESH	Signed32	Read Only
5806	0x16ae	AXIS#.IL.FOLDFTHRESHU	Signed32	Read/Write
5807	0x16af	AXIS#.IL.FOLDWTHRESH	Signed32	Read/Write
5808	0x16b0	AXIS#.IL.FRICTION	Signed32	Read/Write
5809	0x16b1	AXIS#.IL.IFOLD	Signed32	Read Only

ID	Hex	Name	Data Type	Access
5810	0x16b2	AXIS#.IL.KACCFF	Signed32	Read/Write
5811	0x16b3	AXIS#.IL.AINSOURCE	Unsigned8	Read/Write
5812	0x16b4	AXIS#.IL.KVFF	Signed32	Read/Write
5813	0x16b5	AXIS#.IL.LIMITN	Signed32	Read/Write
5814	0x16b6	AXIS#.IL.LIMITP	Signed32	Read/Write
5815	0x16b7	AXIS#.IL.MIFOLD	Signed32	Read Only
5816	0x16b8	AXIS#.IL.OFFSET	Signed32	Read/Write
5817	0x16b9	AXIS#.IL.VCMD	Signed32	Read Only
5818	0x16ba	AXIS#.IL.BW	Unsigned16	Read/Write
5819	0x16bb	AXIS#.IL.KP	Unsigned32	Read Only
5820	0x16bc	AXIS#.IL.DIFOLD	Signed32	Read Only
5821	0x16bd	AXIS#.IL.FBSOURCE	Unsigned8	Read/Write
5822	0x16be	AXIS#.IL.CMDACC	Signed32	Read Only
5823	0x16bf	AXIS#.IL.PWMFREQ	Unsigned32	Read Only
5824	0x16c0	AXIS#.IL.KPLOOKUP	Unsigned32 [256]	Read Only
5825	0x16c1	AXIS#.IL.DI2T	Signed32	Read Only
5826	0x16c2	AXIS#.IL.AINSCALE	Signed32	Read/Write
5827	0x16c3	AXIS#.IL.MI2T	Signed32	Read Only
5828	0x16c4	AXIS#.IL.PWMQUIET	Unsigned8	Read/Write
5829	0x16c5	AXIS#.FBUS.IL.LIMIT	Signed32	Read/Write
5830	0x16c6	AXIS#.IL.CMDMODE	Unsigned8	Read/Write
5831	0x16c7	AXIS#.IL.IUCMDU	Signed32	Read/Write
5832	0x16c8	AXIS#.IL.IVCMDU	Signed32	Read/Write
5833	0x16c9	AXIS#.IL.SETCMD	Unsigned8	Read/Write
5834	0x16ca	AXIS#.IL.IUCMD	Signed32	Read Only
5835	0x16cb	AXIS#.IL.IVCMD	Signed32	Read Only
5836	0x16cc	AXIS#.IL.IWCMD	Signed32	Read Only
5837	0x16cd	AXIS#.IL.COMPTABLE.DATA.SIZE	Signed32	Read/Write
5838	0x16ce	AXIS#.IL.COMPTABLE.ENABLE	Unsigned8	Read/Write
5839	0x16cf	AXIS#.IL.COMPTABLE.MAX	Signed32	Read/Write
5840	0x16d0	AXIS#.IL.COMPTABLE.MIN	Signed32	Read/Write
5841	0x16d1	AXIS#.IL.COMPTABLE.MODULO	Unsigned8	Read/Write
5842	0x16d2	AXIS#.IL.COMPTABLE.MODULOVALUE	Signed32	Read/Write
5844	0x16d4	AXIS#.IL.COMMANGLE	Unsigned32	Read Only

ID	Hex	Name	Data Type	Access
5845	0x16d5	AXIS#.IL.COMPTABLE.INPUT	Signed32	Read Only
5850	0x16da	AXIS#.IL.COGCOMP.TEACH.ACC	Unsigned32	Read/Write
5851	0x16db	AXIS#.IL.COGCOMP.TEACH.DEC	Unsigned32	Read/Write
5852	0x16dc	AXIS#.IL.COGCOMP.TEACH.V	Signed32	Read/Write
5853	0x16dd	AXIS#.IL.COGCOMP.TEACH.STATUS	Signed8	Read Only
5854	0x16de	AXIS#.IL.COGCOMP.TEACH.POSEND	Signed32	Read Only
5855	0x16df	AXIS#.IL.COGCOMP.TEACH.POSSTART	Signed32	Read Only
5856	0x16e0	AXIS#.IL.COGCOMP.TEACH.MOVE	Unsigned8	Read/Write
5860	0x16e4	AXIS#.IL.COGCOMP.MOVETOSTART.ACC	Unsigned32	Read/Write
5861	0x16e5	AXIS#.IL.COGCOMP.MOVETOSTART.DEC	Unsigned32	Read/Write
5862	0x16e6	AXIS#.IL.COGCOMP.MOVETOSTART.V	Signed32	Read/Write
5863	0x16e7	AXIS#.IL.COGCOMP.MOVETOSTART.MOVE	Unsigned8	Read/Write
5864	0x16e8	AXIS#.IL.COGCOMP.AUTOSIZE	Unsigned8	Read/Write
5900	0x170c	AXIS#.PL.CMD	Signed32	Read Only
5901	0x170d	AXIS#.PL.ERR	Signed32	Read Only
5902	0x170e	AXIS#.PL.ERRFTHRESH	Signed32	Read/Write
5903	0x170f	AXIS#.PL.ERRWTHRESH	Signed32	Read/Write
5904	0x1710	AXIS#.PL.FB	Signed32	Read Only
5905	0x1711	AXIS#.PL.FBSOURCE	Unsigned8	Read/Write
5906	0x1712	AXIS#.PL.INTOUTMAX	Signed32	Read/Write
5907	0x1713	AXIS#.PL.KI	Signed32	Read/Write
5908	0x1714	AXIS#.PL.KP	Signed32	Read/Write
5909	0x1715	AXIS#.PL.MODP1	Signed32	Read/Write
5910	0x1716	AXIS#.PL.MODP2	Signed32	Read/Write
5911	0x1717	AXIS#.PL.MODPDIR	Unsigned8	Read/Write
5912	0x1718	AXIS#.PL.MODPEN	Unsigned8	Read/Write
5913	0x1719	AXIS#.PL.AINSCALE	Signed32	Read/Write
5915	0x171b	AXIS#.PL.AINSOURCE	Unsigned8	Read/Write
5916	0x171c	AXIS#.PL.KITHRESH	Signed32	Read/Write
5917	0x171d	AXIS#.PL.OFFSET	Signed32	Read/Write
5920	0x1720	AXIS#.PL.CMD.OFFSET	Signed32	Read Only
5921	0x1721	AXIS#.PL.CMD.MWOFFSET	Signed32	Read/Write
5922	0x1722	AXIS#.PL.CMD.FBUSOFFSET	Signed32	Read/Write
6000	0x1770	AXIS#.VL.ARPF1	Signed32	Read/Write

ID	Hex	Name	Data Type	Access
6001	0x1771	AXIS#.VL.ARPF2	Signed32	Read/Write
6002	0x1772	AXIS#.VL.ARPF3	Signed32	Read/Write
6003	0x1773	AXIS#.VL.ARPF4	Signed32	Read/Write
6004	0x1774	AXIS#.VL.ARPQ1	Signed32	Read/Write
6005	0x1775	AXIS#.VL.ARPQ2	Signed32	Read/Write
6006	0x1776	AXIS#.VL.ARPQ3	Signed32	Read/Write
6007	0x1777	AXIS#.VL.ARPQ4	Signed32	Read/Write
6008	0x1778	AXIS#.VL.ARTYPE1	Signed8	Read/Write
6009	0x1779	AXIS#.VL.ARTYPE2	Signed8	Read/Write
6010	0x177a	AXIS#.VL.ARTYPE3	Signed8	Read/Write
6011	0x177b	AXIS#.VL.ARTYPE4	Signed8	Read/Write
6012	0x177c	AXIS#.VL.ARZF1	Signed32	Read/Write
6013	0x177d	AXIS#.VL.ARZF2	Signed32	Read/Write
6014	0x177e	AXIS#.VL.ARZF3	Signed32	Read/Write
6015	0x177f	AXIS#.VL.ARZF4	Signed32	Read/Write
6016	0x1780	AXIS#.VL.ARZQ1	Signed32	Read/Write
6017	0x1781	AXIS#.VL.ARZQ2	Signed32	Read/Write
6018	0x1782	AXIS#.VL.ARZQ3	Signed32	Read/Write
6019	0x1783	AXIS#.VL.ARZQ4	Signed32	Read/Write
6020	0x1784	AXIS#.FBUS.VL.FF	Signed32	Read/Write
6021	0x1785	AXIS#.VL.CMD	Signed32	Read Only
6022	0x1786	AXIS#.VL.CMDU	Signed32	Read/Write
6023	0x1787	AXIS#.VL.ERR	Signed32	Read Only
6024	0x1788	AXIS#.VL.FB	Signed32	Read Only
6025	0x1789	AXIS#.VL.FBFILTER	Signed32	Read Only
6026	0x178a	AXIS#.VL.FBSOURCE	Unsigned8	Read/Write
6027	0x178b	AXIS#.VL.FF	Signed32	Read Only
6028	0x178c	AXIS#.VL.KI	Signed32	Read/Write
6029	0x178d	AXIS#.VL.KP	Signed32	Read/Write
6030	0x178e	AXIS#.VL.KVFF	Signed32	Read/Write
6031	0x178f	AXIS#.VL.LIMITN	Signed32	Read/Write
6032	0x1790	AXIS#.VL.LIMITP	Signed32	Read/Write
6033	0x1791	AXIS#.VL.LMJR	Unsigned32	Read/Write
6034	0x1792	AXIS#.VL.THRESH	Signed32	Read/Write

ID	Hex	Name	Data Type	Access
6035	0x1793	AXIS#.VL.FBUNFILTERED	Signed32	Read Only
6036	0x1794	AXIS#.VL.VFTHRESH	Signed32	Read Only
6037	0x1795	AXIS#.VL.AINSCALE	Signed32	Read/Write
6038	0x1796	AXIS#.VL.AINACC	Signed32	Read/Write
6039	0x1797	AXIS#.VL.AINDEC	Signed32	Read/Write
6040	0x1798	AXIS#.VL.AINSOURCE	Unsigned8	Read/Write
6041	0x1799	AXIS#.VL.ACCFILTERED	Signed32	Read Only
6042	0x179a	AXIS#.VL.KIMODE	Unsigned8	Read/Write
6100	0x17d4	AXIS#.HOME.ACC	Unsigned32	Read/Write
6101	0x17d5	AXIS#.HOME.AUTOMOVE	Unsigned8	Read/Write
6102	0x17d6	AXIS#.HOME.DEC	Unsigned32	Read/Write
6103	0x17d7	AXIS#.HOME.DIR	Unsigned16	Read/Write
6104	0x17d8	AXIS#.HOME.DIST	Signed32	Read/Write
6105	0x17d9	AXIS#.HOME.CREEPFACTOR	Unsigned32	Read/Write
6106	0x17da	AXIS#.HOME.IPEAK	Signed32	Read/Write
6107	0x17db	AXIS#.HOME.MODE	Unsigned16	Read/Write
6108	0x17dc	AXIS#.HOME.MOVE	Unsigned8	Read/Write
6109	0x17dd	AXIS#.HOME.P	Signed32	Read/Write
6110	0x17de	AXIS#.HOME.PERRTHRESH	Signed32	Read/Write
6111	0x17df	AXIS#.HOME.SET	Unsigned8	Read/Write
6112	0x17e0	AXIS#.HOME.V	Signed32	Read/Write
6113	0x17e1	AXIS#.HOME.MAXDIST	Signed32	Read/Write
6114	0x17e2	AXIS#.HOME.CLEAR	Unsigned8	Read/Write
6115	0x17e3	AXIS#.HOME.MULTITURNMODE	Unsigned8	Read/Write
6116	0x17e4	AXIS#.HOME.OFFSET	Signed32	Read Only
6117	0x17e5	AXIS#.HOME.OFFSETUSER	Signed32	Read/Write
6118	0x17e6	AXIS#.HOME.SWITCHSOURCE	Unsigned8	Read/Write
6119	0x17e7	AXIS#.HOME.SWITCHSTATE	Unsigned8	Read Only
6200	0x1838	AXIS#.MOTOR.AUTOSET	Unsigned8	Read/Write
6201	0x1839	AXIS#.MOTOR.BRAKE	Unsigned8	Read Only
6202	0x183a	AXIS#.MOTOR.BRAKECONTROL	Unsigned8	Read/Write
6203	0x183b	AXIS#.MOTOR.CTF0	Unsigned32	Read Only
6204	0x183c	AXIS#.MOTOR.ICONT	Unsigned32	Read Only
6205	0x183d	AXIS#.MOTOR.IDDATAVALID	Unsigned8	Read Only

ID	Hex	Name	Data Type	Access
6206	0x183e	AXIS#.MOTOR.INERTIA	Unsigned32	Read Only
6207	0x183f	AXIS#.MOTOR.IPEAK	Unsigned32	Read Only
6208	0x1840	AXIS#.MOTOR.KT	Unsigned32	Read Only
6209	0x1841	AXIS#.MOTOR.LQLL	Unsigned32	Read Only
6210	0x1842	AXIS#.MOTOR.PHASE	Unsigned16	Read Only
6211	0x1843	AXIS#.MOTOR.PITCH	Unsigned32	Read/Write
6212	0x1844	AXIS#.MOTOR.POLES	Unsigned16	Read Only
6213	0x1845	AXIS#.MOTOR.R	Unsigned32	Read Only
6214	0x1846	AXIS#.MOTOR.RTYPE	Unsigned8	Read Only
6215	0x1847	AXIS#.MOTOR.TBRAKEAPP	Unsigned16	Read Only
6216	0x1848	AXIS#.MOTOR.TBRAKERLS	Unsigned16	Read Only
6217	0x1849	AXIS#.MOTOR.TEMP	Unsigned32	Read Only
6218	0x184a	AXIS#.MOTOR.TEMPFAULT	Unsigned32	Read Only
6219	0x184b	AXIS#.MOTOR.TEMPWARN	Unsigned32	Read/Write
6220	0x184c	AXIS#.MOTOR.TYPE	Unsigned8	Read Only
6221	0x184d	AXIS#.MOTOR.VMAX	Unsigned16	Read Only
6222	0x184e	AXIS#.MOTOR.VOLTMAX	Unsigned16	Read Only
6223	0x184f	AXIS#.MOTOR.TBRAKETO	Signed32	Read/Write
6224	0x1850	AXIS#.MOTOR.BRAKEIMM	Unsigned8	Read/Write
6225	0x1851	AXIS#.MOTOR.VOLTMIN	Unsigned16	Read/Write
6226	0x1852	AXIS#.MOTOR.VOLTRATED	Unsigned16	Read/Write
6227	0x1853	AXIS#.MOTOR.VRATED	Signed32	Read/Write
6228	0x1854	AXIS#.MOTOR.IMTR	Unsigned16	Read/Write
6229	0x1855	AXIS#.MOTOR.IMID	Unsigned16	Read/Write
6230	0x1856	AXIS#.MOTOR.LDLL	Unsigned32	Read Only
6231	0x1857	AXIS#.MOTOR.LISAT	Unsigned32	Read Only
6232	0x1858	AXIS#.MOTOR.IDMAX	Unsigned32	Read Only
6233	0x1859	AXIS#.MOTOR.PHSADVK1	Signed32	Read Only
6234	0x185a	AXIS#.MOTOR.PHSADVK2	Signed32	Read Only
6235	0x185b	AXIS#.MOTOR.TEMPC	Signed16	Read Only
6236	0x185c	AXIS#.MOTOR.FIELDWEAKENING	Unsigned8	Read/Write
6237	0x185d	AXIS#.MOTOR.NAME	String	Read Only
6238	0x185e	AXIS#.MOTOR.BRAKEPOWERDELAY	Unsigned16	Read Only
6239	0x185f	AXIS#.MOTOR.BRAKEPOWERLOW	Unsigned16	Read Only

ID	Hex	Name	Data Type	Access
6240	0x1860	AXIS#.MOTOR.BRAKEPOWERSAVING	Unsigned8	Read Only
6241	0x1861	AXIS#.MOTOR.KE	Unsigned32	Read Only
6242	0x1862	AXIS#.MOTOR.SERIALNUM	String	Read Only
6243	0x1863	AXIS#.MOTOR.RSOURCE	Unsigned8	Read Only
6244	0x1864	AXIS#.MOTOR.TEMPSOURCE	Unsigned8	Read/Write
6245	0x1865	AXIS#.MOTOR.DISAUTOSET	Unsigned8[64]	Read/Write
6300	0x189c	AXIS#.MT.ACC	Unsigned32[32]	Read/Write
6301	0x189d	AXIS#.MT.CLEAR	Unsigned8[32]	Read/Write
6302	0x189e	AXIS#.MT.CNTL	Unsigned32[32]	Read/Write
6303	0x189f	AXIS#.MT.CONTINUE	Unsigned8	Read/Write
6304	0x18a0	AXIS#.MT.DEC	Unsigned32[32]	Read/Write
6305	0x18a1	AXIS#.MT.MOVE	Unsigned8[32]	Read/Write
6306	0x18a2	AXIS#.MT.MTNEXT	Signed8[32]	Read/Write
6307	0x18a3	AXIS#.MT.P	Signed32[32]	Read/Write
6308	0x18a4	AXIS#.MT.TNEXT	Unsigned16[32]	Read/Write
6309	0x18a5	AXIS#.MT.V	Signed32[32]	Read/Write
6310	0x18a6	AXIS#.MT.VCMD	Signed32	Read Only
6311	0x18a7	AXIS#.MT.FEEDRATE	Unsigned32	Read/Write
6312	0x18a8	AXIS#.MT.CAP	Unsigned8[32]	Read/Write
6313	0x18a9	AXIS#.MT.CLEARALL	Unsigned8	Read/Write
6314	0x18aa	AXIS#.MT.DISALLOWINTERRUPT	Unsigned8[32]	Read/Write
6315	0x18ab	AXIS#.MT.DISALLOWZEROSTARTVEL	Unsigned8[32]	Read/Write
6316	0x18ac	AXIS#.MT.RUNNINGTASK	Signed8	Read Only
6317	0x18ad	AXIS#.MT.TRANSITION	Unsigned8[32]	Read/Write
6318	0x18ae	AXIS#.MT.TYPE	Unsigned8[32]	Read/Write
6400	0x1900	AXIS#.SM.I1	Signed32	Read/Write
6401	0x1901	AXIS#.SM.I2	Signed32	Read/Write
6402	0x1902	AXIS#.SM.MODE	Unsigned16	Read/Write
6403	0x1903	AXIS#.SM.MOVE	Unsigned8	Read/Write
6404	0x1904	AXIS#.SM.T1	Unsigned16	Read/Write
6405	0x1905	AXIS#.SM.T2	Unsigned16	Read/Write
6406	0x1906	AXIS#.SM.V1	Signed32	Read/Write
6407	0x1907	AXIS#.SM.V2	Signed32	Read/Write
6408	0x1908	AXIS#.SM.ACC	Unsigned32	Read/Write

ID	Hex	Name	Data Type	Access
6409	0x1909	AXIS#.SM.DEC	Unsigned32	Read/Write
6500	0x1964	AXIS#.SWLS.EN	Unsigned8	Read/Write
6501	0x1965	AXIS#.SWLS.LIMIT0	Signed32	Read/Write
6502	0x1966	AXIS#.SWLS.LIMIT1	Signed32	Read/Write
6503	0x1967	AXIS#.SWLS.STATE	Unsigned8	Read Only
6600	0x19c8	AXIS#.UNIT.ACCLINEAR	Unsigned8	Read/Write
6601	0x19c9	AXIS#.UNIT.ACCROTARY	Unsigned8	Read/Write
6602	0x19ca	AXIS#.UNIT.PIN	Unsigned32	Read/Write
6603	0x19cb	AXIS#.UNIT.PLINEAR	Unsigned8	Read/Write
6604	0x19cc	AXIS#.UNIT.POUT	Unsigned32	Read/Write
6605	0x19cd	AXIS#.UNIT.PROTARY	Unsigned8	Read/Write
6606	0x19ce	AXIS#.UNIT.VLINEAR	Unsigned8	Read/Write
6607	0x19cf	AXIS#.UNIT.VROTARY	Unsigned8	Read/Write
6608	0x19d0	AXIS#.UNIT.LABEL	String	Read/Write
6700	0x1a2c	AXIS#.WS.ARM	Unsigned8	Read/Write
6701	0x1a2d	AXIS#.WS.DISTMAX	Signed32	Read/Write
6702	0x1a2e	AXIS#.WS.DISTMIN	Signed32	Read/Write
6703	0x1a2f	AXIS#.WS.IMAX	Signed32	Read/Write
6704	0x1a30	AXIS#.WS.MODE	Unsigned8	Read/Write
6705	0x1a31	AXIS#.WS.NUMLOOPS	Unsigned8	Read/Write
6706	0x1a32	AXIS#.WS.STATE	Unsigned8	Read Only
6707	0x1a33	AXIS#.WS.T	Unsigned8	Read/Write
6708	0x1a34	AXIS#.WS.TDELAY1	Unsigned8	Read/Write
6709	0x1a35	AXIS#.WS.TDELAY2	Unsigned8	Read/Write
6710	0x1a36	AXIS#.WS.TDELAY3	Unsigned16	Read/Write
6711	0x1a37	AXIS#.WS.VTHRESH	Signed32	Read/Write
6712	0x1a38	AXIS#.WS.DISARM	Unsigned8	Read/Write
6713	0x1a39	AXIS#.WS.FREQ	Unsigned32	Read/Write
6714	0x1a3a	AXIS#.WS.TDELAY4	Unsigned16	Read/Write
6715	0x1a3b	AXIS#.WS.CHECKT	Unsigned16	Read/Write
6716	0x1a3c	AXIS#.WS.CHECKV	Signed32	Read/Write
6717	0x1a3d	AXIS#.WS.TSTANDSTILL	Unsigned16	Read/Write
6718	0x1a3e	AXIS#.WS.TIRAMP	Unsigned16	Read/Write
6719	0x1a3f	AXIS#.WS.CHECKMODE	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
6800	0x1a90	AXIS#.FAULT1	Unsigned16	Read Only
6801	0x1a91	AXIS#.FAULT2	Unsigned16	Read Only
6802	0x1a92	AXIS#.FAULT3	Unsigned16	Read Only
6803	0x1a93	AXIS#.FAULT4	Unsigned16	Read Only
6804	0x1a94	AXIS#.FAULT5	Unsigned16	Read Only
6805	0x1a95	AXIS#.FAULT6	Unsigned16	Read Only
6806	0x1a96	AXIS#.FAULT7	Unsigned16	Read Only
6807	0x1a97	AXIS#.FAULT8	Unsigned16	Read Only
6808	0x1a98	AXIS#.FAULT9	Unsigned16	Read Only
6809	0x1a99	AXIS#.FAULT10	Unsigned16	Read Only
6900	0x1af4	AXIS#.JOG.ACC	Unsigned32	Read/Write
6901	0x1af5	AXIS#.JOG.DEC	Unsigned32	Read/Write
6902	0x1af6	AXIS#.JOG.MOVEN	Unsigned8	Read/Write
6903	0x1af7	AXIS#.JOG.MOVEP	Unsigned8	Read/Write
6904	0x1af8	AXIS#.JOG.V	Signed32	Read/Write
7000	0x1b58	AXIS#.HWEN.MODE	Unsigned8	Read/Write
7001	0x1b59	AXIS#.HWEN.SOURCE	Unsigned8	Read/Write
7002	0x1b5a	AXIS#.HWEN.STATE	Unsigned8	Read Only
7100	0x1bbc	AXIS#.SETTLE.P	Signed32	Read/Write
7101	0x1bbd	AXIS#.SETTLE.V	Signed32	Read/Write
7200	0x1c20	AXIS#.FAULT6004.ACTION	Unsigned8	Read/Write
7300	0x1c84	AXIS#.FIELDWEAKENING.LOOPBW	Signed32	Read/Write
7301	0x1c85	AXIS#.FIELDWEAKENING.CURRFILTERBW	Signed32	Read/Write
7302	0x1c86	AXIS#.FIELDWEAKENING.VOLTFILTERBW	Signed32	Read/Write
7303	0x1c87	AXIS#.FIELDWEAKENING.VBUSMARGIN	Signed32	Read/Write
7400	0x1ce8	AXIS#.SENSORLESS.BWU	Signed32	Read/Write
7401	0x1ce9	AXIS#.SENSORLESS.BW	Signed32	Read Only
7402	0x1cea	AXIS#.SENSORLESS.FAULTANGLE	Signed32	Read/Write
7403	0x1ceb	AXIS#.SENSORLESS.FAULTTIME	Signed32	Read/Write
7404	0x1cec	AXIS#.SENSORLESS.RPMSTART	Signed32	Read/Write
7405	0x1ced	AXIS#.SENSORLESS.ISTART	Signed32	Read/Write
7406	0x1cee	AXIS#.SENSORLESS.ENPHASELEAD	Unsigned8	Read/Write
7500	0x1d4c	AXIS#.EIP.POSUNIT	Unsigned32	Read/Write
7501	0x1d4d	AXIS#.EIP.PROFUNIT	Unsigned32	Read/Write

ID	Hex	Name	Data Type	Access
7502	0x1d4e	AXIS#.EIP.CMD.CONTROL1	Unsigned8	Read/Write
7503	0x1d4f	AXIS#.EIP.CMD.CONTROL2	Unsigned8	Read/Write
7504	0x1d50	AXIS#.EIP.RSP.STATUS1	Unsigned8	Read Only
7505	0x1d51	AXIS#.EIP.RSP.STATUS2	Unsigned8	Read Only
7506	0x1d52	AXIS#.EIP.RSP.STATUS3	Unsigned8	Read Only
7507	0x1d53	AXIS#.EIP.OPMODE	Unsigned8	Read/Write
7508	0x1d54	AXIS#.EIP.DYNAMICCMDMAP	Unsigned16[16]	Read/Write
7509	0x1d55	AXIS#.EIP.DYNAMICRSPMAP	Unsigned16[16]	Read/Write
7510	0x1d56	AXIS#.EIP.DYNAMICCMDDATA	Unsigned32[16]	Read Only
7511	0x1d57	AXIS#.EIP.DYNAMICRSPDATA	Unsigned32[16]	Read Only
7512	0x1d58	AXIS#.EIP.CMD.BLOCKNUM	Unsigned8	Read/Write
7513	0x1d59	AXIS#.EIP.CMD.CMDTYPE	Unsigned8	Read/Write
7514	0x1d5a	AXIS#.EIP.CMD.RSPTYPE	Unsigned8	Read/Write
7515	0x1d5b	AXIS#.EIP.CMD.DATA	Unsigned32	Read/Write
7516	0x1d5c	AXIS#.EIP.CMD.P	Signed32	Read/Write
7517	0x1d5d	AXIS#.EIP.CMD.V	Signed32	Read/Write
7518	0x1d5e	AXIS#.EIP.CMD.ACC	Unsigned32	Read/Write
7519	0x1d5f	AXIS#.EIP.CMD.DEC	Unsigned32	Read/Write
7520	0x1d60	AXIS#.EIP.CMD.DATA2	Unsigned32	Read/Write
7521	0x1d61	AXIS#.EIP.CMD.ATTRIBUTE	Unsigned8	Read/Write
7522	0x1d62	AXIS#.EIP.RSP.BLOCKNUM	Unsigned8	Read Only
7523	0x1d63	AXIS#.EIP.RSP.RSPTYPE	Unsigned8	Read Only
7524	0x1d64	AXIS#.EIP.RSP.DATA	Unsigned32	Read Only
7525	0x1d65	AXIS#.EIP.RSP.P	Signed32	Read Only
7526	0x1d66	AXIS#.EIP.RSP.V	Signed32	Read Only
7527	0x1d67	AXIS#.EIP.RSP.MOTIONSTAT	Unsigned32	Read Only
7528	0x1d68	AXIS#.EIP.RSP.DATA2	Unsigned32	Read Only
7529	0x1d69	AXIS#.EIP.RSP.ATTRIBUTE	Unsigned8	Read Only
7530	0x1d6a	AXIS#.EIP.PADBYTE	Unsigned8	Read/Write
7531	0x1d6b	AXIS#.EIP.FIXEDCMDMAP	Unsigned16[32]	Read Only
7532	0x1d6c	AXIS#.EIP.FIXEDRSPMAP	Unsigned16[32]	Read Only
7533	0x1d6d	AXIS#.EIP.FIXEDCMDDATA	Unsigned32[32]	Read Only
7534	0x1d6e	AXIS#.EIP.FIXEDRSPDATA	Unsigned32[32]	Read Only
7535	0x1d6f	AXIS#.EIP.DYNAMICCMDINDEX	Unsigned16[16]	Read/Write

ID	Hex	Name	Data Type	Access
7536	0x1d70	AXIS#.EIP.DYNAMICRSPINDEX	Unsigned16[16]	Read/Write
7600	0x1db0	AXIS#.MW.FBUS1	Signed32	Read/Write
7601	0x1db1	AXIS#.MW.FBUS2	Signed32	Read/Write
7602	0x1db2	AXIS#.MW.FBUS3	Signed32	Read/Write
7603	0x1db3	AXIS#.MW.FBUS4	Signed32	Read/Write
7604	0x1db4	AXIS#.MW.FBUS5	Signed32	Read/Write
7605	0x1db5	AXIS#.MW.FBUS6	Signed32	Read/Write
7606	0x1db6	AXIS#.MW.FBUS7	Signed32	Read/Write
7607	0x1db7	AXIS#.MW.FBUS8	Signed32	Read/Write
7608	0x1db8	AXIS#.MW.FBUS9	Signed32	Read/Write
7609	0x1db9	AXIS#.MW.FBUS10	Signed32	Read/Write
7610	0x1dba	AXIS#.MW.FBUS11	Signed32	Read/Write
7611	0x1dbb	AXIS#.MW.FBUS12	Signed32	Read/Write
7612	0x1dbc	AXIS#.MW.FBUS13	Signed32	Read/Write
7613	0x1dbd	AXIS#.MW.FBUS14	Signed32	Read/Write
7614	0x1dbe	AXIS#.MW.FBUS15	Signed32	Read/Write
7615	0x1dbf	AXIS#.MW.FBUS16	Signed32	Read/Write
7700	0x1e14	AXIS#.MW.FBUS1	Signed32	Read/Write
7701	0x1e15	AXIS#.MW.FBUS2	Signed32	Read/Write
7702	0x1e16	AXIS#.MW.FBUS3	Signed32	Read/Write
7703	0x1e17	AXIS#.MW.FBUS4	Signed32	Read/Write
7704	0x1e18	AXIS#.MW.FBUS5	Signed32	Read/Write
7705	0x1e19	AXIS#.MW.FBUS6	Signed32	Read/Write
7706	0x1e1a	AXIS#.MW.FBUS7	Signed32	Read/Write
7707	0x1e1b	AXIS#.MW.FBUS8	Signed32	Read/Write
7708	0x1e1c	AXIS#.MW.FBUS9	Signed32	Read/Write
7709	0x1e1d	AXIS#.MW.FBUS10	Signed32	Read/Write
7710	0x1e1e	AXIS#.MW.FBUS11	Signed32	Read/Write
7711	0x1e1f	AXIS#.MW.FBUS12	Signed32	Read/Write
7712	0x1e20	AXIS#.MW.FBUS13	Signed32	Read/Write
7713	0x1e21	AXIS#.MW.FBUS14	Signed32	Read/Write
7714	0x1e22	AXIS#.MW.FBUS15	Signed32	Read/Write
7715	0x1e23	AXIS#.MW.FBUS16	Signed32	Read/Write
10000	0x2710	AXIS#.SAFE.STO.A	Unsigned8	Read Only

ID	Hex	Name	Data Type	Access
10001	0x2711	AXIS#.SAFE.STO.B	Unsigned8	Read Only
10002	0x2712	AXIS#.SAFE.STO.ACTIVE	Unsigned8	Read Only
10003	0x2713	AXIS#.SAFE.STO.REPORTFAULT	Unsigned8	Read/Write
20000	0x4e20	DIN1.STATE	Unsigned8	Read Only
20001	0x4e21	DIN2.STATE	Unsigned8	Read Only
20002	0x4e22	DIN3.STATE	Unsigned8	Read Only
20003	0x4e23	DIN4.STATE	Unsigned8	Read Only
20004	0x4e24	DIN5.STATE	Unsigned8	Read Only
20005	0x4e25	DIN6.STATE	Unsigned8	Read Only
20006	0x4e26	DIN7.STATE	Unsigned8	Read Only
20007	0x4e27	DIN8.STATE	Unsigned8	Read Only
20008	0x4e28	DIN9.STATE	Unsigned8	Read Only
20009	0x4e29	DIN10.STATE	Unsigned8	Read Only
20010	0x4e2a	DIN11.STATE	Unsigned8	Read Only
20011	0x4e2b	DIN12.STATE	Unsigned8	Read Only
20050	0x4e52	DIN1.INV	Unsigned8	Read/Write
20051	0x4e53	DIN2.INV	Unsigned8	Read/Write
20052	0x4e54	DIN3.INV	Unsigned8	Read/Write
20053	0x4e55	DIN4.INV	Unsigned8	Read/Write
20054	0x4e56	DIN5.INV	Unsigned8	Read/Write
20055	0x4e57	DIN6.INV	Unsigned8	Read/Write
20056	0x4e58	DIN7.INV	Unsigned8	Read/Write
20057	0x4e59	DIN8.INV	Unsigned8	Read/Write
20058	0x4e5a	DIN9.INV	Unsigned8	Read/Write
20059	0x4e5b	DIN10.INV	Unsigned8	Read/Write
20060	0x4e5c	DIN11.INV	Unsigned8	Read/Write
20061	0x4e5d	DIN12.INV	Unsigned8	Read/Write
20100	0x4e84	DIN1.FILTER	Unsigned8	Read/Write
20101	0x4e85	DIN2.FILTER	Unsigned8	Read/Write
20102	0x4e86	DIN3.FILTER	Unsigned8	Read/Write
20103	0x4e87	DIN4.FILTER	Unsigned8	Read/Write
20104	0x4e88	DIN5.FILTER	Unsigned8	Read/Write
20105	0x4e89	DIN6.FILTER	Unsigned8	Read/Write
20106	0x4e8a	DIN7.FILTER	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
20107	0x4e8b	DIN8.FILTER	Unsigned8	Read/Write
20108	0x4e8c	DIN9.FILTER	Unsigned8	Read/Write
20109	0x4e8d	DIN10.FILTER	Unsigned8	Read/Write
20110	0x4e8e	DIN11.FILTER	Unsigned8	Read/Write
20111	0x4e8f	DIN12.FILTER	Unsigned8	Read/Write
21000	0x5208	DOUT1.STATE	Unsigned8	Read Only
21001	0x5209	DOUT2.STATE	Unsigned8	Read Only
21002	0x520a	DOUT3.STATE	Unsigned8	Read Only
21003	0x520b	DOUT4.STATE	Unsigned8	Read Only
21004	0x520c	DOUT5.STATE	Unsigned8	Read Only
21005	0x520d	DOUT6.STATE	Unsigned8	Read Only
21006	0x520e	DOUT7.STATE	Unsigned8	Read Only
21007	0x520f	DOUT8.STATE	Unsigned8	Read Only
21008	0x5210	DOUT9.STATE	Unsigned8	Read Only
21050	0x523a	DOUT1.STATEU	Unsigned8	Read/Write
21051	0x523b	DOUT2.STATEU	Unsigned8	Read/Write
21052	0x523c	DOUT3.STATEU	Unsigned8	Read/Write
21053	0x523d	DOUT4.STATEU	Unsigned8	Read/Write
21054	0x523e	DOUT5.STATEU	Unsigned8	Read/Write
21055	0x523f	DOUT6.STATEU	Unsigned8	Read/Write
21056	0x5240	DOUT7.STATEU	Unsigned8	Read/Write
21057	0x5241	DOUT8.STATEU	Unsigned8	Read/Write
21058	0x5242	DOUT9.STATEU	Unsigned8	Read/Write
21150	0x529e	DOUT1.SOURCE	Unsigned8	Read/Write
21151	0x529f	DOUT2.SOURCE	Unsigned8	Read/Write
21152	0x52a0	DOUT3.SOURCE	Unsigned8	Read/Write
21153	0x52a1	DOUT4.SOURCE	Unsigned8	Read/Write
21154	0x52a2	DOUT5.SOURCE	Unsigned8	Read/Write
21155	0x52a3	DOUT6.SOURCE	Unsigned8	Read/Write
21156	0x52a4	DOUT7.SOURCE	Unsigned8	Read/Write
21157	0x52a5	DOUT8.SOURCE	Unsigned8	Read/Write
21158	0x52a6	DOUT9.SOURCE	Unsigned8	Read/Write
21200	0x52d0	DOUT1.SOURCEID	Unsigned8	Read/Write
21201	0x52d1	DOUT2.SOURCEID	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
21202	0x52d2	DOUT3.SOURCEID	Unsigned8	Read/Write
21203	0x52d3	DOUT4.SOURCEID	Unsigned8	Read/Write
21204	0x52d4	DOUT5.SOURCEID	Unsigned8	Read/Write
21205	0x52d5	DOUT6.SOURCEID	Unsigned8	Read/Write
21206	0x52d6	DOUT7.SOURCEID	Unsigned8	Read/Write
21207	0x52d7	DOUT8.SOURCEID	Unsigned8	Read/Write
21208	0x52d8	DOUT9.SOURCEID	Unsigned8	Read/Write
22000	0x55f0	DIO1.STATE	Unsigned8	Read Only
22001	0x55f1	DIO2.STATE	Unsigned8	Read Only
22002	0x55f2	DIO3.STATE	Unsigned8	Read Only
22003	0x55f3	DIO4.STATE	Unsigned8	Read Only
22004	0x55f4	DIO5.STATE	Unsigned8	Read Only
22005	0x55f5	DIO6.STATE	Unsigned8	Read Only
22050	0x5622	DIO1.DIR	Unsigned8	Read/Write
22051	0x5623	DIO2.DIR	Unsigned8	Read/Write
22052	0x5624	DIO3.DIR	Unsigned8	Read/Write
22053	0x5625	DIO4.DIR	Unsigned8	Read/Write
22054	0x5626	DIO5.DIR	Unsigned8	Read/Write
22055	0x5627	DIO6.DIR	Unsigned8	Read/Write
22100	0x5654	DIO1.INV	Unsigned8	Read/Write
22101	0x5655	DIO2.INV	Unsigned8	Read/Write
22102	0x5656	DIO3.INV	Unsigned8	Read/Write
22103	0x5657	DIO4.INV	Unsigned8	Read/Write
22104	0x5658	DIO5.INV	Unsigned8	Read/Write
22105	0x5659	DIO6.INV	Unsigned8	Read/Write
22150	0x5686	DIO1.STATEU	Unsigned8	Read/Write
22151	0x5687	DIO2.STATEU	Unsigned8	Read/Write
22152	0x5688	DIO3.STATEU	Unsigned8	Read/Write
22153	0x5689	DIO4.STATEU	Unsigned8	Read/Write
22154	0x568a	DIO5.STATEU	Unsigned8	Read/Write
22155	0x568b	DIO6.STATEU	Unsigned8	Read/Write
22200	0x56b8	DIO1.SOURCE	Unsigned8	Read/Write
22201	0x56b9	DIO2.SOURCE	Unsigned8	Read/Write
22202	0x56ba	DIO3.SOURCE	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
22203	0x56bb	DIO4.SOURCE	Unsigned8	Read/Write
22204	0x56bc	DIO5.SOURCE	Unsigned8	Read/Write
22205	0x56bd	DIO6.SOURCE	Unsigned8	Read/Write
22250	0x56ea	DIO1.FILTER	Unsigned8	Read/Write
22251	0x56eb	DIO2.FILTER	Unsigned8	Read/Write
22252	0x56ec	DIO3.FILTER	Unsigned8	Read/Write
22253	0x56ed	DIO4.FILTER	Unsigned8	Read/Write
22254	0x56ee	DIO5.FILTER	Unsigned8	Read/Write
22255	0x56ef	DIO6.FILTER	Unsigned8	Read/Write
22300	0x571c	DIO1.SOURCEID	Unsigned8	Read/Write
22301	0x571d	DIO2.SOURCEID	Unsigned8	Read/Write
22302	0x571e	DIO3.SOURCEID	Unsigned8	Read/Write
22303	0x571f	DIO4.SOURCEID	Unsigned8	Read/Write
22304	0x5720	DIO5.SOURCEID	Unsigned8	Read/Write
22305	0x5721	DIO6.SOURCEID	Unsigned8	Read/Write
22350	0x574e	DIO1.TERM	Unsigned8	Read/Write
22351	0x574f	DIO2.TERM	Unsigned8	Read/Write
22352	0x5750	DIO3.TERM	Unsigned8	Read/Write
22353	0x5751	DIO4.TERM	Unsigned8	Read/Write
22354	0x5752	DIO5.TERM	Unsigned8	Read/Write
22355	0x5753	DIO6.TERM	Unsigned8	Read/Write
23000	0x59d8	AIN1.CUTOFF	Unsigned32	Read/Write
23001	0x59d9	AIN2.CUTOFF	Unsigned32	Read/Write
23050	0x5a0a	AIN1.OFFSET	Signed16	Read/Write
23051	0x5a0b	AIN2.OFFSET	Signed16	Read/Write
23100	0x5a3c	AIN1.VALUE	Signed16	Read Only
23101	0x5a3d	AIN2.VALUE	Signed16	Read Only
23150	0x5a6e	AIN1.DEADBAND	Signed16	Read/Write
23151	0x5a6f	AIN2.DEADBAND	Signed16	Read/Write
23200	0x5aa0	AIN1.DEADBANDMODE	Unsigned16	Read/Write
23201	0x5aa1	AIN2.DEADBANDMODE	Unsigned16	Read/Write
23250	0x5ad2	AIN1.ZERO	Unsigned8	Read/Write
23251	0x5ad3	AIN2.ZERO	Unsigned8	Read/Write
24000	0x5dc0	AOUT1.VALUE	Signed16	Read Only

ID	Hex	Name	Data Type	Access
24001	0x5dc1	AOUT2.VALUE	Signed16	Read Only
24050	0x5df2	AOUT1.SOURCE	Unsigned8	Read/Write
24051	0x5df3	AOUT2.SOURCE	Unsigned8	Read/Write
24100	0x5e24	AOUT1.CUTOFF	Unsigned32	Read/Write
24101	0x5e25	AOUT2.CUTOFF	Unsigned32	Read/Write
24150	0x5e56	FBUS.AOUT1.VALUE	Signed16	Read/Write
24151	0x5e57	FBUS.AOUT2.VALUE	Signed16	Read/Write
24200	0x5e88	AOUT1.OFFSET	Signed16	Read/Write
24201	0x5e89	AOUT2.OFFSET	Signed16	Read/Write
30000	0x7530	FB1.IDENTIFIED	Signed16	Read Only
30001	0x7531	FB2.IDENTIFIED	Signed16	Read Only
30002	0x7532	FB3.IDENTIFIED	Signed16	Read Only
30003	0x7533	FB4.IDENTIFIED	Signed16	Read Only
30004	0x7534	FB5.IDENTIFIED	Signed16	Read Only
30020	0x7544	FB1.LASTIDENTIFIED	Signed16	Read/Write
30021	0x7545	FB2.LASTIDENTIFIED	Signed16	Read/Write
30022	0x7546	FB3.LASTIDENTIFIED	Signed16	Read/Write
30023	0x7547	FB4.LASTIDENTIFIED	Signed16	Read/Write
30024	0x7548	FB5.LASTIDENTIFIED	Signed16	Read/Write
30040	0x7558	FB1.SELECT	Signed16	Read/Write
30041	0x7559	FB2.SELECT	Signed16	Read/Write
30042	0x755a	FB3.SELECT	Signed16	Read/Write
30043	0x755b	FB4.SELECT	Signed16	Read/Write
30044	0x755c	FB5.SELECT	Signed16	Read/Write
30060	0x756c	FB1.HALLSTATE	Unsigned8	Read Only
30061	0x756d	FB2.HALLSTATE	Unsigned8	Read Only
30062	0x756e	FB3.HALLSTATE	Unsigned8	Read Only
30063	0x756f	FB4.HALLSTATE	Unsigned8	Read Only
30064	0x7570	FB5.HALLSTATE	Unsigned8	Read Only
30080	0x7580	FB1.ENCLINES	Unsigned32	Read/Write
30081	0x7581	FB2.ENCLINES	Unsigned32	Read/Write
30082	0x7582	FB3.ENCLINES	Unsigned32	Read/Write
30083	0x7583	FB4.ENCLINES	Unsigned32	Read/Write
30084	0x7584	FB5.ENCLINES	Unsigned32	Read/Write

ID	Hex	Name	Data Type	Access
30100	0x7594	FB1.POLES	Unsigned16	Read/Write
30101	0x7595	FB2.POLES	Unsigned16	Read/Write
30102	0x7596	FB3.POLES	Unsigned16	Read/Write
30103	0x7597	FB4.POLES	Unsigned16	Read/Write
30104	0x7598	FB5.POLES	Unsigned16	Read/Write
30140	0x75bc	FB1.RESKTR	Unsigned16	Read/Write
30141	0x75bd	FB2.RESKTR	Unsigned16	Read/Write
30142	0x75be	FB3.RESKTR	Unsigned16	Read/Write
30143	0x75bf	FB4.RESKTR	Unsigned16	Read/Write
30144	0x75c0	FB5.RESKTR	Unsigned16	Read/Write
30160	0x75d0	FB1.RESREFPHASE	Signed32	Read/Write
30161	0x75d1	FB2.RESREFPHASE	Signed32	Read/Write
30162	0x75d2	FB3.RESREFPHASE	Signed32	Read/Write
30163	0x75d3	FB4.RESREFPHASE	Signed32	Read/Write
30164	0x75d4	FB5.RESREFPHASE	Signed32	Read/Write
30180	0x75e4	FB1.TRACKINGCAL	Unsigned8	Read/Write
30181	0x75e5	FB2.TRACKINGCAL	Unsigned8	Read/Write
30182	0x75e6	FB3.TRACKINGCAL	Unsigned8	Read/Write
30183	0x75e7	FB4.TRACKINGCAL	Unsigned8	Read/Write
30184	0x75e8	FB5.TRACKINGCAL	Unsigned8	Read/Write
30200	0x75f8	FB1.CALTHRESHSINCOS	Unsigned16	Read/Write
30201	0x75f9	FB2.CALTHRESHSINCOS	Unsigned16	Read/Write
30202	0x75fa	FB3.CALTHRESHSINCOS	Unsigned16	Read/Write
30203	0x75fb	FB4.CALTHRESHSINCOS	Unsigned16	Read/Write
30204	0x75fc	FB5.CALTHRESHSINCOS	Unsigned16	Read/Write
30220	0x760c	FB1.MECHPOS	Unsigned32	Read Only
30221	0x760d	FB2.MECHPOS	Unsigned32	Read Only
30222	0x760e	FB3.MECHPOS	Unsigned32	Read Only
30223	0x760f	FB4.MECHPOS	Unsigned32	Read Only
30224	0x7610	FB5.MECHPOS	Unsigned32	Read Only
30240	0x7620	FB1.P	Signed32	Read Only
30241	0x7621	FB2.P	Signed32	Read Only
30242	0x7622	FB3.P	Signed32	Read Only
30243	0x7623	FB4.P	Signed32	Read Only

ID	Hex	Name	Data Type	Access
30244	0x7624	FB5.P	Signed32	Read Only
30260	0x7634	FB1.SIGNALAMPLITUDE	Signed32	Read Only
30261	0x7635	FB2.SIGNALAMPLITUDE	Signed32	Read Only
30262	0x7636	FB3.SIGNALAMPLITUDE	Signed32	Read Only
30263	0x7637	FB4.SIGNALAMPLITUDE	Signed32	Read Only
30264	0x7638	FB5.SIGNALAMPLITUDE	Signed32	Read Only
30280	0x7648	FB1.SIGNALCOS	Signed32	Read Only
30281	0x7649	FB2.SIGNALCOS	Signed32	Read Only
30282	0x764a	FB3.SIGNALCOS	Signed32	Read Only
30283	0x764b	FB4.SIGNALCOS	Signed32	Read Only
30284	0x764c	FB5.SIGNALCOS	Signed32	Read Only
30300	0x765c	FB1.SIGNALSIN	Signed32	Read Only
30301	0x765d	FB2.SIGNALSIN	Signed32	Read Only
30302	0x765e	FB3.SIGNALSIN	Signed32	Read Only
30303	0x765f	FB4.SIGNALSIN	Signed32	Read Only
30304	0x7660	FB5.SIGNALSIN	Signed32	Read Only
30320	0x7670	FB1.SINGLETURNBITS	Unsigned8	Read/Write
30321	0x7671	FB2.SINGLETURNBITS	Unsigned8	Read/Write
30322	0x7672	FB3.SINGLETURNBITS	Unsigned8	Read/Write
30323	0x7673	FB4.SINGLETURNBITS	Unsigned8	Read/Write
30324	0x7674	FB5.SINGLETURNBITS	Unsigned8	Read/Write
30340	0x7684	FB1.MULTITURNBITS	Unsigned8	Read/Write
30341	0x7685	FB2.MULTITURNBITS	Unsigned8	Read/Write
30342	0x7686	FB3.MULTITURNBITS	Unsigned8	Read/Write
30343	0x7687	FB4.MULTITURNBITS	Unsigned8	Read/Write
30344	0x7688	FB5.MULTITURNBITS	Unsigned8	Read/Write
30360	0x7698	FB1.MECHTYPE	Unsigned8	Read/Write
30361	0x7699	FB2.MECHTYPE	Unsigned8	Read/Write
30362	0x769a	FB3.MECHTYPE	Unsigned8	Read/Write
30363	0x769b	FB4.MECHTYPE	Unsigned8	Read/Write
30364	0x769c	FB5.MECHTYPE	Unsigned8	Read/Write
30380	0x76ac	FB1.LINEPITCH	Unsigned32	Read/Write
30381	0x76ad	FB2.LINEPITCH	Unsigned32	Read/Write
30382	0x76ae	FB3.LINEPITCH	Unsigned32	Read/Write

ID	Hex	Name	Data Type	Access
30383	0x76af	FB4.LINEPITCH	Unsigned32	Read/Write
30384	0x76b0	FB5.LINEPITCH	Unsigned32	Read/Write
30400	0x76c0	FB1.BITS	Unsigned16	Read/Write
30401	0x76c1	FB2.BITS	Unsigned16	Read/Write
30402	0x76c2	FB3.BITS	Unsigned16	Read/Write
30403	0x76c3	FB4.BITS	Unsigned16	Read/Write
30404	0x76c4	FB5.BITS	Unsigned16	Read/Write
30420	0x76d4	FB1.FAULTS	Unsigned16[5]	Read Only
30421	0x76d5	FB2.FAULTS	Unsigned16[5]	Read Only
30422	0x76d6	FB3.FAULTS	Unsigned16[5]	Read Only
30423	0x76d7	FB4.FAULTS	Unsigned16[5]	Read Only
30424	0x76d8	FB5.FAULTS	Unsigned16[5]	Read Only
30440	0x76e8	FB1.CALTHRESHRES	Unsigned16	Read/Write
30441	0x76e9	FB2.CALTHRESHRES	Unsigned16	Read/Write
30442	0x76ea	FB3.CALTHRESHRES	Unsigned16	Read/Write
30443	0x76eb	FB4.CALTHRESHRES	Unsigned16	Read/Write
30444	0x76ec	FB5.CALTHRESHRES	Unsigned16	Read/Write
30460	0x76fc	FB1.STOREMULTITURN.ENABLE	Unsigned8	Read/Write
30461	0x76fd	FB2.STOREMULTITURN.ENABLE	Unsigned8	Read/Write
30462	0x76fe	FB3.STOREMULTITURN.ENABLE	Unsigned8	Read/Write
30463	0x76ff	FB4.STOREMULTITURN.ENABLE	Unsigned8	Read/Write
30464	0x7700	FB5.STOREMULTITURN.ENABLE	Unsigned8	Read/Write
30480	0x7710	FB1.STOREMULTITURN.BITS	Unsigned8	Read/Write
30481	0x7711	FB2.STOREMULTITURN.BITS	Unsigned8	Read/Write
30482	0x7712	FB3.STOREMULTITURN.BITS	Unsigned8	Read/Write
30483	0x7713	FB4.STOREMULTITURN.BITS	Unsigned8	Read/Write
30484	0x7714	FB5.STOREMULTITURN.BITS	Unsigned8	Read/Write
30520	0x7738	FB1.EIP.POSUNIT	Unsigned32	Read/Write
30521	0x7739	FB2.EIP.POSUNIT	Unsigned32	Read/Write
30522	0x773a	FB3.EIP.POSUNIT	Unsigned32	Read/Write
30523	0x773b	FB4.EIP.POSUNIT	Unsigned32	Read/Write
30524	0x773c	FB5.EIP.POSUNIT	Unsigned32	Read/Write
30540	0x774c	FB1.INITSIGNED	Unsigned8	Read/Write
30541	0x774d	FB2.INITSIGNED	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
30542	0x774e	FB3.INITSIGNED	Unsigned8	Read/Write
30543	0x774f	FB4.INITSIGNED	Unsigned8	Read/Write
30544	0x7750	FB5.INITSIGNED	Unsigned8	Read/Write
30560	0x7760	FB1.SSITYPE	Unsigned8	Read/Write
30561	0x7761	FB2.SSITYPE	Unsigned8	Read/Write
30562	0x7762	FB3.SSITYPE	Unsigned8	Read/Write
30563	0x7763	FB4.SSITYPE	Unsigned8	Read/Write
30564	0x7764	FB5.SSITYPE	Unsigned8	Read/Write
30660	0x77c4	FB1.FBUSDIR	Unsigned8	Read/Write
30661	0x77c5	FB2.FBUSDIR	Unsigned8	Read/Write
30662	0x77c6	FB3.FBUSDIR	Unsigned8	Read/Write
30663	0x77c7	FB4.FBUSDIR	Unsigned8	Read/Write
30664	0x77c8	FB5.FBUSDIR	Unsigned8	Read/Write
30680	0x77d8	FB1.FBUSP	Signed32	Read Only
30681	0x77d9	FB2.FBUSP	Signed32	Read Only
30682	0x77da	FB3.FBUSP	Signed32	Read Only
30683	0x77db	FB4.FBUSP	Signed32	Read Only
30684	0x77dc	FB5.FBUSP	Signed32	Read Only
30700	0x77ec	FB1.TEMPC	Signed32	Read Only
30701	0x77ed	FB2.TEMPC	Signed32	Read Only
30702	0x77ee	FB3.TEMPC	Signed32	Read Only
30703	0x77ef	FB4.TEMPC	Signed32	Read Only
30704	0x77f0	FB5.TEMPC	Signed32	Read Only
30720	0x7800	FB1.TEMPCEN	Unsigned8	Read/Write
30721	0x7801	FB2.TEMPCEN	Unsigned8	Read/Write
30722	0x7802	FB3.TEMPCEN	Unsigned8	Read/Write
30723	0x7803	FB4.TEMPCEN	Unsigned8	Read/Write
30724	0x7804	FB5.TEMPCEN	Unsigned8	Read/Write
31000	0x7918	CAP1.ARM	Unsigned8	Read/Write
31001	0x7919	CAP2.ARM	Unsigned8	Read/Write
31010	0x7922	CAP1.COUNT	Unsigned16	Read Only
31011	0x7923	CAP2.COUNT	Unsigned16	Read Only
31020	0x792c	CAP1.EDGE	Unsigned8	Read/Write
31021	0x792d	CAP2.EDGE	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
31030	0x7936	CAP1.P	Signed32	Read Only
31031	0x7937	CAP2.P	Signed32	Read Only
31040	0x7940	CAP1.PREEDGE	Unsigned8	Read/Write
31041	0x7941	CAP2.PREEDGE	Unsigned8	Read/Write
31050	0x794a	CAP1.PREMODE	Unsigned8	Read/Write
31051	0x794b	CAP2.PREMODE	Unsigned8	Read/Write
31060	0x7954	CAP1.PRESELECT	Unsigned8	Read/Write
31061	0x7955	CAP2.PRESELECT	Unsigned8	Read/Write
31070	0x795e	CAP1.REARM	Unsigned8	Read/Write
31071	0x795f	CAP2.REARM	Unsigned8	Read/Write
31080	0x7968	CAP1.SOURCE	Unsigned8	Read/Write
31081	0x7969	CAP2.SOURCE	Unsigned8	Read/Write
31090	0x7972	CAP1.STATE	Unsigned8	Read Only
31091	0x7973	CAP2.STATE	Unsigned8	Read Only
31110	0x7986	CAP1.TRIGGER	Unsigned8	Read/Write
31111	0x7987	CAP2.TRIGGER	Unsigned8	Read/Write
31300	0x7a44	CMP1.ARM	Unsigned8[8]	Read/Write
31301	0x7a45	CMP2.ARM	Unsigned8[8]	Read/Write
31320	0x7a58	CMP1.DIR	Unsigned8[8]	Read/Write
31321	0x7a59	CMP2.DIR	Unsigned8[8]	Read/Write
31340	0x7a6c	CMP1.REARM	Unsigned8[8]	Read/Write
31341	0x7a6d	CMP2.REARM	Unsigned8[8]	Read/Write
31360	0x7a80	CMP1.SOURCE	Unsigned8	Read/Write
31361	0x7a81	CMP2.SOURCE	Unsigned8	Read/Write
31380	0x7a94	CMP1.STARTVAL	Signed32[8]	Read/Write
31381	0x7a95	CMP2.STARTVAL	Signed32[8]	Read/Write
31400	0x7aa8	CMP1.STATE	Unsigned8[8]	Read Only
31401	0x7aa9	CMP2.STATE	Unsigned8[8]	Read Only
31420	0x7abc	CMP1.STATES	Unsigned16	Read Only
31421	0x7abd	CMP2.STATES	Unsigned16	Read Only
31440	0x7ad0	CMP1.VAL	Signed32	Read Only
31441	0x7ad1	CMP2.VAL	Signed32	Read Only
31460	0x7ae4	CMP1.WIDTHT	Unsigned32[8]	Read/Write
31461	0x7ae5	CMP2.WIDTHT	Unsigned32[8]	Read/Write

ID	Hex	Name	Data Type	Access
31480	0x7af8	CMP1.WIDTHTYPE	Unsigned8[8]	Read/Write
31481	0x7af9	CMP2.WIDTHTYPE	Unsigned8[8]	Read/Write
31500	0x7b0c	CMP1.WIDTHVAL	Signed32[8]	Read/Write
31501	0x7b0d	CMP2.WIDTHVAL	Signed32[8]	Read/Write
31520	0x7b20	CMP1.MODEN	Unsigned8	Read/Write
31521	0x7b21	CMP2.MODEN	Unsigned8	Read/Write
31540	0x7b34	CMP1.MODVAL1	Signed32	Read/Write
31541	0x7b35	CMP2.MODVAL1	Signed32	Read/Write
31560	0x7b48	CMP1.MODVAL2	Signed32	Read/Write
31561	0x7b49	CMP2.MODVAL2	Signed32	Read/Write
31580	0x7b5c	CMP1.ADVANCET	Unsigned32	Read/Write
31581	0x7b5d	CMP2.ADVANCET	Unsigned32	Read/Write
31600	0x7b70	CMP1.SOURCEVAL	Signed32	Read Only
31601	0x7b71	CMP2.SOURCEVAL	Signed32	Read Only
32000	0x7d00	EEO1.DIR	Unsigned8	Read/Write
32001	0x7d01	EEO2.DIR	Unsigned8	Read/Write
32005	0x7d05	EEO1.LINES	Unsigned32	Read/Write
32006	0x7d06	EEO2.LINES	Unsigned32	Read/Write
32010	0x7d0a	EEO1.MODE	Unsigned8	Read/Write
32011	0x7d0b	EEO2.MODE	Unsigned8	Read/Write
32015	0x7d0f	EEO1.PULSEWIDTH	Signed32	Read/Write
32016	0x7d10	EEO2.PULSEWIDTH	Signed32	Read/Write
32020	0x7d14	EEO1.SOURCE	Unsigned8	Read/Write
32021	0x7d15	EEO2.SOURCE	Unsigned8	Read/Write
32025	0x7d19	EEO1.ZMODE	Unsigned8	Read/Write
32030	0x7d1e	EEO1.ZOFFSET	Unsigned32	Read/Write
32040	0x7d28	EEO1.P	Signed32	Read Only
32041	0x7d29	EEO2.P	Signed32	Read Only
32045	0x7d2d	EEO1.PIN	Unsigned32	Read/Write
32046	0x7d2e	EEO2.PIN	Unsigned32	Read/Write
32050	0x7d32	EEO1.POUT	Unsigned32	Read/Write
32051	0x7d33	EEO2.POUT	Unsigned32	Read/Write
40000	0x9c40	ACTION.RUNNING	Unsigned32	Read Only
40100	0x9ca4	ACTION1.ACTIVE	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
40101	0x9ca5	ACTION2.ACTIVE	Unsigned8	Read/Write
40102	0x9ca6	ACTION3.ACTIVE	Unsigned8	Read/Write
40103	0x9ca7	ACTION4.ACTIVE	Unsigned8	Read/Write
40104	0x9ca8	ACTION5.ACTIVE	Unsigned8	Read/Write
40105	0x9ca9	ACTION6.ACTIVE	Unsigned8	Read/Write
40106	0x9caa	ACTION7.ACTIVE	Unsigned8	Read/Write
40107	0x9cab	ACTION8.ACTIVE	Unsigned8	Read/Write
40108	0x9cac	ACTION9.ACTIVE	Unsigned8	Read/Write
40109	0x9cad	ACTION10.ACTIVE	Unsigned8	Read/Write
40110	0x9cae	ACTION11.ACTIVE	Unsigned8	Read/Write
40111	0x9caf	ACTION12.ACTIVE	Unsigned8	Read/Write
40112	0x9cb0	ACTION13.ACTIVE	Unsigned8	Read/Write
40113	0x9cb1	ACTION14.ACTIVE	Unsigned8	Read/Write
40114	0x9cb2	ACTION15.ACTIVE	Unsigned8	Read/Write
40115	0x9cb3	ACTION16.ACTIVE	Unsigned8	Read/Write
40116	0x9cb4	ACTION17.ACTIVE	Unsigned8	Read/Write
40117	0x9cb5	ACTION18.ACTIVE	Unsigned8	Read/Write
40118	0x9cb6	ACTION19.ACTIVE	Unsigned8	Read/Write
40119	0x9cb7	ACTION20.ACTIVE	Unsigned8	Read/Write
40120	0x9cb8	ACTION21.ACTIVE	Unsigned8	Read/Write
40121	0x9cb9	ACTION22.ACTIVE	Unsigned8	Read/Write
40122	0x9cba	ACTION23.ACTIVE	Unsigned8	Read/Write
40123	0x9cbb	ACTION24.ACTIVE	Unsigned8	Read/Write
40124	0x9cbc	ACTION25.ACTIVE	Unsigned8	Read/Write
40125	0x9cbd	ACTION26.ACTIVE	Unsigned8	Read/Write
40126	0x9cbe	ACTION27.ACTIVE	Unsigned8	Read/Write
40127	0x9cbf	ACTION28.ACTIVE	Unsigned8	Read/Write
40128	0x9cc0	ACTION29.ACTIVE	Unsigned8	Read/Write
40129	0x9cc1	ACTION30.ACTIVE	Unsigned8	Read/Write
40130	0x9cc2	ACTION31.ACTIVE	Unsigned8	Read/Write
40131	0x9cc3	ACTION32.ACTIVE	Unsigned8	Read/Write
40200	0x9d08	ACTION1.CONDITION	Unsigned8	Read/Write
40201	0x9d09	ACTION2.CONDITION	Unsigned8	Read/Write
40202	0x9d0a	ACTION3.CONDITION	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
40203	0x9d0b	ACTION4.CONDITION	Unsigned8	Read/Write
40204	0x9d0c	ACTION5.CONDITION	Unsigned8	Read/Write
40205	0x9d0d	ACTION6.CONDITION	Unsigned8	Read/Write
40206	0x9d0e	ACTION7.CONDITION	Unsigned8	Read/Write
40207	0x9d0f	ACTION8.CONDITION	Unsigned8	Read/Write
40208	0x9d10	ACTION9.CONDITION	Unsigned8	Read/Write
40209	0x9d11	ACTION10.CONDITION	Unsigned8	Read/Write
40210	0x9d12	ACTION11.CONDITION	Unsigned8	Read/Write
40211	0x9d13	ACTION12.CONDITION	Unsigned8	Read/Write
40212	0x9d14	ACTION13.CONDITION	Unsigned8	Read/Write
40213	0x9d15	ACTION14.CONDITION	Unsigned8	Read/Write
40214	0x9d16	ACTION15.CONDITION	Unsigned8	Read/Write
40215	0x9d17	ACTION16.CONDITION	Unsigned8	Read/Write
40216	0x9d18	ACTION17.CONDITION	Unsigned8	Read/Write
40217	0x9d19	ACTION18.CONDITION	Unsigned8	Read/Write
40218	0x9d1a	ACTION19.CONDITION	Unsigned8	Read/Write
40219	0x9d1b	ACTION20.CONDITION	Unsigned8	Read/Write
40220	0x9d1c	ACTION21.CONDITION	Unsigned8	Read/Write
40221	0x9d1d	ACTION22.CONDITION	Unsigned8	Read/Write
40222	0x9d1e	ACTION23.CONDITION	Unsigned8	Read/Write
40223	0x9d1f	ACTION24.CONDITION	Unsigned8	Read/Write
40224	0x9d20	ACTION25.CONDITION	Unsigned8	Read/Write
40225	0x9d21	ACTION26.CONDITION	Unsigned8	Read/Write
40226	0x9d22	ACTION27.CONDITION	Unsigned8	Read/Write
40227	0x9d23	ACTION28.CONDITION	Unsigned8	Read/Write
40228	0x9d24	ACTION29.CONDITION	Unsigned8	Read/Write
40229	0x9d25	ACTION30.CONDITION	Unsigned8	Read/Write
40230	0x9d26	ACTION31.CONDITION	Unsigned8	Read/Write
40231	0x9d27	ACTION32.CONDITION	Unsigned8	Read/Write
40300	0x9d6c	ACTION1.CONDITIONVALUE	Signed32	Read/Write
40301	0x9d6d	ACTION2.CONDITIONVALUE	Signed32	Read/Write
40302	0x9d6e	ACTION3.CONDITIONVALUE	Signed32	Read/Write
40303	0x9d6f	ACTION4.CONDITIONVALUE	Signed32	Read/Write
40304	0x9d70	ACTION5.CONDITIONVALUE	Signed32	Read/Write

ID	Hex	Name	Data Type	Access
40305	0x9d71	ACTION6.CONDITIONVALUE	Signed32	Read/Write
40306	0x9d72	ACTION7.CONDITIONVALUE	Signed32	Read/Write
40307	0x9d73	ACTION8.CONDITIONVALUE	Signed32	Read/Write
40308	0x9d74	ACTION9.CONDITIONVALUE	Signed32	Read/Write
40309	0x9d75	ACTION10.CONDITIONVALUE	Signed32	Read/Write
40310	0x9d76	ACTION11.CONDITIONVALUE	Signed32	Read/Write
40311	0x9d77	ACTION12.CONDITIONVALUE	Signed32	Read/Write
40312	0x9d78	ACTION13.CONDITIONVALUE	Signed32	Read/Write
40313	0x9d79	ACTION14.CONDITIONVALUE	Signed32	Read/Write
40314	0x9d7a	ACTION15.CONDITIONVALUE	Signed32	Read/Write
40315	0x9d7b	ACTION16.CONDITIONVALUE	Signed32	Read/Write
40316	0x9d7c	ACTION17.CONDITIONVALUE	Signed32	Read/Write
40317	0x9d7d	ACTION18.CONDITIONVALUE	Signed32	Read/Write
40318	0x9d7e	ACTION19.CONDITIONVALUE	Signed32	Read/Write
40319	0x9d7f	ACTION20.CONDITIONVALUE	Signed32	Read/Write
40320	0x9d80	ACTION21.CONDITIONVALUE	Signed32	Read/Write
40321	0x9d81	ACTION22.CONDITIONVALUE	Signed32	Read/Write
40322	0x9d82	ACTION23.CONDITIONVALUE	Signed32	Read/Write
40323	0x9d83	ACTION24.CONDITIONVALUE	Signed32	Read/Write
40324	0x9d84	ACTION25.CONDITIONVALUE	Signed32	Read/Write
40325	0x9d85	ACTION26.CONDITIONVALUE	Signed32	Read/Write
40326	0x9d86	ACTION27.CONDITIONVALUE	Signed32	Read/Write
40327	0x9d87	ACTION28.CONDITIONVALUE	Signed32	Read/Write
40328	0x9d88	ACTION29.CONDITIONVALUE	Signed32	Read/Write
40329	0x9d89	ACTION30.CONDITIONVALUE	Signed32	Read/Write
40330	0x9d8a	ACTION31.CONDITIONVALUE	Signed32	Read/Write
40331	0x9d8b	ACTION32.CONDITIONVALUE	Signed32	Read/Write
40400	0x9dd0	ACTION1.RUNCOUNT	Unsigned32	Read Only
40401	0x9dd1	ACTION2.RUNCOUNT	Unsigned32	Read Only
40402	0x9dd2	ACTION3.RUNCOUNT	Unsigned32	Read Only
40403	0x9dd3	ACTION4.RUNCOUNT	Unsigned32	Read Only
40404	0x9dd4	ACTION5.RUNCOUNT	Unsigned32	Read Only
40405	0x9dd5	ACTION6.RUNCOUNT	Unsigned32	Read Only
40406	0x9dd6	ACTION7.RUNCOUNT	Unsigned32	Read Only

ID	Hex	Name	Data Type	Access
40407	0x9dd7	ACTION8.RUNCOUNT	Unsigned32	Read Only
40408	0x9dd8	ACTION9.RUNCOUNT	Unsigned32	Read Only
40409	0x9dd9	ACTION10.RUNCOUNT	Unsigned32	Read Only
40410	0x9dda	ACTION11.RUNCOUNT	Unsigned32	Read Only
40411	0x9ddb	ACTION12.RUNCOUNT	Unsigned32	Read Only
40412	0x9ddc	ACTION13.RUNCOUNT	Unsigned32	Read Only
40413	0x9ddd	ACTION14.RUNCOUNT	Unsigned32	Read Only
40414	0x9dde	ACTION15.RUNCOUNT	Unsigned32	Read Only
40415	0x9ddf	ACTION16.RUNCOUNT	Unsigned32	Read Only
40416	0x9de0	ACTION17.RUNCOUNT	Unsigned32	Read Only
40417	0x9de1	ACTION18.RUNCOUNT	Unsigned32	Read Only
40418	0x9de2	ACTION19.RUNCOUNT	Unsigned32	Read Only
40419	0x9de3	ACTION20.RUNCOUNT	Unsigned32	Read Only
40420	0x9de4	ACTION21.RUNCOUNT	Unsigned32	Read Only
40421	0x9de5	ACTION22.RUNCOUNT	Unsigned32	Read Only
40422	0x9de6	ACTION23.RUNCOUNT	Unsigned32	Read Only
40423	0x9de7	ACTION24.RUNCOUNT	Unsigned32	Read Only
40424	0x9de8	ACTION25.RUNCOUNT	Unsigned32	Read Only
40425	0x9de9	ACTION26.RUNCOUNT	Unsigned32	Read Only
40426	0x9dea	ACTION27.RUNCOUNT	Unsigned32	Read Only
40427	0x9deb	ACTION28.RUNCOUNT	Unsigned32	Read Only
40428	0x9dec	ACTION29.RUNCOUNT	Unsigned32	Read Only
40429	0x9ded	ACTION30.RUNCOUNT	Unsigned32	Read Only
40430	0x9dee	ACTION31.RUNCOUNT	Unsigned32	Read Only
40431	0x9def	ACTION32.RUNCOUNT	Unsigned32	Read Only
40500	0x9e34	ACTION1.SOURCE	Unsigned8	Read/Write
40501	0x9e35	ACTION2.SOURCE	Unsigned8	Read/Write
40502	0x9e36	ACTION3.SOURCE	Unsigned8	Read/Write
40503	0x9e37	ACTION4.SOURCE	Unsigned8	Read/Write
40504	0x9e38	ACTION5.SOURCE	Unsigned8	Read/Write
40505	0x9e39	ACTION6.SOURCE	Unsigned8	Read/Write
40506	0x9e3a	ACTION7.SOURCE	Unsigned8	Read/Write
40507	0x9e3b	ACTION8.SOURCE	Unsigned8	Read/Write
40508	0x9e3c	ACTION9.SOURCE	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
40509	0x9e3d	ACTION10.SOURCE	Unsigned8	Read/Write
40510	0x9e3e	ACTION11.SOURCE	Unsigned8	Read/Write
40511	0x9e3f	ACTION12.SOURCE	Unsigned8	Read/Write
40512	0x9e40	ACTION13.SOURCE	Unsigned8	Read/Write
40513	0x9e41	ACTION14.SOURCE	Unsigned8	Read/Write
40514	0x9e42	ACTION15.SOURCE	Unsigned8	Read/Write
40515	0x9e43	ACTION16.SOURCE	Unsigned8	Read/Write
40516	0x9e44	ACTION17.SOURCE	Unsigned8	Read/Write
40517	0x9e45	ACTION18.SOURCE	Unsigned8	Read/Write
40518	0x9e46	ACTION19.SOURCE	Unsigned8	Read/Write
40519	0x9e47	ACTION20.SOURCE	Unsigned8	Read/Write
40520	0x9e48	ACTION21.SOURCE	Unsigned8	Read/Write
40521	0x9e49	ACTION22.SOURCE	Unsigned8	Read/Write
40522	0x9e4a	ACTION23.SOURCE	Unsigned8	Read/Write
40523	0x9e4b	ACTION24.SOURCE	Unsigned8	Read/Write
40524	0x9e4c	ACTION25.SOURCE	Unsigned8	Read/Write
40525	0x9e4d	ACTION26.SOURCE	Unsigned8	Read/Write
40526	0x9e4e	ACTION27.SOURCE	Unsigned8	Read/Write
40527	0x9e4f	ACTION28.SOURCE	Unsigned8	Read/Write
40528	0x9e50	ACTION29.SOURCE	Unsigned8	Read/Write
40529	0x9e51	ACTION30.SOURCE	Unsigned8	Read/Write
40530	0x9e52	ACTION31.SOURCE	Unsigned8	Read/Write
40531	0x9e53	ACTION32.SOURCE	Unsigned8	Read/Write
40600	0x9e98	ACTION1.SOURCEID	Unsigned8	Read/Write
40601	0x9e99	ACTION2.SOURCEID	Unsigned8	Read/Write
40602	0x9e9a	ACTION3.SOURCEID	Unsigned8	Read/Write
40603	0x9e9b	ACTION4.SOURCEID	Unsigned8	Read/Write
40604	0x9e9c	ACTION5.SOURCEID	Unsigned8	Read/Write
40605	0x9e9d	ACTION6.SOURCEID	Unsigned8	Read/Write
40606	0x9e9e	ACTION7.SOURCEID	Unsigned8	Read/Write
40607	0x9e9f	ACTION8.SOURCEID	Unsigned8	Read/Write
40608	0x9ea0	ACTION9.SOURCEID	Unsigned8	Read/Write
40609	0x9ea1	ACTION10.SOURCEID	Unsigned8	Read/Write
40610	0x9ea2	ACTION11.SOURCEID	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
40611	0x9ea3	ACTION12.SOURCEID	Unsigned8	Read/Write
40612	0x9ea4	ACTION13.SOURCEID	Unsigned8	Read/Write
40613	0x9ea5	ACTION14.SOURCEID	Unsigned8	Read/Write
40614	0x9ea6	ACTION15.SOURCEID	Unsigned8	Read/Write
40615	0x9ea7	ACTION16.SOURCEID	Unsigned8	Read/Write
40616	0x9ea8	ACTION17.SOURCEID	Unsigned8	Read/Write
40617	0x9ea9	ACTION18.SOURCEID	Unsigned8	Read/Write
40618	0x9eaa	ACTION19.SOURCEID	Unsigned8	Read/Write
40619	0x9eab	ACTION20.SOURCEID	Unsigned8	Read/Write
40620	0x9eac	ACTION21.SOURCEID	Unsigned8	Read/Write
40621	0x9ead	ACTION22.SOURCEID	Unsigned8	Read/Write
40622	0x9eae	ACTION23.SOURCEID	Unsigned8	Read/Write
40623	0x9eaf	ACTION24.SOURCEID	Unsigned8	Read/Write
40624	0x9eb0	ACTION25.SOURCEID	Unsigned8	Read/Write
40625	0x9eb1	ACTION26.SOURCEID	Unsigned8	Read/Write
40626	0x9eb2	ACTION27.SOURCEID	Unsigned8	Read/Write
40627	0x9eb3	ACTION28.SOURCEID	Unsigned8	Read/Write
40628	0x9eb4	ACTION29.SOURCEID	Unsigned8	Read/Write
40629	0x9eb5	ACTION30.SOURCEID	Unsigned8	Read/Write
40630	0x9eb6	ACTION31.SOURCEID	Unsigned8	Read/Write
40631	0x9eb7	ACTION32.SOURCEID	Unsigned8	Read/Write
40700	0x9efc	ACTION1.SOURCEPARAM	Signed32	Read/Write
40701	0x9efd	ACTION2.SOURCEPARAM	Signed32	Read/Write
40702	0x9efe	ACTION3.SOURCEPARAM	Signed32	Read/Write
40703	0x9eff	ACTION4.SOURCEPARAM	Signed32	Read/Write
40704	0x9f00	ACTION5.SOURCEPARAM	Signed32	Read/Write
40705	0x9f01	ACTION6.SOURCEPARAM	Signed32	Read/Write
40706	0x9f02	ACTION7.SOURCEPARAM	Signed32	Read/Write
40707	0x9f03	ACTION8.SOURCEPARAM	Signed32	Read/Write
40708	0x9f04	ACTION9.SOURCEPARAM	Signed32	Read/Write
40709	0x9f05	ACTION10.SOURCEPARAM	Signed32	Read/Write
40710	0x9f06	ACTION11.SOURCEPARAM	Signed32	Read/Write
40711	0x9f07	ACTION12.SOURCEPARAM	Signed32	Read/Write
40712	0x9f08	ACTION13.SOURCEPARAM	Signed32	Read/Write

ID	Hex	Name	Data Type	Access
40713	0x9f09	ACTION14.SOURCEPARAM	Signed32	Read/Write
40714	0x9f0a	ACTION15.SOURCEPARAM	Signed32	Read/Write
40715	0x9f0b	ACTION16.SOURCEPARAM	Signed32	Read/Write
40716	0x9f0c	ACTION17.SOURCEPARAM	Signed32	Read/Write
40717	0x9f0d	ACTION18.SOURCEPARAM	Signed32	Read/Write
40718	0x9f0e	ACTION19.SOURCEPARAM	Signed32	Read/Write
40719	0x9f0f	ACTION20.SOURCEPARAM	Signed32	Read/Write
40720	0x9f10	ACTION21.SOURCEPARAM	Signed32	Read/Write
40721	0x9f11	ACTION22.SOURCEPARAM	Signed32	Read/Write
40722	0x9f12	ACTION23.SOURCEPARAM	Signed32	Read/Write
40723	0x9f13	ACTION24.SOURCEPARAM	Signed32	Read/Write
40724	0x9f14	ACTION25.SOURCEPARAM	Signed32	Read/Write
40725	0x9f15	ACTION26.SOURCEPARAM	Signed32	Read/Write
40726	0x9f16	ACTION27.SOURCEPARAM	Signed32	Read/Write
40727	0x9f17	ACTION28.SOURCEPARAM	Signed32	Read/Write
40728	0x9f18	ACTION29.SOURCEPARAM	Signed32	Read/Write
40729	0x9f19	ACTION30.SOURCEPARAM	Signed32	Read/Write
40730	0x9f1a	ACTION31.SOURCEPARAM	Signed32	Read/Write
40731	0x9f1b	ACTION32.SOURCEPARAM	Signed32	Read/Write
40800	0x9f60	ACTION1.TASK	Unsigned8	Read/Write
40801	0x9f61	ACTION2.TASK	Unsigned8	Read/Write
40802	0x9f62	ACTION3.TASK	Unsigned8	Read/Write
40803	0x9f63	ACTION4.TASK	Unsigned8	Read/Write
40804	0x9f64	ACTION5.TASK	Unsigned8	Read/Write
40805	0x9f65	ACTION6.TASK	Unsigned8	Read/Write
40806	0x9f66	ACTION7.TASK	Unsigned8	Read/Write
40807	0x9f67	ACTION8.TASK	Unsigned8	Read/Write
40808	0x9f68	ACTION9.TASK	Unsigned8	Read/Write
40809	0x9f69	ACTION10.TASK	Unsigned8	Read/Write
40810	0x9f6a	ACTION11.TASK	Unsigned8	Read/Write
40811	0x9f6b	ACTION12.TASK	Unsigned8	Read/Write
40812	0x9f6c	ACTION13.TASK	Unsigned8	Read/Write
40813	0x9f6d	ACTION14.TASK	Unsigned8	Read/Write
40814	0x9f6e	ACTION15.TASK	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
40815	0x9f6f	ACTION16.TASK	Unsigned8	Read/Write
40816	0x9f70	ACTION17.TASK	Unsigned8	Read/Write
40817	0x9f71	ACTION18.TASK	Unsigned8	Read/Write
40818	0x9f72	ACTION19.TASK	Unsigned8	Read/Write
40819	0x9f73	ACTION20.TASK	Unsigned8	Read/Write
40820	0x9f74	ACTION21.TASK	Unsigned8	Read/Write
40821	0x9f75	ACTION22.TASK	Unsigned8	Read/Write
40822	0x9f76	ACTION23.TASK	Unsigned8	Read/Write
40823	0x9f77	ACTION24.TASK	Unsigned8	Read/Write
40824	0x9f78	ACTION25.TASK	Unsigned8	Read/Write
40825	0x9f79	ACTION26.TASK	Unsigned8	Read/Write
40826	0x9f7a	ACTION27.TASK	Unsigned8	Read/Write
40827	0x9f7b	ACTION28.TASK	Unsigned8	Read/Write
40828	0x9f7c	ACTION29.TASK	Unsigned8	Read/Write
40829	0x9f7d	ACTION30.TASK	Unsigned8	Read/Write
40830	0x9f7e	ACTION31.TASK	Unsigned8	Read/Write
40831	0x9f7f	ACTION32.TASK	Unsigned8	Read/Write
40900	0x9fc4	ACTION1.TASKID	Unsigned8	Read/Write
40901	0x9fc5	ACTION2.TASKID	Unsigned8	Read/Write
40902	0x9fc6	ACTION3.TASKID	Unsigned8	Read/Write
40903	0x9fc7	ACTION4.TASKID	Unsigned8	Read/Write
40904	0x9fc8	ACTION5.TASKID	Unsigned8	Read/Write
40905	0x9fc9	ACTION6.TASKID	Unsigned8	Read/Write
40906	0x9fca	ACTION7.TASKID	Unsigned8	Read/Write
40907	0x9fcb	ACTION8.TASKID	Unsigned8	Read/Write
40908	0x9fcc	ACTION9.TASKID	Unsigned8	Read/Write
40909	0x9fcd	ACTION10.TASKID	Unsigned8	Read/Write
40910	0x9fce	ACTION11.TASKID	Unsigned8	Read/Write
40911	0x9fcf	ACTION12.TASKID	Unsigned8	Read/Write
40912	0x9fd0	ACTION13.TASKID	Unsigned8	Read/Write
40913	0x9fd1	ACTION14.TASKID	Unsigned8	Read/Write
40914	0x9fd2	ACTION15.TASKID	Unsigned8	Read/Write
40915	0x9fd3	ACTION16.TASKID	Unsigned8	Read/Write
40916	0x9fd4	ACTION17.TASKID	Unsigned8	Read/Write

ID	Hex	Name	Data Type	Access
40917	0x9fd5	ACTION18.TASKID	Unsigned8	Read/Write
40918	0x9fd6	ACTION19.TASKID	Unsigned8	Read/Write
40919	0x9fd7	ACTION20.TASKID	Unsigned8	Read/Write
40920	0x9fd8	ACTION21.TASKID	Unsigned8	Read/Write
40921	0x9fd9	ACTION22.TASKID	Unsigned8	Read/Write
40922	0x9fda	ACTION23.TASKID	Unsigned8	Read/Write
40923	0x9fdb	ACTION24.TASKID	Unsigned8	Read/Write
40924	0x9fdc	ACTION25.TASKID	Unsigned8	Read/Write
40925	0x9fdd	ACTION26.TASKID	Unsigned8	Read/Write
40926	0x9fde	ACTION27.TASKID	Unsigned8	Read/Write
40927	0x9fdf	ACTION28.TASKID	Unsigned8	Read/Write
40928	0x9fe0	ACTION29.TASKID	Unsigned8	Read/Write
40929	0x9fe1	ACTION30.TASKID	Unsigned8	Read/Write
40930	0x9fe2	ACTION31.TASKID	Unsigned8	Read/Write
40931	0x9fe3	ACTION32.TASKID	Unsigned8	Read/Write
41000	0xa028	ACTION1.TASKPARAM	Signed32	Read/Write
41001	0xa029	ACTION2.TASKPARAM	Signed32	Read/Write
41002	0xa02a	ACTION3.TASKPARAM	Signed32	Read/Write
41003	0xa02b	ACTION4.TASKPARAM	Signed32	Read/Write
41004	0xa02c	ACTION5.TASKPARAM	Signed32	Read/Write
41005	0xa02d	ACTION6.TASKPARAM	Signed32	Read/Write
41006	0xa02e	ACTION7.TASKPARAM	Signed32	Read/Write
41007	0xa02f	ACTION8.TASKPARAM	Signed32	Read/Write
41008	0xa030	ACTION9.TASKPARAM	Signed32	Read/Write
41009	0xa031	ACTION10.TASKPARAM	Signed32	Read/Write
41010	0xa032	ACTION11.TASKPARAM	Signed32	Read/Write
41011	0xa033	ACTION12.TASKPARAM	Signed32	Read/Write
41012	0xa034	ACTION13.TASKPARAM	Signed32	Read/Write
41013	0xa035	ACTION14.TASKPARAM	Signed32	Read/Write
41014	0xa036	ACTION15.TASKPARAM	Signed32	Read/Write
41015	0xa037	ACTION16.TASKPARAM	Signed32	Read/Write
41016	0xa038	ACTION17.TASKPARAM	Signed32	Read/Write
41017	0xa039	ACTION18.TASKPARAM	Signed32	Read/Write
41018	0xa03a	ACTION19.TASKPARAM	Signed32	Read/Write

ID	Hex	Name	Data Type	Access
41019	0xa03b	ACTION20.TASKPARAM	Signed32	Read/Write
41020	0xa03c	ACTION21.TASKPARAM	Signed32	Read/Write
41021	0xa03d	ACTION22.TASKPARAM	Signed32	Read/Write
41022	0xa03e	ACTION23.TASKPARAM	Signed32	Read/Write
41023	0xa03f	ACTION24.TASKPARAM	Signed32	Read/Write
41024	0xa040	ACTION25.TASKPARAM	Signed32	Read/Write
41025	0xa041	ACTION26.TASKPARAM	Signed32	Read/Write
41026	0xa042	ACTION27.TASKPARAM	Signed32	Read/Write
41027	0xa043	ACTION28.TASKPARAM	Signed32	Read/Write
41028	0xa044	ACTION29.TASKPARAM	Signed32	Read/Write
41029	0xa045	ACTION30.TASKPARAM	Signed32	Read/Write
41030	0xa046	ACTION31.TASKPARAM	Signed32	Read/Write
41031	0xa047	ACTION32.TASKPARAM	Signed32	Read/Write
41100	0xa08c	ACTION1.TASKTEXT	String	Read/Write
41101	0xa08d	ACTION2.TASKTEXT	String	Read/Write
41102	0xa08e	ACTION3.TASKTEXT	String	Read/Write
41103	0xa08f	ACTION4.TASKTEXT	String	Read/Write
41104	0xa090	ACTION5.TASKTEXT	String	Read/Write
41105	0xa091	ACTION6.TASKTEXT	String	Read/Write
41106	0xa092	ACTION7.TASKTEXT	String	Read/Write
41107	0xa093	ACTION8.TASKTEXT	String	Read/Write
41108	0xa094	ACTION9.TASKTEXT	String	Read/Write
41109	0xa095	ACTION10.TASKTEXT	String	Read/Write
41110	0xa096	ACTION11.TASKTEXT	String	Read/Write
41111	0xa097	ACTION12.TASKTEXT	String	Read/Write
41112	0xa098	ACTION13.TASKTEXT	String	Read/Write
41113	0xa099	ACTION14.TASKTEXT	String	Read/Write
41114	0xa09a	ACTION15.TASKTEXT	String	Read/Write
41115	0xa09b	ACTION16.TASKTEXT	String	Read/Write
41116	0xa09c	ACTION17.TASKTEXT	String	Read/Write
41117	0xa09d	ACTION18.TASKTEXT	String	Read/Write
41118	0xa09e	ACTION19.TASKTEXT	String	Read/Write
41119	0xa09f	ACTION20.TASKTEXT	String	Read/Write
41120	0xa0a0	ACTION21.TASKTEXT	String	Read/Write

ID	Hex	Name	Data Type	Access
41121	0xa0a1	ACTION22.TASKTEXT	String	Read/Write
41122	0xa0a2	ACTION23.TASKTEXT	String	Read/Write
41123	0xa0a3	ACTION24.TASKTEXT	String	Read/Write
41124	0xa0a4	ACTION25.TASKTEXT	String	Read/Write
41125	0xa0a5	ACTION26.TASKTEXT	String	Read/Write
41126	0xa0a6	ACTION27.TASKTEXT	String	Read/Write
41127	0xa0a7	ACTION28.TASKTEXT	String	Read/Write
41128	0xa0a8	ACTION29.TASKTEXT	String	Read/Write
41129	0xa0a9	ACTION30.TASKTEXT	String	Read/Write
41130	0xa0aa	ACTION31.TASKTEXT	String	Read/Write
41131	0xa0ab	ACTION32.TASKTEXT	String	Read/Write
41200	0xa0f0	ACTION1.SOURCETEXT	String	Read/Write
41201	0xa0f1	ACTION2.SOURCETEXT	String	Read/Write
41202	0xa0f2	ACTION3.SOURCETEXT	String	Read/Write
41203	0xa0f3	ACTION4.SOURCETEXT	String	Read/Write
41204	0xa0f4	ACTION5.SOURCETEXT	String	Read/Write
41205	0xa0f5	ACTION6.SOURCETEXT	String	Read/Write
41206	0xa0f6	ACTION7.SOURCETEXT	String	Read/Write
41207	0xa0f7	ACTION8.SOURCETEXT	String	Read/Write
41208	0xa0f8	ACTION9.SOURCETEXT	String	Read/Write
41209	0xa0f9	ACTION10.SOURCETEXT	String	Read/Write
41210	0xa0fa	ACTION11.SOURCETEXT	String	Read/Write
41211	0xa0fb	ACTION12.SOURCETEXT	String	Read/Write
41212	0xa0fc	ACTION13.SOURCETEXT	String	Read/Write
41213	0xa0fd	ACTION14.SOURCETEXT	String	Read/Write
41214	0xa0fe	ACTION15.SOURCETEXT	String	Read/Write
41215	0xa0ff	ACTION16.SOURCETEXT	String	Read/Write
41216	0xa100	ACTION17.SOURCETEXT	String	Read/Write
41217	0xa101	ACTION18.SOURCETEXT	String	Read/Write
41218	0xa102	ACTION19.SOURCETEXT	String	Read/Write
41219	0xa103	ACTION20.SOURCETEXT	String	Read/Write
41220	0xa104	ACTION21.SOURCETEXT	String	Read/Write
41221	0xa105	ACTION22.SOURCETEXT	String	Read/Write
41222	0xa106	ACTION23.SOURCETEXT	String	Read/Write

ID	Hex	Name	Data Type	Access
41223	0xa107	ACTION24.SOURCETEXT	String	Read/Write
41224	0xa108	ACTION25.SOURCETEXT	String	Read/Write
41225	0xa109	ACTION26.SOURCETEXT	String	Read/Write
41226	0xa10a	ACTION27.SOURCETEXT	String	Read/Write
41227	0xa10b	ACTION28.SOURCETEXT	String	Read/Write
41228	0xa10c	ACTION29.SOURCETEXT	String	Read/Write
41229	0xa10d	ACTION30.SOURCETEXT	String	Read/Write
41230	0xa10e	ACTION31.SOURCETEXT	String	Read/Write
41231	0xa10f	ACTION32.SOURCETEXT	String	Read/Write

9.3 AKD2G Explicit Messaging using the MSG Instruction

In Studio 5000 the MSG instruction is used for Explicit Messaging. 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.

9.3.1 Explicit Messaging

On-Demand Messaging

Explicit Messages allow access to 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.

9.3.2 Supported Services

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

9.3.3 Supported Objects

AKD2G supports a number of standard and vendor-specific objects for motion control. See Drive Objects in the AKD2G EtherNet/IP Online Help for more information.

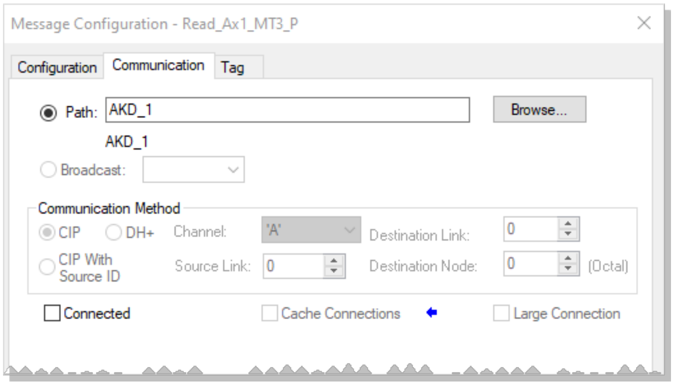
Parameter Object

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 AKD2G EtherNet/IP Objects List for a list of available parameters. See Drive Objects in the AKD2G EtherNet/IP Online Help for more information. Note: This attribute is different than the attributes and attribute IDs listed in AKD2GEtherNet/IP Objects and Attributes.

The following examples will cover 6 possible scenarios

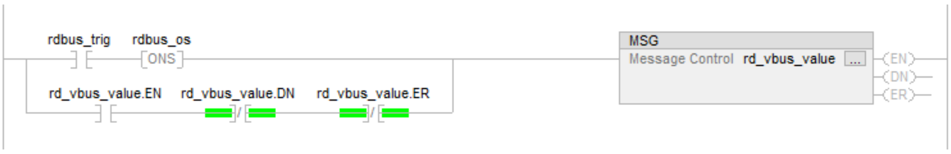
- Example 1: Set an axis-specific non-array parameter
- Example 2: Get an axis-specific non-array parameter
- Example 3: Set an axis-specific array type parameter
- Example 4: Get an axis-specific array type parameter
- Example 5: Set a drive level (axis-independent) parameter
- Example 6: Get a drive-level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

All MSG blocks must be configured under the Communication Path with a specified Path (the target AKD2G drive).



The recommended best practice for triggering the MSG instruction is to use a contact to One Shot the .EnableIn of the MSG instruction followed by a parallel branch to seal-in the instruction on .EnableIn which will continue to be enabled and will execute until 1) the Done (.DN) bit is set (success) or 2) the Error (.ER) bit is set (failed).

Example:



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

Definition:

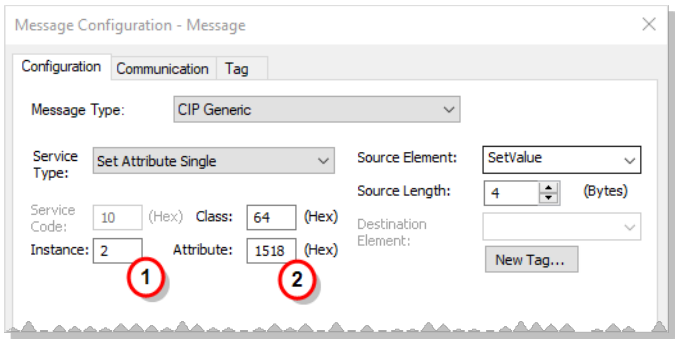
Field	Value
Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10 (Write)
Class	0x64
Instance	Axis Number (1 or 2)
Attribute	Parameter Number (in Hex for Studio 5000 MSG instruction)

Example: AXIS#.GUI.PARAM01

Parameter	ID (Decimal)	ID (Hex)	Data Type	Data Size	Access
AXIS#.GUI.PARAM01	5400	1518	Integer	Signed32	Read/Write

5400,AXIS2.GUI.PARAM01,Integer,4 Byte Signed,ReadWrite,0,0 to 0

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



1. Axis #
2. Parameter Number in Hex (decimal 5400)

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

Definition:

Field	Value
Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0xE (Read)
Class	0x64
Instance	Axis Number (1 or 2)
Attribute	Parameter Number (in Hex for Studio 5000 MSG instruction)

Example: AXIS#.GUI.PARAM01

Parameter	ID (Decimal)	ID (Hex)	Data Type	Data Size	Access
AXIS#.GUI.PARAM01	5400	1518	Integer	Signed32	Read/Write

5400,AXIS2.GUI.PARAM01,Integer,4 Byte Signed,ReadWrite,0,0 to 0

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

9.3.6 Example 3: Set an axis-specific array type parameter

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

Example: AXIS#.MT.P

Parameter	ID (Decimal)	ID (Hex)	Data Type	Data Size	Access
AXIS#.MT.P	6307	18A3	Position	Signed32[32]	Read/Write

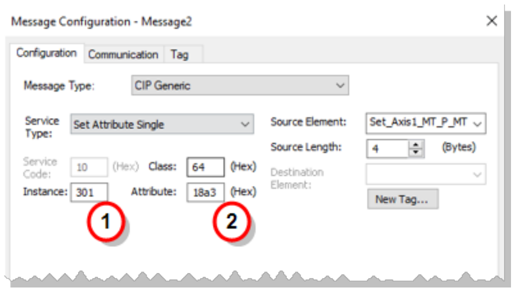
6307,AXIS1.MT.P,Position,4 Byte Signed,ReadWrite,2,0 to 31

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

Example: Axis 1, MT.P Task 3

Field	Value
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)

Source Element was created to hold the value to Set (Write).



1. Axis and Parameter Array Index
2. Parameter Number in Hex (decimal 6307)

9.3.7 Example 4: Get an axis-specific array type parameter

Definition:

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

Example: AXIS#.MT.P

Parameter	ID (Decimal)	ID (Hex)	Data Type	Data Size	Access
AXIS#.MT.P	6307	18A3	Position	Signed32[32]	Access

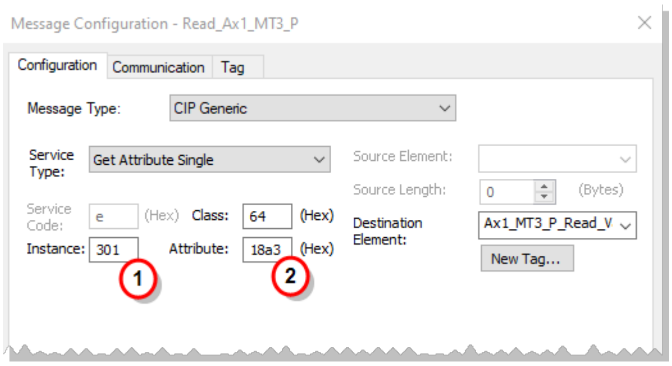
6307,AXIS1.MT.P,Position,4 Byte Signed,ReadWrite,2,0 to 31

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

Example: Axis 1, MT.P task 3

Field	Value
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)

Tag was created for the Destination Element to hold the Read value.



1. Axis and Parameter Array Index
2. Parameter Number in Hex (decimal 6307)

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

Definition:

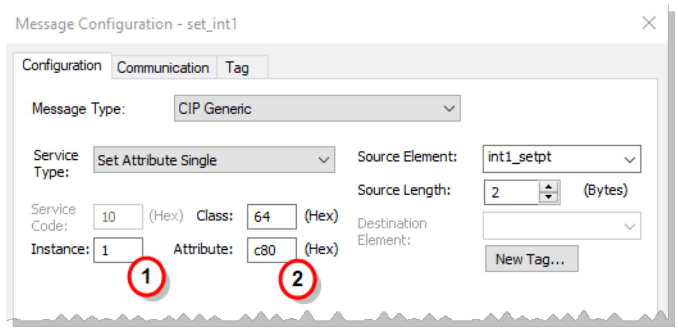
Field	Value
Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10 (Write)
Class	0x64
Instance	Ignored (Set to 1)
Attribute	Parameter Number (in Hex for Studio 5000 MSG instruction)

Example: USER.INT1

Parameter	ID (Decimal)	ID (Hex)	Data Type	Data Size	Access
USER.INT1	3200	C80	Integer	Signed16	Read/Write

3200, USER.INT1, Integer, 2 Byte, Signed, ReadWrite,0,0 to 0

Field	Value
Message Type	CIP Generic
Service Type	Set Attribute Single
Service Code	0x10 (Write)
Class	0x64
Instance	Ignored (Set to 1)
Attribute	0xc80 (3200 decimal)



1. Ignored (Set to 1)
2. Parameter Number in Hex (decimal 3200)

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

Definition:

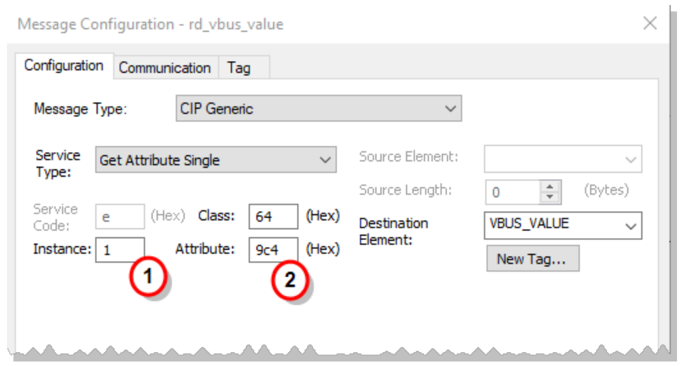
Field	Value
Message Type	CIP Generic
Service Type	Get Attribute Single
Service Code	0xE (Read value)
Class	0x64
Instance	Ignored (Set to 1)
Attribute	Parameter Number (in Hex for Studio 5000 MSG instruction)

Example: VBUS.VALUE

Parameter	ID (Decimal)	ID (Hex)	Data Type	Data Size	Access
VBUS.VALUE	2500	9C4	Float		Read Only

2500,VBUS.VALUE,Float,4 Byte Signed,ReadOnly,0,0 to 0

Field	Value
Message Type	CIP Generic
Service Type	Get Attribute Single
Service Code	0x0E
Class	0x64
Instance	Ignored (Set to 1)
Attribute	0x9c4 (2500 decimal)



1. Ignored (Set to 1)
2. Parameter Number in Hex (decimal 2500)

9.4 Example of AKD2G Dynamic Mapping With Studio 5000

Dynamic Mapping provides the ability to map user selected drive parameters to the upper Bytes of the command and response areas of the cyclic data. For a list of data sizes and tag types see AKD1G Explicit Messaging using the MSG Instruction.

9.4.1 Command Assembly Dynamic Mapping

Axis 1 Byte	Axis 2 Byte	Data	Comment	Command
32-63	96-127	Command Dynamic Map	Data for user mappable parameters. See AXIS#.EIP.DYNAMICCMDMAP.	AXIS#.EIP.DYNAMICCMDDATA

9.4.2 Response Assembly Dynamic Mapping

Axis 1 Byte	Axis 2 Byte	Data	Comment	Command
32-63	96-127	Response Dynamic Map	Data for user mappable parameters. See AXIS#.EIP.DYNAMICRSPMAP.	AXIS#.EIP.DYNAMICRSPDATA

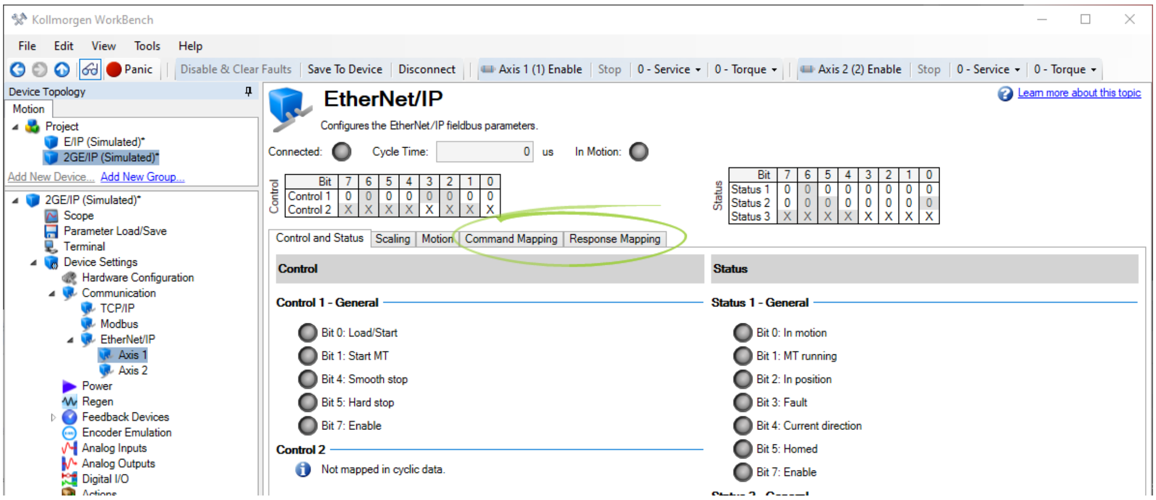
The following commands can be used in Terminal to dynamically map parameters or alternatively use the WorkBenchGUI.

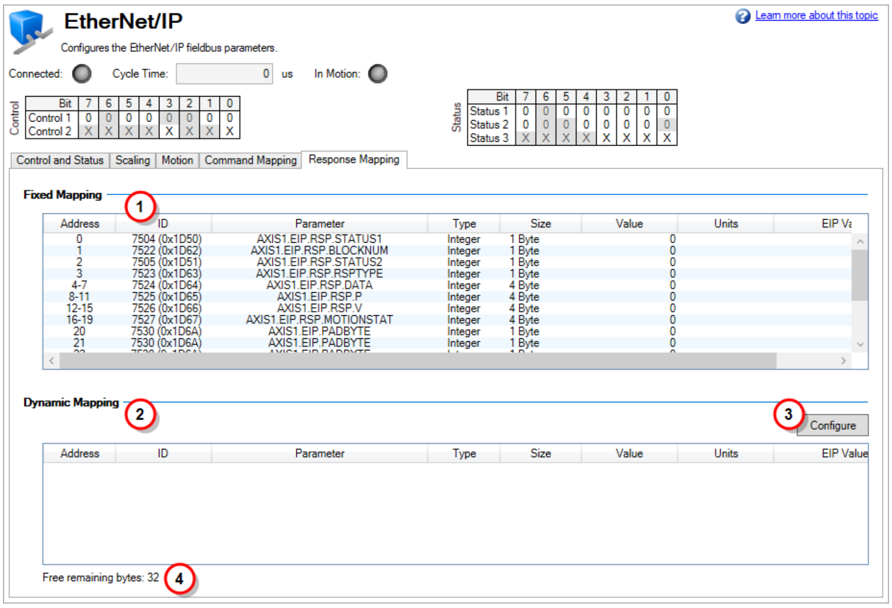
See WorkBench Help for details on the following commands:

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

9.4.3 Using the Workbench GUI to map parameters dynamically

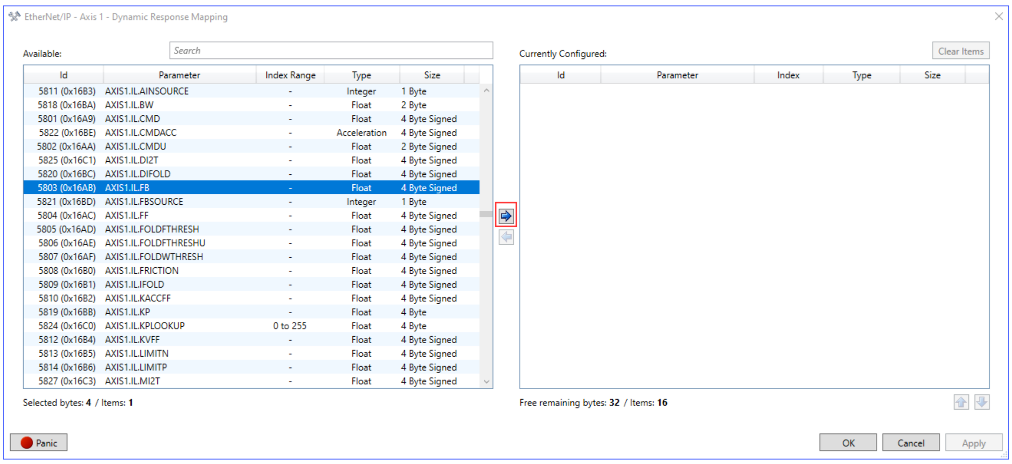
Example: Axis 1 Command and Response Mapping tabs



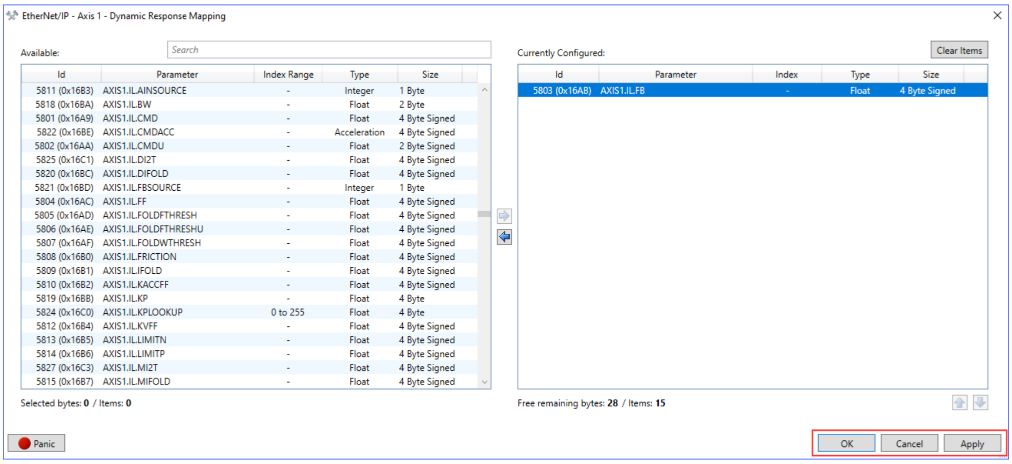


1. Fixed Mapping List
2. Dynamic Mapping List
3. Select Configure to setup Dynamic Mapping
4. Remaining Bytes in Dynamic Mapping

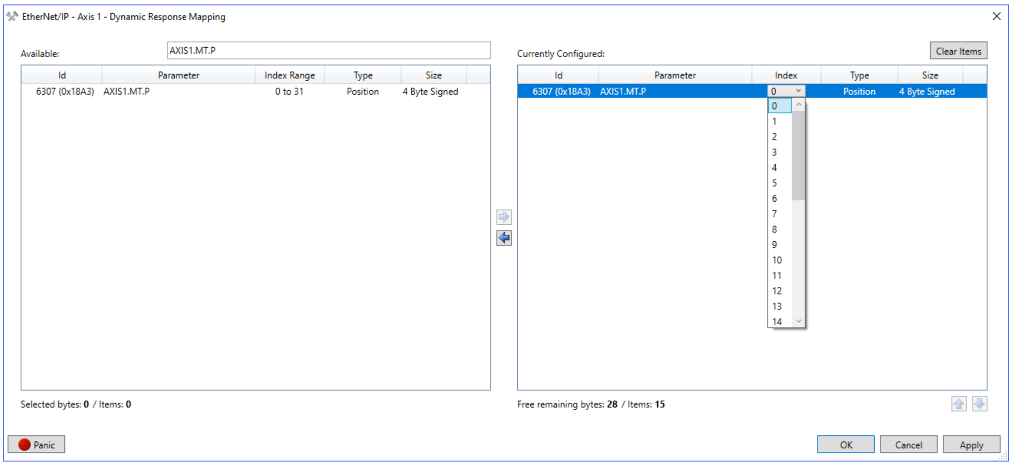
Click to highlight the desired parameter and then press the right arrow key in the center of the screen to move the parameter into the **Currently Configured** list.



Click **Apply** and then click **OK**.



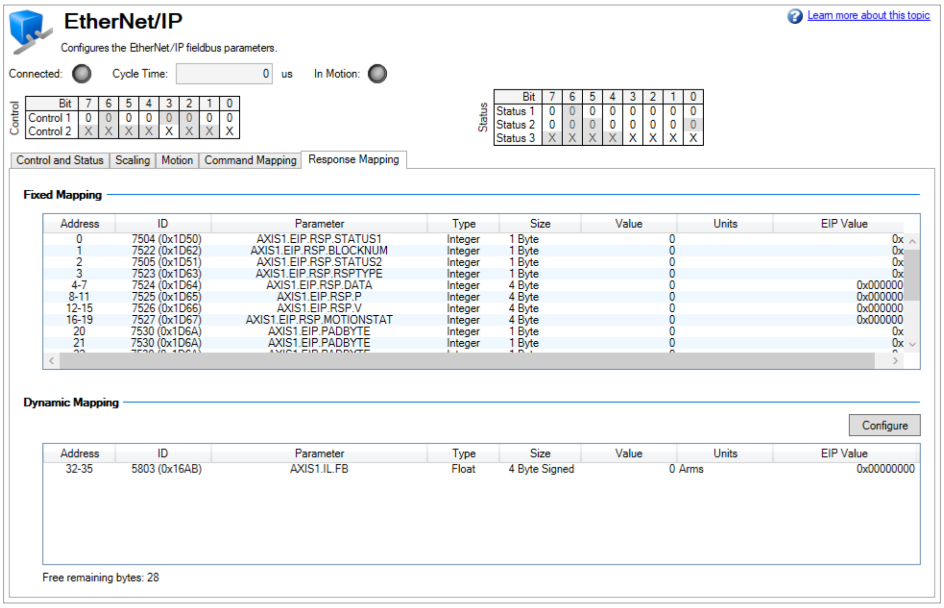
Once array-type parameters are added to the Currently Configured List there will be a listbox under the Index column for selection and assigning the index of the parameter to Dynamically Map (i.e. AXIS1.MT.P for Motion Task 3; index = 3).



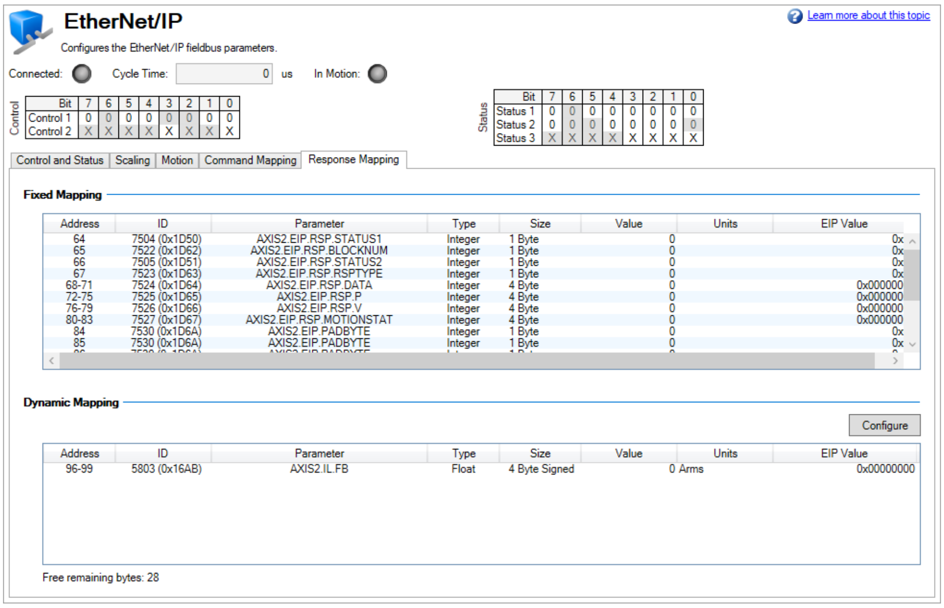
9.4.4 Example of Axis 1 and Axis 2 Current Loop Feedback Dynamically Mapped

Axis 1:

Under the Dynamic Mapping list AXIS1.IL.FB is mapped to the Response Assembly's Bytes 32-35 (4 Bytes total) and the Free Remaining Bytes are now reduced by four Bytes from 32 to 28.



Axis 2:



The Command Mapping tab uses the same conventions and methods.

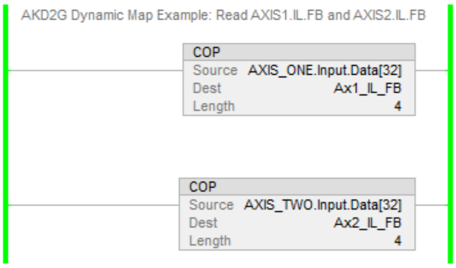
There are Command and Response Mapping tabs for both Axis 1 and Axis 2.

It is up to the programmer to copy the corresponding Bytes of data for each dynamically mapped parameter from the I/O Cyclic Data (Command Assembly and Response Assembly) into their EtherNet/IP Master program to be used.

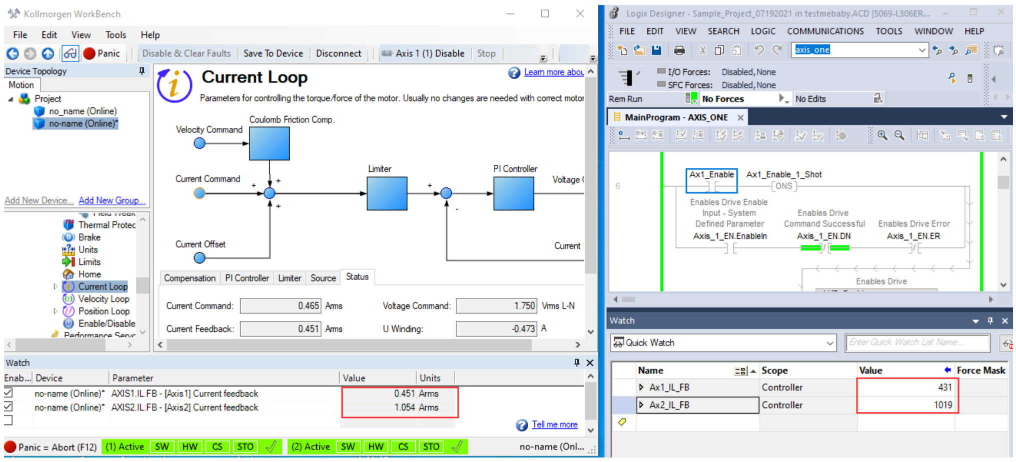
When the parameter is dynamically mapped in the WorkBench GUI as shown above, the data exchange starts immediately as long as the PLC is polling the drive.

Example in Studio 5000

Bytes 32-35 of the AXIS_ONE and AXIS_TWO Axis structure's input data are copied to tags (in this example, declared as DINT, 4 Bytes)



The following demonstrates current feedback values in the Watch Window of Workbench and in the Watch Window of Studio 5000:



10 Software Distribution License