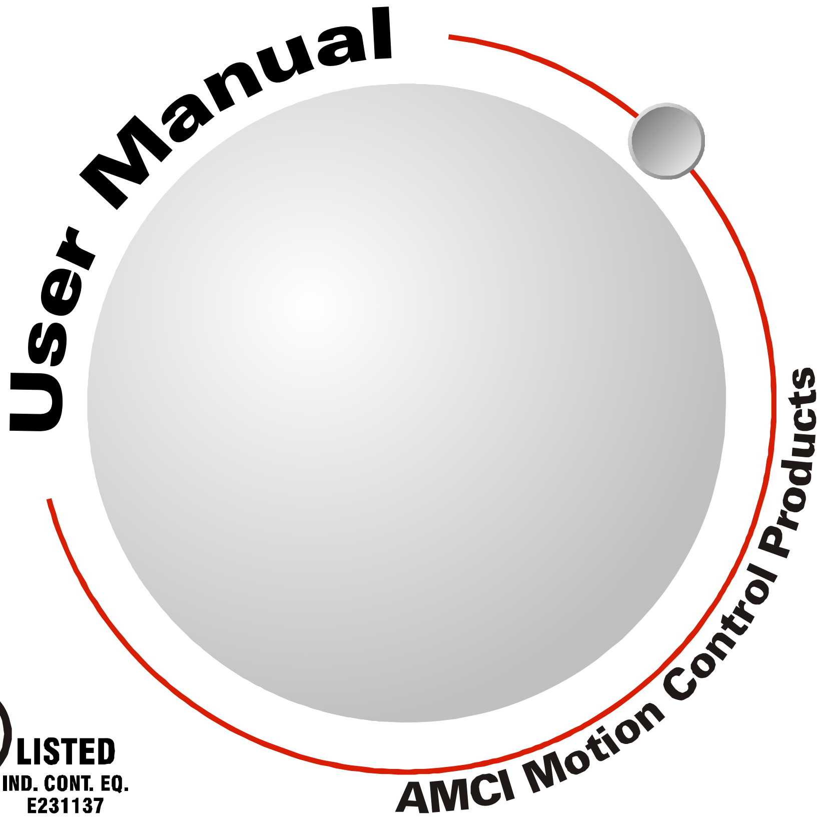
AMCI ADVANCED MICRO CONTROLS INC.

Manual #: 940-0S301

SD17060E-K and SD31045E-K Networked Stepper Indexer/Drivers

with EtherCAT Interface



EtherCAT®

UL® LISTED IND. CONT. EQ. E231137

GENERAL INFORMATION

Important User Information

The products and application data described in this manual are useful in a wide variety of different applications. Therefore, the user and others responsible for applying these products described herein are responsible for determining the acceptability for each application. While efforts have been made to provide accurate information within this manual, AMCI assumes no responsibility for the application or the completeness of the information contained herein.

UNDER NO CIRCUMSTANCES WILL ADVANCED MICRO CONTROLS, INC. BE RESPONSIBLE OR LIABLE FOR ANY DAMAGES OR LOSSES, INCLUDING INDIRECT OR CONSEQUENTIAL DAMAGES OR LOSSES, ARISING FROM THE USE OF ANY INFORMATION CONTAINED WITHIN THIS MANUAL, OR THE USE OF ANY PRODUCTS OR SERVICES REFERENCED HEREIN.

No patent liability is assumed by AMCI, with respect to use of information, circuits, equipment, or software described in this manual.

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

Standard Warranty

ADVANCED MICRO CONTROLS, INC. warrants that all equipment manufactured by it will be free from defects, under normal use, in materials and workmanship for a period of [18] months. Within this warranty period, AMCI shall, at its option, repair or replace, free of charge, any equipment covered by this warranty which is returned, shipping charges prepaid, within eighteen months from date of invoice, and which upon examination proves to be defective in material or workmanship and not caused by accident, misuse, neglect, alteration, improper installation or improper testing.

The provisions of the "STANDARD WARRANTY" are the sole obligations of AMCI and excludes all other warranties expressed or implied. In no event shall AMCI be liable for incidental or consequential damages or for delay in performance of this warranty.

Returns Policy

All equipment being returned to AMCI for repair or replacement, regardless of warranty status, must have a Return Merchandise Authorization number issued by AMCI. Call (860) 585-1254 with the model number and serial number (if applicable) along with a description of the problem during regular business hours, Monday through Friday, 8AM - 5PM Eastern. An "RMA" number will be issued. Equipment must be shipped to AMCI with transportation charges prepaid. Title and risk of loss or damage remains with the customer until shipment is received by AMCI.

24 Hour Technical Support Number

24 Hour technical support is available on this product. If you have internet access, start at www.amci.com. Product documentation and FAQ’s are available on the site that answer most common questions.

If you require additional technical support, call (860) 583-1254. Your call will be answered by the factory during regular business hours, Monday through Friday, 8AM - 5PM Eastern. During non-business hours an automated system will ask you to enter the telephone number you can be reached at. Please remember to include your area code. The system will page an engineer on call. Please have your product model number and a description of the problem ready before you call.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Table of Contents

Notes

About this Manual

Read this chapter to learn how to navigate through this manual and familiarize yourself with the conventions used in it. The last section of this chapter highlights the manual's remaining chapters and their target audience.

Audience

This manual explains the installation and operation of AMCI's Networked Stepper Indexer/Drivers with the EtherCAT® network interface. It is written for the engineer responsible for incorporating these products into a design as well as the engineer or technician responsible for their actual installation.

Applicable Units

This manual applies to the SD17060E-K and SD31045E-K units.

With the exception of power requirements and output current, the information in this manual applies equally to the SD17060E-K and SD31045E-K units. The term "Networked Driver" is used when the information applies to both drivers.

Navigating this Manual

This manual is designed to be used in both printed and on-line forms. Its on-line form is a PDF document, which requires Adobe Acrobat Reader version 7.0+ to open it. You are allowed to select and copy sections for use in other documents and, if you own Adobe Acrobat version 7.0 or later, you are allowed to add notes and annotations. If you decide to print out this manual, all sections contain an even number of pages which allows you to easily print out a single chapter on a duplex (two-sided) printer.

Manual Conventions

Three icons are used to highlight important information in the manual:

NOTE **NOTES** highlight important concepts, decisions you must make, or the implications of those decisions.

CAUTION **CAUTIONS** tell you when equipment may be damaged if the procedure is not followed properly.

WARNING **WARNINGS** tell you when people may be hurt or equipment may be damaged if the procedure is not followed properly.

The following table shows the text formatting conventions:

Format	Description
Normal Font	Font used throughout this manual.
Emphasis Font	Font used for parameter names and the first time a new term is introduced.
Cross Reference	When viewing the PDF version of the manual, clicking on a blue cross reference jumps you to referenced section of the manual.
HTML Reference	When viewing the PDF version of the manual, clicking on a red cross reference opens your default web browser to the referenced section of the AMCI website if you have Internet access.

Trademark Notices

The AMCI logo is a trademark of Advanced Micro Controls Inc. EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany. "Adobe" and "Acrobat" are registered trademarks of Adobe Systems Incorporated.

All other trademarks contained herein are the property of their respective holders.

Revision Record

This manual, 940-0S301 is the second release of this manual. It corrects typographical errors and adds some clarification notes to command and status bits. It was first released on December 25th, 2020.

Revision History

940-0S300: November 25th, 2020 - Initial Release

Manual Layout

You will most likely read this manual for one of two reasons:

- If you are curious about the Networked Stepper Indexer/Driver products from AMCI, this manual contains the information you need to determine if these products are the right products for your application. The first chapter, ***SD17060E-K & SD31045E-K Specifications*** contains all of the information you will need to fully specify the right product for your application.
- If you need to install and use a Networked Stepper Indexer/Driver product from AMCI, then the rest of the manual is written for you. To simplify installation and configuration, the rest of the manual is broken down into *tasks* and *references*. Using a Networked Stepper Indexer/Driver requires you to complete multiple tasks, and the manual is broken down into sections that explain how to complete each one.

Section Title	Page #	Section Description
SD17060E-K & SD31045E-K Specifications	9	Complete specifications for the SD17060E-K and SD31045E-K products.
UL/cUL Recognized Installations	19	Installation guidelines that must be followed in order for an installation of a Networked Driver to meet UL/cUL criterion.
Motion Control	21	Reference information on how the driver can be used to control motion in your application.
Calculating Move Profiles	41	Reference information on calculating detailed move profiles.
Homing an AMCI Networked Driver	51	Reference information on how the home position of the SD17060E-K or SD31045E-K can be set.
Configuration Data Format	57	Reference information on the format of the CANopen over EtherCAT (CoE) data that is used to configure the unit.
Command Mode Data Format	65	Reference information on the format of the network data to and from the SD17060E-K or SD31045E-K that is used to command it.
General Installation Guidelines	85	Reference information on grounding, wiring, and surge suppression, that is applicable to any control system installation.
Installing an AMCI Networked Driver	91	Task instructions covering how to install an SD17060E-K or SD31045E-K on a machine. Includes information on mounting, grounding, and wiring specific to the units.
EtherCAT System Configuration	101	Task instructions that covers how to add a Networked Drive to an EtherCAT network.
Optional: Distributed Clock - SYNC0 Setup	105	Optional Task instructions that covers how to configure an SD17060E-K or SD31045E-K to use the Distributed Clock functionality of the EtherCAT protocol.

SD17060E-K & SD31045E-K SPECIFICATIONS

This manual is designed to get you up and running quickly with either the SD17060E-K or the SD31045E-K from AMCI. As such, it assumes you have some basic knowledge of stepper systems, such as the resolution you want run your motor at, and the reasons why you'd want to use Idle Current Reduction and the reasons why you wouldn't. If these terms or ideas are new to you, we're here to help. AMCI has a great deal of information on our website and we are adding more all the time. If you can't find what you're looking for at http:///www.amci.com, send us an e-mail or call us. We're here to support you with all of our knowledge and experience.

Networked Stepper Indexer/Drivers

The AMCI SD17060E-K is a 6.0Arms micro-stepping driver with a 170Vdc internal bus voltage. The AMCI SD31045E-K is a 4.4Arms micro-stepping driver with a maximum internal bus voltage of 310Vdc. Each Networked Driver attaches to your Ethernet based network and communicates using the EtherCAT protocol.

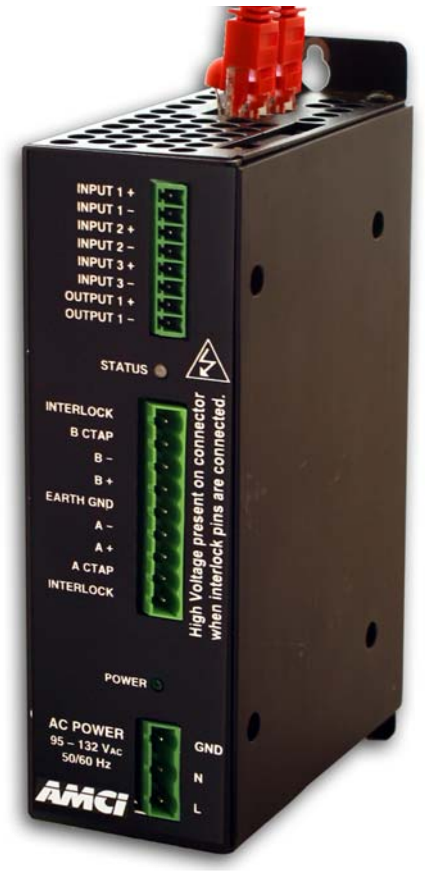


Figure R1.1 SD17060E-K Drive

EtherCAT uses standard Ethernet cabling, but forgoes the full TCP/IP stack typically associated with Ethernet communications. The EtherCAT protocol transfers data to and from multiple slaves with a single packet of information. Data for each slave is located at a known position within the packet. EtherCAT slave devices use a hardware only solution to read and write data to the packet before transmitting the packet to the next slave. This solution leads to a delay between nodes that is typically 4 microseconds or less. This results in a very fast and deterministic network.

An EtherCAT Slave Information (ESI) file is available for the Networked Driver units. The ESI file is required to add the device to the network. Configuration of the Networked Driver is accomplished using the CANopen PDO and SDO objects. EtherCAT supports CANopen through its CANopen over Ethernet (CoE) interface.

Motion commands and status information is transmitted using the output and input variables assigned to the Networked Driver when the EtherCAT system is configured.

Each unit also supports the Distributed Clock (DC) functionality of the EtherCAT system. This allows you to synchronize the start of moves across devices using the SYNC0 signal instead of the SyncManager 2 event.

The combination of host and driver gives you several advantages:

- Sophisticated I/O processing can be performed in the host (PLC or other controller) before sending commands to the Networked Indexer/Driver.
- All motion logic is programmed in the host, eliminating the need to learn a separate motion control language
- Eliminating the separate indexer lowers Total System Cost

Both Networked Drivers can be powered from a nominal 115 Vac, 50/60 Hz source and they will have a 170 Vdc motor bus voltage with this input supply. The SD31045E-K Drivers can also be powered by a nominal 230 Vac 50/60 Hz source and they will have a 310 Vdc motor bus voltage with this input supply.

Networked Stepper Indexer/Drivers (continued)

The output motor current is fully programmable on these Networked Drivers. The SD17060E-K Drivers can be programmed from 1.0 Arms to 6.0 Arms, while the SD31045E-K Drivers can be programmed from 1.0 Arms to 4.4 Arms. It is even possible to change the motor current setting while the move is in progress. These ranges of output currents makes the Networked Drivers compatible with the complete line of stepper motors from AMCI.

In addition to the Motor Current setting, the Motor Steps per Turn, Idle Current Reduction, and Anti-Resonance Circuit features are also fully programmable. If you have used other stepper indexer products from AMCI you will find the programming to be very similar to these products.

All Networked Drivers from AMCI are true RMS motor current control drivers. This means that you will always receive the motor's rated torque regardless of the *Motor_Resolution* setting. (Drivers that control the peak current to the motor experience a 30% decrease in motor torque when microstepping a motor.) The Networked Drivers automatically switch from RMS to peak current control when the motor is idle to prevent overheating the motor.

In addition to power and motor hookups, the Networked Drivers have three DC inputs and one DC output that are used by the indexer. Configuration data from the host sets the function of these points. The output can be configured to be a Fault Output or a general purpose output. Each input can be individually configured as a:

- CW or CCW Limit Switch
- Home Limit Switch
- Capture Encoder Position Input
- Stop Jog or Registration Move Input
- Start Indexer Move
- Emergency Stop Input
- A/B Encoder Inputs
- General Purpose Input

Conformance Markings

The Networked Driver devices meet the requirements for the following conformance markings when installed and operated in accordance with the instructions contained in their product documentation.

UL

Listed component in the USA and Canada. UL file number is E231137.

- US: UL 61800-5-1 Standard For Safety For Adjustable Speed Electrical Power Drive Systems - Part 5-1: Safety Requirements - Electrical, Thermal and Energy
- Canada: CSA C22.2 No. 274 - Adjustable Speed Drives

CE

- Directive 2014/30/EU of the European Parliament and of the Council (EMC), of 26 February 2014 on the harmonisation of the laws of the Member States relating to electromagnetic compatibility; per EN 61800-3:2004/A1:2012.
- Directive 2014/35/EU of the European Parliament and of the Council (Low Voltage), of 26 February 2014 on the harmonisation of the laws of the Member States relating to the making available on the market of electrical equipment designed for use within certain voltage limits; per EN 61010-1:2010.

RoHS

Directive (EU) 2015/863 (RoHS 3) and Directive 2011/65/EU (RoHS 2)

Specifications

Driver Type

Two bipolar MOSFET H-bridges with 20KHz PWM current control.

SD17060E-K Drivers: 170Vdc output bus.
SD31045E-K Drivers: up to 310Vdc output bus.

Physical Dimensions

Width: SD17060E-K Drivers: 2.1 inches max.
SD31045E-K Drivers: 2.7 inches max.

Depth: 4.0 inches max.

Height: 6.2 inches
7.0 inches with mounting tabs

Weight

SD17060E-K Drivers: 2.4 lbs. (1.1 kg.) max.
SD31045E-K Drivers: 2.7 lbs. (1.25 kg.) max.

Inputs

Electrical Characteristics for all Inputs: . . . .
Differential. UL 1577 recognized, 3750 Vrms for 1 minute. Can be wired as single ended inputs.

Accepts 3.5 to 27Vdc without the need for an external current limiting resistor.

Output

Electrical Characteristics:
Open Collector/Emitter. Opto-isolated. (2500 Vac rms (f=60 Hz, t=1 min. duration)
30Vdc, 20 mA max.

The Output can be programmed to be a general purpose output or a Fault Output.

The Fault Output is normally on. Turns off under the following conditions:

Reset	The driver initialization is not yet complete on power up.
Short Circuit	Motor Phase to Phase or Phase to Earth Ground
Over Temp	Heat Sink temperature exceeds 90° C (195° F)
No Motor	The motor interlock terminals are not connected.

Faults are reported in the Network Input Data and can be cleared through the Network Output Data.

Motor Interlock

Two pins that must be shorted together before power will be applied to motor. Can be shorted together through mechanical relay contacts.

Motor Current

SD17060E-K Drivers: Programmable from 1.0 to 6.0 Arms.

SD31045E-K Drivers: Programmable from 1.0 to 4.4 Arms.

Motor Resolution

Programmable to any value from 200 to 32,767 steps per revolution.

Idle Current Reduction

Programmable from 0% to 100% programmed motor current in 1% increments. Motor current is reduced to selected level if there is no motion for 1.5 seconds. Current is restored to full value when motion is started.

Internal Power Fuses

10 Amp Slow Blow. Both Line and Neutral are fused. Fuses are not user replaceable.

Environmental Specifications

Input Power .... SD17060E-K Drivers: 95 to 132Vac, 50/60 Hz, 5.0 Apk max.
SD31045E-K Drivers: 95 to 264Vac, 50/60 Hz, 5.0 Apk max.

Ambient Operating Temperature

.... -4° to +122°F (-20° to +50°C)

Storage Temperature

.... -40° to +185°F (-40° to +85°C)

Humidity .... 0 to 95%, non-condensing

Motor Specifications

Type	2 phase. 4, 6, or 8 lead motor
Insulation	Minimum 500Vdc phase-to-phase and phase-to-case
Inductance	0.3 mH minimum. 2.5 to 45 mH recommended

Connectors

All mating connectors are included with driver.

Connector	Wire	Strip Length	Min./Max. Tightening Torque
I/O	28 - 16 AWG	0.275 inches	1.91/2.23 lb-in (0.22/0.25 Nm)
Motor	24 - 12 AWG	0.275 inches	4.46/5.35 lb-in (0.5/0.6 Nm)
Power	24 - 12 AWG	0.275 inches	4.46/5.35 lb-in (0.5/0.6 Nm)

Indexer Functionality

Standard Functionality

The table below lists the functionality offered by the indexer built into the AMCI Networked Drivers

Feature	Description
Programmable Inputs	Each of the three inputs can be programmed as a Home Limit, Over Travel Limit, Capture Input, Jog Stop, or E-Stop Input. They can also be programmed to accept a quadrature encoder.
Programmable Output	The single output on the Networked Driver can be programmed as a Fault Output or as a general purpose DC output point.
Programmable Parameters	Starting Speed, Running Speed, Acceleration, Deceleration, and Accel/Decel Types are fully programmable.
Homing	Allows you to set the machine to a known position. The Networked Driver can home to a discrete input or to an encoder marker pulse.
Synchronous Moves	Using the Distributed Clock functionality of the EtherCAT system, multiple devices can synchronize the start of their moves to the SYNC0 signal. Moves will begin within ±25 microseconds of each other.
Jog Move	Allows you to jog the motor in either direction based on a bit from your controller.
Relative Move	Allows you to drive the motor a specific number of steps in either direction from the current location.
Absolute Move	Allows you to drive the motor from one known location to another known location.
Registration Move	Allows you to jog the motor in either direction based on a command bit from your controller. When a controlled stop is issued, the move will output a programmable number of steps before coming to a stop.
Assembled Move	Allows you to perform a sequence of relative moves. These moves can be performed with or without stopping between them.
Indexer Move	Allows you to program a move that is run when one of the programmable inputs makes a transition.
Hold Move	Allows you to suspend a move and restart it without losing your position value.
Resume Move	Allows you to restart a previously held move operation.
Immediate Stop	Allows you to immediately stop all motion if an error condition is detected by your host controller.

Table R1.1 Indexer Standard Functionality

Indexer Functionality (continued)

Synchronizing Moves

By default, the Networked Driver products use the SyncManager 2 event to control the transfer of data from the EtherCAT Slave Controller (ECS) to the microprocessor that controls motion. This allows the Networked Driver to execute commands as soon as new data arrives. When more than one axis is updated with a single EtherCAT packet, the time difference between axis updates is very short. Updating multiple axes with a single packet allows the EtherCAT network to out perform the update times of other industrial network protocols.

On very fast machines, or large machines that require more than one transfer to update all axes, the EtherCAT Distributed Clock (DC) functionality can be used to closely synchronize motion over multiple axes if using the SyncMaster2 event proves to be ineffective.

The EtherCAT Distributed Clock functionality is built into the ECS used by the Networked Driver products. The Networked Driver can act as the reference clock for the system if it is the first device in the EtherCAT network. The SYNC0 signal, which is based off of the Distributed Clock, can be use to synchronize the start of moves over multiple devices. The time between when the SYNC0 signal is received by the main processor of the Networked Driver and when the driver begins to cause motion is 520 ± 25 microseconds. The 520 microseconds is the time required to read the data from the ECS once the SYNC0 signal becomes active. The ±25 microseconds is caused by the 20 kHz update frequency of the PWM drivers.

For the Networked Driver, the minimum update time on the SYNC0 signal is two milliseconds. If the task time is less than two milliseconds, the SYNC0 time must be a multiple of the task time.

Encoder Functionality

Two of the three inputs available on the Networked Drive can be configured to accept AB quadrature signals from an incremental encoder. These signals are always decoded using X4 decoding. When configured this way, the indexer section of the Networked Driver has the following additional functionality.

Feature	Description
Stall Detection	When the Networked Driver is configured to accept an encoder, the encoder can be used to verify motion when a move command is issued.
Encoder Registration Move	Uses the count from a quadrature encoder to determine when a move begins to decelerate and stop. Commonly used in spooling/despooling applications.
Electronic Gearing	The Networked Driver can be configured to control the position of a motor based on feedback from an external encoder. The ratio of encoder pulses to motor pulses is full programmable and can be changed on-the-fly.

Table R1.2 Indexer Encoder Functionality

Stall Detection

When Stall Detection is enabled, the Networked Driver monitors the encoder for position changes, regardless of whether or not a move is in progress. If the error between the encoder position and the motor position exceeds forty-five degrees, the Networked Driver responds in the following manner:

- The stall is reported in the network input data.
- The motor position becomes invalid. (The machine must be homed or the motor position preset before Absolute moves can be run again.
- If a move was in progress, the move is stopped.

Note that a move does not have to be in progress for stall detection to be useful. By enabling stall detection, the Networked Driver can notify the system if the motor shaft moves more than forty-five degrees while the motor should be holding the load. For example, if the motor should hold a load against gravity, but the Idle Current Reduction parameter is set to high, the Networked Driver will issue a stall detection error if the load begins to fall.

Indexer Functionality (continued)

Encoder Functionality (continued)

Encoder Registration Move

The encoder position acts as a registration mark in an Encoder Registration Move. Typically, the encoder is mounted separately from the motor. With this mode, the Networked Driver will drive the motor until the desired encoder count is reached, at which time the motor begins to decelerate and stop. This functionality is useful in spool winding or unwinding applications, where the motor drives the spool and the encoder measures the material length as it enters or exits the spool. As the spool diameter increases or decreases, the programmed distance remains the same because you are measuring material, not the number of turns.

Electronic Gearing

In this mode, the stepper motor follows the rotation of an external encoder. This encoder is typically attached to another motor. The ratio of encoder pulses to stepper pulses is programmable over a wide range. This mode electronically couples the two motors together through a programmable gear ratio.

Driver Functionality

This table summarizes the features of the stepper motor driver portion of the SD17060E-K and SD31045E-K.

Feature	Benefits
170 Vdc or 310 Vdc Output Bus	A high voltage output bus means you can derive more high speed torque (power) from your stepper motor.
RMS Current Control	RMS current control give the Networked Driver the ability to drive the motor at its fully rated power when microstepping. Peak current controllers typically experience a 30% drop in power when microstepping a motor.
Programmable Motor Current	RMS current supplied to the motor can be programmed in 0.2 amp increments. This allows you to use a Networked Driver with the full line of AMCI stepper motors.
Programmable Idle Current Reduction	Extends motor life by reducing the motor current when not running. This extends the life of the motor by reducing its operating temperature.
Programmable Current Loop Gain	Allows you to tailor the driver circuitry to the motor’s impedances, thereby maximizing your motor’s performance.
Programmable Motor Steps/Turn	Allows you to scale your motor count to a real world value. (counts per inch, counts per degree, etc.)
Anti-Resonance Circuitry	This circuitry gives the Networked Driver the ability to modify motor current waveforms to compensate for mechanical resonance in your system. This will give you smooth performance over the entire speed range of the motor.
Motor Interlock	Safety feature that prevents power from being applied to the motor connector if a wire jumper does not exist between two of its pins. A *mechanical contact* can be used to disable the current to the motor. The voltage between the two interlock pins never exceeds 5Vdc, but 115/230Vac may be present between either of the Interlock pins and either of the power supply inputs.
Wiring Short Detection	Safety feature that removes power from the motor if a short is detected in one of the windings of the motor.
Over Temperature Detection	Protects the Networked Driver from damage by removing power from the motor if the internal temperature of the driver exceed a safe operating threshold.

Table R1.3 Driver Functionality

Driver Functionality (continued)

Idle Current Reduction

Idle Current Reduction allows you to prolong the life of your motor by reducing its idling temperature. Values for this parameter range from 0% (no holding torque when idle) to 100%.

Idle current reduction should be used whenever possible. By reducing the current, you are reducing the I^2R losses in the motor. Therefore, the temperature drop in the motor is exponential, not linear. This means that even a small reduction in the idle current can have a large effect on the temperature of the motor.

NOTE Note that the reduction values are "to" values, not "by" values. Setting a motor current to 4 Arms and the current reduction to 25% will result in an idle current of 1 Apk. (The Networked Driver always switches from RMS to peak current control when the motor is idle to prevent motor damage due to excessive heating.)

Current Loop Gain

This feature gives you the ability to adjust the gain of the power amplifiers in the Networked Driver to match the electrical characteristics of your motor. The value of this parameter can range from 1 to 40 with 40 representing the largest gain increase. In general, using a larger gain will increase high speed torque but the motor will run louder. A lower gain will offer quieter low speed operation at the cost of some high speed torque.

The use of this feature is completely optional and you can leave the Current Loop Gain at its default setting of "1" for standard motor performance.

Available Inputs

Each Networked Driver has three DC inputs that accept 3.5 to 27Vdc signals. (5 to 24Vdc nominal) They can be wired as differential, sinking, or sourcing inputs. How the Networked Driver uses these inputs is fully programmable as well as their active states. (Each input can be programmed to act as a Normally Open (NO) or Normally Closed (NC) input.)

Home Input

Many applications require that the machine be brought to a known position before normal operations can begin. This is commonly called "homing" the machine or bringing the machine to its "home" position. Each Networked Driver allows you to define this starting position in two ways. The first is with a Position Preset Command. The second is with a sensor mounted on the machine. When you define one of the inputs as the Home Input, you can issue commands to the Networked Driver that will cause the unit to seek this sensor. How the Networked Driver actually finds the home sensor is described in the reference chapter ***Homing an AMCI Networked Driver*** starting on page 51.

CW Limit Switch or CCW Limit Switch

Each input can be defined as a CW or CCW Limit Switch. When used this way, the inputs are used to define the limits of mechanical travel. For example, if you are moving in a clockwise direction and the CW Limit Switch activates, all motion will immediately stop. At this point, you will only be able to move in the counter-clockwise direction.

Start Indexer Move Input

Indexer Moves are programmed through the Network Data like every other move. The only difference is that Indexer Moves are not run until a Start Indexer Move Input makes a inactive-to-active state transition. This allows a Networked Driver to run critically timed moves that cannot be reliably started from the network due to data transfer lags.

If Inputs 1 and 2 are programmed as quadrature encoder inputs and Input 3 is programmed as a Start Indexer Move Input, then the quadrature encoder position data will be captured whenever Input 3 transitions. An inactive-to-active state transition will also trigger an Indexer Move if one is pending.

Available Inputs (continued)

Emergency Stop Input

When an input is defined as an Emergency Stop, or E-Stop Input, motion will immediately stop when this input becomes active. The driver remains enabled and power is supplied to the motor. No move can begin while this input is active.

Stop Jog or Registration Move Input

When an input is configured as a Stop Jog or Registration Move Input, triggering this input during a Jog Move or Registration will bring the move to a controlled stop. The controlled stop is triggered on an inactive-to-active state change on the input. Only Jog Moves and Registration Moves can be stopped with this input type, all other moves will ignore this input.

If Inputs 1 and 2 are programmed as quadrature encoder inputs and Input 3 is programmed as a Stop Jog or Registration Move Input, then the quadrature encoder position data will be captured when Input 3 makes an inactive-to-active transition to bring a Jog or Registration Move to a controlled stop. The encoder position data is not captured if a Jog or Registration Move is not in progress. If you want to capture encoder position data on every transition of Input 3, configure it as a Start Indexer Move Input.

Capture Encoder Position Input

As described in the ***Start Indexer Move Input*** and ***Stop Jog or Registration Move Input*** sections above, a Networked Driver can be configured to capture the encoder position value on a transition on Input 3.

Encoder Feedback

Each Networked Driver can be configured to accept a quadrature encoder instead of discrete inputs. Inputs 1 and 2 can be programmed to accept the ±A and ±B signals from a quadrature encoder. When using an encoder, you can wire the ±Z signal to Input 3. The ±Z signal can them be used to home the machine to the Z pulse if Input 3 is configured as a Home Input. You can also use the Z pulse to capture the encoder position on every completed turn by programming Input 3 as a Stop Jog or Registration Move Input or a Capture Encoder Position Input.

Additional information on the ways to use an encoder with a Networked Driver is described in the ***Encoder Functionality*** section on page 13.

General Purpose Input

If your application does not require all three inputs, you can configure the unused inputs as General Purpose Inputs. The inputs are not used by the Networked Driver, but the input state is reported in the network data.

Network Interface Description

The network interface is located on the top of the drive, towards the tabbed mounting surface. The network status LED's are also located in this area.

Ethernet Interface

The Ethernet Interface on the SD17060E-K and SD31045E-K drivers has two standard RJ-45 jacks that accept any standard 100Base-TX cable. Both ports are also auto MDI-X capable. This means that a standard cable can be used when connecting the Ethernet Driver to any device. Crossover cables are never needed.

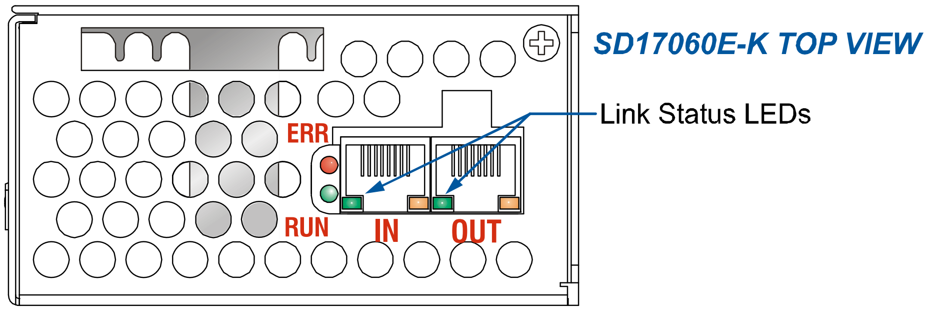


Figure R1.2 Ethernet Interface Location

The "IN" network jack in the diagram above must be attached to the upstream device in the EtherCAT network. (Closer to the network master.) The "OUT" jack must be attached to the next (downstream) device in the network of the Networked Driver is not the last device in the network.

Status LED's

Two status LED's are located on the front panel of the Networked Driver. EtherCAT status LED's are located on the top of the device.

Front Panel Status LED's

- **Power:** This green LED should be on when power is applied to the drive. If the LED is off, the input voltage is below 95Vac or one or both of the internal fuses are blown. The fuses are not field replaceable.

- **Status:**

Steady Green	Driver OK
Steady Red	Short Circuit Fault
	Over Temperature Fault
	Interlock Missing
Blinking Green ...	Successful write to Flash memory. Cycle power to the drive.
Blinking Red	Failure to write to Flash memory. Cycle power to the drive before attempting another write.

A Networked Driver will only detect motor errors when the motor current is enabled.

Status LED's (continued)

Top Panel Status LED's

There are two LED's on each of the RJ-45 jacks. The left LED is not used. The right LED shows the link status of the Ethernet connection. They are on when there is a physical connection between the Networked Driver and the previous or next device in the network. They blink when data is being transmitted over the network.

There are two additional LED's to the left of the jacks. The green RUN LED indicates the logical state of the device. The red ERR LED indicates an error state in the EtherCAT protocol.

EtherCAT Run LED

The green RUN LED indicates the logical state of the device.

LED State	Description
Off	Device in the EtherCAT Init state
Fast Blink (4 Hz)	Device in the EtherCAT Pre-Operational (Pre-Op) state
Slow Blink (1 Hz)	Device in the EtherCAT Safe-Operational (Safe-Op) state
Steady Green	Device in the EtherCAT Operational (Op) state.

Table R1.4 Module RUN LED States

EtherCAT ERR LED

The ERR LED location houses a red LED that indicates an error state in the EtherCAT protocol.

Red LED State	Description
Off	No errors in device operation
Single blink 200 ms ON / 1 s OFF	Problems with synchronization such as with the Distributed Clock (DC) PLL.
Double Blink 200 ms Pulses / 1 s between	SYNC manager watchdog timeout. The master has not updated the output data in the configured time interval. The Networked Driver has switched to the Safe-OP state.
Slow Blink (1 Hz)	All other issues, such as cable disconnect. The Networked Driver has switched to the Safe-OP state.

Table R1.5 Red ERR LED States

UL/cUL Recognized Installations

The SD17060E-K and SD31045E-K drivers are Underwriter Laboratory Inc.® listed devices. It is listed as "Industrial Control Equipment" under the control number 60GB. The UL file number is E231137. These units are appropriate for UL and cUL applications.

UL Required Information

If your installation is to meet UL requirements, you must be aware of the following information when using the SD17060E-K or SD31045E-K.

- Maximum surrounding air temperature is +50°C
- The driver does not incorporate internal motor overload protection.
- The driver does not provide motor over temperature protection.
- The driver does not provide overspeed protection.
- The driver shall be used in pollution degree 1 or 2 environments. If the driver is mounted in an enclosure, this enclosure must meet these requirements.
- All wiring to the driver shall be R/C (AVLV2), minimum rating of 80°C, 300V, except secondary low-voltage circuit wiring.
- Use 75°C copper conductors only.
- Terminals shall be tighten to manufacturer's recommended torques
- Power Connector shall be rated for a minimum 12A, 600V, in a pollution degree 2 environment†
- Motor Connector shall be rated for a minimum 16A, 600V, in a pollution degree 2 environment†
- I/O Connector shall be rated for a minimum 8A, 300V, in a pollution degree 2 environment†

† The mating connectors supplied with these units meet these requirements. A complete connector kit, including all connectors and protective boots, can be ordered from AMCI under the part number *AK-DRIVES*.

Additional mating connectors can be ordered directly from Phoenix Contact under the following part numbers:

Location	Phoenix Contact Part Number
Power	1737822
Motor	1912919
I/O	1803633

Table R2.1 Mating Connectors

Notes

REFERENCE 3
MOTION CONTROL

When a move command is sent to a Networked Driver, the unit calculates the entire profile before starting the move or issuing an error message. This chapter explains how the profiles are calculated and the different available moves.

Definitions

Units of Measure

Distance: Every distance is measured in steps. When you configure the unit, you will specify the number of steps you want to complete one rotation of the motor shaft. It is up to you to determine how many steps are required to travel the appropriate distance in your application.

Speed: All speeds are measured in steps/second. Since the number of steps needed to complete one shaft rotation is determined by your programming, it is up to you to determine how many steps per second is required to rotate the motor shaft at your desired speed.

Acceleration: The typical unit of measure for acceleration and deceleration is steps/second/second, or steps/second^2. However, when programming a Networked Driver, all acceleration and deceleration values must be programmed in the unit of measure of steps/second/millisecond.

- To convert from steps/second^2 to steps/second/millisecond, divide the value by 1000. This must be done when converting from a value used in the equations to a value programmed into a Networked Driver.
- To convert from steps/second/millisecond to steps/second^2, multiply the value by 1000. This must be done when converting from the value programmed into a Networked Driver to the value used in the equations.

Motor Position

Motor Position is defined in counts. The range is -2,147,483,648 to +2,147,483,647.

Home Position

The Home Position is any position on your machine that you can sense and stop at. There are two ways to defining the Home Position. The first is using the Preset Position command to set the Motor Position register to a known value. The second method is using one of the *Find Home* commands. If you use the unit's *Find Home* commands, the motor position and encoder position registers will automatically be set to zero once the home position is reached. Defining a Home Position is completely optional. Some applications, such as those that use the Networked Driver for speed control, don't require position data at all.

Count Direction

Clockwise moves will always increase the motor position register reported back to the host. Some of the moves, such as the Jog Move, have a positive and negative command. A positive command, such as the +Jog Move command, will result in a clockwise rotation of the shaft.

Starting Speed

The Starting Speed is the speed that most moves will begin and end at. This value is set while configuring the unit and it has a valid range of 1 to 1,999,999 steps/second. This value is typically used to start the move above the motor's low frequency resonances and, in micro-stepping applications, to limit the amount of time needed for acceleration and deceleration. AMCI does not specify a default value in this manual because it is very dependent on motor size and attached load.

Definitions (continued)

Target Position

The Target Position is the position that you want the move to end at. There are two ways to define the Target Position, with relative coordinates or absolute coordinates.

Relative Coordinates

Relative coordinates define the Target Position as an offset from the present position of the motor. Most moves use relative coordinates.

- The range of values for the Target Position when it is treated as an offset is ±8,388,607 counts. Positive offsets will result in clockwise moves, while negative offsets result in counter-clockwise moves.
- The Motor Position value reported back to the host can exceed ±8,388,607 counts. The only way to move beyond ±8,388,607 counts is with multiple relative moves or jog commands.

Absolute Coordinates

Absolute coordinates treat the Target Position as an actual position on the machine. Note that you must set the Home Position on the machine before you can run an Absolute Move. (See ***Home Position*** on the previous page.)

- The range of values for the Target Position when it is treated as an actual position on the machine is ±8,388,607 counts. The move will be clockwise if the Target Position is greater than the Current Position and negative if the Target Position is less than the Current Position.
- The Motor Position value reported back to the host can exceed ±8,388,607 counts. However, you cannot move beyond ±8,388,607 counts with an Absolute Move. The only way to move beyond ±8,388,607 counts is with multiple relative moves or jog commands.

Definition of Acceleration Types

With the exception of Registration Moves, all move commands, including homing commands, allow you to define the acceleration type used during the move. The Networked Driver supports three types of accelerations and decelerations. The type of acceleration used is controlled by the *Acceleration Jerk* parameter.

Detailed move profile calculations, including the effect of the *Acceleration Jerk* parameter, can be found in the reference section, ***Calculating Move Profiles***, starting on page 41.

Linear Acceleration

When the Acceleration Jerk parameter equals zero, the axis accelerates (or decelerates) at a constant rate until the programmed speed is reached. This offers the fastest acceleration, but consideration must be given to insure the smoothest transition from rest to the acceleration phase of the move. The smoothest transition occurs when the configured Starting Speed is equal to the square root of the programmed Linear Acceleration. Note that other values will work correctly, but you may notice a quick change in velocity at the beginning of the acceleration phase.

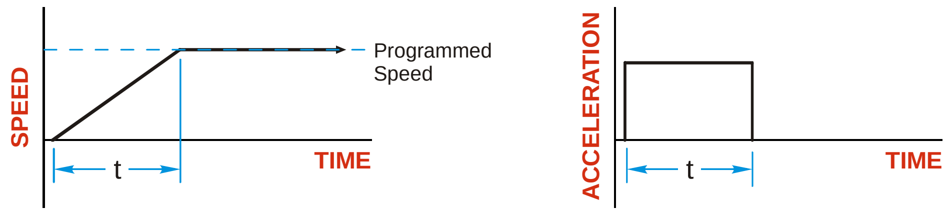


Figure R3.1 Linear Acceleration

Definition of Acceleration Types (continued)

Triangular S-Curve Acceleration

When the Acceleration Jerk parameter equals one, the axis accelerates (or decelerates) at a constantly changing rate that is slowest at the beginning and end of the acceleration phase of the move. The Triangular S-Curve type offers the smoothest acceleration, but it takes twice as long as a Linear Acceleration to achieve the same velocity.

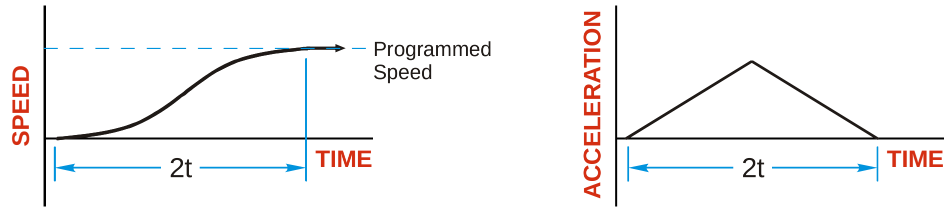


Figure R3.2 Triangular S-Curve Acceleration

Trapezoidal S-Curve Acceleration

When the Acceleration Jerk parameter is in the range of 2 to 5,000, Trapezoidal S-Curve acceleration is used. The Trapezoidal S-Curve acceleration is a good compromise between the speed of Linear acceleration and the smoothness of Triangular S-Curve acceleration. Like the Triangular S-Curve, this acceleration type begins and ends the acceleration phase smoothly, but the middle of the acceleration phase is linear. Figure R3.3 shows a trapezoidal curve when the linear acceleration phase is half of the total acceleration time. With this setting, the Trapezoidal S-Curve acceleration only requires 33% more time to achieve the same velocity as a Linear Acceleration.

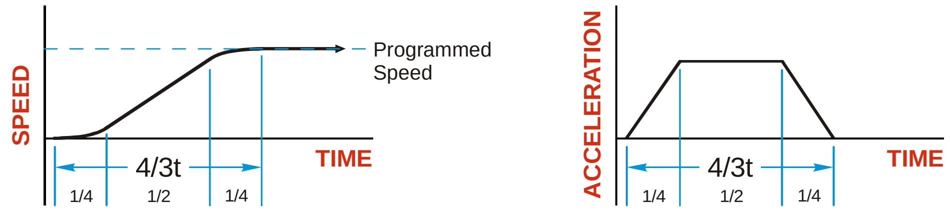


Figure R3.3 Trapezoidal S-Curve Acceleration

A Simple Move

As shown in the figure below, a move from A (Current Position) to B (Target Position) consists of several parts.

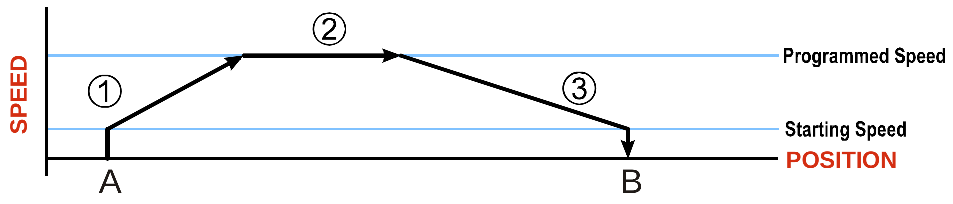


Figure R3.4 A Trapezoidal Profile

1)The move begins at point A, where the motor jumps from rest to the configured *Starting Speed*. The motor then accelerates at the programmed *Acceleration Value* until the speed of the motor reaches the *Programmed Speed*. Both the Acceleration Value and the Programmed Speed are programmed when the move command is sent to the Networked Driver.

2)The motor continues to run at the Programmed Speed until it reaches the point where it must decelerate before reaching point B.

3)The motor decelerates at the *Deceleration Value*, which is also programmed by the move command, until the speed reaches the Starting Speed, which occurs at the Target Position (B). The motor stops at this point. Note that the acceleration and deceleration values can be different in the move.

Figure R3.4 above shows a Trapezoidal Profile. A Trapezoidal Profile occurs when the Programmed Speed is reached during the move. This occurs when the number of steps needed to accelerate and decelerate are less than the total number of steps in the move.

Figure R3.5 below shows a Triangular Profile. A Triangular Profile occurs when the number of steps needed to accelerate to the Programmed Speed and decelerate from the Programmed Speed are greater than the total number of steps in the move. In this case, the profile will accelerate as far as it can before it has to decelerate to reach the Starting Speed at the Target Position. The Programmed Speed is never reached.

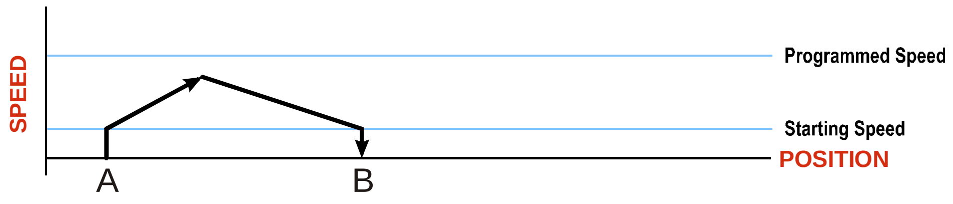


Figure R3.5 A Triangular Profile

Controlled and Immediate Stops

Once a move is started, there are several ways to stop the move before it comes to an end. These stops are broken down into two types:

- **Controlled Stop:** The axis immediately begins decelerating at the move's programmed deceleration value until it reaches the configured Starting Speed. The axis stops at this point. The motor position value is still considered valid after a Controlled Stop and the machine does not need to be homed again before Absolute Moves can be run.
- **Immediate Stop:** The axis immediately stops motion regardless of the speed the motor is running at. Since it is possible for the inertia of the load attached to the motor to pull the motor beyond the stopping point, the motor position value is considered invalid after an Immediate Stop. The machine must be homed or the position must be preset before Absolute Moves can be run again.

Host Control

Hold Move Command: This command can be used with some moves to bring the axis to a Controlled Stop. The move can be resumed and finished, or it can be aborted. Not all moves are affected by this command. The section ***Basic Move Types***, starting on page 25, describes each move type in detail, including if the move is affected by this command.

Immediate Stop Command: When this command is issued from the host, the axis will come to an Immediate Stop. The move cannot be restarted and the machine must be homed again before Absolute Moves can be run.

Hardware Control

CW Limit and CCW Limit Inputs: In most cases, activating these inputs during a move will bring the axis to an Immediate Stop. The exceptions are the *CW/CCW Find Home* commands, the *CW/CCW Jog Move* commands, and the *CW/CCW Registration Move* commands. The *Find Home* commands are explained in the reference section, ***Homing an AMCI Networked Driver***, which starts on page 51. The ***CW/CCW Jog Move*** commands are fully explained on page 27, and the ***CW/CCW Registration Move*** commands are fully explained on page 29.

Emergency Stop Input: It is possible to configure an input as an Emergency Stop Input. When an Emergency Stop Input is activated, the axis will come to an Immediate Stop, regardless of the direction of travel. Note that power is not removed from the motor.

Basic Move Types

Relative Move

Relative Moves move an offset number of steps (n) from the Current Position (A). A trapezoidal profile is shown to the right, but Relative Moves can also generate triangular profiles. The command's Target Position is the move's offset. The offset can be in the range of ±8,388,607 counts. Positive offsets will result in clockwise moves, while negative offsets result in counter-clockwise moves.

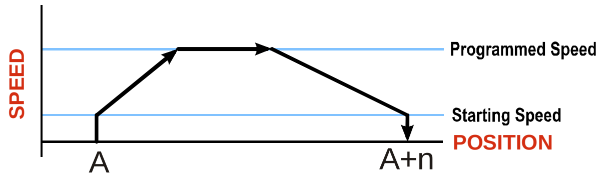


Figure R3.6 Relative Move

NOTE

1) You do not have to preset the position or home the machine before you can use a Relative Moves. That is, the Position_Invalid status bit can be set.

2) Relative Moves allow you to move your machine without having to calculate absolute positions. If you are indexing a rotary table, you can preform a relative move of 30° multiple times without recalculating new target positions in your controller. If you perform the same action with Absolute Moves, you would have to calculate your 30° position followed by your 60° position, followed by your 90° position, etc.

Basic Move Types (continued)

Relative Move (continued)

Relative Moves can be brought to a Controlled Stop by using the Hold Move Command from your host controller. When the command is accepted, the axis will immediately decelerate at the programmed rate and stop. When stopped successfully, the Networked Driver will set a *In_Hold_State* bit in the input data table. The Relative Move can be restarted with the Resume Move command from the host controller or the move can be aborted by starting another move. The Resume Move command allows you to change the move's Programmed Speed, Acceleration Value and Type, and the Deceleration Value and Type. The Target Position cannot be changed with the Resume Move Command.

Controlled Stop Conditions

- The move completes without error.
- You toggle the Hold_Move control bit in the Network Output Data. Note that your holding position will most likely not be the final position you commanded. You can resume a held Relative Move by using the Resume Move command. The use of the Hold_Move and Resume_Move bits is further explained in the ***Controlling Moves In Progress*** section starting on page 37.

Immediate Stop Conditions

- The Immediate Stop bit makes a 0→1 transition in the Network Output Data.
- An inactive-to-active transition on an input configured as an E-Stop Input.
- A CW or CCW Limit Switch is reached. If the limit that is reached is the same as the direction of travel, for example, hitting the CW limit while running a CW move, a *Reset Errors* command must be issued before moves are allowed in that direction again. If the limit that is reached is opposite the direction of travel, a *Reset Errors* command does not have to be issued.

Absolute Move

Absolute Moves move from the Current Position (A) to a given position (B). (The Networked Driver calculates the direction and number of steps needed to move to the given position and moves that number of steps.) A trapezoidal profile is shown to the right, but Absolute Moves can also generate triangular profiles. The command's Target Position must be in the range of ±8,388,607 counts. The move will be clockwise if the Target Position is greater than the Current Position and counter-clockwise if the Target Position is less than the Current Position.

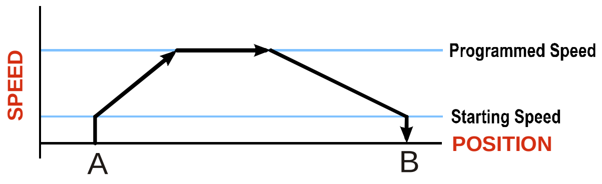


Figure R3.7 Absolute Move

NOTE

1) The *Home Position* of the machine must be set before running an Absolute Move. See the reference section, ***Homing an AMCI Networked Driver***, which starts on page 51, for information on homing the machine.

2) The Motor Position must be valid before you can use an Absolute Move. The Motor Position becomes valid when you preset the position or home the machine.

3) Absolute Moves allow you to move your machine without having to calculate relative positions. If you are controlling a rotary table, you can drive the table to any angle without having to calculate the distance to travel. For example an Absolute Move to 180° will move the table to the correct position regardless of where the move starts from.

Basic Move Types (continued)

Absolute Move (continued)

Controlled Stop Conditions

- The move completes without error.
- You toggle the Hold_Move control bit in the Network Output Data. Note that your holding position will most likely not be the final position you commanded. You can resume a held Absolute Move by using the Resume_Move bit or the move can be aborted by starting another move. The use of the Hold_Move and Resume_Move bits is explained in the ***Controlling Moves In Progress*** section starting on page 37.

Immediate Stop Conditions

- The Immediate Stop bit makes a 0→1 transition in the Network Output Data.
- An inactive-to-active transition on an input configured as an E-Stop Input.
- A CW or CCW Limit Switch is reached. If the limit that is reached is the same as the direction of travel, for example, hitting the CW limit while running a CW move, a *Reset Errors* command must be issued before moves are allowed in that direction again. If the limit that is reached is opposite the direction of travel, a *Reset Errors* command does not have to be issued.

CW/CCW Jog Move

Jog Moves move in the programmed direction as long as the command is active. Two commands are available. The CW Jog Move will increase the motor position count while the CCW Jog Move will decrease the motor position count. These commands are often used to give the operator manual control over the axis.

Jog Moves are also used when you are interested in controlling the speed of the shaft instead of its position. One such application is driving a conveyor belt. To accommodate these applications, the running speed, acceleration, and deceleration of the Jog Move can be changed *while the move is in progress*.

The CW Limit and CCW Limit inputs behave differently for CW/CCW Jog Moves and CW/CCW Registration Moves than all other move types. Like all moves, activating a limit will bring the move to an Immediate Stop. Unlike other moves, a Jog or Registration move can be started when an end limit switch is active provided that the commanded direction is opposite that of the activated switch. For example, a CW Jog Move can be issued while the CCW limit switch is active. This allows you to move off of an activated end limit switch.

Basic Move Types (continued)

CW/CCW Jog Move (continued)

As shown below, a Jog Moves begins at the programmed Starting Speed, accelerates at the programmed rate to the Programmed Speed and continues until a stop condition occurs. If it is a *Controlled Stop Condition*, the Networked Driver will decelerate the motor to the starting speed and stop without losing position. If it is an *Immediate Stop Condition*, the motion stops immediately and the position becomes invalid.

It is possible to change the speed of a Jog Move without stopping the motion. The Programmed Speed, Acceleration, and Deceleration parameters can be changed during a Jog Move. When the Programmed Speed is changed, the motor will accelerate or decelerate to the new Programmed Speed using the new accelerate/decelerate parameter values. If you write a Programmed Speed to the unit that is less than the starting speed, the Jog Move will continue at the previously programmed speed.

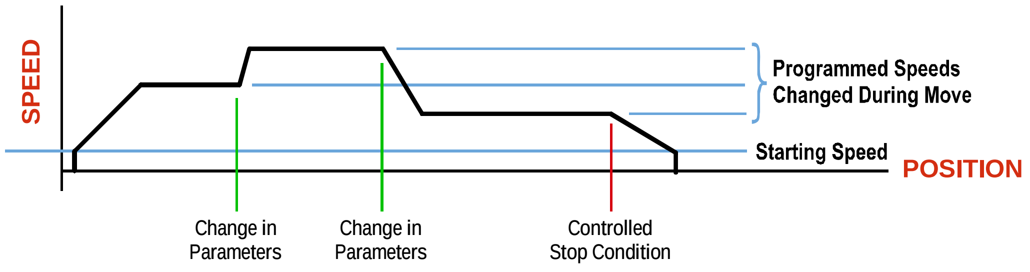


Figure R3.8 Jog Move

Controlled Stop Conditions

- The Jog Move Command bit is reset to “0”.
- An inactive-to-active transition on an input configured as a *Stop Jog or Registration Move* Input.
- You toggle the Hold_Move control bit in the Network Output Data. The use of the Hold_Move and Resume_Move bits is explained in the ***Controlling Moves In Progress*** section starting on page 37.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Output Data.
- A inactive-to-active transition on an input configured as an E-Stop Input.
- A CW or CCW Limit Switch is reached. If the limit that is reached is the same as the direction of travel, for example, hitting the CW limit while running a CW move, a *Reset Errors* command must be issued before moves are allowed in that direction again. If the limit that is reached is opposite the direction of travel, a *Reset Errors* command does not have to be issued.

NOTE Note that it is possible to *start* a move while a CW or CCW Limit Switch is active as long as the direction of travel is *opposite* that of the activated Limit Switch. For example, it is possible to start a CW Jog Move while the CCW Limit Switch is active.

Basic Move Types (continued)

CW/CCW Registration Move

Similar to a Jog Move, a Registration Move will travel in the programmed direction as long as the command is active. CW Registration Moves increase the motor position count while the CCW Registration Moves decrease the motor position count. When the command terminates under Controlled Stop conditions, the Networked Driver will output a programmed number of steps as part of bringing the move to a stop. Note that all position values programmed with a Registration Move are relative values, not absolute machine positions.

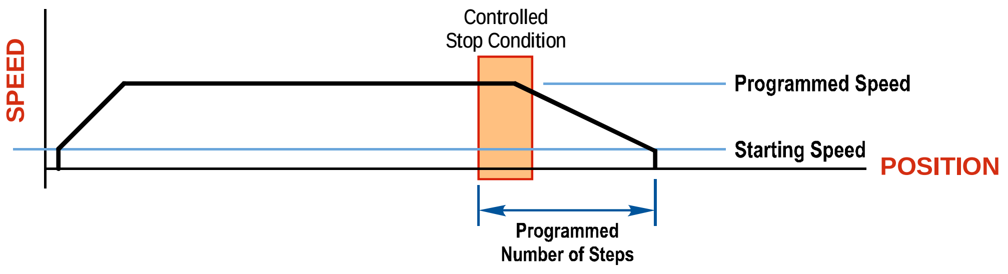


Figure R3.9 Registration Move

NOTE If the Programmed Number of Steps are less than the number of steps needed to bring the axis to a stop based on the Programmed Speed and Deceleration values set with the command, the Networked Driver will decelerate at the programmed Deceleration value until it has output the Programmed Number of Steps and then stop the move without further deceleration.

An additional feature of the Registration Moves is the ability to program the driver to ignore the Controlled Stop conditions until a minimum number of steps have occurred. This value is programmed through the Minimum Registration Move Distance parameter, which is set when you command the Registration Move. The figure below shows how the Minimum Registration Move Distance parameter affects when the Stop Condition is applied to the move. As shown in the second diagram, Controlled Stop conditions are level triggered, not edge triggered. If a Controlled Stop Condition occurs before the Minimum Registration Move Distance is reached and stays active, the move will begin its controlled stop once the Minimum Registration Move Distance is reached.

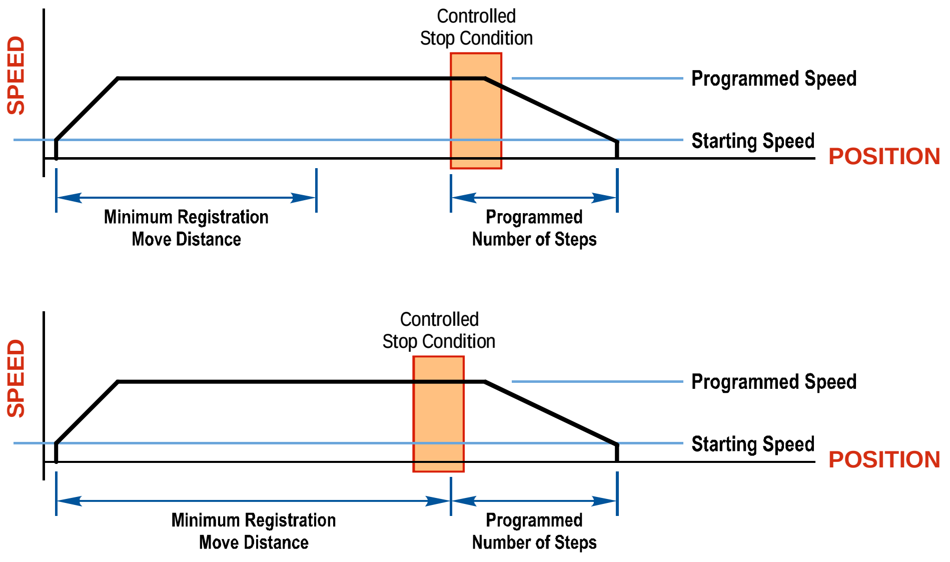


Figure R3.10 Min. Registration Move Distance

Basic Move Types (continued)

CW/CCW Registration Move (continued)

Controlled Stop Conditions

- The Registration Move Command bit is reset to “0”.
- A positive transition on an input configured as a *Stop Jog or Registration Move* Input.

  NOTE Starting a Registration Move with a *Stop Jog or Registration Move* Input in its active state will result in a move of (*Minimum Registration Distance* + *Programmed Number of Steps*).

- You toggle the Hold_Move control bit in the Network Output Data. The Networked Driver responds by using the programmed Deceleration value to bring the move to a stop, without using the value of the Programmed Number of Steps parameter. A Registration Move does not go into the Hold State if the Hold_Move control bit is used to stop the move and it cannot be restarted.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Output Data.
- An inactive-to-active transition on an input configured as an E-Stop Input.
- A CW or CCW Limit Switch is reached. If the limit that is reached is the same as the direction of travel, for example, hitting the CW limit while running a CW move, a *Reset Errors* command must be issued before moves are allowed in that direction again. If the limit that is reached is opposite the direction of travel, a *Reset Errors* command does not have to be issued.

NOTE Note that it is possible to *start* a move while a CW or CCW Limit Switch is active as long as the direction of travel is *opposite* that of the activated Limit Switch. For example, it is possible to start a CW Registration Move while the CCW Limit Switch is active.

Encoder Registration Move

When the Networked Driver is configured to use a quadrature encoder, the position value from the encoder can be used as a registration mark. Once the encoder count is reached, the move will begin to decelerate and stop when the move slows to the configured Starting Speed. Absolute and relative type move types are supported. When programming an absolute move, the target position defines the encoder position that must be reached before the move begins to decelerate and stop. When programming a relative move, the sign of the target position determines the direction of the move and its magnitude determines the distance traveled.

NOTE
1) The encoder is typically mounted separate from the motor. For example, the encoder can measure the length of a flexible and/or stretchable material, or material on a spool. The motor is used to drive the material for the length measured by the encoder, regardless of the slipping, flexing, or stretching of the material as it is driven by the motor.
2) This is an open-loop move. The encoder does not provide feedback in a closed-loop, servo control fashion. The encoder position is used to decide when the move deceleration begins, not the final position of the move.
3) You do not have to preset the position or home the machine before you can use a relative Encoder Registration Move.

Basic Move Types (continued)

Encoder Registration Move (continued)

The figure below represents either a relative Encoder Registration Move of 11,000 counts or an absolute Encoder Registration Move to position 16,000. The figure shows that the encoder position you program in the move defines the point at which the motor begins to decelerate and stop. *It does not define the stopping position as it does in other move types.* The endpoint of the move depends on the speed of the motor when the programmed encoder position is reached and the deceleration values. This behavior is different from Absolute and Relative Moves where the position you program into the move is the end point of the move.

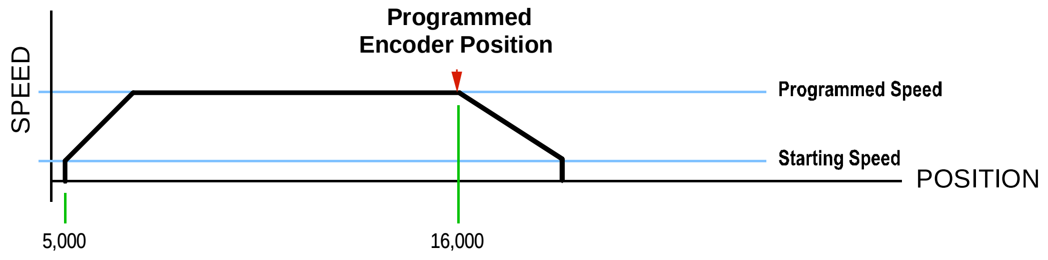


Figure R3.11 Encoder Registration Move

Controlled Stop Conditions

- The move completes without error
- You toggle the Hold_Move control bit in the Network Output Data. Note that your holding position will most likely not be the final position you commanded. You can resume a held Encoder Registration Move by using the Resume_Move bit. The use of the Hold_Move and Resume_Move bits is explained in the ***Controlling Moves In Progress*** section starting on page 37.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Output Data.
- A positive transition on an input configured as an E-Stop Input.
- A CW/CCW Limit Switch is reached. If the limit that is reached is the same as the direction of travel, for example, hitting the CW limit while running a CW move, a *Reset Errors* command must be issued before moves are allowed in that direction again. If the limit that is reached is opposite the direction of travel, a *Reset Errors* command does not have to be issued.

Assembled Moves

All of the moves explained so far must be run individually to their completion or must be stopped before another move can begin. The Networked Driver also gives you the ability to pre-assemble more complex profiles from a series of relative moves that are then run with a single command. Each Assembled Move can consist of 2 to 16 segments. Assembled Moves are programmed through a hand shaking protocol that uses the input and output registers assigned to the unit. The protocol is fully described in the ***Assembled Move Programming*** section on page 35.

Two types of Assembled Moves exist in a Networked Driver:

- **Blend Move -** A Blend Move gives you the ability to string multiple relative moves together and run all of them sequentially without stopping the shaft between moves. A Blend Move can be run in either direction, and the direction is set when the move command is issued.
- **Dwell Move -** A Dwell Move gives you the ability to string multiple relative moves together, and the Networked Driver will stop between each move for a programed *Dwell Time*. Because motion stops between each segment, a Dwell Move allows you to reverse direction during the move.

Assembled Moves (continued)

Blend Move

Each Relative Move defines a *segment* of the Blend Move. The following restrictions apply when programming Blend Moves.

1)Each segment of the Blend Move must be written to the Networked Driver before the move can be initiated.

- The Networked Driver supports Blend Moves with up to sixteen segments.

2)Each segment is programmed as a relative move. Blend Moves cannot be programmed with absolute coordinates.

3)All segments run in the same direction. The sign of the target position is ignored and only the magnitude of the target position is used. The move's direction is controlled by the bit pattern used to start the move. If you want to reverse direction during your move, consider using the ***Dwell Move*** which is explained starting on page 33.

4)The Programmed Speed of each segment must be greater than or equal to the Starting Speed.

5)The Programmed Speed can be the same between segments. This allows you to chain two segments together.

6)For all segments except for the last one, the programmed position defines the end of the segment. For the last segment, the programmed position defines the end of the move.

7)Once you enter a segment, that segment's programmed acceleration and deceleration values are used to change the speed of the motor.

8)The blend segment must be long enough for the acceleration or deceleration portions of the segment to occur. If the segment is not long enough, the motor speed will jump to the speed of the next segment without acceleration or deceleration.

The figure below shows a three segment Blend Move that is run twice. It is first run in the clockwise direction, and then in the counter-clockwise direction.

NOTE The deceleration value programmed with segment 3 is used twice in the segment. Once to decelerate from the Programmed Speed of segment 2 and once again to decelerate at the end of the move.

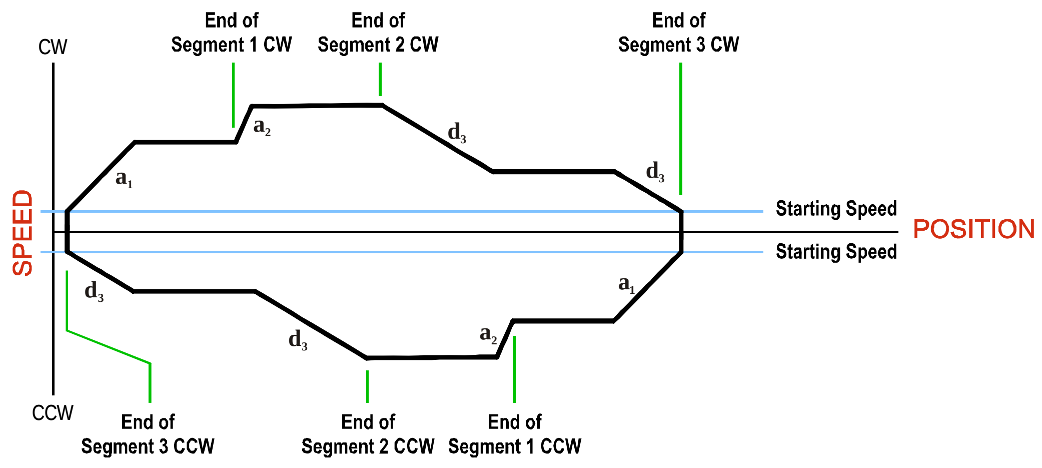


Figure R3.12 Blend Move

Assembled Moves (continued)

Blend Move (continued)

NOTE

1) You do not have to preset the position or home the machine before you can use a Blend Move. Because the Blend Move is based on Relative Moves, it can be run from any location.
2) The Blend Move is stored in the internal memory of the Networked Driver and can be run multiple times once it is written to the unit. The Blend Move data stays in memory until power is removed, the unit is sent new Configuration Data, or a new Blend or Dwell Move is written to the unit. As described in ***Saving an Assembled Move in Flash*** on page 36, it is also possible to save a Blend Move to flash memory. This move is restored on power up and can be run as soon as you configure the Networked Driver and enter Command Mode.
3) There are two control bits used to specify which direction the Blend Move is run in. This gives you the ability to run the Blend Move in either direction.

Controlled Stop Conditions

- The move completes without error.
- You toggle the Hold_Move control bit in the Network Output Data. When this occurs, the Networked Driver decelerates the move at the deceleration rate of the present segment to the Starting Speed and ends the move. Note that your final position will most likely not be the one you commanded. A Blend Move that is brought to a controlled stop with the Hold_Move bit cannot be restarted. The use of the Hold_Move bit is explained in the ***Controlling Moves In Progress*** section starting on page 37.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Output Data.
- A positive transition on an input configured as an E-Stop Input.
- A CW or CCW Limit Switch is reached. If the limit that is reached is the same as the direction of travel, for example, hitting the CW limit while running a CW move, a *Reset Errors* command must be issued before moves are allowed in that direction again. If the limit that is reached is opposite the direction of travel, a *Reset Errors* command does not have to be issued.

Dwell Move

A Dwell Move gives you the ability to string multiple relative moves together and run all of them sequentially with a single start condition. Like a Blend Move, a Dwell Move is programmed into a Networked Driver as a series of relative moves before the move is started.

Unlike a Blend Move, the motor is stopped between each segment of the Dwell Move for a programed *Dwell Time*. The Dwell Time is programmed as part of the command that starts the move. The Dwell Time is the same for all segments. Because the motor is stopped between segments, the motor direction can be reversed during the move. The sign of the target position for the segment determines the direction of motion for that segment. Positive segments will result in clockwise shaft rotation while a negative segment will result in a counter-clockwise shaft rotation.

Assembled Moves (continued)

Dwell Move (continued)

The following figure shows a drilling profile that enters the part in stages and reverses direction during the drilling operation so chips can be relieved from the bit. You *could* accomplish this Dwell Move with a series of six relative moves that are sent down to the Networked Driver sequentially. The two advantages of a Dwell Move in this case are that the Networked Driver will be more accurate with the Dwell Time then you can be in your control program, and Dwell Moves simplify your program's logic.

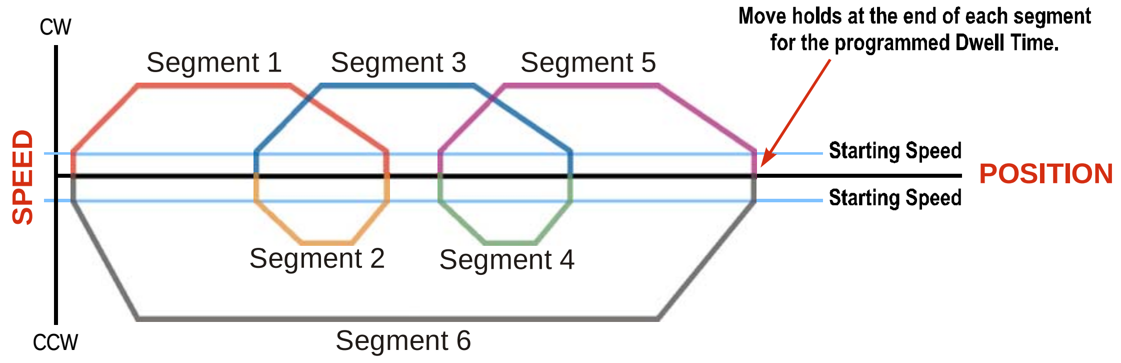


Figure R3.13 Dwell Move

NOTE

1) You do not have to preset the position or home the machine before you using a Dwell Move. Because the Dwell Move is based on Relative Moves, it can be run from any location.

2) The Dwell Move is stored in the internal memory of the Networked Driver and can be run multiple times once it is written to the unit. The Dwell Move data stays in memory until power is removed, the unit is sent new Configuration Data, or a new Blend or Dwell Move is written to the Networked Driver. As described in ***Saving an Assembled Move in Flash*** on page 36, it is also possible to save a Dwell Move to flash memory. This move is restored on power up and can be run as soon as you configure your Networked Driver and enter Command Mode.

Controlled Stop Conditions

- The move completes without error.
- You toggle the Hold_Move control bit in the Network Output Data. When this occurs, the Networked Driver decelerates the move at the deceleration rate of the present segment to the Starting Speed and ends the move. Note that your final position will most likely not be the one you commanded. A Dwell Move that is brought to a controlled stop with the Hold_Move bit cannot be restarted.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Output Data.
- A positive transition on an input configured as an E-Stop Input.
- A CW or CCW Limit Switch is reached. If the limit that is reached is the same as the direction of travel, for example, hitting the CW limit while running a CW move, a *Reset Errors* command must be issued before moves are allowed in that direction again. If the limit that is reached is opposite the direction of travel, a *Reset Errors* command does not have to be issued.

Assembled Move Programming

All of the segments in a Blend or Dwell Move must be written to the Networked Driver before the move can be run. Segment programming is controlled with two bits in the Network Output Data and two bits in the Network Input Data. Blend and Dwell Moves are programmed in exactly the same way. When you start the move, a bit in the command data determines which type of Assembled Move is run. In the case of a Blend Move, the signs of the segment's Target Positions are ignored and all segments are run in the same direction. In the case of a Dwell Move, the signs of the segment's Target Positions determine the direction of the segment. For Dwell Moves, the Dwell Time is sent to the Networked Driver as part of the command.

Control Bits – Output Data

- **Program_Assembled bit –** A 0→1 transition on this bit tells the Networked Driver that you want to program a Blend or Dwell Move Profile. The Networked Driver will respond by setting the *In_Assembled_Mode* bit in the Network Input Data. At the beginning of the programming cycle, the Networked Driver will also set the *Waiting_For_Assembled_Segment* bit to signify that it is ready for the first segment.
- **Read_Assembled_Data bit –** A 0→1 transition on this bit tells the Networked Driver that the data for the next segment is available in the remaining data words.

Control Bits – Input Data

- **In_Assembled_Mode bit –** The Networked Driver sets this bit to tell you that it is ready to accept segment programming data in the remaining output data words. The actual transfer of segment data is controlled by the *Waiting_For_Assembled_Segment* and *Read_Assembled_Data* bits.
- **Waiting_For_Assembled_Segment bit –** A 0→1 transition on this bit from the Networked Driver is the signal to the host that the Networked Driver is ready to accept the data for the next segment.

Programming Routine

1) The host sets the *Program_Assembled* bit in the Network Output Data.
2) The Networked Driver responds by setting both the *In_Assembled_Mode* and *Waiting_For_Assembled_Segment* bits in the Network Input Data.
3) When the host detects that the *Waiting_For_Assembled_Segment* bit is set, it writes the data for the first segment in the Network Output Data and sets the *Read_Assembled_Data* bit.
4) The Networked Driver checks the data, and when finished, resets the *Waiting_For_Assembled_Segment* bit. If an error is detected, it also sets the *Command_Error* bit.
5) When the host detects that the *Waiting_For_Assembled_Segment* bit is reset, it resets the *Read_Assembled_Data* bit.
6) The Networked Driver detects that the *Read_Assembled_Data* bit is reset, and sets the *Waiting_For_Assembled_Segment* bit to signal that it is ready to accept data for the next segment.
7) Steps 3 to 6 are repeated for the remaining segments until the entire move profile has been entered. The maximum number of segments per profile is sixteen.
8) After the last segment has been transferred, the host exits Assembled Move Programming Mode by resetting the *Program_Assembled* bit.
9) The Networked Driver resets the *In_Assembled_Mode* and the *Waiting_For_Assembled_Segment* bits.

Assembled Move Programming (continued)

Saving an Assembled Move in Flash

The Networked Driver also contains the *Save_Assembled_to_Flash* bit that allows you to store the Assembled Move in flash memory. This allows you to run the Assembled Move right after power up, without having to go through a programming sequence first. To use this bit, you follow the above programming routine with the *Save_Assembled_to_Flash* bit set. When you reach step 9 in the sequence, the Networked Driver responds by resetting the *In_Assembled_Mode* and *Transmit Blend Move Segments* bits as usual and then flashes the Error LED. If the LED flashes green at 4 Hz (fast blink), the write to flash memory was successful. If it flashes green at 1 Hz (slow blink), then there was an error in writing the data. In either case, power must be cycled to the Networked Driver before you can continue. With a limit of 10,000 write cycles, the design decision that requires you to cycle power to the Networked Driver was made to prevent an application from damaging the module by continuously writing to it.

Indexed Moves

All of the moves that have been explained in the chapter up to this point can be started by a transition on one of the inputs instead of a command from the network. If the *Indexed Move* bit is set when the command is issued, the Networked Driver will not run the move until the configured input makes an inactive-to-active transition. This allows you to run time critical moves that cannot be reliably started from the network because of messaging time delays.

- The input must be configured as a *Start Indexed Move Input.*
- The move begins with an inactive-to-active transition on the input. Note that an active-to-inactive transition on the input will not stop the move.
- The move command must stay in the Network Output Data while performing an Indexed Move. The move will not occur if you reset the command word before the input triggers the move.
- The move can be run multiple times as long as the move command data remains unchanged in the Network Output Data. The move will run on every inactive-to-active transition on the physical input if a move is not currently in progress. Once a move is triggered, the Start Indexed Move Input is ignored by the Networked Driver until the triggered move is finished.
- As stated above, a move can be run multiple times as long at the move command data remains unchanged. If you wish to program a second move and run it as an Indexed Move type, then you must have a 0→1 transition on the move command bit before the new parameters are accepted. The easiest way to accomplish this is by writing a value of 16#0000 to the command word between issuing move commands.
- A Jog Move that is started as an Indexed Move will come to a controlled stop when the command bit in the Network Output Data is reset to zero.
- It is possible to perform an indexed Registration Move by configuring two inputs for their respective functions. The first input, configured as a *Start Indexed Move Input*, starts the move and the second, configured as a *Stop Jog or Registration Move Input* causes the registration function to occur.
- You cannot issue a Hold Command with the Indexed Bit set and have the Hold Command trigger on the inactive-to-active transition of a physical input. Hold Commands are always acted upon as soon as they are accepted from the Network Output Data.
- You cannot issue an Immediate Stop Command with the Indexed Bit set and have the Immediate Stop Command trigger on the inactive-to-active transition of a physical input. Immediate Stop Commands are always acted upon as soon as they are accepted from the Network Output Data. If you need this functionality, consider programming the physical input as an E-Stop Input.
- You cannot issue a Clear Error Command with the Indexed Bit set and have the Clear Error Command trigger on the inactive-to-active transition of a physical input. Clear Error Commands are always acted upon as soon as they are accepted from the Network Output Data.

Controlling Moves In Progress

Each Networked Driver has the ability to place a running move on hold and later resume the move if an error did not occur while the move was in its Hold state. One potential application for this feature is bringing a move to a controlled stop when your controller senses an end-of-stock condition. The move can be put in its Hold state until the stock is replenished and then the move can be resumed.

Note that you do not have to resume a move once it has been placed in its Hold state. You can place a move in its Hold state to prematurely end the move with a controlled stop and issue a new move of any type from the stopped position.

The figure below shows a profile of a move that is placed in its Hold state and later resumed.

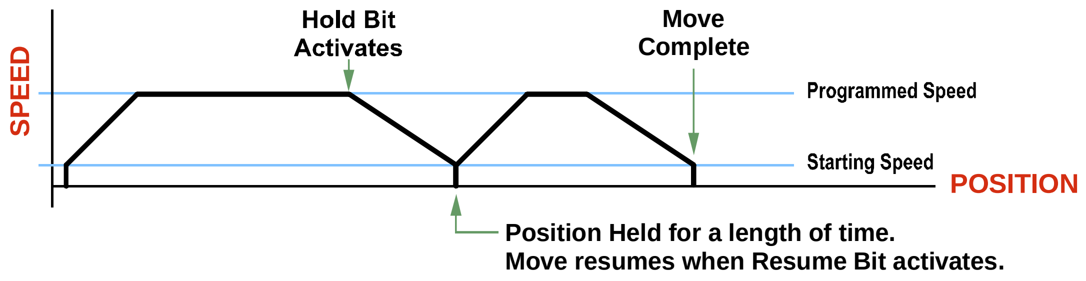


Figure R3.14 Hold/Resume a Move Profile

Find Home Moves

A Find Home command can be placed in a Hold state but cannot be resumed. This give you the ability to bring a Find Home command to a controlled stop if an error condition occurs.

Jog Moves

Jog Moves can be placed in a Hold state and resumed if error conditions, such as programming errors, have not occurred. New Acceleration, Deceleration, and Programmed Speed parameters can be written to the Networked Driver while a Jog Move is in its hold state. If these parameters are accepted without error, the move can be resumed and it will use the new parameter values.

Registration Moves

Registration Moves can be brought to a controlled stop with the Hold bit, but they cannot be restarted.

Absolute, Relative, and Encoder Registration Moves

Absolute, Relative, and Encoder Registration Moves can be placed in a Hold state and resumed if error conditions, such as programming errors, have not occurred. New Acceleration, Deceleration, and Programmed Speed parameters can be written to the Networked Driver while these moves are in their hold states. If the parameters are accepted without error, the move can be resumed and it will use the new parameter values. Note that a change to the Target Position is ignored.

Assembled Moves

A Blend or Dwell Move can be placed in a Hold state but cannot be resumed. This give you the ability to prematurely end an Assembled Move with a controlled stop. The Assembled Move is not erased from memory and can be run again without having to reprogram it.

Electronic Gearing

The final form of motion control available with a Networked Driver is Electronic Gearing. A quadrature encoder is required but it is not mounted on the motor controlled by the Networked Driver. The encoder is typically mounted on a second motor, but it can be mounted anywhere, including on something as simple as a hand crank.

This mode is sometimes referred to as *encoder following*, because the motor will change position in response to a change in position of the encoder. AMCI refers to it as Electronic Gearing because the Networked Driver has three parameters that allow you to set any turns ratio you want between the encoder and the motor.

Motor_Resolution

This is the same parameter explained at the beginning of this chapter. In Electronic Gearing mode, this parameter sets the number of encoder counts required to complete one rotation of the shaft of the motor driven by the Networked Driver. It has a range of 200 to 32,767. If you set the Motor_Resolution parameter to four times the number of lines of your encoder, then by default your motor will complete one rotation for every rotation of the encoder.

ELGearing Multiplier and Divisor

The ratio of these two parameters sets the number of motor rotations per encoder rotation. Each parameter has a range of 1 to 255.

How It Works

The Networked Driver always uses 4X decoding when counting pulses from the encoder. If you set the Motor_Resolution parameter to four times the number of encoder lines, and set both of the ELGearing Multiplier and Divisor parameters to 1, then the motor will complete one rotation for every rotation of the encoder's shaft.

Once placed in Electronic Gearing mode, the Networked Driver monitors the Jog Move command bits in the Network Output Registers. When either of these bits are set, the encoder inputs are monitored for a change in position. When a change is sensed, the Networked Driver will begin to turn the motor within 50 microseconds. An increase in encoder counts will result in clockwise rotation. A decrease in encoder counts will result in counter-clockwise rotation.

The values of the ELGearing Multiplier and Divisor can be changed while electronic gearing motion is occurring. The Networked Driver will accelerate or decelerate the motor to match the new ratio.

Encoder position data can be captured and reported to the host controller while in Electronic Gearing mode by configuring Input 3 as a Capture Encoder Position input.

NOTE You must set the Acceleration and Deceleration parameters before you can put the Networked Driver in Electronic Gearing mode.

NOTE You cannot use Stall Detection when using Electronic Gearing. Stall detection requires that the encoder be mounted to the motor you are controlling. Electronic Gearing requires that the encoder not be mounted on the motor you are controlling.

Electronic Gearing (continued)

Controlled Stop Conditions

- The encoder stops moving.
- Both of the Jog Move command bits equal zero.
- Electronic Gearing moves cannot be brought to a controlled stop by using the Hold_Move control bit in the Network Output Registers.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Output Data.
- A positive transition on an input configured as an E-Stop Input.
- A CW or CCW Limit Switch is reached.

Advanced Ratio Control

The ELGearing Multiplier and Divisor values give you a great deal of control over the ratio of motor turns per encoder turn, but you can achieve even finer control by adjusting the Motor_Resolution parameter.

The Z pulse is not used to correct the encoder position once per turn, so you can actually program the Motor_Resolution parameter to any value you want within its valid range. For example, if your encoder outputs 4,096 pulse per turn (a 1,024 line encoder) and you set the Motor_Resolution parameter to 8,192, you will have built a 2:1 gear down into your system before applying the ELGearing Multiplier and Divisors. (Two rotations of the encoder = 8,192 counts = 1 motor rotation.)

This technique allows you to set a median gear ratio in your system that you can adjust on-the-fly by using the ELGearing Multiplier and Divisor parameters.

Notes

REFERENCE 4
CALCULATING MOVE PROFILES

This reference was added because some of our customers must program very precise profiles. Understanding this section is not necessary before programming the Networked Driver and it can be considered optional. Two different approaches are presented here. The constant acceleration example takes given parameters and calculates the resulting profile. The variable acceleration example starts with a desired speed profile and calculates the required parameters

The equations in this reference use a unit of measure of steps/second/second (steps/second^2) for acceleration and deceleration. However, when programming the Networked Driver, all acceleration and deceleration values must be programmed in the unit of measure of steps/second/millisecond.

- To convert from steps/second^2 to steps/second/millisecond, divide the value by 1000. This must be done when converting from a value used in the equations to a value programmed into the Networked Driver.
- To convert from steps/second/millisecond to steps/second^2, multiply the value by 1000. This must be done when converting from the value programmed into the Networked Driver to the value used in the equations.

Constant Acceleration Equations

When you choose to use constant accelerations, the speed of the move will increase linearly towards the Programmed Speed. This is the fastest form of acceleration, resulting in the fastest move between two points at its programmed speed. For the smoothest transition from the starting speed, the starting speed should be equal to the square root of the acceleration in steps/sec^2. For example, if the choose acceleration is 20,000 steps/sec^2, the smoothest transition occurs when the starting speed is 141. ($141^2 \approx 20{,}000$)

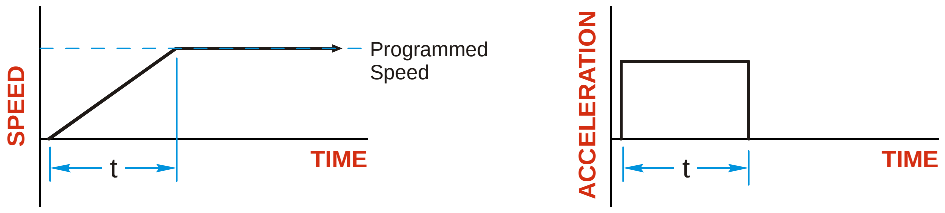
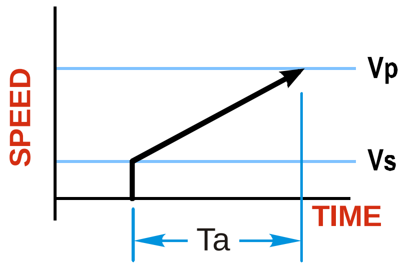


Figure R4.1 Constant Acceleration Curves

Variable Definitions

The following variables are used in these equations:



- **V_S** = Configured Starting Speed of the move
- **V_P** = Programmed Speed of the move
- **a** = Acceleration value. Must be in the units of steps/second^2
- **d** = Deceleration value. Must be in the units of steps/second^2
- **T_A or T_D** = Time needed to complete the acceleration or deceleration phase of the move
- **D_A or D_D** = Number of Steps needed to complete the acceleration or deceleration phase of the move

Constant Acceleration Equations (continued)

Figure R4.1 gives the equations to calculate Time, Distance, and Acceleration values for a constant acceleration move.

Acceleration Type	T_A or T_D (Time to Accelerate or Decelerate)	D_A or D_D (Distance to Accelerate or Decelerate)	a (Average Acceleration)
Linear	$T_A = (V_P - V_S)/a$	$D_A = T_A*(V_P + V_S)/2$	$a = (V_P^2 - V_S^2)/2D_A$

Table R4.1 Acceleration Equations

If the sum of the D_A and D_D values of the move is *less than* the total number of steps in the move, your move will have a Trapezoidal profile.

If the sum of the D_A and D_D values of the move is *equal to* the total number of steps in the move, your move will have a Triangular profile and your move will reach the Programmed Speed before it begins to decelerate.

If the sum of the D_A and D_D values of the move is *greater than* the total number of steps in the move, your move will have a Triangular profile and it *will not* reach the Programmed Speed before it begins to decelerate.

As an example, lets assume the values in table R4.2 for a move profile.

Name	Value	Networked Driver Parameter Values
Acceleration (a)	20,000 steps/sec^2	20
Deceleration (d)	25,000 steps/sec^2	25
Starting Speed (V_S)	141 steps/sec	141
Programmed Speed (V_P)	100,000 steps/sec	100,000

Table R4.2 Sample Values

From figure R4.1:

Time to accelerate: $T_A = (V_P - V_S)/a = (100{,}000 - 141)/20{,}000 = 4.993$ seconds
Time to decelerate: $T_D = (V_P - V_S)/d = (100{,}000 - 141)/25{,}000 = 3.994$ seconds
Distance to Accelerate: $D_A = T_A*(V_P + V_S)/2 = 4.993 * (100{,}000 + 141)/2 = 250{,}002$ steps
Distance to Decelerate: $D_D = T_D*(V_P + V_S)/2 = 3.994 * (100{,}000 + 141)/2 = 199{,}982$ steps

Total Distance needed to accelerate and decelerate: 250,002 + 199,982 = 449,984 steps

If a move with the above acceleration, deceleration, starting speed, and programmed speed has a length greater than 449,984 steps, the Networked Driver will generate a Trapezoidal profile. If the move is equal to 449,984 steps, the Networked Driver will generate a Triangular profile and the unit will output one pulse at the programmed speed. If the move is less than 449,984 steps, the Networked Driver will generate a Triangular profile and the programmed speed will not be reached.

In the case of a Triangular profile where the programmed speed is not reached, it is fairly easy to calculate the maximum speed (V_M) attained during the move. Because the move is always accelerating or decelerating, the total distance traveled is equal to the sum of D_A and D_D.

$D_A = T_A*(V_M + V_S)/2$ and $T_A = (V_M - V_S)/a$. By substitution:
$D_A = (V_M - V_S)/a * (V_M + V_S)/2 = (V_M^2 - V_S^2)/2a$. By the same method,
$D_D = (V_M^2 - V_S^2)/2d$.

Therefore, total distance traveled =

$$D_A + D_D = (V_M^2 - V_S^2)/2a + (V_M^2 - V_S^2)/2d.$$

In the case where the acceleration and deceleration values are equal, this formula reduces to:

$$D_A + D_D = (V_M^2 - V_S^2)/a$$

Constant Acceleration Equations (continued)

Continuing the example from table R4.2, assume a total travel distance of 300,000 steps.

$$
\begin{aligned}
D_A + D_D &= \frac{V_M^2 - V_S^2}{2a} + \frac{V_M^2 - V_S^2}{2d} \\
300{,}000 \text{ steps} &= \frac{V_M^2 - 141^2}{2(20{,}000)} + \frac{V_M^2 - 141^2}{2(25{,}000)} \\
300{,}000 \text{ steps} &= \frac{V_M^2 - 20{,}000}{40{,}000} + \frac{V_M^2 - 20{,}000}{50{,}000} \\
300{,}000 \text{ steps} &= \frac{5}{5}\left(\frac{V_M^2 - 20{,}000}{40{,}000}\right) + \frac{4}{4}\left(\frac{V_M^2 - 20{,}000}{50{,}000}\right) \\
300{,}000 \text{ steps} &= \frac{5V_M^2 - 100{,}000}{200{,}000} + \frac{4V_M^2 - 80{,}000}{200{,}000} \\
300{,}000\ (200{,}000) &= 9V_M^2 - 180{,}000 \\
\frac{60{,}000.18 \times 10^6}{9} &= V_M^2 \\
V_M &= 81{,}650 \text{ steps/sec}
\end{aligned}
$$

Once you have calculated the maximum speed, you can substitute this value into the time and distance formulas in table R4.1 to calculate time spent and distance traveled while accelerating and decelerating.

Total Time Equations

For Trapezoidal Profiles you must first determine the number of counts that you are running at the Programmed Speed. This value, (D_P below), is equal to your D_A and D_D values subtracted from your total travel. You can then calculate your total profile time, (T_T below), from the second equation.

D_P = (Total Number of Steps) – (D_A + D_D)

$T_T = T_A + T_D + D_P/V_P$

For Triangular Profiles, the total time of travel is simply:

$T_T = T_A + T_D$

S-Curve Acceleration Equations

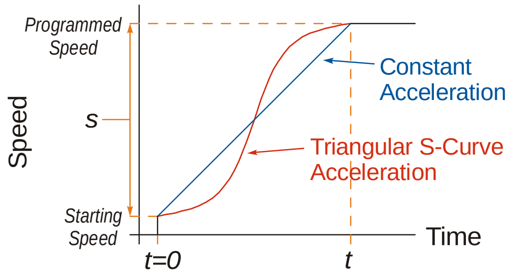
When the Acceleration Jerk parameter value is in the range of 1 to 5,000, the Networked Driver uses this value to smoothly change the acceleration value applied during the move. In this case, the speed of the move does not increase linearly, but exponentially, resulting in an "S" shaped curve. This limits mechanical shocks to the system as the load accelerates. Just as constant acceleration will result in a trapezoidal or triangular speed profile, the Acceleration Jerk parameter will result in a trapezoidal or triangular acceleration phase.

In order to keep the Acceleration Jerk parameter value that is programmed into the Networked Driver below sixteen bits, the Acceleration Jerk parameter programmed into the driver does not have units of steps/sec^3. The Acceleration Jerk parameter equals ({100 * jerk in steps/sec^3} / acceleration in steps/sec^2). This translates to the jerk property in steps/sec^3 equalling ({Acceleration Jerk parameter/100} * acceleration in steps/sec^2). With the range of values for the Acceleration Jerk parameter being 1 to 5,000, the jerk value ranges from 0.01*a* to 50*a* where "*a*" is the acceleration value in steps/sec^2. For example, if the acceleration is programmed to 20,000 steps/sec^2, then the value of the jerk property used by the unit can be programmed to be between 200 steps/sec^3 (0.01*20,000) and 1,000,000 steps/sec^3 (50*20,000). This statement applies to the Deceleration Parameter as well. If the Acceleration and Deceleration parameters are different, the calculated jerk values will also differ.

When using variable accelerations, the starting speed does not have to be equal to the square root of the programmed acceleration value. Variable acceleration provides smooth transitions at the beginning and end of the acceleration phase.

Triangular S-Curve Acceleration

Figure R4.2 shows the speed profile of a move during its acceleration phase. The figure shows the desired triangular S-curve acceleration in red along with the equivalent constant acceleration in blue. The equivalent constant acceleration is equal to the change in speed divided by the time it takes to achieve this change in speed. This is the value that would have to be used if the Jerk parameter was left at zero and we will use this information to calculate the S-curve acceleration and the value of the Jerk Parameter.



s = Programmed Speed – Starting Speed

$$\text{Acceleration} = \frac{\text{speed}}{\text{time}} \qquad \text{jerk} = \frac{\text{acceleration}}{\text{time}} \qquad \text{Unit's Acceleration Jerk Parameter(J)} = \frac{100j}{a}$$

$$a = \frac{s}{t} \qquad j = \frac{a}{t} \qquad j = \frac{Ja}{100}$$

$$at = s \qquad jt = a$$

Figure R4.2 Move Profile Example

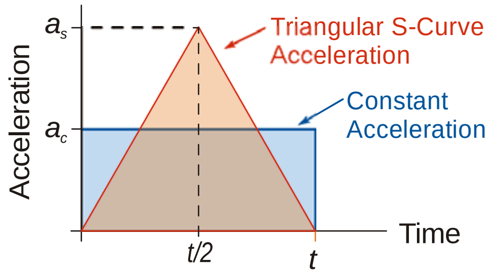


Figure R4.3 Triangular Acceleration

Speed is equal to acceleration multiplied by the time it is applied. This is shown graphically in figure R4.3 as the area of the blue rectangle. In order for the Triangular S-curve acceleration to reach the same speed in the same amount of time, the area of the triangle must equal the area of the square. Area of a triangle is one half of the base length multiplied by the height. Therefore:

$$a_c t = \frac{a_s t}{2} \quad \text{Area of rectangle = Area of triangle}$$

$$a_s = 2a_c$$

This means that a triangular S-curve acceleration profile requires twice the programmed maximum acceleration as a constant acceleration profile to achieve the same speed in the same amount of time.

S-Curve Acceleration Equations (continued)

Triangular S-Curve Acceleration (continued)

The value of the Acceleration Jerk parameter can now be easily calculated.

$$j = \frac{a_s}{t/2} \qquad (j = a/t)$$

$$j = \frac{2a_s}{t}$$

$$\frac{Ja_s}{100} = \frac{2a_s}{t} \qquad \left(j = \frac{Ja}{100}\right)$$

$$Ja_st = 200a_s$$

$$J = \frac{200}{t} \qquad \text{Acceleration Jerk parameter = 200 / acceleration time}$$

This value represents the ideal Acceleration Jerk parameter value for a triangular S-curve acceleration. Setting the value lower than this will result in a longer acceleration period, while setting the value above this will result in a trapezoidal S-curve acceleration.

When $a_s = a_c$

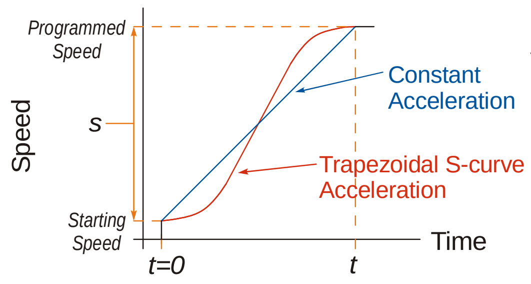
The above examples assume that you can increase the programmed acceleration value to keep the acceleration time the same. If your constant acceleration value is the maximum your system will allow, then using S-curve accelerations will lengthen the time needed to accelerate to your desired speed.

In the case of Triangular S-curve accelerations where the Acceleration Jerk parameter is optimized at 200/t, the value of “t” must be twice that of the acceleration period when constant acceleration is used. For example, assume a equivalent constant acceleration of 20,000 steps/sec^2 that is applied for 2.0 seconds. If the acceleration value must remain at 20,000 steps/sec^2, then the acceleration phase will take 4.0 seconds and the Acceleration Jerk parameter should be set to 50 (200/4.0)

S-Curve Acceleration Equations (continued)

Trapezoidal S-Curve Acceleration

Figure R4.4 shows the speed profile of a move during its acceleration phase. The figure shows the desired trapezoidal S-curve acceleration in red along with the equivalent constant acceleration in blue. The equivalent constant acceleration is equal to the change in speed divided by the time it takes to achieve the change in speed. This is the value that would have to be used if the Acceleration Jerk parameter was left at zero and we will use this information to calculate the S-curve acceleration and the value of the Acceleration Jerk Parameter.



S = Programmed Speed – Starting Speed

$$\text{Acceleration} = \frac{\text{speed}}{\text{time}} \qquad \text{jerk} = \frac{\text{acceleration}}{\text{time}} \qquad \text{Unit's Acceleration Jerk Parameter(J)} = \frac{100j}{a}$$

$$a = \frac{s}{t} \qquad j = \frac{a}{t}$$

$$at = s \qquad jt = a \qquad \Rightarrow \quad j = \frac{Ja}{100}$$

Figure R4.4 Move Profile Example

In this example, the period of constant acceleration is 50% of the acceleration phase.

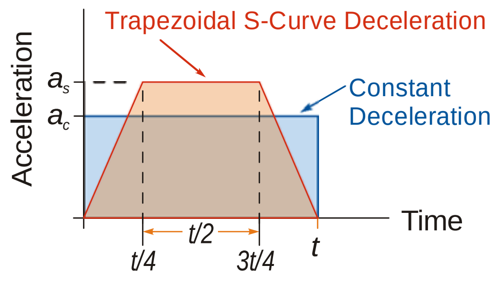


Figure R4.5 Trapezoidal Acceleration

Speed is equal to acceleration multiplied by the time it is applied. This is shown graphically in figure R4.5 as the area of the blue rectangle. In order for the Trapezoidal S-curve acceleration to reach the same speed in the same amount of time, the area of the polygon must equal the area of the rectangle.

$$\frac{a_s t}{2} + \frac{a_s t}{4} = a_c t \quad \text{Area of polygon = Area of rectangle}$$

$$\frac{2a_s t}{4} + \frac{a_s t}{4} = a_c t$$

$$\frac{3a_s t}{4} = a_c t$$

$$a_s = \frac{4}{3} a_c$$

This means that a trapezoidal S-curve acceleration profile that is has a period of constant acceleration equal to half of the total phase time, requires its programmed acceleration value to be 4/3 that of the constant acceleration value used to achieve the same speed in the same amount of time.

S-Curve Acceleration Equations (continued)

Trapezoidal S-Curve Acceleration (continued)

The value of the Acceleration Jerk parameter can now be easily calculated.

$$j = \frac{a_s}{t/4} \qquad (j = a/t)$$

$$j = \frac{4a_s}{t}$$

$$\frac{Ja_s}{100} = \frac{4a_s}{t} \qquad \left(j = \frac{Ja}{100}\right)$$

$$Ja_s t = 400a_s$$

$$J = \frac{400}{t} \qquad \text{Acceleration Jerk Parameter = 400 / acceleration time}$$

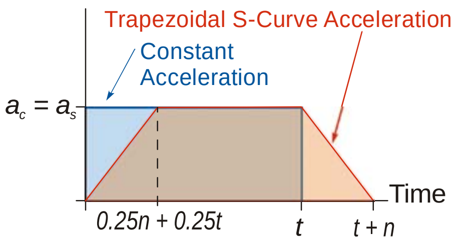
This value represents the ideal Acceleration Jerk parameter value for a trapezoidal S-curve acceleration with a constant acceleration for half of the phase. Setting the value lower than this will result in a shorter constant period, while setting the value greater than this will result in a longer constant period.

Another example of a trapezoidal S-curve acceleration is when the linear acceleration occurs for one third of the time. In this case, the programmed acceleration must be the constant acceleration value multiplied by 3/2 and the Acceleration Jerk parameter must be set to 300/t.

When $a_s = a_c$

The above examples assume that you can increase the programmed acceleration value to keep the time of the acceleration phase the same. If your constant acceleration value is the maximum your system will allow, then using S-curve accelerations will lengthen the time needed to accelerate to your desired speed.

In the case of trapezoidal S-curve accelerations, calculating the percentage increase in time is shown in figure R4.6. The time added to the acceleration phase is equal to the time spent increasing the acceleration during the phase. As shown in the figure, when the Trapezoidal S-curve is programmed to spend 50% of its time at the programmed acceleration value, the time spent in the acceleration phase will be 133.33% of the time spent if a constant acceleration were used.



$a_c(t) = a_c(.5n + .5t) + a_c(.25n + .25t)$

$a_c(t) = a_c((.5n + .5t) + (.25n + .25t))$

$t = .75n + .75t$

$0.25t = .75n$

$t = 3n$

$t/3 = n \Rightarrow t+n = t + t/3 = 4/3t = 1.3333t$

Figure R4.6 Trapezoidal S-curve Time Increase Example

In this case the value of the Acceleration Jerk parameter should be based on the new, longer time. For example, assume an equivalent constant acceleration of 15,000 steps/sec^2 that is applied for 2.0 seconds. If the acceleration value must remain at 15,000 steps/sec^2, then the acceleration phase will take 2.667 seconds (2.0×1.333) and the Acceleration Jerk parameter should be set to 150 (400/2.667)

Similarly, if the Trapezoidal S-curve acceleration is to spend 33.3% of its time at constant acceleration, and the programmed acceleration value cannot be increased, the time spent accelerating will increase by 50% and the Acceleration Jerk parameter should be adjusted accordingly.

S-Curve Acceleration Equations (continued)

Determining Waveforms by Values

If your programmed acceleration and deceleration values are the same, then your move's acceleration and decelerations will be identical. If these two programmed values are different, use the above methods to determine the Acceleration Jerk parameter for either the move's acceleration or deceleration phases and use the following calculations to determine the shape of the other phase.

Two examples are given below. Both assume a change in speed between the Starting Speed and Programmed Speed of 30,000 steps/sec and an acceleration of 58,000 steps/sec^2. The first example uses an Acceleration Jerk parameter value of 20 and the second a value of 400.

Triangular or Trapezoidal S-curve accelerations are always symmetrical. We'll use this fact to calculate the profile up to one-half of the change in speed. At that point, doubling the time and distance will yield the total time and distance traveled.

Example 1, Jerk = 20

$$S_m = \frac{30{,}000 \text{ steps/sec}}{2} = 15{,}000 \text{ steps/sec}$$

S_m = midpoint of change in speed

J = Acceleration Jerk parameter

j = physical jerk property

a_f = calculated final acceleration

$$J = \frac{100j}{a} \Rightarrow j = \frac{Ja}{100}$$

$$j = \frac{20(58{,}000 \text{ steps/sec}^2)}{100}$$

$$j = 11{,}600 \text{ steps/sec}^3$$

Just as displacement $= \frac{1}{2}at^2$, Speed $= \frac{1}{2}jt^2$

$$15{,}000 \text{ steps/sec} = \frac{11{,}600 \text{ steps/sec}^3(t^2)}{2}$$

$$t^2 = \frac{15{,}000 \text{ steps/sec}}{5{,}800 \text{ stesp/sec}^3}$$

$$t = 1.608 \text{ seconds}$$

Just as speed = at, acceleration = jt

$$a_f = 11{,}600 \text{ steps/sec}^3(1.608 \text{ sec})$$

$$a_f = 18{,}655 \text{ steps/sec}^2$$

Because a_f is less than or equal to the programmed acceleration of 58,000 steps/sec^2, the resulting acceleration is a Triangular S-curve. Total time to accelerate is twice the value calculated above, or 3.216 seconds.

S-Curve Acceleration Equations (continued)

Determining Waveforms by Values (continued)

Example 2, Jerk = 400

$$S_m = \frac{30{,}000 \text{ steps/sec}}{2} = 15{,}000 \text{ steps/sec}$$

$$J = \frac{100j}{a} \Rightarrow j = \frac{Ja}{100}$$

$$j = \frac{400(58{,}000 \text{ steps/sec}^2)}{100}$$

$$j = 232{,}000 \text{ steps/sec}^3$$

S_m	=	midpoint of change in speed
J	=	Acceleration Jerk parameter
j	=	physical jerk property
a_f	=	calculated final acceleration

Just as displacement = $\frac{1}{2}at^2$, speed = $\frac{1}{2}jt^2$

$$15{,}000 \text{ steps/sec} = \frac{232{,}000 \text{ steps/sec}^3(t^2)}{2}$$

$$t^2 = \frac{15{,}000 \text{ steps/sec}}{116{,}000 \text{ steps/sec}^3}$$

$$t = 0.3596 \text{ seconds}$$

Just as speed = at, acceleration = jt

$$a_f = 232{,}000 \text{ steps/sec}^3(0.3596 \text{ sec})$$

$$a_f = 83{,}427 \text{ steps/sec}^2$$

Because a_f is greater than the programmed acceleration of 58,000 steps/sec^2, the resulting acceleration is a trapezoidal S-curve. As shown in figure R4.7, two additional calculations must be made. The first is the time (t_1) it takes to jerk to the programmed acceleration value. The second is the time (t_2) it takes to accelerate to half of the required change in speed (S_m).

$$232{,}000 \text{ steps/sec}^3(t_1) = 58{,}000 \text{ steps/sec}^2 \qquad jt = a$$

$$t_1 = 0.25 \text{ seconds}$$

Determine speed at t_1: Speed = $\frac{1}{2}jt^2$

$$S_1 = \frac{232{,}000 \text{ steps/sec}^3(0.25)^2}{2}$$

$$S_1 = 7{,}250 \text{ steps/sec}$$

Determine remaining change in speed and required time based on programmed acceleration

$$S_2 = S_m - S_1 = (15{,}000 - 7{,}250) \text{ steps/sec}$$

$$S_2 = 7{,}750 \text{ steps/sec}$$

$$S_2 = a_c(t_2) \Rightarrow t_2 = S_2 / a_c$$

$$t_2 = \frac{7{,}750 \text{ steps/sec}}{58{,}000 \text{ steps/sec}^2}$$

$$t_2 = 0.1336 \text{ seconds}$$

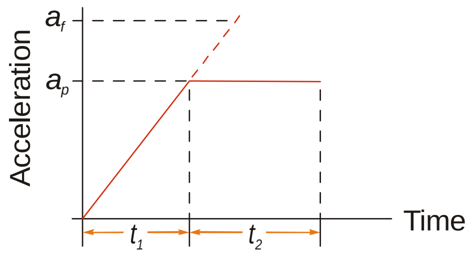


Figure R4.7 Calculating Trapezoidal S-Curve

The time for this acceleration phase is 2(t1 + t2), which equals 2(0.2500 sec + 0.1336 sec) or 0.7672 seconds. Time spent in the constant acceleration period is (2(0.1336))/0.7672) or 34.8% of the entire phase.

Notes

REFERENCE 5

HOMING AN AMCI NETWORKED DRIVER

This chapter explains the various ways of homing a Networked Driver. Inputs used to home the unit are introduced and diagrams that show how the unit responds to a homing command are given.

Definition of Home Position

The Home Position is any position on your machine that you can sense and stop at. Once at the Home Position, the motor position register of a Networked Driver must be set to an appropriate value. If you use the unit's *CW/CCW Find Home* commands, the motor position register will automatically be set to zero once the home position is reached. *The Encoder Position register will also be reset to zero if the encoder is available and enabled.*

NOTE Defining a Home Position is completely optional. Some applications, such as those that use a Networked Driver for speed control, don't require position data at all.

With the exception of Absolute Moves, a Networked Driver can still perform all of its move commands if the Home Position is not defined.

Position Preset

One of the ways to define the Home Position is to issue the Preset Position command to the Networked Driver. When a Networked Driver is configured to use an incremental encoder, the motor position and the encoder position are preset separately. The motor and encoder position values can be preset anywhere in the range of –8,388,607 to +8,388,607.

CW/CCW Find Home Commands

The other choice is to use the driver's Find Home commands to order the Networked Driver to find the Home Position based on sensors brought into the unit. The CW Find Home command begins searching by rotating the motor shaft in the positive direction and ends when the home sensor triggers while the Networked Driver is rotating in the positive direction at the starting speed. The CCW Find Home command operates in the same way but starts and ends with motion in the negative direction.

Homing Inputs

Up to three physical inputs can be used to home the driver. A Backplane Proximity bit is available in the network output data that can be used to control when the home input is acted upon. This is typically used in applications where the home input triggers multiple times in a machine cycle, and the system needs to control which trigger is acted upon.

Physical Inputs

- **Home Input:** This input is used to define the actual home position of the machine.
- **CW Limit Switch Input:** This input is used to prevent overtravel in the clockwise direction.
- **CCW Limit Switch Input:** This input is used to prevent overtravel in the counter-clockwise direction.

Backplane Inputs

- **Backplace_Proximity_Bit:** A Networked Driver can be configured to ignore changes on the physical homing input until the Backplace_Proximity_Bit makes a 0→1 transition. The Networked Driver will home on the next inactive-to-active change on the physical input once this transition occurs. You must program your host to control the state of this bit.

NOTE This bit is disabled by default, and must be activated when you configure the Networked Driver before is can be used. If you decide to activate this bit when you configure the unit and then never set it to a "1", the Networked Driver will never act on the physical Home Input.

Homing Configurations

A Networked Driver must have one of its DC inputs configured as the home input before one of the *CW/CCW Find Home* commands can be issued.

NOTE

1) You do not have to configure and use CW or CCW Limits. If you choose to configure the module this way, then the Networked Driver has no way to automatically prevent overtravel during a homing operation. You must prevent overtravel by some external means, or ensure that the homing command is issued in the direction that will result in reaching the homing input directly.

2) You can use a bit in the Network Output Data (the Backplace_Proximity_Bit) as a home proximity input. Using this bit is completely optional and prevents the Home Input from being acted upon until the Backplace_Proximity_Bit makes a 0→1 transition.

Homing Profiles

NOTE

1) The CW Find Home command is used in all of these examples. The CCW Find Home command will generate the same profiles in the opposite direction.

2) The homing diagrams show CW and CCW direction on their vertical axes. These are the directions that will occur if using an AMCI motor that is wired as shown in this manual. It is possible to reverse the direction of rotation by swapping one connections on one of the motor windings.

Home Input Only Profile

Figure R5.1 below shows the move profile generated by a +Find Home command when you use the Home Input without the Backplace_Proximity_Bit.

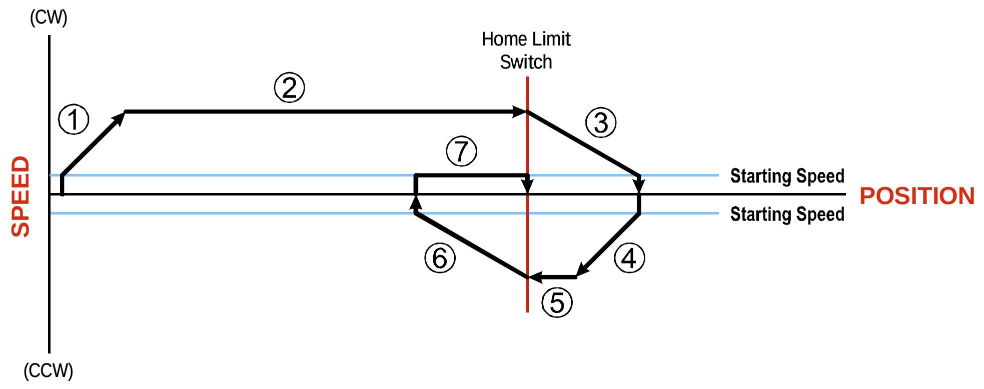


Figure R5.1 Home Input Profile

1) Acceleration from the configured Starting Speed to the Programmed Speed
2) Run at the Programmed Speed until the Home Input activates
3) Deceleration to the Starting Speed and stop, followed by a two second delay.
4) Acceleration to the Programmed Speed opposite to the requested direction.
5) Run opposite the requested direction until the Home Input transitions from Active to Inactive
6) Deceleration to the Starting Speed and stop, followed by a two second delay.
7) Return to the Home Input at the configured Starting Speed. Stop when the Home Input transitions from inactive to active.

NOTE If the Home Input is active when the command is issued, the move profile begins at step 5 above.

Homing Profiles (continued)

Profile with Network Backplace_Proximity_Bit

Figure R5.2 below shows the move profile generated by a +Find Home command when you use the Home Input with Network Backplace_Proximity_Bit.

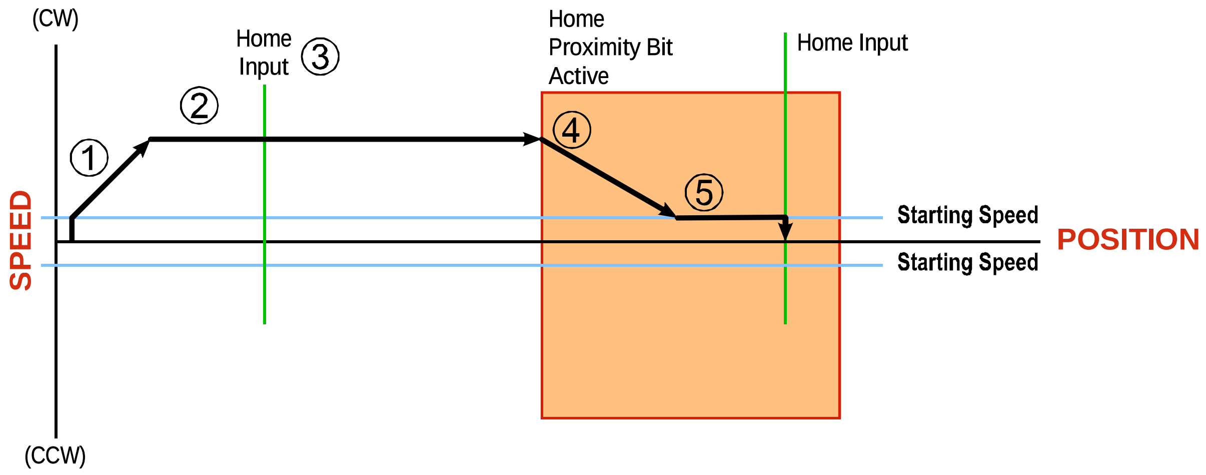


Figure R5.2 Homing with Proximity

1) Acceleration from the configured Starting Speed to the Programmed Speed
2) Run at the Programmed Speed
3) Ignores the Home Input because Backplace_Proximity_Bit has not made a 0→1 transition.
4) Deceleration towards the Starting Speed when the Backplace_Proximity_Bit transitions from 0 to 1. The axis will stop as soon as the Home Input becomes active.
5) The Starting Speed is the minimum speed the profile will run at. If the axis decelerates to the Starting Speed before reaching the Home Input, it will continue at this speed.

NOTE Figure R5.2 shows the Backplane Backplace_Proximity_Bit staying active until the Networked Driver reaches its home position. This is valid, but does not have to occur. As stated in step 4, the Networked Driver starts to hunt for the home position as soon and the Backplane Backplace_Proximity_Bit makes a 0→1 transition

Homing Profiles (continued)

Profile with Overtravel Limit

Figure R5.3 below shows the move profile generated by a +Find Home command when you use:

- CW Overtravel Limit
- Home Input without the Backplace_Proximity_Bit

The profile is generated when you encounter an overtravel limit in the direction of travel. (In this example, hitting the CW limit while traveling in the CW direction.)

NOTE The Networked Driver will stop and issue a *Home Invalid* error to your host if you activate the overtravel limit associated with travel in the opposite direction. i.e. Activating the CCW limit during a CW Find Home command. This can occur if the overtravel limits are not wired to the Networked Driver correctly, or if both overtravel limits are activated while the unit is trying to find the home position.

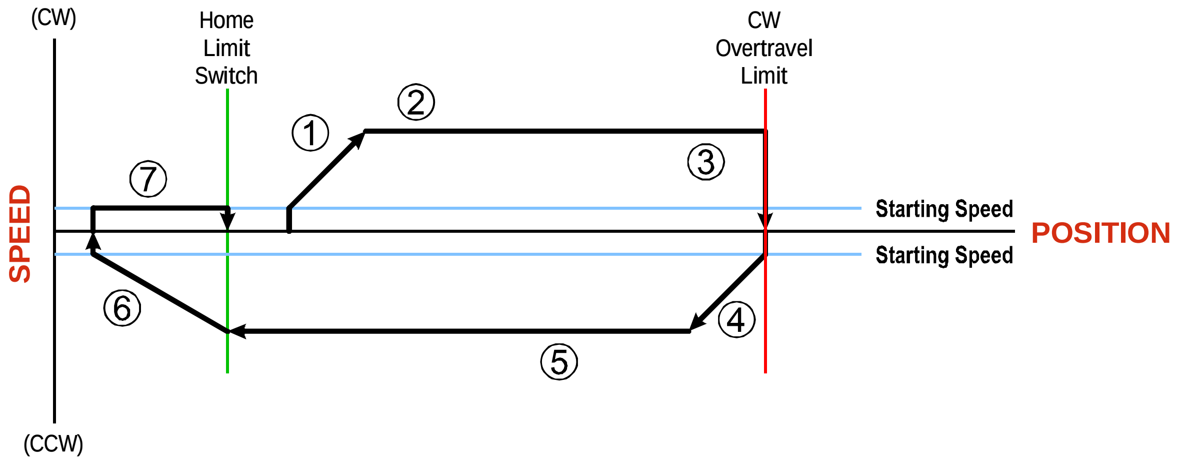


Figure R5.3 Profile with Overtravel Limit

1) Acceleration from the configured Starting Speed to the Programmed Speed
2) Run at the Programmed Speed
3) Hit CW Limit and immediately stop, followed by a two second delay.
4) Acceleration to the Programmed Speed opposite to the requested direction.
5) Run opposite the requested direction until the Home Input transitions from Active to Inactive
6) Deceleration to the Starting Speed and stop, followed by a two second delay.
7) Return to the Home Input at the configured Starting Speed. Stop when the Home Input transitions from Inactive to Active.

NOTE If the overtravel limit is active when the Find Home Command is active, the profile will begin at step 4.

Controlling Find Home Commands In Progress

Controlled Stops

- The move completes without error.
- You toggle the Hold_Move control bit in the Network Output Data. This will abort the command and the axis will decelerate at the programmed rate until it reaches the Starting Speed. At this point, the motor will stop. Note that Find Home commands cannot be restarted once held.

Immediate Stops

- The Immediate Stop bit makes a 0→1 transition in the Network Output Data.
- An inactive-to-active transition on an input configured as an E-Stop Input.
- The overtravel limit associated with travel in the opposite direction is activated. i.e. Activating the CCW limit during a CW Find Home command. This can occur if the overtravel limits are not wired to the Networked Driver correctly, or if both overtravel limits are activated while the unit is trying to find the home position.

Notes

REFERENCE 6
CONFIGURATION DATA FORMAT

This chapter covers the format of the configuration data for an AMCI Networked Driver. Configuration data is stored in registers available through the CANopen over Ethernet (CoE) interface.

CoE Registers

The SD17060E-K and SD31045E-K units use the Service Data Objects (SDO) of the CANopen protocol to configure the unit. Typically, this configuration is specified in the system software and is written down to the Networked Driver when communications is established. The TwinCAT system also allows you to programmatically read or change these values using the FB_EcCoeSdoRead and FB_EcCoeSdoWrite instructions.

Output Data Format

The correct format for the Network Output Data is shown below.

CANopen Index:SubIndex	Data Size	Configuration Data	Range
8010:01	UINT	CFG_Word_0	See below
8010:02	UINT	CFG_Word_1	See below
8010:03	UDINT	Starting_Speed	1 to 1,999,999 steps/sec.
8010:04	UINT	Motor_Resolution	200 to 32,767
8010:05	UINT	Reserved	0
8010:06	UINT	Encoder_Resolution	0 to 32,767
8010:07	UINT	Idle_current_reduction	0 to 100%
8010:08	UINT	Motor_Current (X10)	10 to 60. Represents 1.0 to 6.0 Arms
8010:09	UINT	Current Loop Gain	1 to 40

Table R6.1 Network Output Data Format: Configuration Data

CFG_Word_0 Format

Configuration Word 0

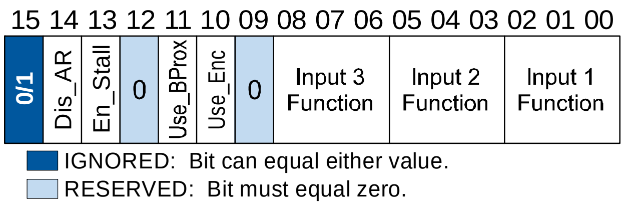


Figure R6.1 Configuration Word 0 Format

Bit 15: Reserved – State ignored.

Bit 14: Disable_Antiresonance – "1" disables the antiresonance feature. "0" enables the anti-resonance feature of the Networked Driver. The Anti-resonance feature will provide smoother operation in most cases. If you are still experiencing resonance problems with this feature enabled, disable this feature and test the machine again.

Bit 13: Enable_Stall_Detection – "0" disables motor stall detection. "1" enables motor stall detection. Only valid when an encoder is used and it is attached to the motor. The Encoder_Resolution parameter must also be programmed and must be four times the line count of your encoder.

CFG_Word_0 Format (continued)

Bit 12: **Reserved –** Must equal zero.

Bit 11: **Use_Backplane_Proximity –** “0” when the Backplace_Proximity_Bit is not used when homing the Networked Driver. “1” when the Backplace_Proximity_Bit is used when homing the Networked Driver. Note that this bit is not the Backplace_Proximity_Bit, but enables or disables its operation. Do not use the Backplace_Proximity_Bit if you only want to home to the Home Limit Switch. (Leave this bit equal to “0”.)

Bit 10: **Use_Encoder –** “0” when Quadrature Encoder is not used. “1” to enable a Quadrature Encoder. You must also configure Inputs 1 and 2 to accept quadrature signals and enter a value in Word 6 for the Encoder_Resolution parameter in addition to setting this bit.

Bit 9: **Reserved –** Must equal zero.

Bits 8-6: **Input 3 Function –** See Table Below

Bits 5-3: **Input 2 Function –** See Table Below

Bits 2-0: **Input 1 Function –** See Table Below

Bits				
8	7	6		
5	4	3		
2	1	0	Function	Available On
0	0	0	General Purpose Input	All Inputs - The input is not used in any functions of the Networked Driver, but it’s status is reported in the Network Data. This allows the input to be used as a discrete DC input to your controller.
0	0	1	CW Limit	All Inputs
0	1	0	CCW Limit	All Inputs
0	1	1	Start Indexed Move	All Inputs
0	1	1	Start Indexed Move / Capture Encoder Value	All Inputs, or Input 3 when Inputs 1 & 2 are configured as quadrature encoder inputs. The encoder position value is captured whenever Input 3 transitions. An inactive-to-active state transition will also trigger an Indexed Move if one is pending in the Networked Driver.
1	0	0	Stop Jog or Registration Move	All Inputs
1	0	0	Stop Jog or Registration Move & Capture Encoder Value	All Inputs, or Input 3 when Inputs 1 & 2 are configured as quadrature encoder inputs. The encoder position value is captured when Input 3 triggers a controlled stop to a Jog or Registration move.
1	0	1	Emergency Stop	All Inputs
1	1	0	Home	All Inputs when using discrete sensors. Input 3 only when using quadrature encoder. Use this configuration value when homing to the Z-Pulse of an encoder.
1	1	1	Quadrature Encoder Input	Inputs 1 and 2 only. Input 1 is channel A. Input 2 is channel B.

Table R6.2 Configuration Data: Input Function Selections

NOTE When using a quadrature encoder, you must set bit 10, the Use_Encoder Bit, in addition to programming the inputs to accept quadrature signals. You must also program the Encoder_Resolution parameter in configuration word 6.

CFG_Word_0 Format (continued)

To Make These Settings...		...Set These Bits To															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input 1①	General Purpose				Bit 12 is Reserved. It must always be set to zero.			Bit 9 is Reserved. It must always be set to zero.							0	0	0
	CW Limit														0	0	1
	CCW Limit														0	1	0
	Start Indexed Move†														0	1	1
	Stop Jog/Reg. Move†														1	0	0
	E-Stop														1	0	1
	Home														1	1	0
	Encoder Channel A														1	1	1
Input 2①	General Purpose											0	0	0			
	CW Limit											0	0	1			
	CCW Limit											0	1	0			
	Start Indexed Move②											0	1	1			
	Stop Jog/Reg. Move②											1	0	0			
	E-Stop											1	0	1			
	Home											1	1	0			
	Encoder Channel B											1	1	1			
Input 3①	General Purpose								0	0	0						
	CW Limit								0	0	1						
	CCW Limit								0	1	0						
	Start Indexed Move②								0	1	1						
	Stop Jog/Reg. Move②								1	0	0						
	E-Stop								1	0	1						
	Home								1	1	0						
Use_Encoder③	Disabled						0										
	Enabled						1										
Use_Backplane_ Proximity	Disabled					0											
	Enabled					1											
Stall_Detection④	Disabled			0													
	Enabled			1													
Antiresonance	To Enable		0														
	To Disable		1														
Resulting Bit Pattern:		**0**			**0**			**0**									
Hexadecimal Digits for CoE Configuration Setting:																	

① Configuring two or more inputs to have the same function, such as two CW Limit Switches, will cause a configuration error. (An error does not occur if two or more inputs are configured as General Purpose Inputs.)

② If the encoder is enabled, the encoder position will be trapped and reported when the input makes an inactive-to-active transition.

③ Enabling the encoder will cause a Configuration Error if Input 1 & 2 are not programmed as Encoder Channel A & B respectively.

④ Enabling Stall_Detection will cause a Configuration Error if the encoder is not enabled. (Bit 10 must equal “1”.)

Table R6.3 Configuration Word 0 Bits

CFG_Word_1 Format

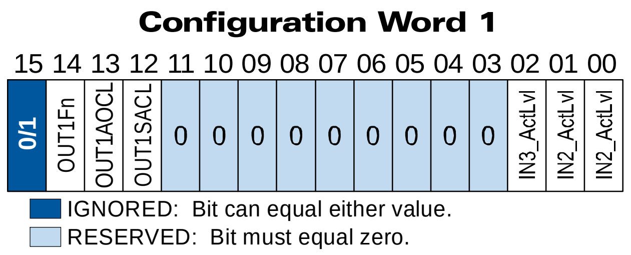


Figure R6.2 Configuration Mode: Config Word Format

Bit 15: Reserved – State ignored.

Bit 14: OUT1_Function – “0” configures Output 1 to be a Fault Output. The output will conduct current until a fault occurs. “1” configures Output 1 to be a general purpose output whose state is determined by a bit in the Command Mode Network Output Data. This output is in an ON state when power is applied to the Networked Driver and it has not yet been configured.

Bit 13: OUT1_Action_on_Connection_Lost – This bit is only acted upon if Output 1 is configured as a general purpose output. (Bit 14 above set to “1”.) A “0” on this bit will keep Output 1 at its last value if the network connection is lost. A “1” on this bit will set the state of Output 1 to the value specified in Bit 12 of this word.

Bit 12: OUT1_State_at_Connection_Lost – This bit is only acted upon if Output 1 is configured as a general purpose output. (Bit 14 above set to “1”.) When bit 13 of this word is set, Output 1 will be set to the state of this bit if the network connection is lost.

Bits 11 - 7: Reserved – Must equal zero.

Bit 6: DC_Sync – When set to “0”, the Networked Driver begins a move as soon as data is read from the EtherCAT Slave Controller (ESC). When set to “1”, the Networked Driver will synchronize the move to the SYNC0 output of the system’s Distributed Clock. This allows you to synchronize moves over multiple devices.

Bits 5 - 3: Reserved – Must equal zero.

Bit 2: IN3_Active_Level – Determines the active state of Input 3. Set to “0” if your sensor has Normally Closed (NC) contacts and the input is active when there is no current flow through it. Set to “1” if your sensor has Normally Open (NO) contacts and current flows through the input when it is active.

Bit 1: IN2_Active_Level – Determines the active state of Input 2. Set to “0” if your sensor has Normally Closed (NC) contacts and the input is active when there is no current flow through it. Set to “1” if your sensor has Normally Open (NO) contacts and current flows through the input when it is active.

Bit 0: IN1_Active_Level – Determines the active state of Input 1. Set to “0” if your sensor has Normally Closed (NC) contacts and the input is active when there is no current flow through it. Set to “1” if your sensor has Normally Open (NO) contacts and current flows through the input when it is active.

NOTE If you are not using an input, sets its Active_Level bit to “1”. The input will always report as inactive in the network data.

NOTE The AMCI Networked Driver does not report the state of the inputs when they are configured as encoder inputs (AB). The driver reports the state of the *Active Level Bit* for the input.

CFG_Word_1 Format (continued)

To Make These Settings...		...Set These Bits To															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input 1	Active Low†																0
	Active High†																1
Input 2	Active Low†															0	
	Active High†															1	
Input 3	Active Low†														0		
	Active High†														1		
DC_Sync	Disabled										0						
	Enabled										1						
OUT1_State_ at_Conn_Lost	Output Off				0												
	Output Conducting				1												
OUT1_Action_ On_Conn_Lost	Retain Last Value			0													
	Use state of bit 12			1													
OUT1_Function	Fault Output		0														
	GP Output		1														
Resulting Bit Pattern:		**0**				**0**	**0**	**0**	**0**	**0**		**0**	**0**	**0**			
Hexadecimal Digits for CoE Configuration Setting:																	

† Active Low: The Networked Driver will report that the input is active when it is not conducting current.
Active High: The Networked Driver will report that the input is active when it is conducting current.

Table R6.4 Configuration Word 1 Bits

Current Loop Gain

This feature gives you the ability to adjust the gain of the power amplifiers in the Networked Driver to match the electrical characteristics of your motor. The value of this parameter can range from 1 to 40 with 40 representing the largest gain increase. In general, using a larger gain will increase high speed torque but the motor will run louder. A lower gain will offer quieter low speed operation at the cost of some high speed torque.

The use of this feature is completely optional and you can leave the Current Loop Gain at its default setting of "5" for standard motor performance.

Assuming a stable line voltage of 115Vac, the following gains can be used for AMCI motors. These gain settings are factory suggestions and are average settings for our motors. Your system may benefit from increasing or decreasing these settings. In general, increase the setting by one or two counts to improve high speed performance or decrease the settings by one or two for quieter low speed operation.

		115 Vac GAIN SETTINGS						
	MOTOR CURRENT →	1 A	1.5 A	2 A	3 A	4 A	5 A	6 A
MOTOR PART #	SM2340-130			2	3	4		
	SM2340-240			4	5	6		
	SM34-450			6	9	11		
	SM34-850			11	14	17		
	SM34-1100			12	17	21		
	SM23-130 (Series)	4	6	7				
	SM23-130 (Parallel)			2	3	4		
	SM23-240 (Series)	6	8	10				
	SM23-240 (Parallel)			4	5	6		
	SM42-1800				10	11	12	14

Table R6.5 Optional 115Vac Current Loop Gain Settings

When using the SD31045E-K with a stable 230 Vac line voltage, the following gains can be used for AMCI motors. These gain settings are factory suggestions and are average settings for our motors. Your system may benefit from increasing or decreasing these settings. In general, increase the setting by one or two counts to improve high speed performance or decrease the settings by one or two for quieter low speed operation.

		230Vac GAIN SETTINGS					
	MOTOR CURRENT →	1 A	1.5 A	2 A	3 A	4 A	4.4 A
MOTOR PART #	SM2340-130			1	1	1	
	SM2340-240			2	3	3	
	SM34-450			3	4	5	
	SM34-850			6	7	8	
	SM34-1100			8	9	10	
	SM23-130 (Series)	2	3	3			
	SM23-130 (Parallel)			1	1	1	
	SM23-240 (Series)	3	4	5			
	SM23-240 (Parallel)			2	3	3	
	SM42-1800				5	6	7

Table R6.6 Optional 230Vac Current Loop Gain Settings

Notes on Other Configuration Words

- Changes to the Idle Current only take effect at *the end of the first move after re-configuration.*
- Motor current can be set to any value within their respective ranges. (1.0 to 6.0 amps for the SD17060E-K and 1.0 to 4.5 amps for the SD31045E-K.) When setting the current with command data, only even numbered values are accepted.

Invalid Configurations

The following configurations are invalid:

1) Setting any of the reserved bits in the configuration words.
2) Setting any parameter to a value outside of its valid range.
3) Configuring the two inputs to have the same function, such as two CW Limit Switches. (An error does not occur if both are configured as General Purpose Inputs.)
4) Setting the Input Configuration bits for Input 3 to “111”. See table R6.3 on page 59 for more information.
5) You configure the Networked Driver to use an encoder, but you do not configure Inputs 1 and 2 as Quadrature Encoder Inputs A and B.
6) You configure the Networked Driver to use an encoder, but you do not set the Encoder_Resolution value in word 6.
7) Setting the Enable_Stall_Detection Bit without configuring the Networked Driver to use an encoder.

Notes

REFERENCE 7
COMMAND MODE DATA FORMAT

This chapter covers the format of the Network Output Data used to command the SD17060E-K or the SD31045E-K and the format of the Network Input Data that contains the responses from the device. The parameter names of the input and output data are defined by the device ESI file. The size of each data element is also defined by the ESI file.

Command Bits Must Transition

Commands are only accepted when a command bit makes a 0→1 transition. The easiest way to do this is to write a value of zero into the CMD_word0 before writing the next command.

The command bits are split between two, 16 bit words, CMD_word0 and CMD_word1. Only one bit in CMD_word0 can make a 0→1 transition at a time.

Output Data Format

The following table shows the format of the output network data words when writing command data to the SD17060E-K or SD31045E-K.

ESI File Name	Data Size	Function
CMD_word0	UINT	Command Word 0
CMD_word1	UINT	Command Word 1
Position	DINT	Command Parameters Word meaning depends on the command set to the Networked Driver
Velocity	UDINT	
Acceleration	UINT	
Deceleration	UINT	
Motor_Current	UINT	
Jerk	UINT	

Figure R7.1 Command Data Format

CMD_word0

CMD_word0

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	Prst_Enc	Run_AMov	Rd_AData	Prg_Assm	RSet_Err	Prst_Pos	Jog_CCW	Jog_CW	Home_CCW	Home_CW	I-Stop	Resm_Mv	Hold_Mv	Rel_Mv	Abs_Mv

Figure R7.2 CMD_word0 Format

Bit 15: Mode_Select – "0" for Command Mode programming.

NOTE It is possible to write configuration data to the unit using the output words, but the use of this method is discouraged. Configuration data should be written to the Networked Driver using the CoE interface as described in the previous chapter.

Bit 14: Preset_Encoder – When set to "1" the Networked Driver will preset the Encoder Position to the value stored in the DINT Position output register.

Bit 13: Run_Assembled_Move – When set to "1" the Networked Driver will run the Assembled Move already stored in memory.

- **Assembled_Move_Type – CMD_word1, Bit 9:** This bit determines the type of move that is run. When this bit equals "0", a Blend Move is run. When this bit equals "1", a Dwell Move is run. When starting a Dwell Move, the Dwell Time is programmed with the UINT Jerk output register of the Command Data. The value is programmed in milliseconds and can range from 0 to 65,536.
- **Reverse_Blend_Direction – CMD_word1, Bit 4:** This bit is used to determine the direction that the Blend Move will be run in. When this bit equals "0", the Blend Move runs in the clockwise direction. When this bit equals "1", the Blend Move is run in the counter-clockwise direction.

Bits 11 & 12: Program_Assembled & Read_Assembled_Data – These bits are used to program the segments of an Assembled Move before the move can be run. Their use is explained in the ***Assembled Move Programming*** section of this manual starting on page 35.

Bit 10: Reset_Errors – When set to "1" the Networked Driver will clear all existing errors and reset the *Move_Complete* bit. A driver fault will be cleared if the Clear_Driver_Fault bit, (CMD_word1, bit 10), is set to "1" when the Reset Errors command is issued.

Bit 9: Preset_Position – When set to "1" the Networked Driver will preset the Motor Position to the value stored in the DINT Position output register and reset the Move_Complete bit in the Network Input Data.

Bit 8: Jog_CCW – When set to "1" the Networked Driver will run a Jog Move in the counter-clockwise direction. The full explanation of a ***CW/CCW Jog Move*** can be found starting on page 27.

- **Registration_Move – CMD_word1, Bit 7:** When this bit equals "0", and a Jog Move command is issued, it will run as a standard Jog Move. When this bit equals "1" and a Jog Move command is issued, the move will run as a Registration Move.
- **Enable_Electronic_Gearing – CMD_word1, Bit 6:** When this bit equals "1" the Networked Driver will switch its operation to ***Electronic Gearing*** mode as described on page 38. While in this mode, the two Jog Move bits are used to enable motor motion. One of these two bits must equal "1" before the motor will follow a change in encoder position.

CMD_word0 (continued)

Bit 7: **Jog_CW –** When set to “1” the Networked Driver will run a Jog Move in the clockwise direction. The full explanation of a ***CW/CCW Jog Move*** can be found starting on page 27.

- **Registration_Move – CMD_word1, Bit 7:** When this bit equals “0”, and a Jog Move command is issued, it will run as a standard Jog Move. When this bit equals “1” and a Jog Move command is issued, the move will run as a Registration Move.
- **Enable_Electronic_Gearing – CMD_word1, Bit 6:** When this bit equals “1” the Networked Driver will switch its operation to ***Electronic Gearing*** mode as described on page 38. While in this mode, the two Jog Move bits are used to enable motor motion. One of these two bits must equal “1” before the motor will follow a change in encoder position.

Bit 6: **Find_Home_CCW –** When set to “1” the Networked Driver will attempt to move to the Home Limit Switch in the counter-clockwise direction. A full explanation of homing can be found in the ***Homing an AMCI Networked Driver*** reference chapter starting on page 51.

Bit 5: **Find_Home_CW –** When set to “1” the Networked Driver will attempt to move to the Home Limit Switch in the clockwise direction. A full explanation of homing can be found in the ***Homing an AMCI Networked Driver*** reference chapter starting on page 51.

Bit 4: **Immediate_Stop –** When set to “1” the Networked Driver will stop all motion without deceleration. The Motor Position value will become invalid if this bit is set during a move. Setting this bit when a move is not in progress will not cause the Motor Position to become invalid.

Bit 3: **Resume_Move –** Set to “1” to resume a move that you previously placed in a hold state. Use of the Resume_Move and Hold_Move bits can be found in the ***Controlling Moves In Progress*** section of this manual starting on page 37. Note that a move in its hold state does not need to be resumed. The move is automatically cancelled if another move is started in its place.

Bit 2: **Hold_Move –** Set to “1” to hold a move. The move will decelerate to its programmed Starting Speed and stop. The move can be completed by using the Resume_Move bit. Use of the Hold_Move and Resume_Move bits can be found in the ***Controlling Moves In Progress*** section of this manual starting on page 37.

Bit 1: **Relative_Move –** Set to “1” to perform a Relative Move using the data in the rest of the Command Data. The full explanation of a ***Relative Move*** can be found starting on page 25.

Bit 0: **Absolute_Move –** Set to “1” to perform an Absolute Move using the data in the rest of the Command Data. The full explanation of an ***Absolute Move*** can be found starting on page 26.

CMD_word1

CMD_word1

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
En_Driver	0	OUT1_State	0	BP_Prox	Clr_Drv_Flt	AsMv_Type	Index_Cmd	Reg_Move	En_ElGear	SvA_to_Flash or Current_Key2	Rev_BlendDir	0	Enc_Reg_Mv	Current_Key1	Current_Key0

Figure R7.3 CMD_word1 Format

Bit 15: Enable_Driver – "0" to disable the motor current, "1" to enable motor current. A valid configuration must be written to the Networked Driver before the driver can be enabled.

Bit 14: Reserved – Must be set to "0".

Bit 13: OUT1_Set_State – When the output is configured as a general purpose output point instead of the Fault Output, this bit controls the state of the output. When this bit equals a "1", the output is on and conducts current.

Bit 12: Reserved – Must be set to "0".

Bit 11: Backplace_Proximity_Bit – When the Networked Driver is configured to use the Backplace_Proximity_Bit, the unit will ignore the state of the Home Input as long as this bit equals "0". This bit must equal "1" before a transition on the Home Input can be used to home the machine. Further information on using the Backplace_Proximity_Bit can be found in the ***Profile with Network Backplace_Proximity_Bit*** section found on page 53.

Bit 10: Clear_Driver_Fault – If this bit is set when a Reset Errors Command is issued, (CMD_word0, Bit 10) the Networked Driver will attempt to clear driver errors such as a missing interlock jumper or motor short fault. Note that the driver must be disabled (CMD_word1, Bit 15 = 0), when using this command.

Bit 9: Assembled_Move_Type – When this bit equals "0", a Blend Move is started when the Run Assembled Move bit, (CMD_word1, Bit 13) makes a 0→1 transition. When this bit equals "1", a Dwell Move is started on the transition. The direction of a Blend Move is controlled by the Reverse_Blend_Direction bit, (CMD_word1, Bit 4). In a Dwell Move, the Dwell Time between segments is programmed with the UINT Jerk register of the command data.

Bit 8: Indexed_Command – If this bit is set when a move command is issued, the Networked Driver will not run the move immediately, but will instead wait for an inactive-to-active transition on an input configured as a *Start Indexer Move* input.

Bit 7: Registration_Move – When this bit equals "0", and a Jog Move command is issued, it will run as a standard Jog Move. When this bit equals "1" and a Jog Move command is issued, the move will run as a Registration Move.

Bit 6: Enable_Electronic_Gearing – Set to "1" to put the Networked Driver in Electronic Gearing mode. Set to "0" for normal operation. A full description of Electronic Gearing mode starts on page 23.

Bit 5: Save_Assembled_to_Flash - Set this bit to save the data of a programmed Assembled Move. This bit is only acted upon this way when the Program_Assembled bit (CMD_word0, Bit 11 makes a 1→0 transition as explained in the ***Assembled Move Programming*** section of this manual starting on page 35.
OR
Motor_Current_Key2 - See ***Description of Motor Current Keys*** on the following page.

CMD_word1 (continued)

Bit 4: **Reverse_Blend_Direction –** When you command a Blend Move to run, this bit determines the direction of rotation. Set to "0" for a clockwise Blend Move, '1' for a counter-clockwise Blend Move.

Bit 3: **Reserved –** Must be set to "0".

Bit 2: **Encoder_Registration_Move –** This bit is used with the Relative_Move (CMD_word0, bit 1) or Absolute_Move (CMD_word0, bit 0) bits. Set this bit to "1" to command an Encoder Registration Move. This move will run until the encoder count reaches the programmed relative or absolute value and then the move will begin to decelerate and stop. Must equal "0" for normal relative or absolute moves. A full description of an ***Encoder Registration Move*** can be found starting on page 14.

Bit 1: **Motor_Current_Key1 –** See the description below.

Bit 0: **Motor_Current_Key0 –** See the description below.

Description of Motor Current Keys

With the exceptions of the CW/CCW Registration Move commands, it is possible to change the motor current "on the fly" while in Command Mode. To do this, place the new motor current in the UINT Motor_Current register and set the key bits as shown below.

- CMD_word1: Bit 5, (Key2) – "1"
- CMD_word1: Bit 1, (Key1) – "0"
- CMD_word1: Bit 0, (Key0) – "1"

This feature does not work with CW/CCW Registration Move commands because the last two words in the command data are used to store the Minimum Registration Distance parameter. For all other commands, the Networked Driver will respond by changing the motor current and writing the new value into the Network Input Data, Word 8. Once the motor current has been set, reset the key bits to all zeros to prevent further changes to the motor current. The motor current can be changed at any time, including during a move.

NOTE Setting the motor current to an invalid value does not force a command error. The value will be ignored and the Networked Driver will use the motor current value programmed while in Configuration Mode. Invalid values are:

- Values whose last digit is not even
- Values outside the range of 1.0 to 6.0 amps on the SD17060E-K
- Values outside the range of 1.0 to 4.4 amps on the SD31045E-K

Command Blocks

The following section lists the output data format for the sixteen different commands.

Absolute Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0001
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Absolute Target Position	Steps	–8,388,608 to +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.1 Absolute Move Command Block

Relative Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0002
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Relative Target Position	Steps	–8,388,608 to +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.2 Relative Move Command Block

Command Blocks (continued)

Hold Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0004
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R7.3 Hold Move Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Resume Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0008
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.4 Resume Move Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values from the previous command. This is typically the case when resuming a move, the words are listed as “Unused” to highlight that the target position of a held move cannot be changed when the move is resumed.

Command Blocks (continued)

Immediate Stop

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	*CMD_word0*		16#0010
CMD_Word1	UINT	*CMD_word1*		See pg. 68
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R7.5 Immediate Stop Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Find Home CW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	*CMD_word0*		16#0020
CMD_Word1	UINT	*CMD_word1*		See pg. 68
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.6 Find Home CW Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Command Blocks (continued)

Find Home CCW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0040
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.7 Find Home CCW Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Jog CW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0080
CMD_Word1	UINT	***CMD_word1***		See pg. 68 Bits 7 & 6 must equal “00”
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.8 Jog Move CW Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Command Blocks (continued)

Registration Move CW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0080
CMD_Word1	UINT	***CMD_word1***		See pg. 68 Bits 7 & 6 must equal “10”
Position	DINT	Stopping Distance	Steps	0 to +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current†	UINT	Minimum Registration Move Distance	steps	UDINT value between 0 and +8,388,607
Jerk†	UINT			

Table R7.9 Registration Move CW Command Block

† Motor Current and Jerk parameters cannot be programmed as part of a Registration Move. These two words are used to store the UDINT value of the Minimum Registration Move Distance. The structured text example below splits the thirty-two bit Minimum Registration Move Distance (nMRMDist) into the two sixteen bit registers. This code was written with TwinCAT software.

```
Motor_Current := UDINT_TO_UINT(nMRMDist);  //Motor_Current stores the lower 16 bits.
Jerk := UDINT_TO_UINT(SHR(nMRMDist, 16));  //Jerk stores the upper 16 bits.
```

Jog CCW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0100
CMD_Word1	UINT	***CMD_word1***		See pg. 68 Bits 7 & 6 must equal “00”
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.10 Jog CCW Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Command Blocks (continued)

Registration Move CCW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0100
CMD_Word1	UINT	***CMD_word1***		See pg. 68 Bits 7 & 6 must equal “10”
Position	DINT	Stopping Distance	Steps	0 to +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current†	UINT	Minimum Registration Move Distance	steps	UDINT value between 0 and +8,388,607
Jerk†	UINT			

Table R7.11 Registration Move CCW Command Block

† Motor Current and Jerk parameters cannot be programmed as part of a Registration Move. These two words are used to store the UDINT value of the Minimum Registration Move Distance. The structured text example below splits the thirty-two bit Minimum Registration Move Distance (nMRMDist) into the two sixteen bit registers. This code was written with TwinCAT software.

```
Motor_Current := UDINT_TO_UINT(nMRMDist);  //Motor_Current stores the lower 16 bits.
Jerk := UDINT_TO_UINT(SHR(nMRMDist, 16));  //Jerk stores the upper 16 bits.
```

Command Blocks (continued)

Encoder Follower Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0080 or 16#0100
CMD_Word1	UINT	***CMD_word1***		See pg. 68 Bit 6 must equal “1”
Position†	DINT	Upper 16 bits: EG Numerator		Value between 1 and 255
		Lower 16 bits: EG Denominator		Value between 1 and 255
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Motor Current	0.1 amps	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Unused		See Note Below

Table R7.12 Encoder Follower Move Command Block

Unused words are ignored by the Network Driver and can be any value, including parameter values from the previous command.

† The 32 bit position register is used to store the Electronic Gearing Numerator and Denominator values. Both of these values are USINT values. The structured text example below combines the two values into the one 32 bit register. This code was written with TwinCAT software.

```
//Denominator stored in lower 16 bits.
Position := USINT_TO_DINT(nEGDenominator);
//Numerator converted, shifted and ANDed into the upper 16 bits.
Position := Position AND SHL(USINT_TO_DINT(nEGNumerator), 16);
```

Preset Position

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0200
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Position Preset Value	Steps	–8,388,608 to +8,388,607
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R7.13 Preset Position Command Block

Unused words are ignored by the Network Driver and can be any value, including parameter values in the previous command.

Presetting the position will also reset the *Move_Complete* status bit in the Network Input Data.

Command Blocks (continued)

Reset Errors

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	*CMD_word0*		16#0400
CMD_Word1	UINT	*CMD_word1*		See pg. 68 Set bit 10 to clear driver faults
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R7.14 Reset Errors Command Block

Unused words are ignored by the Network Driver and can be any value, including parameter values in the previous command.

- Issuing a Reset Errors command will reset the *Move_Complete* status bit in the Network Input Data.
- Issuing a Reset Errors command will not reset the *Position_Invalid* or *Configuration_Error* bits.

Run Assembled Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	*CMD_word0*		16#2000
CMD_Word1	UINT	*CMD_word1*		See pg. 68 Blend Move: Bit 9 = "0" Dwell Move: Bit 9 = "1" The Blend Move direction is controlled by Bit 4.
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Motor Current	0.1 A	SD17060E-K: 10 to 60 SD31045E-K: 10 to 44 *Even numbers only.*
Jerk	UINT	Unused with Blend Move Dwell Time with Dwell Move	milliseconds	0 to 65,535

Table R7.15 Run Assembled Move Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Command Blocks (continued)

Preset Encoder Position

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#4000
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Encoder Preset Value	Counts	–8,388,608 to +8,388,607
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R7.16 Preset Encoder Position Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Programming Blocks

The following blocks are used to program an Assembled Move. Both of the moves, Blend Move, and Dwell Move, are programmed exactly the same way. The bit configuration used when starting the move determines which type of Assembled Move is run.

First Block

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0800
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R7.17 Assembled Move First Programming Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values from the previous command.

Once the first block is transmitted, the Networked Driver responds by setting bits 8 and 9 in STATUS_word0. (See ***STATUS_word0 Format*** starting on page 80.) Once these are set, you can then start transmitting Segment Blocks.

Segment Block

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#1800
CMD_Word1	UINT	***CMD_word1***		See pg. 68
Position	DINT	Relative Target Position	Steps	–8,388,608 to +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R7.18 Assembled Move Segment Programming Block

Note that each Segment Block starts with bits 11 and 12 set in the CMD_word0 word (16#1800). When the Networked Driver sees bit 12 of CMD_word0 set, it will accept the block and reset bit 9 in STATUS_word0. When your program sees this bit reset, it must respond by resetting bit 12 of CMD_word0. The Networked Driver will respond to this by setting bit 9 in STATUS_word0 and the next Segment Block can be written to the Networked Driver. You can write a maximum of sixteen Segment Blocks for each Assembled Move.

Input Data Format

The correct format for the Network Input Data when the Networked Driver is in Command Mode is shown below.

ESI File Name	Data Size	Function
STATUS_word0	UINT	Status Word 0
STATUS_word1	UINT	Status Word 1
Motor_Position	DINT	Current commanded motor position
Encoder_Position	DINT	Current encoder position. (Zero if encoder is not available or disabled.)
Trapped_Encoder_Position	DINT	Last captured encoder position
Motor_Current	UINT	Motor current of last move.†
Jerk	UINT	Jerk value of last move.

† Note that this is the value of the most recently programmed motor current. It is not the actual motor current being sent to the motor when the input data was read.

Table R7.19 Network Input Data Format: Command Mode

STATUS_word0 Format

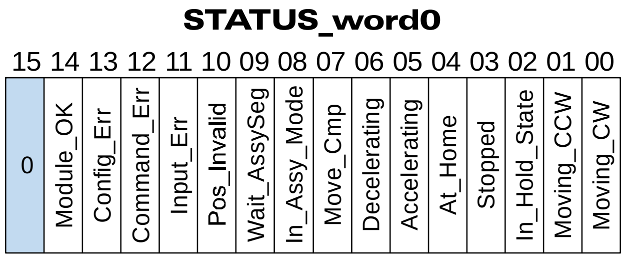


Figure R7.4 Command Mode: Status_word0 Format

Bit 15: Mode_Flag – Set to "0" in Command Mode.

Bit 14: Module_OK – "1" when the Networked Driver is operating without a fault, "0" when an internal fault condition exists.

Bit 13: Configuration_Error – "1" on power up before a valid configuration has been written to the Networked Driver or after any invalid configuration has been written to the driver. "0" when the Networked Driver has a valid configuration written to it.

Bit 12: Command_Error – "1" when an invalid command has been written to the Networked Driver. This bit can only be reset by the Reset_Errors bit, CMD_word0, Bit 10. Note that setting the motor current to a value outside the range of 1.0 to 6.0 amps on the SD17060E-K, or 1.0 to 4.4 amps on the SD31045E-K, or an odd number within these ranges, does not force a command error. The driver simply ignores the new value and uses the last accepted value for the motor current.

Input Data Format (continued)

STATUS_word0 Format (continued)

Bit 11: Input_Error – "1" when:

- ➤Emergency Stop input has been activated
- ➤Either of the End Limit Switches activates during any move operation except for homing
- ➤Starting a Jog Move in the same direction as an active End Limit Switch
- ➤If the opposite End Limit Switch is reached during a homing operation.
- ➤A SyncManager 2 timeout event has occurred. This is indicative of a networking error that occurred during the EtherCAT Pre-OP or OP states.

This bit is reset by a ***Reset Errors*** command. The format of the command is given on page 77.

Bit 10: Position_Invalid – "1" when:

- ➤A configuration change is written to the unit through the C0E SDO interface.
- ➤The motor position has not been preset
- ➤The machine has not been homed
- ➤An Immediate or Emergency Stop has occurred
- ➤An End Limit Switch has been reached
- ➤A motor stall has been detected.

Absolute moves cannot be performed while the position is invalid.

Bit 9: Waiting_For_Assembled_Segment – The Networked Driver sets this bit to tell the host that it is ready to accept the data for the next segment of your assembled move profile. Its use is explained in the ***Assembled Move Programming*** section of this manual starting on page 35.

Bit 8: In_Assembled_Mode – The Networked Driver sets this bit to signal the host that it is ready to accept assembled move profile programming data. Its use is explained in the ***Assembled Move Programming*** section of this manual starting on page 35.

Bit 7: Move_Complete – Set to "1" when the present Absolute, Relative, Jog, Registration, or Assembled Move command completes without error. This bit is reset to "0" when the next move command is written to the Networked Driver, when the position is preset, or a Reset Errors command is issued to the unit. This bit is also set along with the Command_Error bit (Bit 12 of this word), when any Jog Move or Registration Move parameters are outside of their valid ranges. This bit is not set on a command error for any other type of command. Finally, this bit is not set at the end of a homing operation.

Bit 6: Decelerating – Set to "1" when the present move is decelerating. Set to "0" at all other times.

Bit 5: Accelerating – Set to "1" when the present move is accelerating. Set to "0" at all other times.

Bit 4: At_Home – Set to "1" when a homing command has completed successfully, "0" at all other times.

Bit 3: Stopped – Set to "1" when the motor is not in motion. Note that this is stopped for any reason, not just a completed move. For example, an Immediate Stop command during a move will set this bit to "1", but the Move_Complete Bit, (bit 7 above) will not be set.

Bit 2: In_Hold_State – Set to "1" when a move command has been successfully brought into a Hold State. Hold States are explained is the Controlling Moves In Progress section starting on page 22.

Bit 1: Moving_CCW – Set to "1" when the motor is rotating in a counter-clockwise direction.

Bit 0: Moving_CW – Set to "1" when the motor is rotating in a clockwise direction.

Input Data Format (continued)

STATUS_word1 Format

STATUS_word1

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Drive_Enabled	Stall_Detected	OUT1_State	0	Heartbeat_Bit	Limit_Condition	Inv_Jog_Par	0	Driver_Fault	0	SafeOp	Temp_90C	0	IN3Active	IN2_Active	IN1_Active

Figure R7.5 Command Mode: STATUS_word1 Format

Bit 15: **Drive_Is_Enabled –** Set to "1" when the motor driver section of the Networked Driver is enabled and current is available to the motor. Set to "0" when the motor driver section is disabled. If this bit is set to "1", the motor current remains present when an E-Stop input is active. This bit will remain set if the motor current has been removed because motion is not occurring and the Idle Current Reduction is programmed to its 0% setting. Motor current is removed if there is a Driver_Fault (Bit 7 below) regardless of the state of this bit.

Bit 14: **Stall_Detected –** Set to "1" when a motor stall has been detected.

Bit 13: **OUT1_State –** Present actual state of Output 1. When this bit is set to "1", the output is in its on state and conducts current.

Bit 12: **Reserved Bit –** Will always equal zero.

Bit 11: **Heartbeat_Bit –** This bit will change state approximately every 500 milliseconds. Monitor this bit to verify that the unit and network connection are operating correctly.

Bit 10: **Limit_Condition –** This bit is set if an End Limit Switch is reached during a move. This bit will be reset when the Limit Switch changes from its active to inactive state, or when a Reset Errors Command is issued.

Bit 9: **Invalid_Jog_Change –** Set during a Jog Move if parameters are changed to invalid values. Parameters that can be changed during a Jog Move are Programmed Speed, Acceleration, and Deceleration. Set while in Electronic Gearing mode if the Numerator or Denominator are set outside their range of 1 to 255.

Bit 8: **Reserved –** Will always equal zero.

Bit 7: **Driver_Fault –** If the driver section of the Networked Driver is enabled, this bit will be a "1" during a Over temperature Fault, a Short Circuit Fault, or when the Interlock Jumper is missing. This fault can be cleared by issuing a ***Reset Errors*** programming block with the Clear_Driver_Fault bit, (CMD_word1, bit 10) set to "1". For additional information, see ***Notes on Clearing a Driver Fault*** on page 83. Note that the driver fault bit is set at power up if the interlock is missing and the driver is immediately enabled.

Bit 6: **Reserved –** Will always equal zero.

Bit 5: **SafeOp –** Set to "1" when the network is in Safe_Op mode. Moves cannot be commanded while in this mode. Reset to "0" when the network is in Operational mode. Moves can be commanded when in Op mode.

Bit 4: **Temperature_Above_90C –** This bit is set to "1" when the processor internal temperature exceeds 90°C. At this point, the heatsink temperature is typically above 80°C. If this bit trips often and you want to lower the operating temperature of the unit, consider changing how the device is mounted, or installing a fan to force additional airflow through the device.

Bit 3: **Reserved –** Will always equal zero.

Input Data Format (continued)

STATUS_word1 Format (continued)

Bit 2: **IN3_Active –** "1" when Input 3 is in its active state. The active state of the input is programmed as explained in the ***CFG_Word_1 Format*** section starting on page 60.

Bit 1: **IN2_Active –** "1" when Input 2 is in its active state. The active state of the input is programmed as explained in the ***CFG_Word_1 Format*** section starting on page 60.

Bit 0: **IN1_Active –** "1" when Input 1 is in its active state. The active state of the input is programmed as explained in the ***CFG_Word_1 Format*** section starting on page 60.

NOTE The Networked Driver does not report the state of the inputs when they are configured as encoder inputs (AB). The driver reports the state of the *Active Level Bit* for the input which is set in Configuration Mode.

Notes on Clearing a Driver Fault

A Driver Fault occurs when there is an over temperature condition, a short circuit condition in the motor, or the Interlock jumper is missing.

When a Driver Fault occurs, the driver sets bit 7 of STATUS_word1 in the Network Input Data. (See ***STATUS_word1 Format*** on page 82 for a full description of STATUS_word1.) *Once you have cleared the fault condition,* you can reset the Driver Fault with the following Command Block:

Reset Driver Fault

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	***CMD_word0***		16#0400
CMD_Word1	UINT	***CMD_word1***		16#0400
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R7.20 Reset Driver Fault Command Block

Unused words are ignored by the Networked Driver and can be any value, including parameter values in the previous command.

Once the command block is accepted by the Networked Driver, it will respond by resetting bit 7 in STATUS_word1 of the Network Input Data.

NOTE After this procedure, there will still be no current to the motor. This is because the Enabled_Driver Bit, (bit 15 of CMD_Word1 in the Network Output Data), must be reset when writing down the Reset Driver Fault Command Block. Setting this bit in the next command block will re-enable the motor.

Notes

REFERENCE 8

GENERAL INSTALLATION GUIDELINES

This chapter gives general information on installing electronic controls in an industrial environment including the importance of proper wiring, grounding, and surge suppression. If you are responsible for installing an SD17060E-K or SD31045E-K, make sure you are familiar with these practices and follow them when installing the unit.

These instructions assume a solidly grounded system, which is used in a vast majority of modern industrial systems. As defined by the IEEE, a solidly grounded system is one in which the neutral of a generator, power transformer, or grounding transformer is directly connected to the system ground or earth. The installation of ungrounded systems is not covered by these instructions.

WARNING This chapter is presented as a tool in the hopes of avoiding common installation problems. It is not a substitute for the safety practices called out in local electrical codes or, in the United States, the National Electrical Code published by the National Fire Protection Association. If *any* conflicts exist, local and national codes must be followed. *It is the responsibility of the user* to determine what installation practices must be followed to conform to all local and national codes.

Safety is the single most important objective of any electrical installation. The system must be safe to use and it must be able to detect unsafe conditions and remove power when these conditions occur.

The second important objective of an electrical installation is proper and consistent operation. Proper operation can be achieved through:

- Proper Grounding
- Suppression of electrical noise generated by the machine.
- Suppression of electrical noise coming in from the environment. (Other machines, power surges, etc.)

Grounding

Proper grounding is the single most important consideration for a safe installation. Proper grounding also ensures that unwanted electrical currents, such as those induced by electromagnetic noise, will be quickly shunted to ground instead of flowing through the machine.

Earth ground connections on electronic controls are one of two types.

- **Protective Earth Connection:** Point that must be tied to Earth for the safe operation of the device. (Protection against electric shock should the device be damaged.)

- **Functional Earth Connection:** Point that must be tied to Earth to improve noise immunity of the device.

For all AMCI Networked Drivers, the Protective Earth Connection is any mounting surface of the unit. (The SD31045E-K Drivers also have a grounding lug on their heatsinks.) The Functional Earth Connection of the Networked Drivers is the ground terminal (GND) on the power connector.

Grounding Electrode System

The Grounding Electrode System is the common name for the building's earth ground infrastructure. This system *defines* earth-ground potential within a building and is the central ground for all electrical equipment.

Ground Bus

AMCI strongly suggests the use of a ground bus in each system enclosure. The ground bus is simply a metal bar with studs or tapped holes for accepting grounding connections in the enclosure. The ground bus is the only component in the enclosure that is directly connected to your Grounding Electrode System. Therefore, the ground bus is becomes the central grounding point for the enclosure and its equipment.

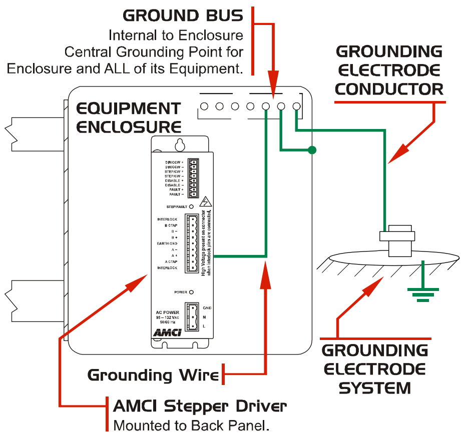
Grounding (continued)

Grounding Electrode Conductor

Connection of the ground bus to the Grounding Electrode System is made with the Grounding Electrode Conductor. The Grounding Electrode Conductor should be the shortest length of stranded wire or braid possible. The American NEC allows a length of up to six feet, but shorter is better. A shorter length will help shunt high frequency noise to the Grounding Electrode System.

In extremely noisy environments, it is sometimes advisable to run two wires from the ground bus to the GES. The length of these two wires should differ by 20% to 30% so they offer different impedances to the noise frequencies being shunted to earth ground.

The minimum sizes for the Grounding Electrode Conductor are #8 AWG stranded wire or 1" braid, but larger sizes may be needed depending on the total ground fault current your system can generate. The actual size is determined by the size of the largest conductor in the system.


GROUND BUS
Internal to Enclosure
Central Grounding Point for Enclosure and ALL of its Equipment.
GROUNDING ELECTRODE CONDUCTOR
EQUIPMENT ENCLOSURE
GROUNDING ELECTRODE SYSTEM
Grounding Wire
AMCI Stepper Driver
Mounted to Back Panel.
Figure R8.1 Grounding Components

The American NEC allows the use of a solid wire for the Grounding Electrode Conductor but this should be avoided. High frequency currents (electrical noise) travel along the surface of a conductor. Heavy gauge stranded wire and braid both have large surface areas, solid conductors do not. Therefore, stranded wire and braid have a much lower effective resistance to electrical noise.

Grounding Wires

Grounding wires are used to connect the pieces of equipment to your ground bus. AMCI strongly suggests using #8 AWG stranded wire or 1" wire braid for all grounding connections. The American NEC allows the use of solid wire for grounding wires but this should be avoided for the same reasons they should not be used for the Grounding Electrode Conductor.

Avoiding Grounding Problems

As stated before, there are two reasons to properly ground a system. The first reason is to protect human life. A properly designed grounding system will be able to detect potential shock currents and remove power from the system. The second reason is to improve system reliability by shunting electrical noise currents to earth instead of allowing them to flow through the system where they can disrupt the operation of electronic controls.

The entire grounding system, even though conductive, is not meant to carry current except under fault conditions.

NOTE **Under normal operating conditions, the grounding system should not carry any current at all.**

Ground Fault Interrupt (GFI) circuits depend on this condition to work correctly. Any ground current flowing through a GFI is considered a fault condition and the GFI immediately opens to remove power from the faulted circuit.

By this definition, electrical noise and other transient currents such as inductive spikes are considered fault currents. The fact that these currents are very short lived and of relatively small power allows them to be safely shunted without tripping a Ground Fault Interrupt circuit.

Avoiding Grounding Problems (continued)

Grounding problems occur when the grounding system carries currents during normal operation or when it cannot shunt high frequency electrical currents to earth.

AMCI generally encounters three types of grounding problems in systems that use AMCI equipment.

- **Ground Loops:** A ground loop occurs when AC or DC return currents can travel through the system ground path in addition to their normal return path. This can cause damaging currents to electronics that share the power supply.

  Ground loops in AC systems can occur when the neutral conductor is bonded to the Grounding Electrode System in more than one place. Under the right conditions, AC line currents may end up flowing through the protective ground system, which may include exposed metal surfaces and therefore represent a touch shock hazard. Designing an effective ground fault detection system is well beyond the scope of this manual. Please refer to the NEC and other design guidelines for your area.
- **Ground Shift:** Remote machines or monitoring stations will usually be connected to a different point on the Grounding Electrode System than a local system. A voltage potential can exist between these stations due to resistance between the grounding electrodes and earth. A similar problem exists on machines that are not properly bonded together. The resistance of a poor bond in the system will result in a voltage potential across the connection when current is forced through it. Incorrectly installed sensor or communications cables that run between these systems can be damaged by these Ground Shift potentials.
- **High Impedance Grounds:** The grounding system shows a high impedance to high frequency noise currents and these signals are not properly shunted to earth. The high impedance to high frequency noise is caused by capacitance and inductance in the system. Use #8 AWG stranded wire or 1” wire braid for all ground connections and keep connections as short as possible to minimize capacitance and inductance in the grounding system.

Surge (EMI) Suppression

Incoming Power

In many systems, three phase power is brought to the machine and the control system is powered from one of these phases. In these cases, good quality, clean AC power can only be achieved by properly conditioning the three phase power, not just the single phase used by the control system. This is generally achieved by placing surge suppressors across each phase where the three phase enters the system and at all inductive loads that are on the three phase branch. This includes inductive loads *on any other machine* that may be powered by the same feeder circuit.

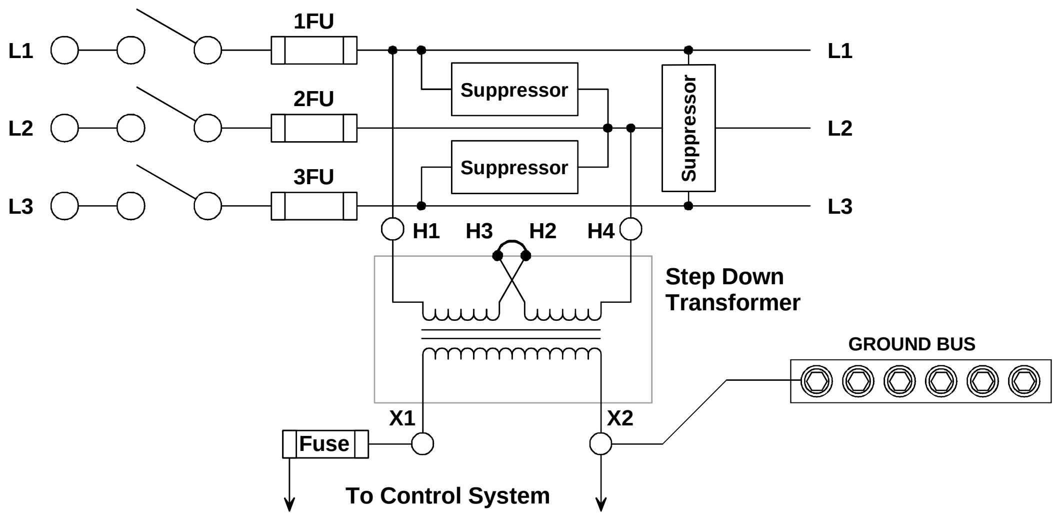


Figure R8.2 Surge Suppression on Power Inputs

Surge (EMI) Suppression (continued)

Inductive Loads

NOTE All inductive devices in the system, such as AC and DC motors, motor starters, contactors, relays and solenoids, must have surge suppression devices installed across their coils.

NOTE Because the Networked Driver is responsible for commutating the stepper motor, surge suppressors cannot be installed at the motor connector. The best way to avoid noise issues when using a stepper motor is proper cable installation. Keep the motor wires as short as possible. Consider twisting the winding pairs together to minimize radiated noise. Use shielded cable if you must extend the cable and tie the shield to earth ground at the Networked Driver motor connector only.

System Environment

When designing the physical layout of your system it is important to review the environment your control system will be placed in. This should include reviews of both the physical and electrical environments.

- Check the quality of the AC power that will feed the machine.
- Check for adequate surge suppression on machines that will be near your new system.

A review of the electrical environment includes checking the quality of the AC power that will feed your new system as well as checking the quality of the point where your system will be connected to the building's Grounding Electrode System.

When checking the quality of the AC power for your system, be sure to check every machine that may be powered by the same feeder circuit. It is in AMCI's experience that nuisance faults can be caused by another system on the same feeder circuit that does not have adequate surge suppression devices.

System Layout Considerations

The first step to achieving a clean system layout is classifying the wires and cables used in your system based on the amount of power they carry. Refer to the following table to help you categorize your wires and cables.

Category	Description	Examples
1	**Control and AC Power -** High power conductors that are more tolerant of electrical noise but are also more likely to carry electrical noise than the other categories. Corresponds to IEEE levels 3 & 4	AC power lines High power digital AC I/O lines High power digital DC I/O lines High power I/O lines typically connect to devices such as mechanical switches, relays, and solenoids. Stepper Motor wiring also falls into this category.
2	**Signal and Communication -** Low power conductors that are less tolerant of electrical noise but are also less likely to carry electrical noise than category 1. Corresponds to IEEE levels 1 & 2	Analog I/O lines Low power digital AC I/O lines Low power digital DC I/O lines Ethernet or other Communications Cables Low power I/O lines typically connect to devices such as proximity switches, photo sensors, and encoders.
3	**Intra-enclosure -** Similar characteristics to category 2 conductors, these conductors interconnect system components within an enclosure.	Low voltage DC power cables Communications Cables

Table R8.1 Conductor Categories

Wiring Categories for the SD17060E-K or SD31045E-K

- AC Power Input: Class 1 (AC power line)
- DC Input cables (also included encoder cabling): Class 2 (Low power digital DC input line)
- DC output cable: Class 2 (Low Power digital DC output line)
- Motor Outputs: Class 1 (High Power DC outputs)

System Layout Considerations (continued)

Minimize Voltages in the System Enclosure

You will want to minimize the voltage in the enclosure that houses your control system to minimize EMI and voltage transients. In many cases, three phase is used to power the machine's motors and the control system is powered by one of the phases. Ideally, the three phase power, it's disconnect, fuses, filtering, and all three phase controls such as motor starters, are housed in their own enclosure. The single phase used to power the control system is then brought into a separate enclosure built exclusively for the control system.

Power Supply Sizing

A properly sized power supply is vital to system operation. The best guideline that we can give you is to buy the best supply your budget allows and consider the surge requirements of the components you are attaching to the supply.

Component Placement

Once you have established the proper categories of all of the system's wires and cables and determined all of your components you can determine proper component placement and conductor routing within the system enclosure.

CAUTION The following guidelines are for noise immunity only. Follow all local codes for safety requirements.

There are three general guidelines to follow when placing components:

- Keep as much physical space between the classes of conductors as possible.
- Minimize the distance that conductors run in parallel with each other to minimize capacitive coupling.
- If conductors must cross, they should do so at right angles to minimize inductive coupling.

If you have a PLC in your system, be sure to consider the spacing of the modules within the rack. Do not place an AC output module next to low level DC input module unless absolutely necessary. For example, consider placing low level input modules on the left side of the rack, high power AC output modules on the right side, and medium power I/O in the center. This should help you maximize the spacing of I/O wiring within the enclosure.

System Layout Considerations (continued)

Conduits to Enclosure

When designing the layout of you system be sure to include enough conduits to house the different categories of cabling. To guard against coupling noise from one conductor to another, follow the spacing guidelines in table R8.2 below. These spacing values should be followed for routing cables both inside and outside of an enclosure. Of course, sometimes these guidelines cannot be followed, such as when the connection points on a controller are spaced closer together than these guidelines recommend.

Category	Guidelines
1	These conductors can be routed in the same cable tray, raceway, or conduit as machine power conductors of up to 600Vac. Power conductors cannot feed larger than 100HP devices.
2	➤ If they must cross power feeds, they must do so at right angles. ➤ Route in a raceway or conduit separate from all category 1 conductors and properly shield where applicable. ➤ Any metal wireway or conduit that houses the conductor must be bonded along its entire length and bonded to the enclosure at its entry point. ➤ If in a continuous metal wireway or conduit, route at least 3" from category 1 conductors of less than 20A, 6" from category 1 conductors of greater than 20A and 12" from any category 1 conductor of any amperage of the circuit is greater than 100 kVA. ➤ If not in a continuous metal wireway or conduit, route at least 6" from category 1 conductors of less than 20A, 12" from category 1 conductors of greater than 20A and 24" from any category 1 conductor of any amperage of the circuit is greater than 100 kVA. ➤ Route at least 5 feet away from high-voltage enclosures or sources of rf/microwave radiation.
3	Route these conductors external to all raceways in the enclosure or in a raceway separate from all category 1 conductors. Use the same spacing rules given for category 2 conductors.

Table R8.2 Conductor Routing Guidelines

Installing an AMCI Networked Driver

The Networked Driver is designed to be mounted in an enclosure that is close to the motor. On small machines, this may be the same enclosure that the rest of the control system in mounted in. On large machines, this is typically a small enclosure that only houses the Networked Driver.

➤ A minimum of two conduits are required for the enclosure that houses a Networked Driver. One conduit is for the DC I/O and ethernet cable, and the other is for AC power and the motor connections. Three conduits may be required if the code in your area does not allow you to run the motor connections with AC power or if the noise generated on the motor connections is injected into your AC lines.

➤ If 230Vac is the only line voltage available in a system that is using the SD17060E-K Drivers, a step down transformer that is rated for a minimum of 800VA must be installed. Consider installing this transformer in the enclosure with the SD17060E-K Networked Driver. This will prevent any noise placed on the AC line by the SD17060E-K from leaving the enclosure.

➤ For the SD31045E-K Drivers, if 115Vac is the only line voltage available in the system and you require the 310Vdc motor bus, a step up transformer that is rated for a minimum of 800VA must be installed. Consider installing this transformer in the enclosure with the SD31045E-K. This will prevent any noise placed on the AC line by the SD31045E-K from leaving the enclosure.

➤ Like all motors, stepper motors are noise generators and must be properly grounded when installed.

➤ Keep the motor wires as short as possible. Consider twisting the winding pairs together to minimize radiated noise. Use shielded cable if you must extend the cable and tie the shield to earth ground at the motor connector of the Networked Driver only.

TASK 1

INSTALLING AN AMCI NETWORKED DRIVER

This chapter gives detailed information on installing an AMCI Networked Driver. The person responsible for installing the driver should be familiar with the general installation guidelines given in the previous chapter.

1.1 Location

AMCI Networked Drivers are suitable for use in industrial environments that meet the following criteria:

- Only non-conductive pollutants normally exist in the environment, but an occasional temporary conductivity caused by condensation is expected.
- Transient voltages are controlled and do not exceed the impulse voltage capability of the product’s insulation.

Note that these criteria apply to the system as a whole. These criteria are equivalent to the *Pollution Degree 2* and *Over Voltage Category II* designations of the International Electrotechnical Commission (IEC).

1.2 Safe Handling Guidelines

1.2.1 Prevent Electrostatic Damage

CAUTION Electrostatic discharge can damage the Networked Driver. Follow these guidelines when handling the unit.

1) Touch a grounded object to discharge static potential before handling the module.
2) Work in a static-safe environment whenever possible.
3) Wear an approved wrist-strap grounding device.
4) Do not touch the pins of the bus connector or I/O connector.
5) Do not disassemble the module
6) Store the module in its anti-static bag and shipping box when it is not in use.

1.2.2 Prevent Debris From Entering the Unit

WARNING While mounting a devices, be sure that all debris (metal chips, wire strands, tapping liquids, etc.) is prevented from falling into the driver. Debris may cause damage to the unit or unintended machine operation with possible personal injury.

1.2.3 Remove Power Before Servicing

WARNING Remove power before removing or installing a Networked Driver.

1.3 Airflow and Wiring Space

To ensure adequate space for wiring and convectional cooling, you need:

- 2.0" (50 mm) of space above and below the driver
- 1.5" (37 mm) of space to the left and right of the driver
- 1.0" (25 mm) of space in front of the driver

NOTE Drivers should be mounted with the orientation shown in the outline drawings below. Holes are in the top and bottom of the driver cover for convectional cooling. If you have an active cooling system such as enclosure fans, the drivers should still be mounted as shown below, but you should be able to mount the drivers closer together.

1.4 Outline Drawings

1.4.1 SD17060E-K Outline Drawing

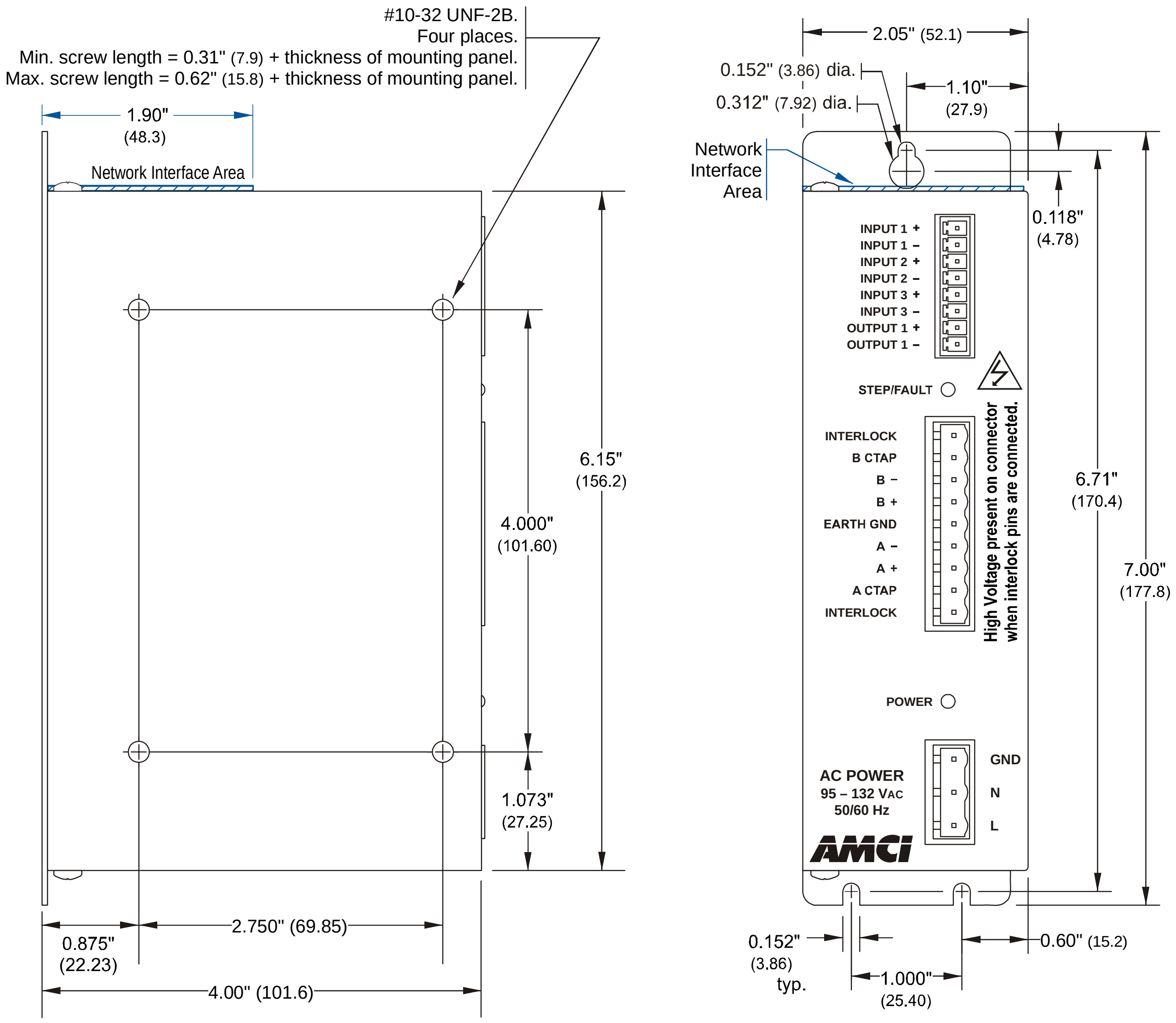


Figure T1.1 SD17060E-K Outline Drawing

1.4 Outline Drawings (continued)

1.4.2 SD31045E-K Outline Drawing

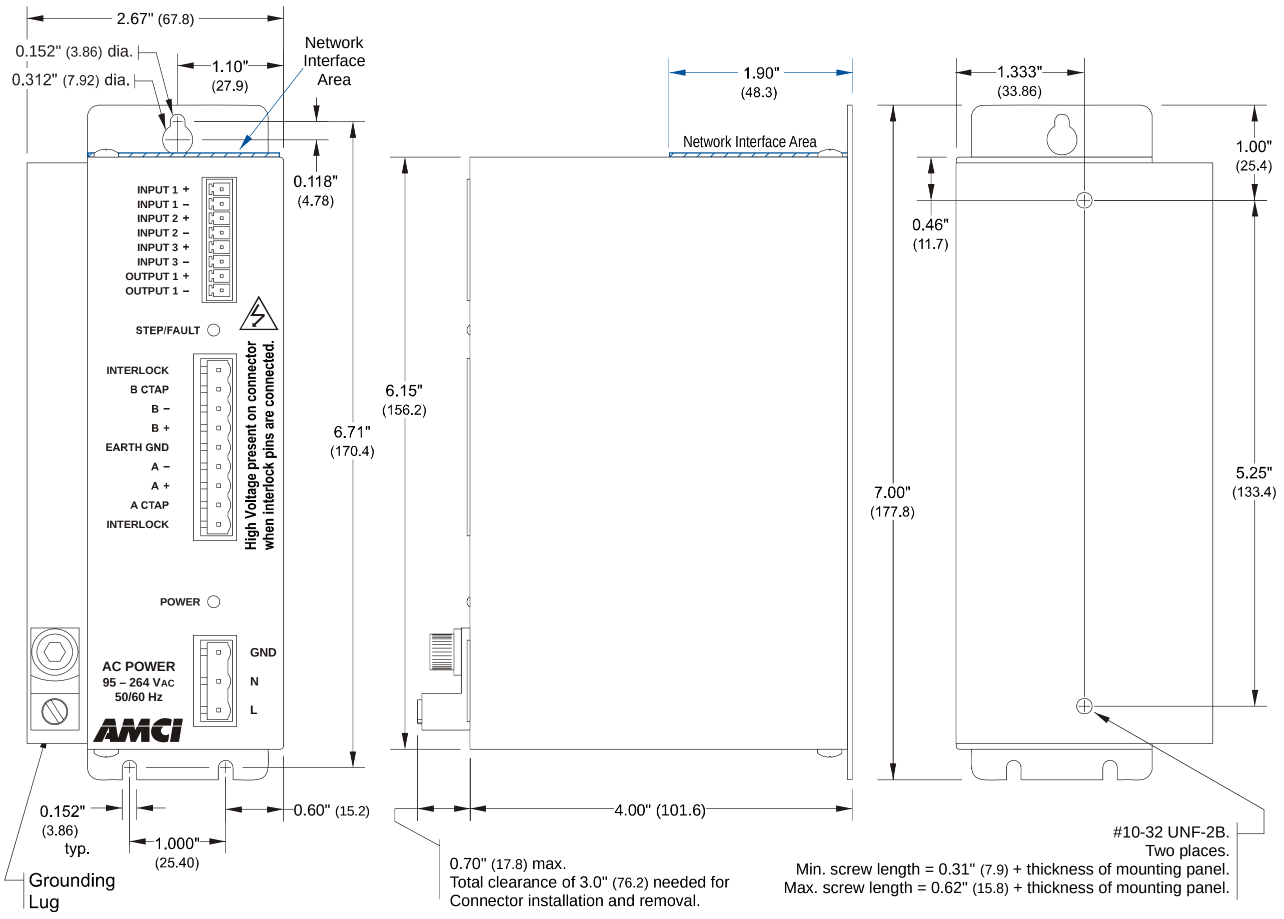


Figure T1.2 SD31045E-K Outline Drawing

1.5 Mounting Methods

There are two ways to mount an AMCI Networked Driver. The first method is with the #10-32 screws into the side panel of the SD17060E-K or the back panel of the SD31045E-K. The second method is by the mounting tabs. Mounting tabs are for #6 screws.

WARNING When using the panel mounting method that uses the #10-32 screws, minimum and maximum screw lengths must be observed to prevent a screw from shorting to the PC board or components.

NOTE There are airflow holes in the top and bottom of the enclosure. To ensure adequate convectional airflow, the drive must be mounted in the orientation shown in the drawing.

1.6 Grounding and Powering the System

WARNING The chassis must be connected to earth ground. Failure to properly ground the chassis leaves the potential for severe electrical hazard and/or problems with normal operation.

The Networked Driver must be grounded for proper operation. The **GND** connection on the power connector is connected to the enclosure of the Networked Driver and is a sufficient grounding point for most applications. When mounting the Networked Driver on a surface that is electrically conductive and grounded, you should take steps to ensure that the two are electrically bonded together. If necessary, remove paint from the surfaces of the panel that contact the mounting bolts to ensure adequate electrical bonding.

AC power connections are made to the Networked Driver using the power connector kit that ships with the driver. This kit includes the power connector and rubber boot. Figure T1.3 below shows how to properly wire and ground the driver.

NOTE For clarity, the rubber boot is not shown in the figure. When installing the power cable, slide the rubber boot onto the cable before wiring the connector. When you're sure the wiring is correct, slide the boot over the connector to cover the screw heads.

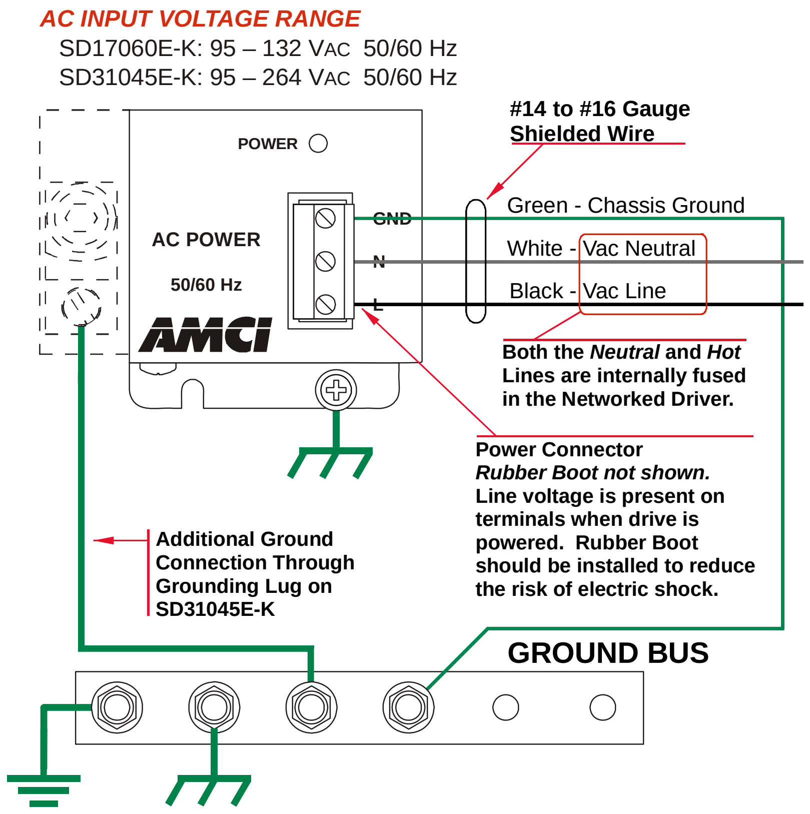

AC INPUT VOLTAGE RANGE
SD17060E-K: 95 – 132 VAC 50/60 Hz
SD31045E-K: 95 – 264 VAC 50/60 Hz
#14 to #16 Gauge Shielded Wire
POWER
AC POWER
50/60 Hz
AMCI
GND
N
L
Green - Chassis Ground
White - Vac Neutral
Black - Vac Line
Both the Neutral and Hot Lines are internally fused in the Networked Driver.
Power Connector Rubber Boot not shown. Line voltage is present on terminals when drive is powered. Rubber Boot should be installed to reduce the risk of electric shock.
Additional Ground Connection Through Grounding Lug on SD31045E-K
GROUND BUS

Figure T1.3 Power and Grounding Connections

NOTE

1) Input power must be 95 to 132 Vac, 50/60 Hz for the SD17060E-K and 95 to 264 Vac, 50/60 Hz for the SD31045E-K. Input power must able to supply 800VA for proper operation.
2) Never attempt to power an SD17060E-K with 230Vac. Doing so will damage the driver and void its warranty.

Both the Neutral and the Line power connections are internally fused in the AMCI Networked Drivers. These fuses are not field replaceable, so external fuses or circuit breakers should also be used. They must be rated for at least 10 amps.

1.7 Installing the Stepper Motor

1.7.1 Outline Drawings

Outline drawings for all of our motors can be found on our website, *www.amci.com*, in the *PDF Documents* section. They're available as Adobe Acrobat pdf files. A document that is simply called *wiring* lists all of the wiring color codes for all AMCI motors. If you do not have internet access contact AMCI and we will fax the information to you.

1.7.2 Mounting the Motor

All AMCI motor have flanges on the front of the motor for mounting. This flange also acts as a heatsink, so motors should be mounted on a large, unpainted metal surface. Mounting a motor in this fashion will allow a significant amount of heat to be dissipated away from the motor, which will increase the motor's life by reducing its operating temperature. If you cannot mount the motor on a large metal surface, you may need to install a fan to force cooling air over the motor.

Motors should be mounted using the heaviest hardware possible. AMCI motors can produce high torques and accelerations that may weaken and shear inadequate mounting hardware.

NOTE

1) The motor case must be grounded for proper operation. This is usually accomplished through its mounting hardware. If you suspect a problem with your installation, such as mounting the motor to a painted surface, then run a bonding wire from the motor to a solid earth ground point near it. Use a minimum #8 gauge stranded wire or 1/2" wire braid as the grounding wire
2) Do not disassemble *any* stepper motor. A significant reduction in motor performance will result.

1.7.3 Connecting the Load

Care must be exercised when connecting your load to the stepper motor. Even small shaft misalignments can cause large loading effects on the bearings of the motor and load. The use of a flexible coupler is *strongly* recommended whenever possible.

1.7.4 Extending the Motor Cable

Even though it is possible to extend the cable length an additional forty feet, AMCI recommends installing the Networked Driver as close to the motor as possible. This will decrease the chances of forming a ground loop, and has the added benefit of limiting the amount of power loss in the motor cable. If you must extend the cable, you should use a cable with twisted pairs 18 AWG or larger and an overall shield. Belden 9554 (eight wire), 9553 (six wire) and 9552 (four wire) meet these specifications for 18 AWG+. For long runs, consider using 14 AWG wire. Belden 1070A (eight wire), 1527A (six wire) and 1069A (four wire) meet these specifications for 14 AWG.

1.7.5 Installing the Motor Cable

NOTE

1) All of the motor connections are high power, high voltage signals. Cable from the motor can be installed in conduit along with ac/dc power lines or high power ac/dc I/O. It cannot be installed in conduit with low power cabling such as I/O cabling or Ethernet cabling attached to the Networked Driver.
2) If you decide to extend the motor cable, treat the shield as a signal carrying conductor when installing the motor cable. Do not connect the shield to earth ground at any junction box.

1.8 Connecting the Motor

1.8.1 Motor Connector

The motor connector is shown in figure T1.4. The two Interlock terminals are a safety feature. When these two terminals are not connected, the driver will not power the motor outputs, the Status LED turns on red, and it activates the Fault Output. For normal operation, these two terminals must be connected together with a short wire.

The two center tap pins, A CTAP and B CTAP, are there for wiring convenience only. They are electrically isolated from the rest of the driver and are not used to power the motor. The **EARTH GND** pin is for the shields of the motor cable. This pin is directly connected to the GND pin of the power connector on the Networked Driver.

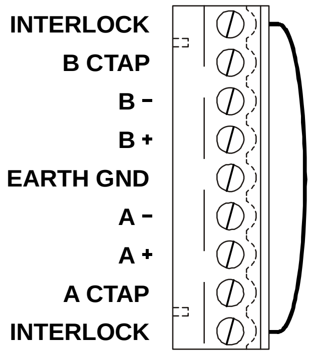


Figure T1.4 Motor Connector with Interlock Jumper

NOTE When powered, the motor connector represents a shock hazard because it has 170/310 Vdc present on its terminals. A rubber boot that is included with the connector must be installed but is not shown in the following figures for clarity. When installing the motor cable, slide the rubber boot onto the cable before wiring the connector. When you're sure the wiring is correct, slide the boot over the connector to cover the screw heads.

WARNING Always remove power from the Networked Driver before connecting or disconnecting the motor.

NOTE
1) Never connect the motor leads to ground or to a power supply.
2) Always connect the cable shield to the Earth Ground terminal of the Motor Connector on the Networked Driver.

1.8.2 Interlock Wiring

The motor interlock is designed to prevent 170/310 Vdc power from being supplied to the motor connector unless its mate is installed. As shown in figure T1.4 above, a short wire has to be installed between the two outer pins of the mating connector or the motor will not receive power when it is plugged into the Networked Driver.

NOTE It is possible to use the Interlock terminals as an alternative way of disabling motor current. This is accomplished by wiring the two pins through a *mechanical contact*. The wires to the Interlock pins should be kept as short as possible, which means using a mechanical relay in the cabinet that houses the Networked Driver if you need to control the Interlock from a source outside the cabinet. Only mechanical relays should be used. The voltage between the two interlock pins will never exceed 5 Vdc, but 115/230 Vac may be present between either of the two interlock pins and either of the power supply pins.

When the Networked Driver detects a break in the Interlock wiring, it will issue a *Driver Fault* error. This error is latched and can only be cleared after the interlock connection is re-established. The fault can be cleared over the network by following the procedure outlined in ***Notes on Clearing a Driver Fault*** which starts on page 83 or by cycling power to the driver.

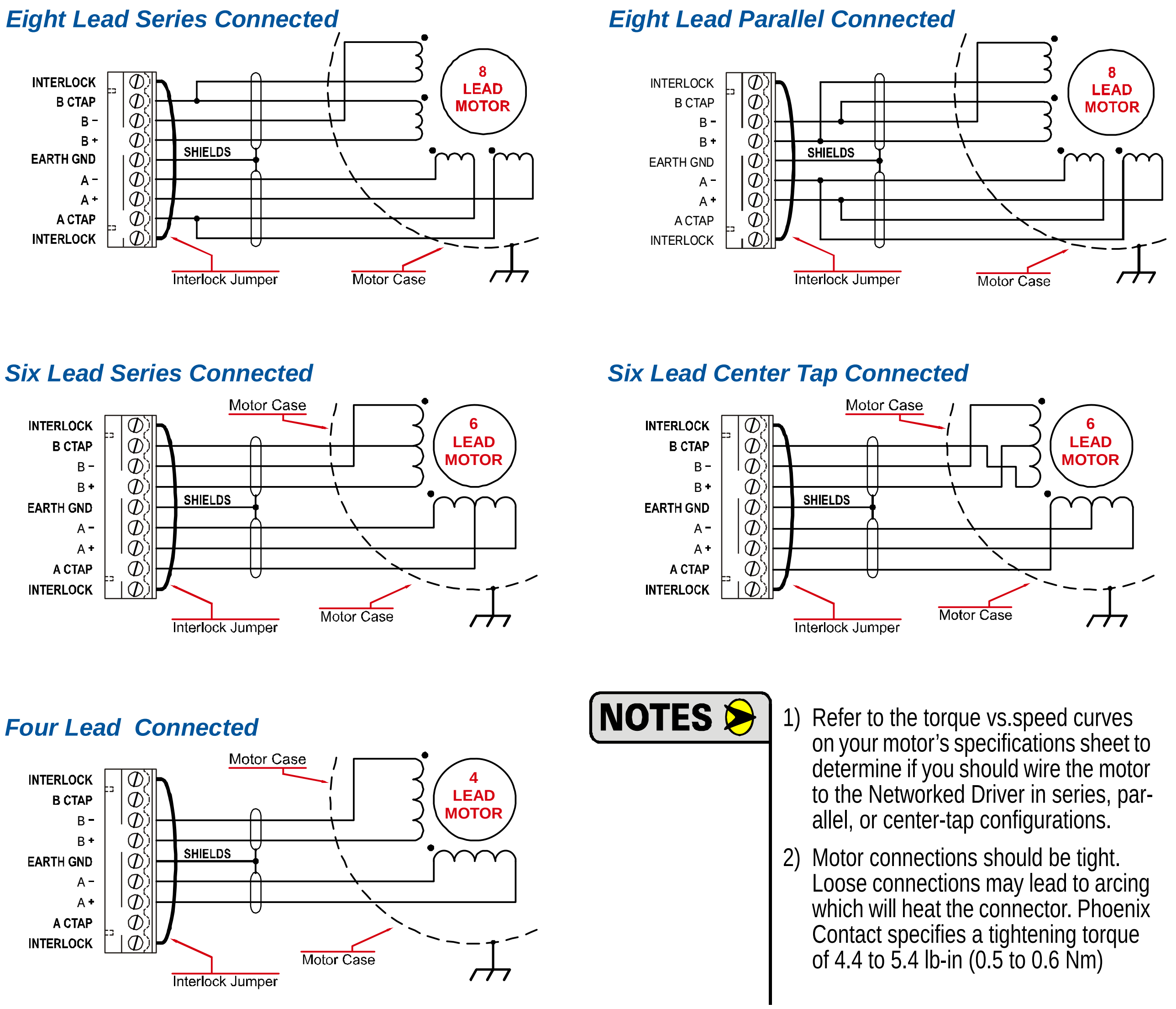
1.8 Connecting the Motor (continued)

1.8.3 Motor Wiring

A Networked Driver will work with many different motors, including those not sold by AMCI. This section assumes that you have already chosen your motor and you are looking for wiring information. No wire colors are given in the figures below because there is no single industry wide color coding scheme for stepper motors. You must refer to your motor data sheets for this information.

A wiring document for all of the motors ever sold by AMCI is available on our website. This single document contains all of the information necessary to connect any AMCI motor to any AMCI driver. At the time of this manual revision, the wiring manual can be found in the *PDF Documents* section of the website. It is under the *Stepper Motor* heading, and link is simply called "wiring".

Figure T1.5 shows how to wire a motor to a Networked Driver in series, parallel, or center-tap configurations.



NOTES

1) Refer to the torque vs.speed curves on your motor's specifications sheet to determine if you should wire the motor to the Networked Driver in series, parallel, or center-tap configurations.
2) Motor connections should be tight. Loose connections may lead to arcing which will heat the connector. Phoenix Contact specifies a tightening torque of 4.4 to 5.4 lb-in (0.5 to 0.6 Nm)

Figure T1.5 Motor Wiring

1.9 Digital Input Wiring

1.9.1 Cable Shields

Because they are low power signals, cabling from the sensor to the Networked Driver should be done using a twisted pair cable with an overall shield. The shield should be grounded at the end when the signal is generated, which is the sensor end. If this is not practical, the shield should be grounded to the same ground bus as the Networked Driver.

1.9.2 Input Wiring

All three inputs are differential, and can be wired to sinking or sourcing sensors without requiring a pull up or pull down resistor. They accept 3.5 to 27 Vdc without the need for an external current limiting resistor. If using a 48 Vdc sensor, a 5 kilohm resistor that is wired in series with the input pin is required. Figure T1.7 below shows how to wire a discrete DC sourcing or sinking sensor to an input on the Networked Driver.

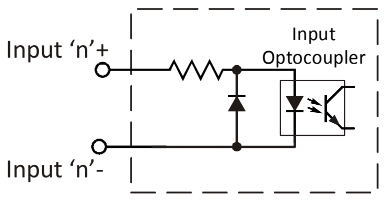


Figure T1.6 Input Schematic

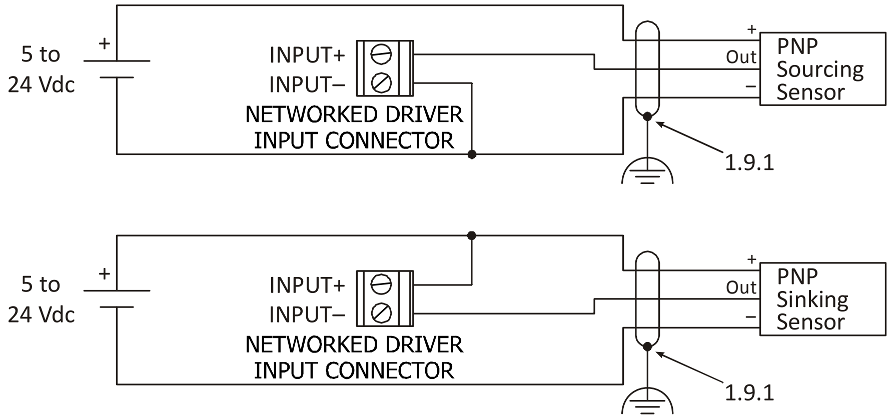


Figure T1.7 Wiring - Input 3

Encoders typically have differential outputs, but must be wired to the driver in a specific way. More information on wiring an encoder is given later in this chapter.

1.9 I/O Wiring (continued)

1.9.3 Output Wiring

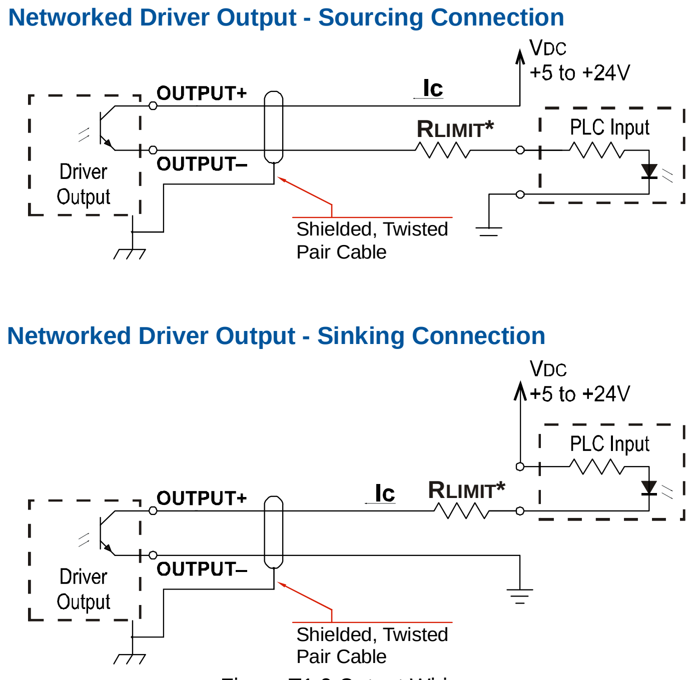

Networked Driver Output - Sourcing Connection
VDC
+5 to +24V
OUTPUT+
Ic
RLIMIT*
PLC Input
Driver Output
OUTPUT–
Shielded, Twisted Pair Cable
Networked Driver Output - Sinking Connection
VDC
+5 to +24V
PLC Input
OUTPUT+
Ic
RLIMIT*
Driver Output
OUTPUT–
Shielded, Twisted Pair Cable

Figure T1.8 Output Wiring

The output on the Networked Driver is an optically isolated transistor that is capable of driving a typical PLC input. Both ends are uncommitted, so it can be wired as a sinking or sourcing output.

Electrical Specifications

VDC max: 30Vdc	$V_{CE\ SAT}$ = 1Vdc @ 20 mA
Ic max: 20 mA	Power Dissipation = 20 mW max.

RLIMIT

A resistor may be needed to limit the current through the output. The value, and power rating of the resistor is dependent on the value of Vdc, the voltage drop across the input, and the current requirements of the input.

1.9.4 Encoder Wiring

The figure below shows how to wire a 5Vdc differential encoder to a Networked Driver.

- The ±A channel must be wired to Input 1 for proper operation
- The ±B channel must be wired to Input 2 for proper operation
- The ±Z channel is optional. If used, it must be wired to Input 3. The Z channel is only used when homing the Networked Driver.

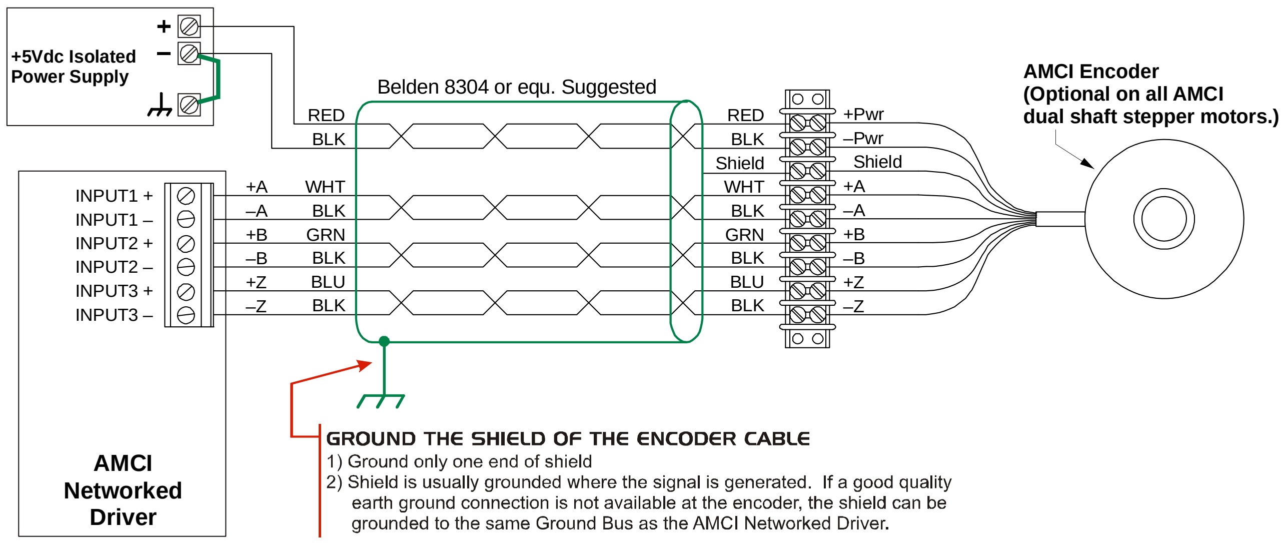

+5Vdc Isolated Power Supply
+
–
Belden 8304 or equ. Suggested
AMCI Encoder (Optional on all AMCI dual shaft stepper motors.)
RED
BLK
RED
BLK
Shield
WHT
BLK
GRN
BLK
BLU
BLK
+Pwr
–Pwr
Shield
+A
–A
+B
–B
+Z
–Z
INPUT1 +
INPUT1 –
INPUT2 +
INPUT2 –
INPUT3 +
INPUT3 –
+A
–A
+B
–B
+Z
–Z
WHT
BLK
GRN
BLK
BLU
BLK
AMCI Networked Driver
GROUND THE SHIELD OF THE ENCODER CABLE
1) Ground only one end of shield
2) Shield is usually grounded where the signal is generated. If a good quality earth ground connection is not available at the encoder, the shield can be grounded to the same Ground Bus as the AMCI Networked Driver.

Figure T1.9 Sample Encoder Wiring

1.10 Ethernet Connections

The Ethernet connectors are located on the top of the SD17060E-K and SD31045E-K units. The connectors are standard RJ-45 jacks that will accept any standard 100base-TX cable. Because the ports run at 100 Mbit speeds, Category 5, 5e, or 6 cable should be used.

The ports are auto MDI-X capable. This means that a standard cable can be used to connect the SD17060E-K or SD31045E-K to any device, including a personal computer.

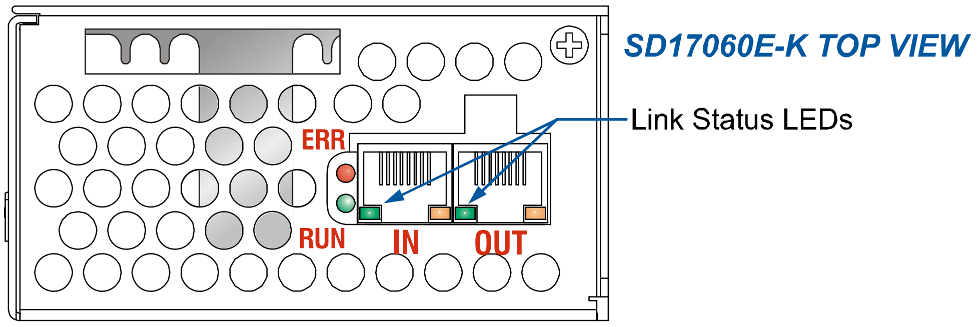


Figure T1.10 Ethernet Port Location

The network jack labeled “IN”, in the diagram above, must be attached to the upstream device in the EtherCAT network. (Closer to the network master.) The “OUT” jack must be attached to the next (down stream) device in the network of the Networked Driver is not the last device in the network.

The EtherCAT Error and Run LED’s are fully described on page 17 in the ***Status LED’s*** section of the Specifications reference.

TASK 2
ETHERCAT SYSTEM CONFIGURATION

This chapter outlines how to add an SD17060E-K or SD31045E-K unit to an EtherCAT system. The TwinCAT version 3 software is used as an example.

2.1 Install the ESI file

2.1.1 Obtain the ESI file

All AMCI ESI files are located on our website at the following address:

- ***http://www.amci.com/industrial-automation-support/configuration-files/***

Simply download the ZIP file and extract it.

2.1.2 Install the ESI file

Once extracted, the xml file must be copied or moved to the appropriate system directory. For version 3 of TwinCAT, the default directory is:

- C:\TwinCAT\3.1\Config\Io\EtherCAT\

2.1.3 Restart the Programming System If Needed

If the TwinCAT program was running when the ESI file was copied to the appropriate system directory, you may have to restart the TwinCAT program before it will recognize the new ESI file.

2.2 Add the Networked Driver to the Project

NOTE This section assumes that the TwinCAT software is in Config Mode. The example uses an SD17060E-K. All of the information in this section applies to the SD31045E-K as well.

2.2.1 Scan for the Networked Driver

1) Attach the Networked Driver to the network and power up the device.
2) At this point, the device is in its INIT state. The power and status LED's on the front of the device will be on. Both of the EtherCAT status LED's on the top of the unit will be off.
3) Right click on the EtherCAT adapter that the Networked Driver is attached to. In the drop down menu that opens, select the "Scan" option. (If the "Scan" option is not available, the TwinCAT software is not in Config Mode.)

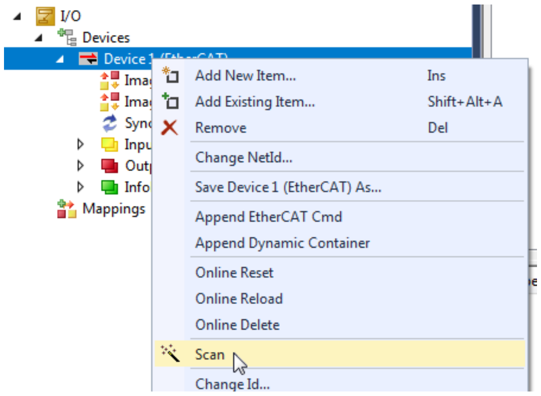
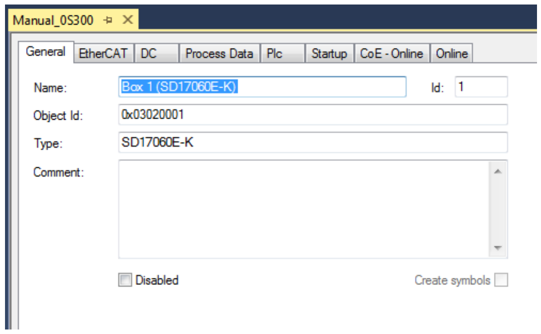


Figure T2.1 Scan for the Networked Driver

2.2.2 Rename the Device

The Networked Driver will appear in the device tree and the name will typically begin with "Box".

1) Click on the unit in the device tree.
2) If needed, click on the "General" tab in the window that opens.
3) The device name for the Networked Driver can be changed in the *Name:* field.



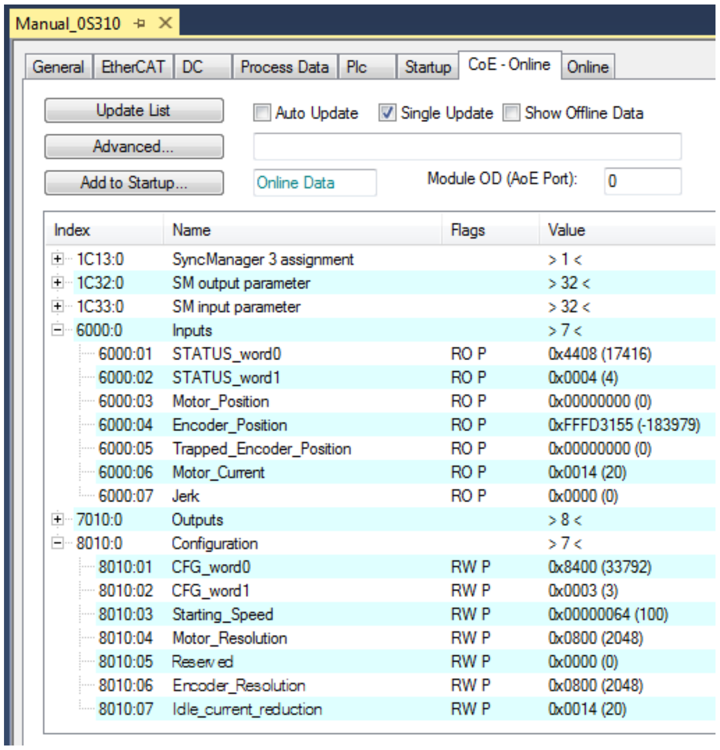
Name: Box 1 (SD17060E-K) Id: 1
Object Id: 0x03020001
Type: SD17060E-K
Comment:
Disabled
Create symbols

Figure T2.2 Rename the Networked Driver

2.2 Add the Networked Driver to the Project (continued)

2.2.3 Configure the Networked Driver

1) Click on the “CoE - Online” tab.
2) Click on the [+] button next to the 6000:0 Index field (Inputs) to expand it. Note the value of the STATUS_word1 register (6000:01). This value is typically 0x4408.
3) Click on the [+] button next to the 8010:0 Index field to expand it.
4) The configuration data is available under the nine sub-index fields of the 8010:0 Index. *Configuration data can be temporarily changed here*. Double click on any field to open the *Set Value Dialog* screen to set the parameter to the desired value. Table R6.3, ***Configuration Word 0 Bits*** found on page 59, allows you to calculate the proper hexadecimal value of CFG_word0 for your application.
5) Once you set a parameter, check the value of the STATUS_word1 register (6000:01). If the value has bit 13 set to “1”, there is an error in your configuration setting. (An example is a STATUS_word1 register change from 0x4408 to 0x6408.)
6) To make these changes permanent, you must define values under the “Startup” tab. Click on the “Startup” tab.
7) Right click on the blank surface and select “Add New Item...” in the resulting pop up menu.
8) Click on the [+] button next to the 8010:0 Index field to expand it.
9) Double click on the field that must be changed. This opens a dialog that will allow you to set the field with a decimal or hexadecimal value.
10) Click the [OK] button in the dialog box to set the new value.
11) Click the [OK] button in the “Edit CANopen Startup Entry” screen to accept the new startup parameter setting.



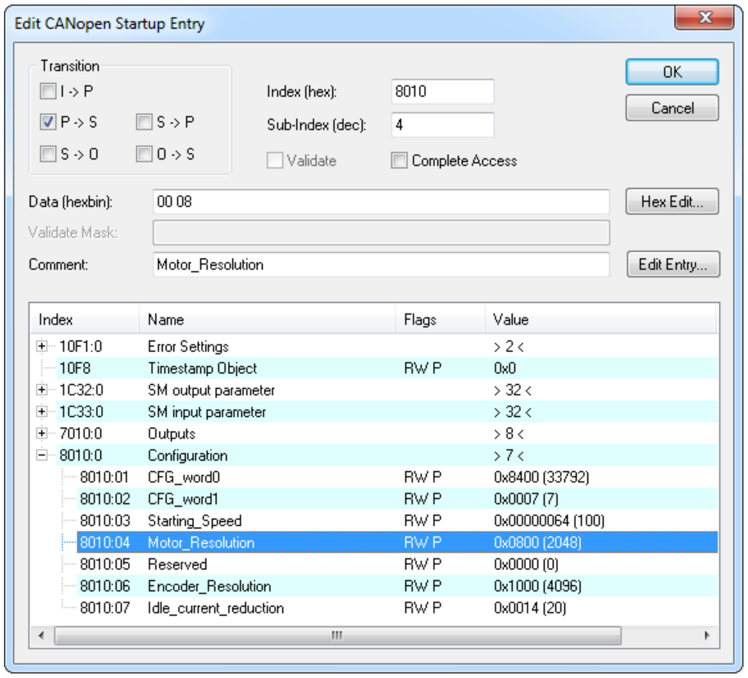
[ ] Auto Update [✓] Single Update [ ] Show Offline Data

Online Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
1C13:0	SyncManager 3 assignment		> 1 <
1C32:0	SM output parameter		> 32 <
1C33:0	SM input parameter		> 32 <
6000:0	Inputs		> 7 <
6000:01	STATUS_word0	RO P	0x4408 (17416)
6000:02	STATUS_word1	RO P	0x0004 (4)
6000:03	Motor_Position	RO P	0x00000000 (0)
6000:04	Encoder_Position	RO P	0xFFFD3155 (-183979)
6000:05	Trapped_Encoder_Position	RO P	0x00000000 (0)
6000:06	Motor_Current	RO P	0x0014 (20)
6000:07	Jerk	RO P	0x0000 (0)
7010:0	Outputs		> 8 <
8010:0	Configuration		> 7 <
8010:01	CFG_word0	RW P	0x8400 (33792)
8010:02	CFG_word1	RW P	0x0003 (3)
8010:03	Starting_Speed	RW P	0x00000064 (100)
8010:04	Motor_Resolution	RW P	0x0800 (2048)
8010:05	Reserv ed	RW P	0x0000 (0)
8010:06	Encoder_Resolution	RW P	0x0800 (2048)
8010:07	Idle_current_reduction	RW P	0x0014 (20)

Figure T2.3 Networked Driver Configuration Registers



Transition
[ ] I -> P
[✓] P -> S [ ] S -> P
[ ] S -> O [ ] O -> S

Index (hex): 8010
Sub-Index (dec): 4
[ ] Validate [ ] Complete Access

Data (hexbin): 00 08
Validate Mask:
Comment: Motor_Resolution

Index	Name	Flags	Value
10F1:0	Error Settings		> 2 <
10F8	Timestamp Object	RW P	0x0
1C32:0	SM output parameter		> 32 <
1C33:0	SM input parameter		> 32 <
7010:0	Outputs		> 8 <
8010:0	Configuration		> 7 <
8010:01	CFG_word0	RW P	0x8400 (33792)
8010:02	CFG_word1	RW P	0x0007 (7)
8010:03	Starting_Speed	RW P	0x00000064 (100)
8010:04	Motor_Resolution	RW P	0x0800 (2048)
8010:05	Reserved	RW P	0x0000 (0)
8010:06	Encoder_Resolution	RW P	0x1000 (4096)
8010:07	Idle_current_reduction	RW P	0x0014 (20)

Figure T2.4 Networked Driver Configuration Registers

2.3 Create a DUT

AMCI sample programs use a DUT (Defined User Type) to group the I/O variables associated with each Networked Driver. Once the DUT is defined, a separate variable is created for each device on the machine.

1) If needed, create a PLC for the project.
2) Expand out the PLC tree until the DUT field is available.
3) Right click on DUT and select Add → DUT.
4) In the window that opens, name the DUT and give it a Structure data type.
5) The following is the variable layout used in the sample programs. This layout can be used as is, or can be modified to fit your specific application.

```
TYPE DUT_AMCI_SD17060E-K :
STRUCT
  //Outputs:
    nCMD0 AT %Q*          : UINT;
    nCMD1 AT %Q*          : UINT := 16#8000;
    nTargetPos AT %Q*     : DINT;
    nTargetVel AT %Q*     : UDINT;
    nAccel AT %Q*         : UINT;
    nDecel AT %Q*         : UINT;
    nMtrCurr AT %Q*       : UINT;
    nJerk AT %Q*          : UINT := 0;
  //Inputs:
    nStatus0 AT %I*       : UINT;
    nStatus1 AT %I*       : UINT;
    nMotorPos AT %I*      : DINT;
    nEncPos AT %I*        : DINT;
    nTrappedEnc AT %I*    : DINT;
    nLastCurrent AT %I*   : UINT;
    nLastJerk AT %I*      : UINT;
END_STRUCT
END_TYPE
```

2.4 Create Variables Based on the DUT

The location of the actual variable(s) used in the system's program depends on programming style and program complexity. Typically, variables are declared in the Main program or in a Global Variable List.

```
st_Axis1_SD17060E-K : DUT_AMCI_SD17060E-K;
st_Axis2_SD17060E-K : DUT_AMCI_SD17060E-K;
```

2.5 Link Variable Names to I/O Words

1) From the menu, select *Build* → *Build Solution*. (Ctrl+Shift+B). The build will fail with a message that at least one variable must be linked to a task variable. Click [OK] to close the message.
2) In the I/O tree, expand the Networked Driver until the input and output words are visible.
3) Right click on the STATUS_word0 input word. In the pop up menu that opens, select "Change Link..."

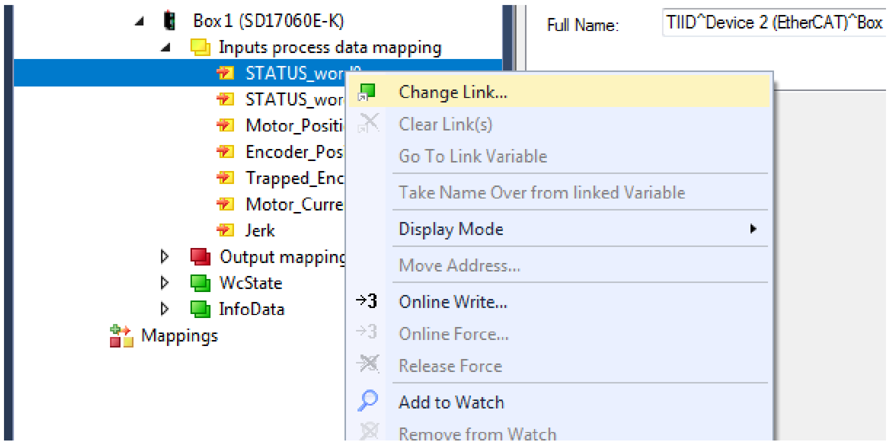
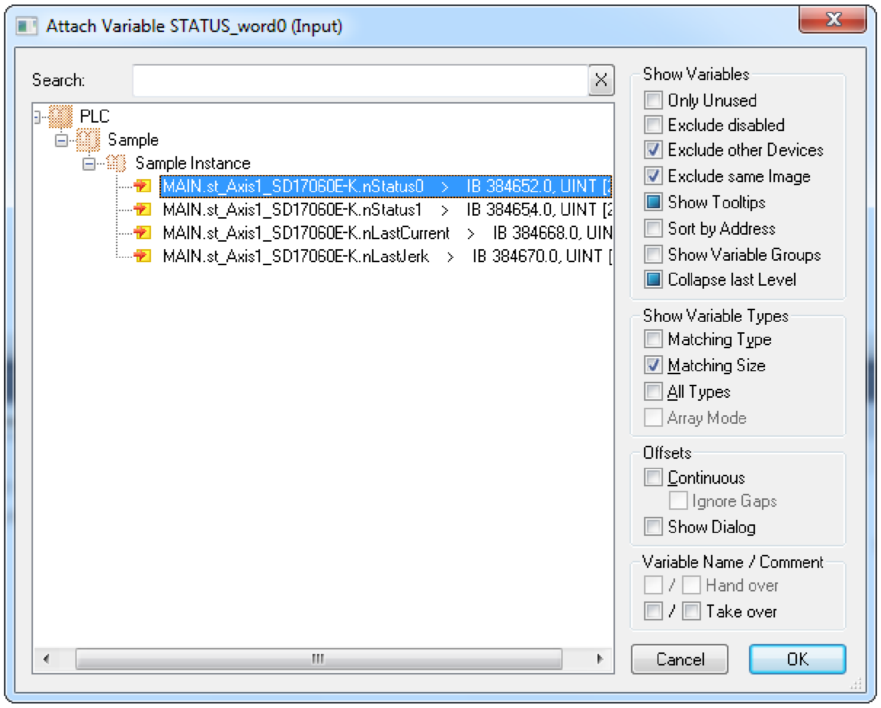


Figure T2.5 Change Link Pop up Menu

2.5 Link Variable Names to I/O Words (continued)

4) In the *Attach Variable* window that opens, select the variable to link to the STATUS_word0 input word and click [OK].



- PLC
  - Sample
    - Sample Instance
      - MAIN.st_Axis1_SD17060E-K.nStatus0 > IB 384652.0, UINT [
      - MAIN.st_Axis1_SD17060E-K.nStatus1 > IB 384654.0, UINT [
      - MAIN.st_Axis1_SD17060E-K.nLastCurrent > IB 384668.0, UIN
      - MAIN.st_Axis1_SD17060E-K.nLastJerk > IB 384670.0, UINT [

Show Variables
- ☐ Only Unused
- ☐ Exclude disabled
- ☑ Exclude other Devices
- ☑ Exclude same Image
- ■ Show Tooltips
- ☐ Sort by Address
- ☐ Show Variable Groups
- ■ Collapse last Level

Show Variable Types
- ☐ Matching Type
- ☑ Matching Size
- ☐ All Types
- ☐ Array Mode

Offsets
- ☐ Continuous
- ☐ Ignore Gaps
- ☐ Show Dialog

Variable Name / Comment
- ☐ / ☐ Hand over
- ☐ / ☐ Take over

Figure T2.6 Change Link Pop up Menu

5) Repeat steps three and four for the remaining input and output words of the Networked Driver.

TASK 3 (OPTIONAL)
DISTRIBUTED CLOCK - SYNC0 SETUP

This chapter outlines the steps need to configure the Networked Driver to synchronize to the Sync0 signal from the Distributed Clock (DC) feature instead of the SyncManager 2 event. This configuration is optional.

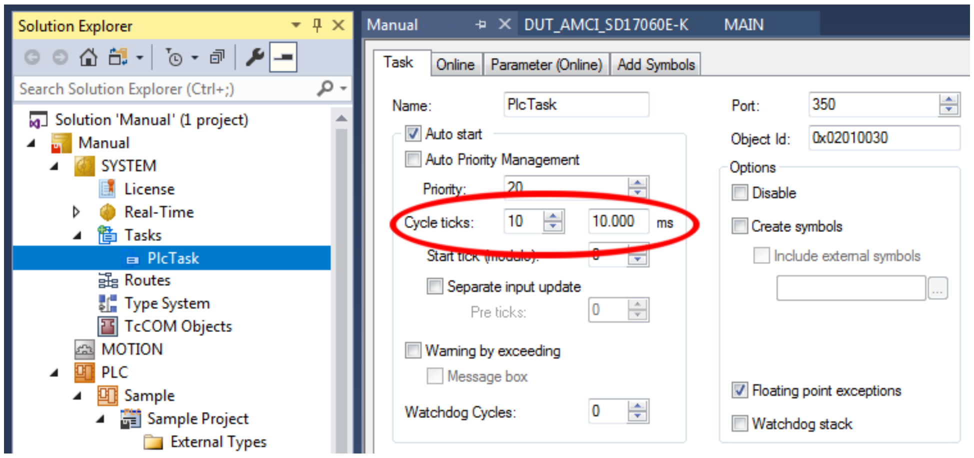
By default, the networked Driver acts on the data from the EtherCAT master as soon as it is transmitted to the device. This mode of operation is called "SM-Synchron". The data transfer is synchronized with the SyncManager 2 event. This synchronization is more than adequate for most machine types.

On very high speed machines, or large machines that require more than one EtherCAT transfer to update all of the I/O, it may be necessary to use the EtherCAT Distributed Clock mechanism to synchronize the start of moves with other motion axes or other devices on the EtherCAT network.

Using the Distributed Clock feature adds complexity to the system that may not be needed. On large machines that require more than one EtherCAT transfer to update all of the I/O, configure the network to update all of the axes with one transfer. This may eliminate the need to use the DC functionality on large machines.

3.1 Verify Main PLC Task Timing

1) In the *Solutions Explorer,* expand out *System* → *Tasks* so that the tasks are visible. Double click on the task assigned to the main PLC. This task is typically named *PlcTask*. In the windows that open, note the value for Cycle Ticks and its time in milliseconds. These values are typically 10 ticks and 10.000 milliseconds.
   - ➤ You can change the timing of the main task here, but this will affect the timing of every program that runs under this task.



Name: PlcTask
Port: 350
Object Id: 0x02010030
[✓] Auto start
[ ] Auto Priority Management
Priority: 20
Cycle ticks: 10 10.000 ms
Start tick (modulo): 0
[ ] Separate input update
Pre ticks: 0
[ ] Warning by exceeding
[ ] Message box
Watchdog Cycles: 0
Options
[ ] Disable
[ ] Create symbols
[ ] Include external symbols
[✓] Floating point exceptions
[ ] Watchdog stack

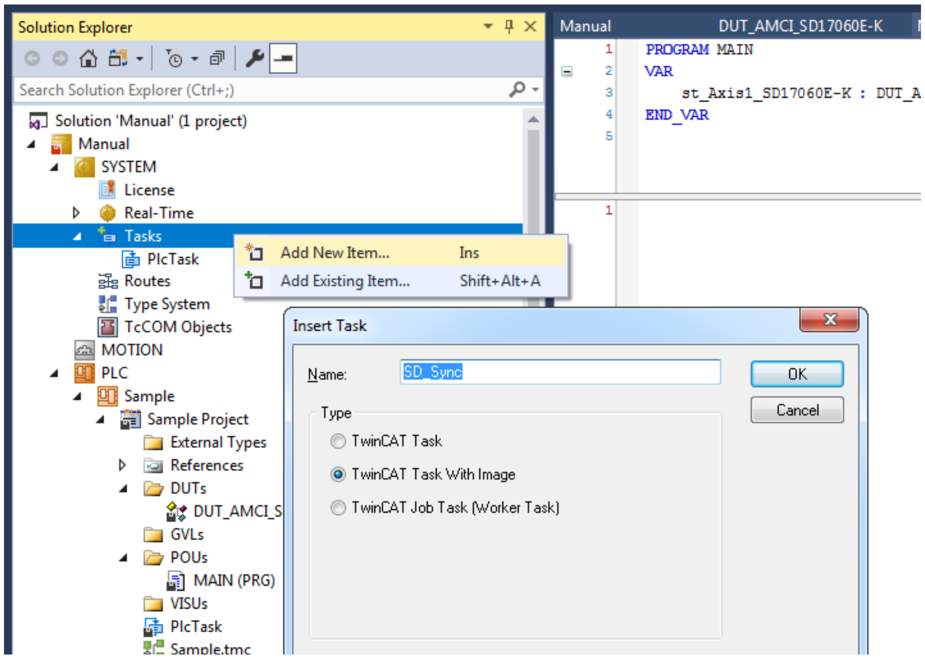
Figure T3.1 Main PLC Task Timing

2) If this value is between 2 and 50 milliseconds and the correct update time for the Networked Driver, skip to step 3.3. When setting the *SYNC 0 -> Sync Unit Cycle* multiplier in step 6 of 3.3, use a value of 1.
3) If the timing in step 1 above is not correct for the Networked Driver, but the correct time is an integer multiple of this timing, then skip to step 3.3. When setting the *SYNC 0 -> Sync Unit Cycle* multiplier in step 6 of 3.3, use the correct integer multiplier.
4) If the timing is step 1 above is not correct for the Networked Driver, and the correct time is *not* an integer multiple of this timing, then proceed with step 3.2.

3.2 Create New PLC Task

You only have to follow this step if the Main PLC task timing verified in step 3.1 above is not the correct update time for the Networked Driver and the correct time is not an integer multiple of the Main PLC task timing.

1) In the *Solutions Explorer*, expand out *System* → *Tasks* so that the tasks are visible. Right click on *Tasks* and select "Add New Item...".
2) In the window that opens, name the new task and select "TwinCAT Task With Image" as the *Type*. Click the [OK] button to accept the values.



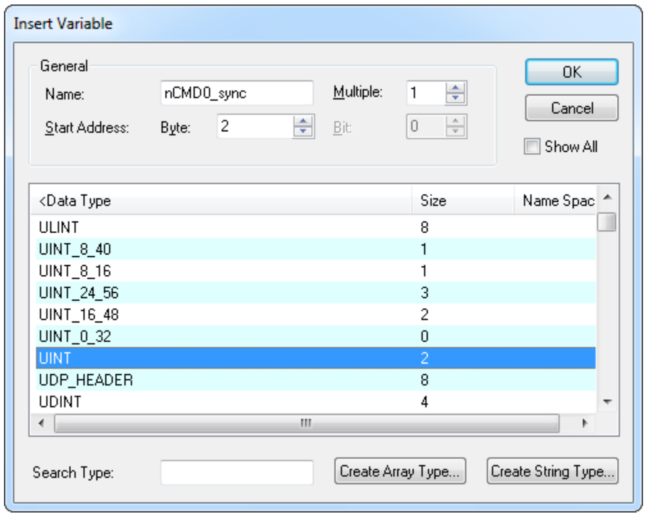
```
PROGRAM MAIN
VAR
    st_Axis1_SD17060E-K : DUT_A
END_VAR
```

Name: SD_Sync

Type: TwinCAT Task With Image

Figure T3.2 Create New Task

3) In the window that open, enter the desired update time in Cycle Ticks. By default, each Cycle Tick is 1 millisecond.
4) If needed, expand out the new task so that the Outputs icon is visible.
5) Right click on the Outputs icon and select "Add New Item...". The *Insert Variable* window will open.
6) Name the new variable. Under >*Data Type*, select "UINT". Click the [OK] button to close the *Insert Variable* window.



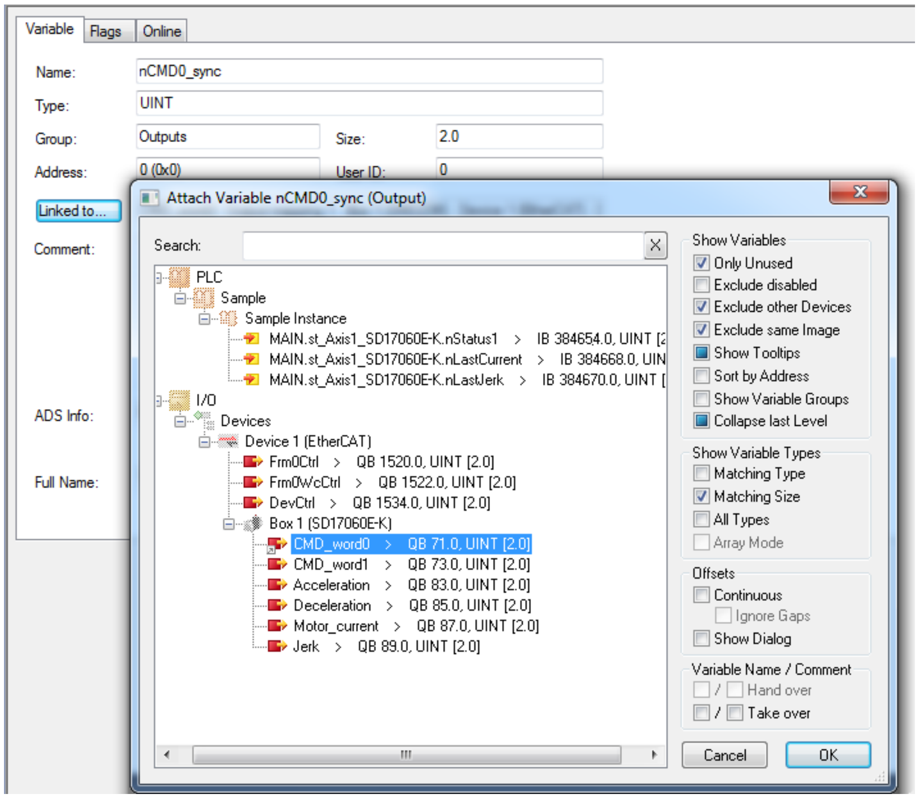
Name: nCMD0_sync
Multiple: 1
Start Address: Byte: 2 Bit: 0

<Data Type	Size	Name Spac
ULINT	8	
UINT_8_40	1	
UINT_8_16	1	
UINT_24_56	3	
UINT_16_48	2	
UINT_0_32	0	
UINT	2	
UDP_HEADER	8	
UDINT	4	

Figure R8.3 Setting Task Variable Name and Size

3.2 Create New PLC Task (continued)

7) In the *Solutions Explorer*, click on the name of the new variable to select it. The information on the variable will appear in a pane. Click on the [Linked to...] button in this pane to open the *Attach Variable* window.
8) Double click on the CMD_word0 variable of the correct Networked Driver. This will link the CMD_word0 of the unit to the variable created for the task. The Networked Driver will update at the time specified by the new task.



Variable | Flags | Online

Name: nCMD0_sync
Type: UINT
Group: Outputs Size: 2.0
Address: 0 (0x0) User ID: 0
Linked to...
Comment:
ADS Info:
Full Name:

Attach Variable nCMD0_sync (Output)

Search:

```
PLC
  Sample
    Sample Instance
      MAIN.st_Axis1_SD17060E-K.nStatus1  >  IB 384654.0, UINT [2
      MAIN.st_Axis1_SD17060E-K.nLastCurrent  >  IB 384668.0, UIN
      MAIN.st_Axis1_SD17060E-K.nLastJerk  >  IB 384670.0, UINT [
I/O
  Devices
    Device 1 (EtherCAT)
      Frm0Ctrl  >  QB 1520.0, UINT [2.0]
      Frm0WcCtrl  >  QB 1522.0, UINT [2.0]
      DevCtrl  >  QB 1534.0, UINT [2.0]
      Box 1 (SD17060E-K)
        CMD_word0  >  QB 71.0, UINT [2.0]
        CMD_word1  >  QB 73.0, UINT [2.0]
        Acceleration  >  QB 83.0, UINT [2.0]
        Deceleration  >  QB 85.0, UINT [2.0]
        Motor_current  >  QB 87.0, UINT [2.0]
        Jerk  >  QB 89.0, UINT [2.0]
```

Show Variables
- [✓] Only Unused
- [ ] Exclude disabled
- [✓] Exclude other Devices
- [✓] Exclude same Image
- [■] Show Tooltips
- [ ] Sort by Address
- [ ] Show Variable Groups
- [■] Collapse last Level

Show Variable Types
- [ ] Matching Type
- [✓] Matching Size
- [ ] All Types
- [ ] Array Mode

Offsets
- [ ] Continuous
- [ ] Ignore Gaps
- [ ] Show Dialog

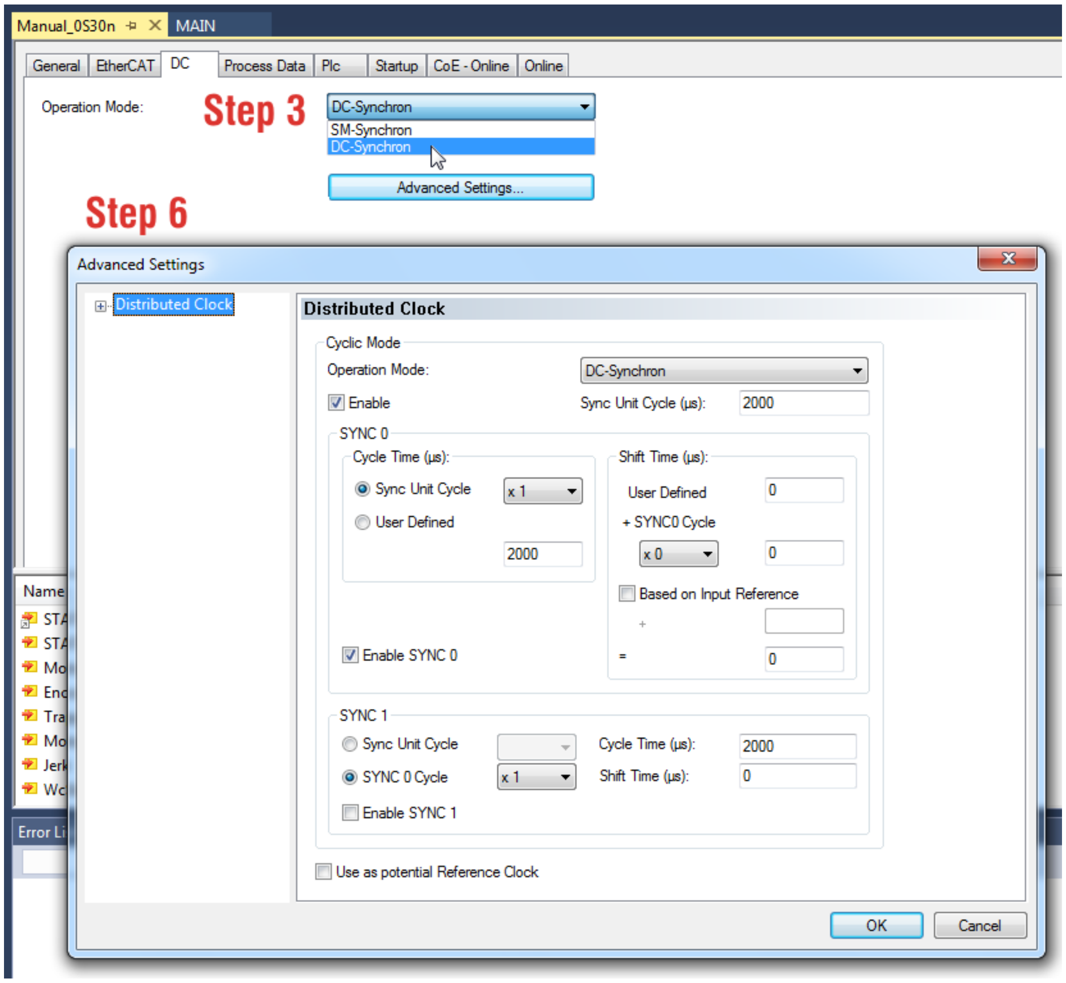
Variable Name / Comment
- [ ] / [ ] Hand over
- [ ] / [ ] Take over

Cancel | OK

Figure T3.3 Linking Task Variable to the Networked Driver

3.3 Set Operational Mode

1) If necessary, expand the I/O Device tree until the Networked Driver is visible.
2) Double click on the Networked Driver to open the setting window for the device.
3) Click on the “DC” tab.
4) Under the Operation Mode setting, click on the drop down menu and select “DC_Synchron”.
5) Click on the [Advanced Settings...] button.
6) Verify or set the following:
   - *Cyclic Mode -> Enable* checkbox is checked.
   - *Sync Unit Cycle (μs)* is equal to the time of the correct task. If not, the variables are not linked correctly.
   - *SYNC 0 -> Sync Unit Cycle* radio button is enabled. In the drop down menu, set the multiplier as needed so that the *Sync Unit Cycle (μs)* time multiplied by the multiplier value is between 2 and 50 milliseconds.
   - *Enable SYNC 0* checkbox is checked.
   - *Enable SYNC 1* checkbox is not checked.
7) Click [OK] to close the window.



Operation Mode: DC-Synchron

Distributed Clock

Cyclic Mode
Operation Mode: DC-Synchron
[✓] Enable   Sync Unit Cycle (µs): 2000

SYNC 0
Cycle Time (µs): [✓ Sync Unit Cycle] x 1 / User Defined 2000
Shift Time (µs): User Defined 0; + SYNC0 Cycle x 0 0
[ ] Based on Input Reference
+
= 0
[✓] Enable SYNC 0

SYNC 1
Sync Unit Cycle / [✓ SYNC 0 Cycle] x 1
Cycle Time (µs): 2000
Shift Time (µs): 0
[ ] Enable SYNC 1

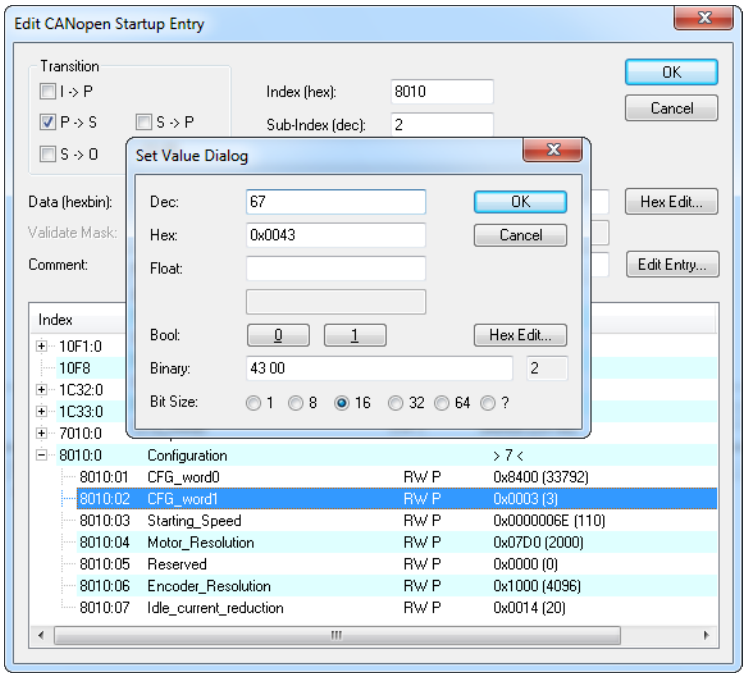
[ ] Use as potential Reference Clock

Figure T3.4 Distributed Clock Settings

3.4 Set CFG_word1 Value to Enable SYNC0

Continuing from step 3.3...

1) Click on the “CoE - Online” tab in the settings window.
2) Expand the 8010:0 tree to see the value of the 8010:02 register. (CFG_word1 value)
3) This value must have bit 6 set to enable the synchronous mode in the Networked Driver. If this value is greater than or equal to 64, then the bit is set and you are finished with this step of the procedure. If the value is less than 64, then the bit is not set. This new value must be specified in the Startup tab in order to make the change permanent.
4) Click on the “Startup” tab.
5) Right click on the blank surface and select “Add New Item...” in the resulting pop up menu.
6) Click on the [+] button next to the 8010:0 Index field to expand it.
7) Double click on the 8010:02 field. This opens a dialog that will allow you to set the field with a decimal or hexadecimal value. If setting in decimal, the new value is the present value from step 3 + 64.
8) Click the [OK] button in the dialog box to set the new value.
9) Click the [OK] button in the Edit CANopen Startup Entry screen to accept the new startup parameter setting.



Edit CANopen Startup Entry

Transition: [ ] I -> P; [✓ P -> S]; [ ] S -> P; [ ] S -> O

Index (hex): 8010

Sub-Index (dec): 2

Data (hexbin):

Validate Mask:

Comment:

Set Value Dialog

Dec: 67

Hex: 0x0043

Float:

Binary: 43 00 2

Bit Size: ( ) 1 ( ) 8 [✓ 16] ( ) 32 ( ) 64 ( ) ?

Index			
10F1:0			
10F8			
1C32:0			
1C33:0			
7010:0			
8010:0	Configuration		> 7 <
8010:01	CFG_word0	RW P	0x8400 (33792)
8010:02	CFG_word1	RW P	0x0003 (3)
8010:03	Starting_Speed	RW P	0x0000006E (110)
8010:04	Motor_Resolution	RW P	0x07D0 (2000)
8010:05	Reserved	RW P	0x0000 (0)
8010:06	Encoder_Resolution	RW P	0x1000 (4096)
8010:07	Idle_current_reduction	RW P	0x0014 (20)

Figure T3.5 Set DC_Sync Bit in Configuration Word 1

ADVANCED MICRO CONTROLS INC.

20 GEAR DRIVE, TERRYVILLE, CT 06786 T: (860) 585-1254 F: (860) 584-1973

www.amci.com

LEADERS IN ADVANCED CONTROL PRODUCTS