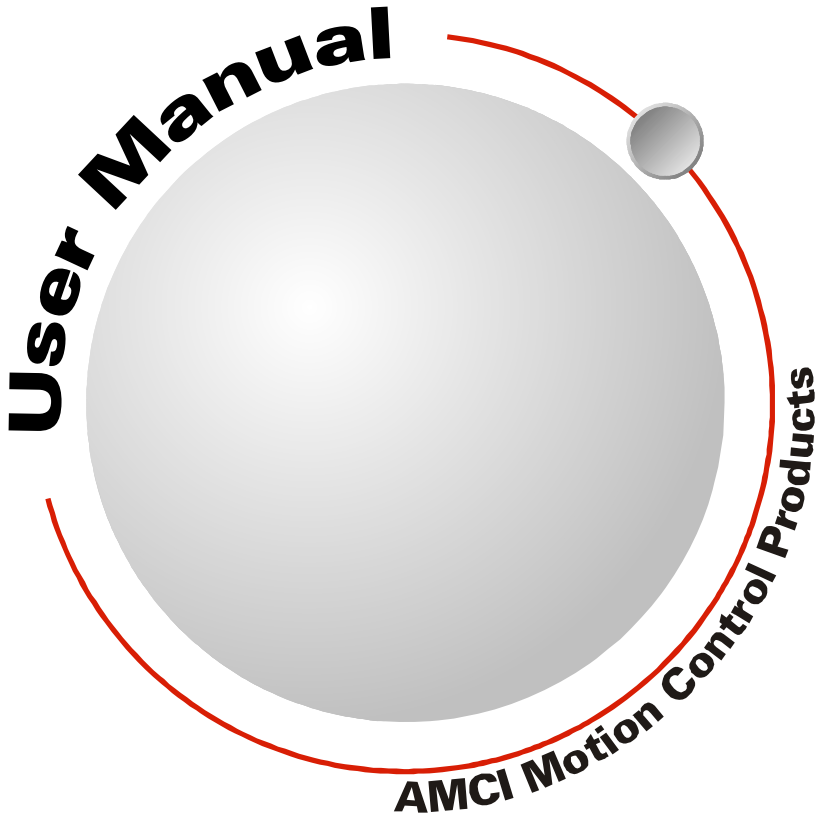
ADVANCED MICRO CONTROLS INC.

Manual #: 940-0S220

SMD17E2

Integrated Stepper Indexer/Driver/Motor

with Integral 2-Port Ethernet Switch
Device Level Ring functionality for EtherNet/IP
Media Redundancy Protocol for PROFINET



EtherNet/IP™

PROFINET® INDUSTRIAL ETHERNET

Modbus

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.

We Want Your Feedback

Manuals at AMCI are constantly evolving entities. Your questions and comments on this manual are both welcomed and necessary if this manual is to be improved. Please direct all comments to: Technical Documentation, AMCI, 20 Gear Drive, Terryville CT 06786, or fax us at (860) 584-1973. You can also e-mail your questions and comments to *techsupport@amci.com*

Table of Contents

Reference: Motion Control (continued)

Reference: Calculating Move Profiles

Reference: Homing an SMD17E2

Reference: Configuration Mode Data Format

Reference: Command Mode Data Format

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 the SMD17E2 Integrated Stepper Indexer/Driver/Motors from AMCI. 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.

These devices support the EtherNet/IP, Modbus TCP, and PROFINET protocols. Each unit contains a two port Ethernet switch, which simplifies network wiring. When the EtherNet/IP protocol is used, the unit can act as a node in a Device Level Ring (DLR). When PROFINET is enabled, the unit supports the Media Redundancy Protocol (MRP) and can be incorporated in PROFINET installations that use a redundant ring topology.

Applicable Units

This manual applies to all of the units in the SMD17E2 family.

Model Number	Description
SMD17E2-60-M12	Size 17 motor, 60 oz-in holding torque with sealed M12 connectors for an IP50 rating.
SMD17E2-80-M12	Size 17 motor, 80 oz-in holding torque with sealed M12 connectors for an IP50 rating.
SMD17E2-60A-M12	Same as SMD17E-60-M12 with an integrated absolute multi-turn encoder.
SMD17E2-80A-M12	Same as SMD17E-80-M12 with an integrated absolute multi-turn encoder.
SMD17E2-60E-M12	Same as SMD17E-60-M12 with an integrated incremental encoder.
SMD17E2-80E-M12	Same as SMD17E-80-M12 with an integrated incremental encoder.
SMD17E2-60-M12S	Size 17 motor, 60 oz-in holding torque with sealed M12 connectors and shaft seal for an IP64 rating.
SMD17E2-80-M12S	Size 17 motor, 80 oz-in holding torque with sealed M12 connectors and shaft seal for an IP64 rating.
SMD17E2-60A-M12S	Same as SMD17E2-60-M12S with an integrated absolute multi-turn encoder.
SMD17E2-80A-M12S	Same as SMD17E2-80-M12S with an integrated absolute multi-turn encoder.
SMD17E2-60E-M12S	Same as SMD17E2-60-M12S with an integrated incremental encoder.
SMD17E2-80E-M12S	Same as SMD17E2-80-M12S with an integrated incremental encoder.

Part Number Description

NOTE This manual also applies to all products in the SMD17A2 product line. SMD17A2 products are identical to their SMD17E2 counterparts, but are factory configured for the PROFINET protocol instead of the EtherNet/IP protocol.

Trademark Notices

The AMCI logo is a trademark of Advanced Micro Controls Inc.

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

Revision Record

This manual, 940-0S220, is the first revision of this manual. It was released June 18th, 2018.

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 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:

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

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

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 the first time a new term is introduced.
Cross Reference	When viewing the PDF version of the manual, clicking on the cross reference text jumps you to referenced section.
HTML Reference	When viewing the PDF version of the manual, clicking on the HTML reference text will open your default web browser to the referenced web page.

Manual Layout

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

- If you are curious about the Integrated Stepper Indexer/Driver/Motor 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, ***SMD17E2 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 an Integrated Stepper Indexer/Driver/Motor 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 *references* and *tasks*. Using an Integrated Stepper Indexer/Driver/Motor product requires you to complete multiple tasks, and the manual is broken down into sections that explain how to complete each one.

Section Title	Section Description
SMD17E2 Specifications	Complete specifications for the SMD17E2 products.
Motion Control	Reference information on how the SMD17E2 can be used to control motion in your application.
Calculating Move Profiles	Reference information on calculating detailed move profiles.
Homing an SMD17E2	Reference information on how to set the home position of the SMD17E2.
Configuration Mode Data Format	Reference information on the format of the network data to and from the SMD17E2 that is used to configure it.
Command Mode Data Format	Reference information on the format of the network data to and from the SMD17E2 that is used to command it.
Installing the SMD17E2	Task instructions covering how to install an SMD17E2 on a machine. Includes information on mounting, grounding, and wiring specific to the units.
Set the IP Address and Protocol	Task instructions that covers the options for setting the IP address on an SMD17E2.
Implicit Communications with an EDS	Task instructions that cover how to add an SMD17E2 to an EtherNet/IP host that supports the use of EDS files.
Implicit Communications Without an EDS	Task instructions for adding an SMD17E2 to a project as a generic device. This configuration is for EtherNet/IP hosts that do not support EDS files while supporting implicit communications.
EtherNet/IP Explicit Messaging	Task instructions for adding message instructions to you host controller program that write data to the SMD17E2 through message instructions.
Modbus TCP Configuration	Task instructions for communicating with an SMD17E2 using the Modbus TCP protocol.
PROFINET Network Configuration	Task instructions for communicating with an SMD17E2 using the PROFINET protocol.
Optional: ***Configure Your Network Interfaces***	Instructions for the optional task of configuring network interfaces on your computer or laptop.

Manual Sections

Notes

REFERENCE 1

SMD17E2 SPECIFICATIONS

This manual is designed to get you up and running quickly with an SMD17E2 product 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.

The SMD17E2 Family

The SMD17E2 units are part of a growing product line from AMCI with a simple concept: a stepper indexer, driver, and motor that can be attached to any popular industrial network. Each SMD17E2 attaches to your Ethernet network and communicates using the EtherNet/IP, Modbus TCP, or PROFINET protocols.

Each unit has two Ethernet ports which are internally connected through an onboard, two port, 10/100 Mbps ethernet switch. These ports allow you to wire your network in a "daisy-chain" fashion, which may lower network wiring costs and complexities.

The two ports also allow the SMD17E2 products to function as members of a redundant Device Level Ring (DLR) network when using the EtherNet/IP protocol or as clients in a Media Redundancy Protocol (MRP) network when using PROFINET.

In DLR environments, the SMD17E2 units act as Beacon-Based Ring Nodes. All SMD17E2 units can process beacon packets at the default rate of every 400 microseconds. Beacon-based nodes can respond faster to network changes than nodes that only process Announce packets.

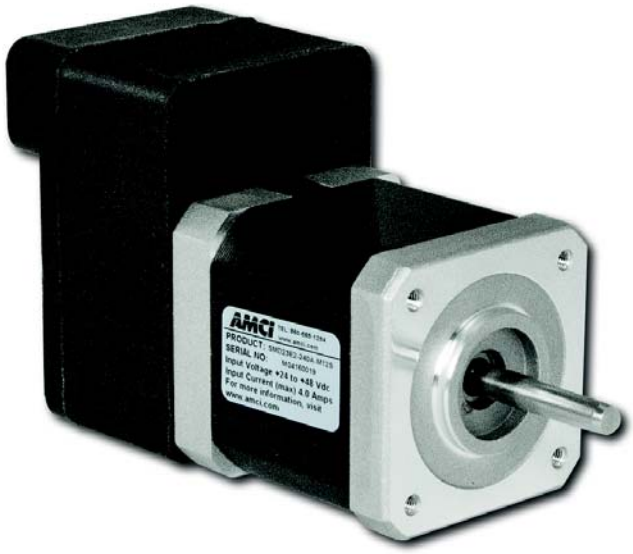


Figure R1.1 IP50 Rated SMD17E2

Each unit can be ordered with an optional incremental or absolute multi-turn encoder. This encoder gives you the additional functionality of position verification and stall detection. The absolute multi-turn encoder allows you to track machine position with power removed, eliminating the need to home the machine after cycling power.

The SMD17E2 Family (continued)

Part Numbering System

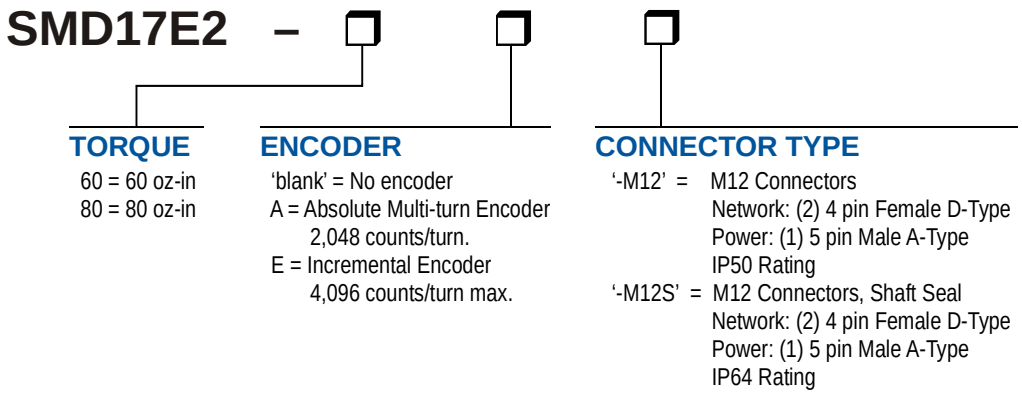


Figure R1.2 Part Numbering System

General Functionality

Each member of the SMD17E2 family has three integrated parts:

- An indexer that accepts commands over an Ethernet connection using the EtherNet/IP, Modbus TCP, or PROFINET protocol
- A 2.0 Arms micro-stepping driver that accepts 24 to 48 Vdc as its input power source
- A high torque size 17 stepper motor (60 or 80 oz-in holding torque).

An incremental or absolute multi-turn encoder is also available for applications that require position feedback or verification.

The availability of the EtherNet/IP, Modbus TCP, or PROFINET protocols makes the SMD17E2 units easy to integrate into a wide variety of control systems. This 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 SMD17E2 unit
- All motion logic is programmed in the host, eliminating the need to learn a separate motion control language
- The integral two port Ethernet switch simplifies network cabling
- The DLR interface eliminates single point failures in EtherNet/IP environments
- The MRP interface eliminates single point failures in PROFINET environments
- The elimination of the separate indexer and driver lowers total system cost.

An SMD17E2 is powered by a nominal 24 to 48 Vdc power source, and can accept surge voltages of up to 60 Vdc without damage. The output motor current is fully programmable from 0.1 Arms to 2.0 Arms which makes the SMD17E2 suitable to a wide range of applications. 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 programming an SMD17E2 to be very similar to these products.

The SMD17E2 contains a true RMS motor current control driver. This means that you will always receive the motor's rated torque regardless of the *Motor Steps/Turn* setting. (Drivers that control the peak current to the motor experience a 30% decrease in motor torque when microstepping a motor.) The combination of an ultra-low inductance motor and a high-power, true RMS driver gives unprecedented torque vs. speed performance for any DC application.

The SMD17E2 Family (continued)

General Functionality (continued)

The SMD17E2 units have two DC inputs that are used by the indexer. Configuration data from the host sets the function of these inputs. Each input can be individually configured as a:

- CW or CCW Limit Switch
- Home Limit Switch
- Capture Position Input (Will capture encoder position on units with the internal encoder.)
- Stop Jog or Registration Move Input
- Start Indexer Move
- Emergency Stop Input
- General Purpose Input

Encoder Functionality

All SMD17E2 units can be ordered with an internal incremental or absolute multi-turn encoder. Incremental encoders can be programmed to 1,024, 2,048, or 4,069 counts per turn. Absolute encoders have a fixed resolution of 2,048 counts per turn and encode a total of 2^{21} turns. (32 bits total.) Using an encoder gives you the ability to:

- Verify position during or after a move
- Detect motor stall conditions
- Maintain machine position when power is removed if using an absolute encoder.
- Close the position/velocity loop with the encoder position instead of the motor position when following a virtual axis.

The motor position can be preset to the encoder position with a single command. SMD17E2 units with absolute encoders allow you to preset the encoder position and save the resulting offset in Flash memory.

Network Data Format

In order to support any host that communicates with the supported protocols, the format of the data read from and written to the SMD17E2 while in command mode is completely programmable. The format of the network input and output data can be programmed separately.

The smallest data size used by the SMD17E2 is the sixteen bit word, however some parameters and data values can exceed this size. For these thirty-two bit values, the default data format is referred to as the *multi-word* format. The data value is split between the hundreds digit and the thousands digit. For example, a value of 12,345 would have 12 placed in the first (lower addressed) word, and 345 placed in the second (higher addressed) word. This format greatly simplifies setting parameter values when programming command blocks.

The other data format is a signed thirty-two bit integer format. When using the thirty-two bit format, there is one additional parameter named *Data Endian*. Its use is best explained with an example. The value of 123,456 equals 0001:E240 in hexadecimal. When storing and transmitting this data, some host controllers will store the least significant word (16#E240) in the lower addressed word in their data tables, while others will store the most significant word (16#0001) stored in the lower addressed word in their data tables. These controllers expect thirty-two bit values to be returned to them using the same format. Least significant word first is called *little endian*, most significant word first is called *big endian*. Rockwell Automation controllers use *little endian* format, the default Modbus format is *big endian*, and the default PROFINET format is *big endian*.

NOTE The range of values when using the multi-word format is -32,768,000 to 32,767,999. When used in continuous rotation applications, it is possible to overflow these values. When a position value overflows, the value of the associated data words will become indeterminate. AMCI strongly suggests using the signed 32-bit integer format for continuous rotation applications.

Specifications

Network Interface

10/100baseT. Two switched ports.
Supports EtherNet/IP, Modbus TCP, and PROFINET. EtherNet/IP-DLR and PROFINET-MRP extensions also supported.

Driver Type

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

Physical Dimensions

See Outline Drawings, starting on page 87

Weight

SMD17E2-60-M12 ..... 0.90 lbs. (0.41 kg.)
SMD17E2-80-M12 ..... 1.07 lbs. (0.49 kg.)
All weights are without mating connectors

Maximum Shaft Loads

Radial: 6.5 lbs (29 N) at center of flat on shaft

Axial: 5.6 lbs (25 N)

Maximum Operating Temperature

203°F /95°C (Note that this is the operating temperature of the motor, not maximum ambient temperature. An Over Temperature fault occurs at this point and current is removed from the motor.)

Over Temperature Fault

Over temperature faults are reported in the Network Input Data.

Inputs

Electrical Characteristics:
IN1 and IN2: Single ended sinking.

Accept 3.5 to 27Vdc without the need for an external current limiting resistor. Optoisolated, 1500 Vac/dc isolation.

Specifications

Motor Current

Programmable from 0.1 to 2.0 Arms in 0.1 A steps.

$DCPower_{AUX}$ Current

70 mA @ 24Vdc, 40mA @48Vdc

Motor Counts per Turn

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

Internal Encoder (Optional)

Incremental encoder option supplies 1,024, 2,048, or 4,096 counts per turn.

Absolute encoder option supplies 2,048 counts per turn, 32 bit max. counts.

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.

Environmental Specifications

Input Power ..... 24 to 48 Vdc, surge to 60 Vdc without damage to unit.

Ambient Operating Temperature

..... -40° to 122°F (-40° to 50°C)

Storage Temperature

..... -40° to 185°F (-40° to 85°C)

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

IP Rating ..... IP50 or IP64 with shaft seal.

Status LED's

See *Status LED's* section starting on page 19.

Connectors and Cables

All mating connectors are available separately under the following AMCI part numbers.

Connector	AMCI Part #	Wire	Strip Length	Connection Type
Ethernet	MS-28	18 AWG max.	0.197 inches	Screw Terminals
I/O	MS-31	18 AWG max.	0.197 inches	Screw Terminals

Cable	AMCI Part #	Length
Ethernet	CNER-5M	5 meter
Power & I/O	CNPL-2M	2 meter
Power & I/O	CNPL-5M	5 meter

Indexer Functionality

The table below lists the functionality offered by the indexer built into the SMD17E2.

Feature	Description
EtherNet/IP, PROFINET, Modbus TCP	Allows easy setup and communication with a wide range of host controllers such as the latest PLC's from Allen-Bradley.
EtherNet/IP-DLR	SMD17E2 units have Device Level Ring functionality, which adds redundancy to the EtherNet/IP protocol.
PROFINET-MRP	SMD17E2 units have Media Redundancy Protocol support, which adds redundancy to the PROFINET protocol.
Programmable Inputs	Each of the inputs can be programmed as a Home Limit, Over Travel Limit, Capture Input, Manual Jog Stop, Start Indexer Move, E-Stop, or a General Purpose Input.
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. An SMD17E2 homes to a discrete input and can use a bit in the Network Data as a home proximity input.
Jog Move	Allows you to drive the motor in either direction as long as the command is active.
Synchrostep Move	Allows you to treat the SMD17E2 as a motion axis by constantly updating position and velocity data to the unit.
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 an input bit from your host controller. When a controlled stop is issued, the move will output a programmable number of steps before coming to a stop.
Blend Move	Allows you to perform a sequence of relative moves without stopping between them.
Dwell Move	Allows you to perform a sequence of relative moves with a stop between each move that has a programmable length of time. Used to create highly accurate move profiles that avoid network latency issues.
Indexer Move	Allows you to program a move that is held in memory. The move is run when one of the programmable inputs makes a transition. Note that an Indexer Move requires a connection to a host controller to program the move.
Hold Move	Allows you to suspend a move, and optionally 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.
Stall Detection	When an SMD17E2 is purchased with an encoder option, the encoder can be used to verify motion when a move command is issued.

Table R1.1 Indexer Functionality

Indexer Functionality (continued)

Stall Detection with SMD17E2 Units

Stall Detection is one of the additional features available to you when you order an encoder option on an SMD17E2. When Stall Detection is enabled, the SMD17E2 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 SMD17E2 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. As described later in this chapter, there is an auxiliary power pin that powers the electronics of an SMD17E2 but does not power the motor. The primary use of this feature is to keep the unit on the network while power is removed from the motor. When using the $DCPower_{AUX}$ pin, the SMD17E2 cannot sense when power has been removed from the $DCPower_{MAIN}$ pin. By enabling stall detection, the SMD17E2 can notify the system if the motor shaft moves more than forty-five degrees while power is removed from the motor.

Driver Functionality

This table summarizes the features of the stepper motor driver portion of an SMD17E2.

Feature	Benefits
RMS Current Control	RMS current control give an SMD17E2 the ability to drive the motor at its fully rated power regardless of the programmed steps per turn. There is no reduction in power when microstepping that may occur with other drivers.
Programmable Motor Current	RMS current supplied to the motor can be programmed from 0.1 to 2.0 amps in 0.1 amp increments. Reducing the motor current to the minimum needed for your application will significantly reduce the motors operating temperature
Programmable Idle Current Reduction	Extends motor life by reducing the motor current when motion is not occurring. This extends the life of the motor by reducing its operating temperature.
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 feedback circuitry and algorithm gives the SMD17E2 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.
Over Temperature Detection	An SMD17E2 sets a warning bit in the network data when the temperature of the unit approaches its safe operating threshold.
Over Temperature Protection	Protects your SMD17E2 from damage by removing power from the motor if the internal temperature of the driver exceeds the safe operating threshold of 203°F/95°C.

Table R1.2 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, which results in an exponential, not linear, drop in motor temperature. This means that even a small reduction in the idle current can have a significant effect on the temperature of the motor.

NOTE Note that the reduction values are "to" values, not "by" values. Setting a motor current to 2 Arms and the current reduction to 25% will result in an idle current of 0.5 Apk. (The SMD17E always switches from RMS to peak current control when the motor is idle to prevent motor damage due to excessive heating.)

Available Discrete Inputs

The SMD17E2 has two discrete, sinking, DC inputs that accept 3.5 to 27Vdc signals. (5 to 24Vdc nominal) How your SMD17E2 uses these inputs is fully programmable. The active state of each input is also programmable. Programming their active states allow them to act as Normally Open (NO) or Normally Closed (NC) contacts.

Home Input

Many applications require that the machine be brought to a known position before normal operation can begin. This is commonly called "homing" the machine or bringing the machine to its "home" position. An SMD17E2 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 SMD17E2 that will cause the unit to seek this sensor. How the SMD17E2 actually finds the Home sensor is described in the ***Homing an SMD17E2*** chapter starting on page 53.

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 jog 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 an SMD17E2 to run critically timed moves that cannot be reliably started from the network due to data transfer lags.

If the unit was ordered with the encoder option, and one of the discrete DC inputs is programmed as a Start Indexer Move Input, then the encoder position data will be captured whenever the DC input makes a transition. An inactive-to-active state transition on the DC input will also trigger an Indexer Move if one is pending.

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. Any type of move, including a Jog or Registration Move, cannot begin while this input is active.

Available Discrete Inputs (continued)

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 Move 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 this way, all other moves ignore this input.

If the unit was order with an integral encoder, the encoder position data will be captured when the DC input makes an inactive-to-active transition if it is configured as a Stop Jog or Registration Move Input. 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 a DC input, 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, an SMD17E2 can be configured to capture the encoder position value on a transition of a discrete DC input.

General Purpose Input

If your application does not require one or both of the inputs, you can configure the unused inputs as General Purpose Inputs. The inputs are not used by the SMD17E2, but their on/off state is reported in the network data and is available to your host controller.

Optional Encoder

The SMD17E2 can be ordered with an integral encoder. An input can be configured to capture the encoder value when the input makes an inactive to active transition. This captured value is written to the host controller. Two encoder options are available:

Incremental Encoder

The incremental encoder can be programmed to 1,024, 2,048, or 4,096 counts per turn. The SMD23/4E2 has an internal thirty-two bit counter associated with the encoder. The incremental encoder are primarily used for position verification and stall detection.

Absolute Multi-turn Encoder

The absolute encoder has a fixed resolution of 2,048 counts per turn. The absolute encoder is a multi-turn device that encodes a total of 2^{21} turns, yielding a full thirty-two bits of position resolution. The absolute encoder can be used for position verification and stall detection, but its primary advantage is that it eliminates the need to home the axis after cycling power to the drive.

Like many intelligent absolute encoders on the market today, the absolute encoder in the SMD17E2 uses a battery backed circuit to count zero crossings while power is removed from the rest of the device. The circuit will accurately track position as long as the shaft acceleration is limited to 160,000 degrees/sec^2, (444.4 rev/sec^2), or less. Battery life is a minimum of 10 years in the absence of power. The battery cannot be replaced in the field.

Status LED's

Each SMD17E2 has two status LED's that show module and network status. As shown in figure R1.3, these LED's are located on the rear cover.

Module Status (MS) LED

The Module LED is a bi-color red/green LED that show the general status of the unit.

- **Steady Green:** Unit OK
- **Steady Red:** An Overtemperature Fault exists.
- **Blinking Green:** Successful write to flash memory. Power must be cycled to the unit before additional commands can be written to it.
- **Blinking Red:** Failed write to flash memory. You must cycle power to the unit to clear this fault.
- **Alternating Red/Green:** Communications failure. There is a communications error between the main processor and the ethernet co-processor within the unit. You must cycle power to the SMD17E2 to attempt to clear this fault.

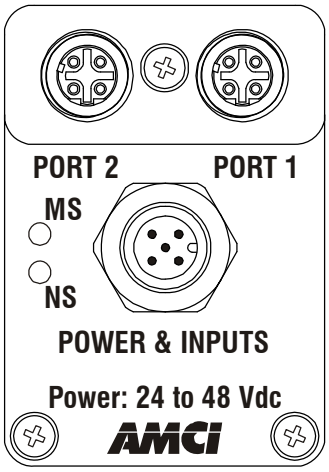


Figure R1.3 Rear Cover Status LED's

Power Up Behavior

- **Blinking Green:** The unit will blink the Module Status LED green during initialization.
- **Blinking Red:** The unit will blink the Module Status LED red three times if there is an error with the internal absolute encoder.

Network Status (NS) LED

The Network Status LED is a bi-color red/green LED. The state of the LED depends on the protocol the SMD17E2 is configured to for.

LED State	EtherNet/IP Definition	Modbus TCP Definition	PROFINET Definition
Off	No Power	No power or no TCP connections	No power, duplicate IP address on the network, or no connection to IO Controller.
Alternating Red/Green	Power up Self-Test	Power up Self-Test	Power up Self-Test
Flashing Green	Ethernet connection, but no CIP connections	Indicates number of connections with 2 second delay between group. The SMD17E2 supports up to 3 concurrent connections.	On-line, Stop state. A connection with the IO Controller is established and it is in its STOP state.
Steady Green	Valid Ethernet network and CIP connections	Should not occur. LED should always flash when network is connected.	On-line, Run state. A connection with the IO Controller is established and it is in its RUN state.
Flashing Red	Network Connection Timeout	Not implemented in Modbus TCP.	Not Implimented
Steady Red	Duplicate IP address on network.		Not Implimented.

Table R1.3 Network Status LED States

SMD17E2 Connectors

Input Connector

As shown in figure R1.4, the Input Connector is located on the back of the unit near its center. All digital input and power supply connections are made at this connector. The connector is a standard five pin A-coded M12 connector that is rated to IP67 when the mate is properly attached. Figure R1.5 shows the pinout of the connector when viewed from the back of the SMD17E2.

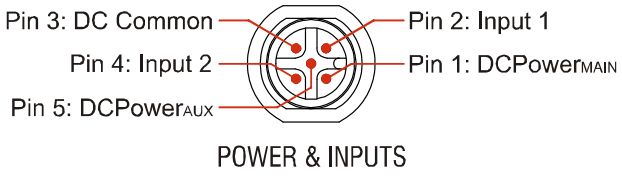


Figure R1.5 M12 Input Connector

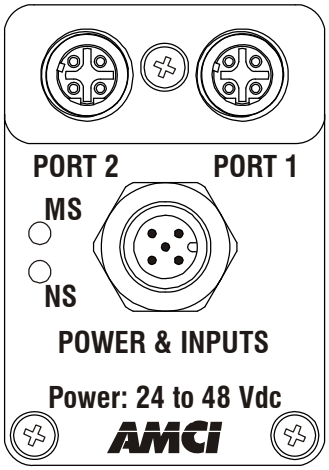


Figure R1.4 SMD17E2 Connector Locations

Digital inputs are single ended and referenced to the DC Common pin. (Sinking inputs.)

There are two power pins. $DCPower_{MAIN}$ powers both the control electronics and the motor. $DCPower_{AUX}$ powers only the control electronics. Using the $DCPower_{AUX}$ pin is optional. If your application requires you to cut power to your motor under some conditions, using the $DCPower_{AUX}$ pin allows you to cut power to your motor without losing your network connection.

NOTE If the unit was ordered with an encoder, the $DCPower_{AUX}$ pin will also maintain power to the encoder. If the motor shaft is rotated while motor power is removed, the encoder position will update. (The motor position will not update.) Once power is restored to the motor, a Preset Position command can be issued to restore the correct motor position without having to go through a homing sequence. If Stall Detection is enabled on the SMD17E2, it will also be able to tell the system if the motor shaft rotated more than forty-five degrees with power removed.

SMD17E2 Connectors (continued)

Ethernet Connectors

Figure R1.4 also show the placement of the sealed Ethernet Connector(s), while figure R1.6 shows the connector pinout when viewed from the back of the SMD17E2. The Ethernet port on the SMD17E2 is an "auto-sense" port that will automatically switch between 10baseT and 100baseT depending on the network equipment it is attached to. The port also has "auto switch" capability. This means that a standard cable can be used when connecting the SMD17E2 to any device, including a personal computer.

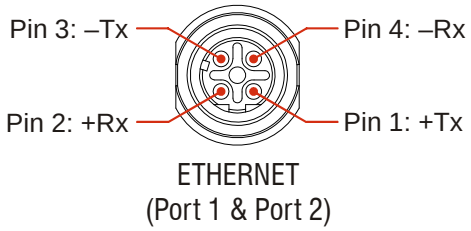


Figure R1.6 Ethernet Connector Pinout

The connector is a standard four pin D-coded female M12 connector that is rated to IP67 when the mate is properly attached.

Torque and Power Curves

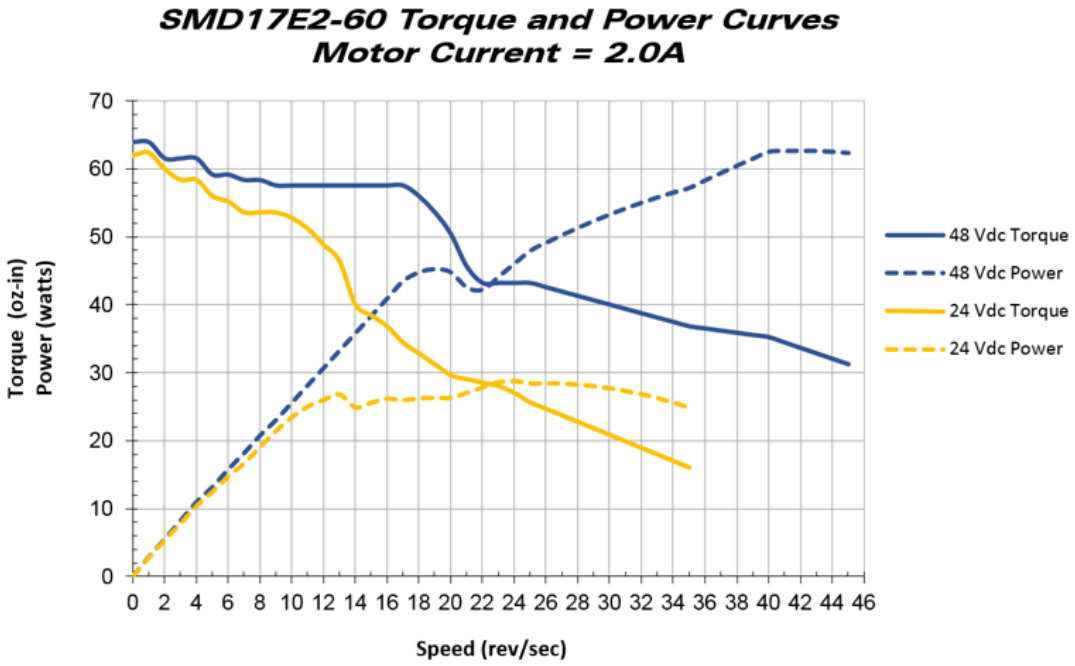


Figure R1.7 SMD17E2-60 Torque and Power Curves

Torque and Power Curves (continued)

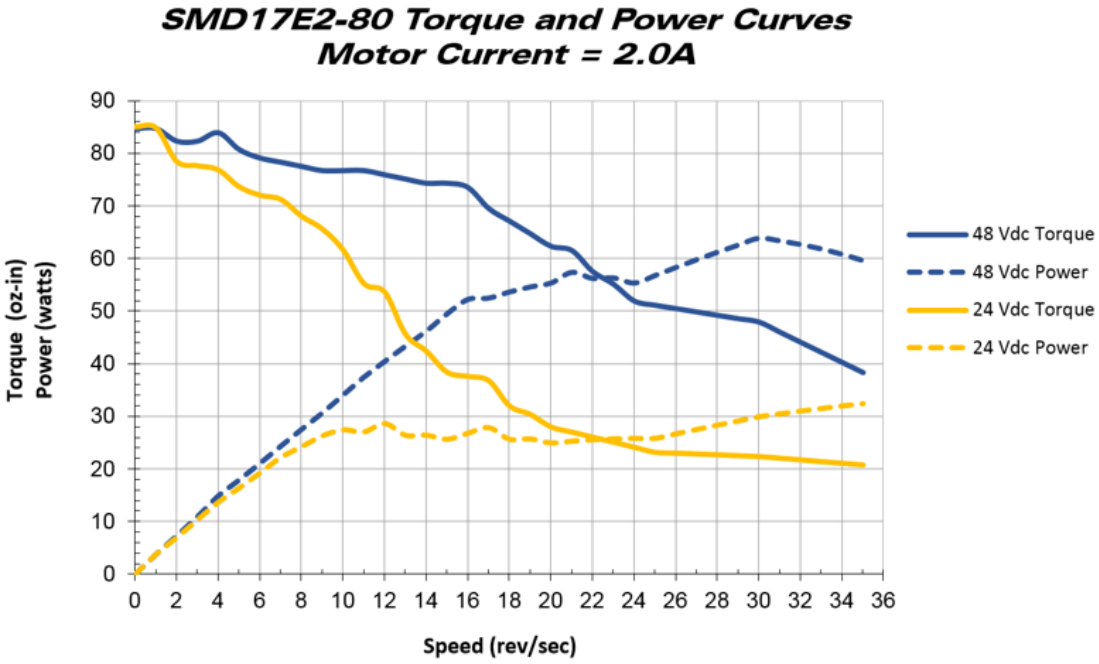


Figure R1.8 SMD17E2-80 Torque and Power Curves

Power Supply Sizing

The power supply can be sized based on the power the motor must generate during its operation. As a general guideline, your supply should be able to produce 150% to 175% of the power the motor can produce. The previous power and torque curves can be used to determine the maximum power the motor can generate over its speed range.

Note that the power value you should use is the *maximum* power value over the range of speeds that the motor will be operated at. The power generated by the motor decreases towards the end of its usable speed. Therefore, the power generated at your machine's operating point may be less than the maximum the motor can generate at a lower speed.

Example 1: An SMD17E2-60 will be running at a maximum of 12 RPS and a 48 Vdc supply will be used. Based on the power curve in figure R1.7 on page 21, the combinations will generate a maximum of 31 Watts. Therefore a 48 Vdc supply with a power range of 47 W to 55 W can be used in the application.

Example 2: An SMD17E2-80 will be running at a maximum of 24 RPS and a 48 Vdc supply will be used. Based on the power curve in figure R1.8 above, the power at this speed is 55 W, but the maximum power over the entire speed range is 58 W, which occurs at 21 RPS. Therefore, the 58 Watt value should be used, and the 48 Vdc supply should be able to generate 87 W to 102 W.

NOTE

1) SMD17E2 units have an additional pin that supplies power to the control electronics and encoder only. This allows you to remove power from the motor without losing the network connection.

2) Note that the unit is not powered from two isolated sources. There is only a single common in the system, and the electronics and motor sections are not electrically isolated.

Power Supply Sizing (continued)

Table R1.4 below shows the suggested power supply sizes based on the maximum power the motor can generate over its entire speed range.

		SMD17E2-60			SMD17E2-80		
		Motor Power	150% Supply	175% Supply	Motor Power	150% Supply	175% Supply
Supply Voltage	24 Vdc	29 W	44 W	51 W	33 W	50 W	58 W
	48 Vdc	63 W	95 W	111 W	64 W	96 W	112 W

Table R1.4 Suggested Power Supply Ratings

Regeneration (Back EMF) Effects

All motors generate electrical energy when the mechanical speed of the rotor is greater than the speed of the rotating magnetic fields set by the drive. This is known as regeneration, or back EMF. Designers of systems with a large mass moment of inertia or high deceleration rates must take regeneration effects into account.

The stepper motors used in the SMD17E2 units are all low inductance motors. Back EMF is typically not an issue unless there is a gearhead attached to the motor and it is driven by hand. In these instances, the motor acts as a generator. With the speed of the motor multiplied by the ratio of the gearhead, this can lead to large enough voltage spikes to damage the attached power supply.

The first line of defense against regenerative events is an appropriately sized power supply. The additional capacitance typically found in a larger supplies can be used to absorb the regenerative energy. If your application has high deceleration rates, then a supply that can deliver 175% of peak motor power should be used.

The second line of defense is a regeneration resistor, also known as a braking resistor. Braking resistors, and their control circuitry, are built into AMCI AC powered drives. They are not included in the SMD products because of the limited ability to dissipate the heat generated by the resistor. An external braking resistor and control circuitry can be added to the system if needed.

Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17E2 provided that the manufacturer follows the connector and Ethernet standards. AMCI has reviewed the following connectors and ethernet cordsets for compatibility with the SMD17E2.

Connectors

AMCI #	Binder #	Description
MS-28	99-3729-810-04	Mating connector for Ethernet Connector. Male, 4 pin D-coded. Screw terminal connections. 6 to 8 mm dia. cable. Straight, IP67 rated when properly installed.
MS-31	99-0436-12-05	Mating connector for Power Connector. Female, 5 pin A-coded. Screw terminal connections. 6 to 8 mm dia. cable. Straight, IP67 rated when properly installed.

Table R1.5 Compatible Connectors

Compatible Connectors and Cordsets (continued)

Ethernet Cordset

AMCI Part #	Description
CNER-5M	4-position, 24 AWG, shielded. EIA/TIA 568B color coded. Connectors: Straight M12, D-coded, Male to RJ45. Shield attached to both connectors. Cable length: 5 m

Table R1.6 Ethernet Cordset

Power Cordsets

AMCI Part #	Description
CNPL-2M	5-position, 18 AWG. Connector: Straight M12, A-coded, Female to 2 inch flying leads, 0.28” stripped. Cable length: 2 m
CNPL-5M	5-position, 18 AWG. Connector: Straight M12, A-coded, Female to 2 inch flying leads, 0.28” stripped. Cable length: 5 m

Table R1.7 Power Cordsets

When a move command is sent to an SMD17E2, the unit calculates the entire profile before starting the move or issuing an error message. This chapter explains 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 an SMD17E2, 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 unit.
- 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 unit to the value used in the equations.

Motor Position

Motor Position is defined in counts, and its limits are based on the data format you choose when configuring the unit. The default multi-word format limits the Motor Position range from -32,768,000 to +32,767,999. If you choose the thirty-two bit double integer format, the range is -2,147,483,648 to +2,147,483,647. In continuous rotation applications, you should choose the double integer format.

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 SMD17E2 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 clockwise and counter-clockwise command. A counter-clockwise command, such as the CCW Jog Move command, will result in a decrease in motor position.

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 SMD17E2 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 exceeds ±8,388,607 counts. If the starting position or the ending position is outside of the ±8,388,607 count range, relative moves or jog commands must be used.

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 ±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 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 SMD17E2 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 43.

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.

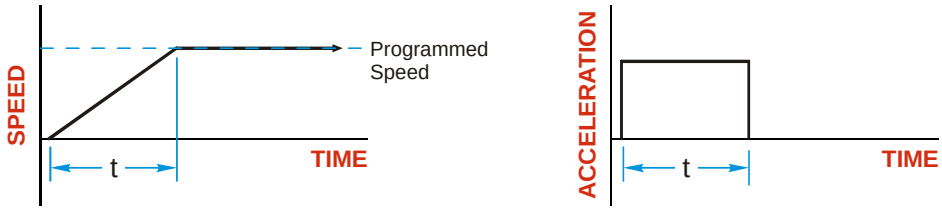


Figure R2.1 Linear Acceleration

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.

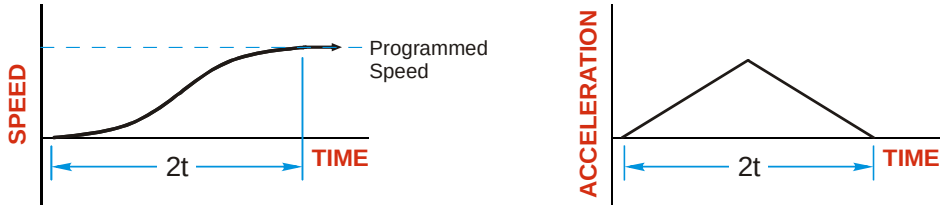


Figure R2.2 Triangular S-Curve Acceleration

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 R2.3 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.

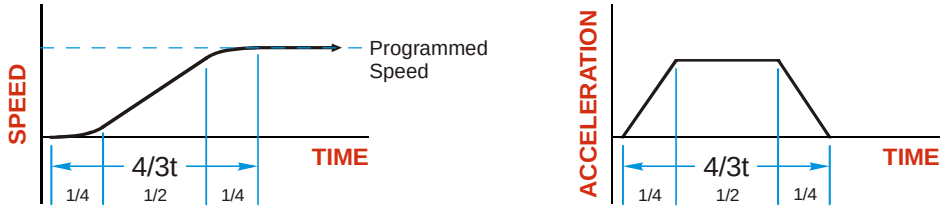


Figure R2.3 Trapezoidal S-Curve Acceleration

An acceleration jerk setting of 2 will result in the smallest amount of constant acceleration during a trapezoidal S-curve acceleration. An acceleration jerk setting of 5000 will result in the largest amount of constant acceleration during a trapezoidal S-curve acceleration. See ***S-Curve Acceleration Equations***, which starts on page 46, for a methods of calculating the Acceleration Jerk parameter.

A Simple Move

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

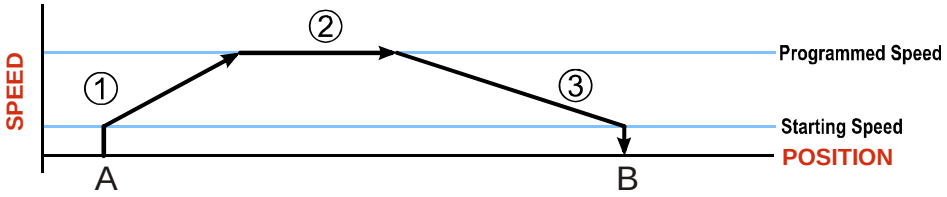


Figure R2.4 A Trapezoidal Profile

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 SMD17E2.
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 R2.4 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 R2.5 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.

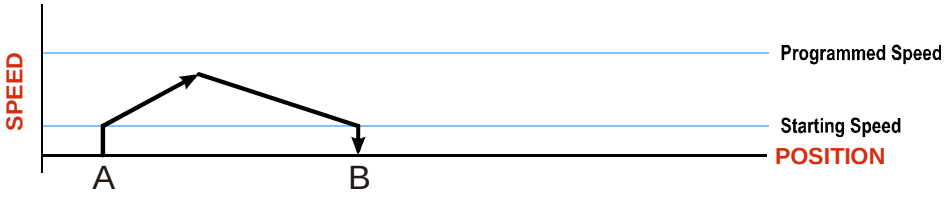


Figure R2.5 A Triangular Profile

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 30, 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. Note that power is not removed from the motor.

Hardware Control

Stop Jog or Registration Move Input: Triggering this input type during a Jog Move or Registration Move 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 this way, all other moves ignore this input.

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 SMD17E2***, which starts on page 53. The ***CW/CCW Jog Move*** commands are fully explained on page 32, and the ***CW/CCW Registration Move*** commands are fully explained on page 33.

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. The move cannot be restarted and the machine must be homed again before Absolute Moves can be run. 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.

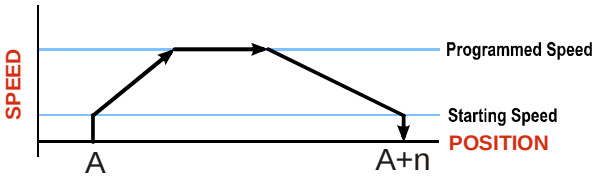


Figure R2.6 Relative Move

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.

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 SMD17E2 will set an *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 41.

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 CWW 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.

Basic Move Types (continued)

Absolute Move

Absolute Moves move from the Current Position (A) to a given position (B). (The SMD17E2 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 can 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.

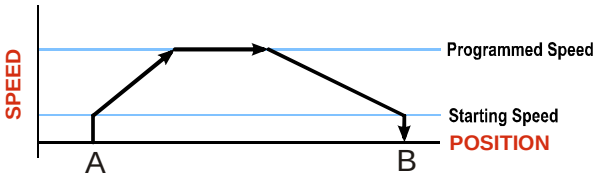


Figure R2.7 Absolute Move

NOTE

1) The *Home Position* of the machine must be set before running an Absolute Move. See the reference section, ***Homing an SMD17E2***, which starts on page 53, 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.

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 41.

Immediate Stop Conditions

- The Immediate Stop bit makes a 0→1 transition in the Network Input Data.
- An inactive-to-active transition on an input configured as an E-Stop Input.
- A CW or CWW 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.

Basic Move Types (continued)

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.

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 SMD17E2 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.

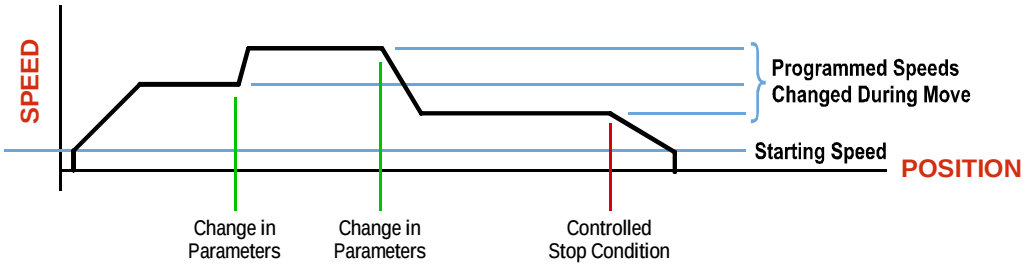


Figure R2.8 Jog Move

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 41.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Input Data.
- A inactive-to-active transition on an input configured as an E-Stop Input.
- A CW or CWW 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 SMD17E2 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.

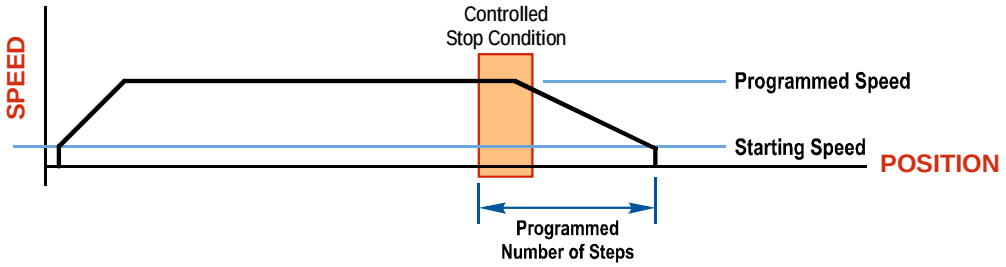


Figure R2.9 Registration Move

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 SMD17E2 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.

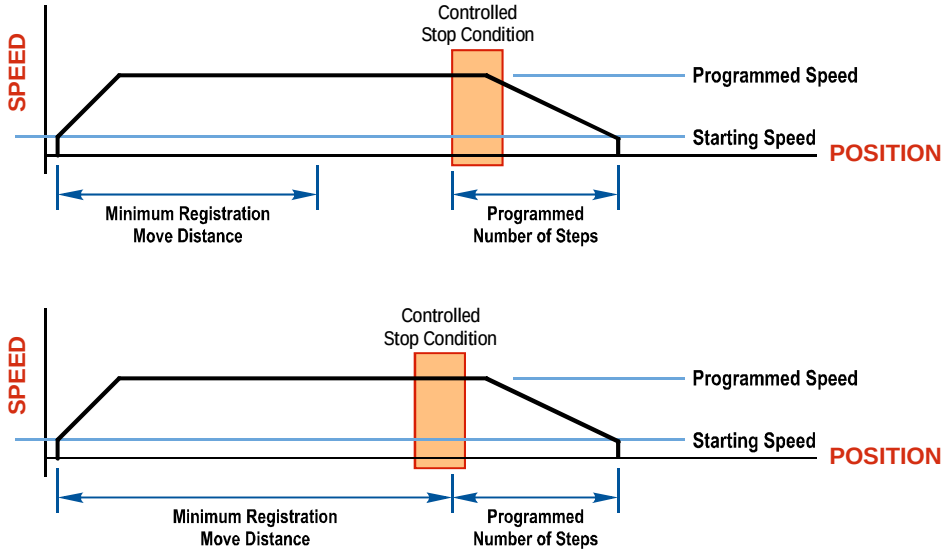


Figure R2.10 Min. Registration Move Distance

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 SMD17E2 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 with the Resume Move command.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Input Data.
- An inactive-to-active transition on an input configured as an E-Stop Input.
- A CW or CWW 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.

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 SMD17E2 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 38.

Two types of Assembled Moves exist in an SMD17E2:

- **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 command is issued. The direction of motion cannot be reversed with a Blend Move.
- **Dwell Move -** A Dwell Move gives you the ability to string multiple relative moves together, and the SMD17E2 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) All segments of the Blend Move must be written to the unit before the move can be initiated.
   - ➤ The SMD17E2 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 36.
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.

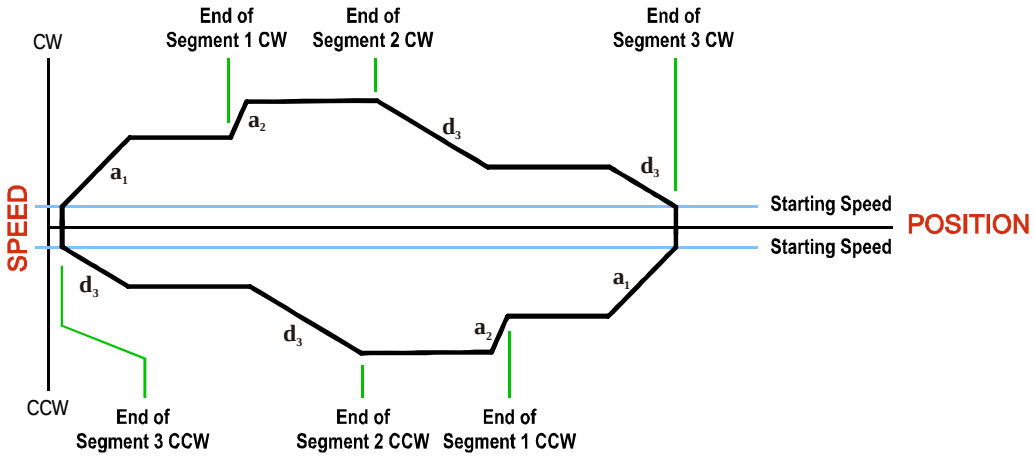


Figure R2.11 Blend Move

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 SMD17E2 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 38, 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 SMD17E2 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 SMD17E2 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 41.

Immediate Stop Conditions

- The Immediate_Stop bit makes a 0→1 transition in the Network Input Data.
- A positive transition on an input configured as an E-Stop Input.
- A CW or CWW 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 an SMD17E2 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 SMD17E2 sequentially. The two advantages of a Dwell Move in this case are that the unit will be more accurate with the Dwell Time then you can be in your control program, and Dwell Moves simplify your program's logic.

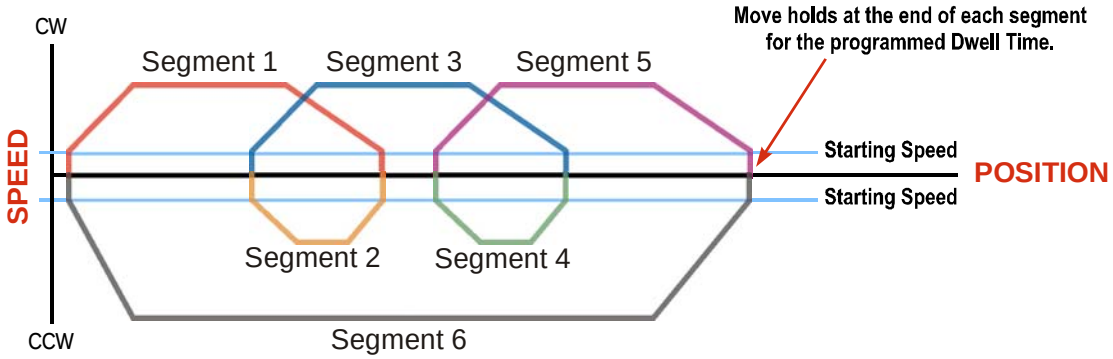


Figure R2.12 Dwell Move

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 SMD17E2 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 unit. As described in ***Saving an Assembled Move in Flash*** on page 38, 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 SMD17E2 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 SMD17E2 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 Input Data.
- A positive transition on an input configured as an E-Stop Input.
- A CW or CWW 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 SMD17E2 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 sign of each segment's Target Position is ignored and all segments are run in the same direction. In the case of a Dwell Move, the sign of each segment's Target Position determines the direction of the segment. For Dwell Moves, the Dwell Time is sent to the unit as part of the command.

Control Bits – Output Data

- **Program_Assembled bit –** A 0→1 transition on this bit tells the SMD17E2 that you want to program a Blend or Dwell Move Profile. The unit will respond by setting the *In_Assembled_Mode* bit in the Network Input Data. At the beginning of the programming cycle, the unit 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 SMD17E2 that the data for the next segment is available in the remaining data words.

Control Bits – Input Data

- **In_Assembled_Mode bit –** The SMD17E2 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 SMD17E2 is the signal to the host that the unit 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 SMD17E2 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 SMD17E2 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 SMD17E2 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 unit resets the *In_Assembled_Mode* and the *Waiting_For_Assembled_Segment* bits.

Saving an Assembled Move in Flash

The SMD17E2 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 SMD17E2 responds by resetting the *In_Assembled_Mode* and *Transmit Blend Move Segments* bits as usual and then it will flash the Status LED. If the LED is flashing green, the write to flash memory was successful. If it flashes red, then there was an error in writing the data. In either case, power must be cycled to the SMD17E2 before you can continue. With a limit of 10,000 write cycles, the design decision that requires you to cycle power to the unit 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 SMD17E2 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 SMD17E2 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.

Synchrostep (Virtual Axis) Moves

On controllers that support motion axis programming, such as the Rockwell Automation ControlLogix platforms, an SMD17E2 can be tied to the motion axis, with the host controller periodically sending position and velocity data to the unit as part of the axis update. The loop is closed by the SMD17E2 by controlling the velocity of the motor. Both linear and circular axes are supported.

Linear and Circular Synchrostep Moves have the following parameters and characteristics:

- Position and Velocity are programmed as 32 bit double integer values.
- The loop can be closed with respect to the internal motor position.
- The loop can be closed with respect to the encoder position on units that have the encoder option.
- A proportional constant is available to control the sharpness of the control.
  AMCI suggests a value of 1 or 2 when using the motor position to control the loop.
  AMCI suggests a value between 10 and 50 when using the encoder position to control the loop.
- A network delay value that can be used to offset some of the network delays incurred when communicating with the SMD17E2. Programmed in milliseconds, using this value is optional and defaults to zero. If used, a good starting point is the update time of the connection in milliseconds. The range of this parameter is 0 to 20.

Circular Synchrostep Moves have one additional parameter. This parameter is often called the "Position Unwind" value, and it defines the point at which the position rolls over and returns to zero. On the SMD17E2 units, this parameter has a range of 21 to 65,535.

NOTE When using the SMD17E2 as an axis follower, it is best to run the virtual axis as a high priority event driven periodic task.

NOTE When using the SMD17E2 as an axis follower, it is best to configure the unit to have a starting speed of 1. This will reduce jitter in the motor position at slow speeds or when the position is near its target.

NOTE When using the SMD17E2 as an axis follower, the programmed acceleration and deceleration values determine the response time of the internal closed loop algorithm. A suggested starting value for the Acceleration and Deceleration Parameters is Motor Resolution / 10.

NOTE When using the SMD17E2 as a linear axis follower, use caution in systems that always rotate in the same direction. If the position value overflows, it will switch from the maximum positive value to the maximum negative value and unpredictable motion may occur.

A sample program that demonstrates virtual axis programming on the ControlLogix platform is available on the AMCI website at ***https://www.amci.com/industrial-automation-support/sample-programs/***. It can be found in the *Stepper Motor Control* section of that page. The sample includes the setup required to run the axis as a high priority event driven periodic task. If you are using a different host controller that supports motion axis programming, feel free to contact AMCI technical support for assistance in programming your controller.

Blended Moves offer you similar functionality to linear motion axes when the move profile is well defined before the move begins. Consider using this functionality for simple linear profiles. A Blend Move is programmed into the SMD17E2 before it is run and the unit precisely controls position and velocity throughout the entire move. Additional information on ***Blend Move*** functionality can be found starting on page 35.

Controlling Moves In Progress

Each SMD17E2 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.

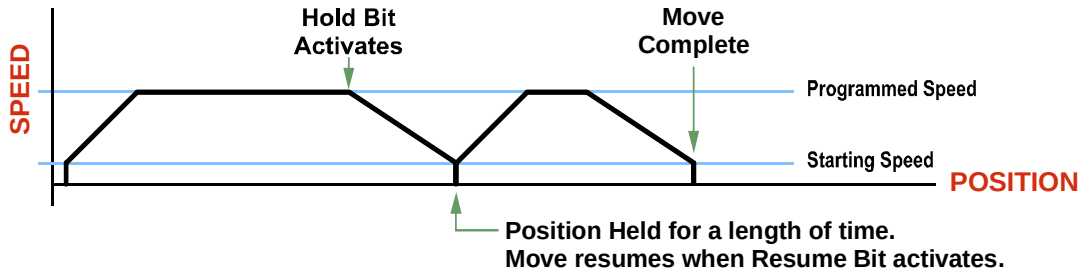


Figure R2.13 Hold/Resume a Move Profile

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 SMD17E2 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 and Relative Moves

Absolute and Relative 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 SMD17E2 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.

Notes

REFERENCE 3

CALCULATING MOVE PROFILES

This reference was added for customers that must program very precise profiles. Understanding this section is not necessary before programming the SMD17E2 and therefore 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 SMD17E2, 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 SMD17E2.
- 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 SMD17E2 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. ($141^2 \approx 20{,}000$)

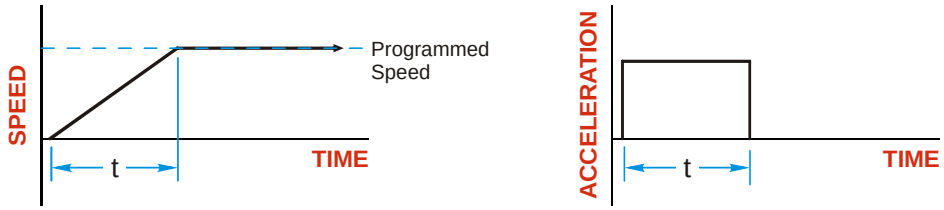
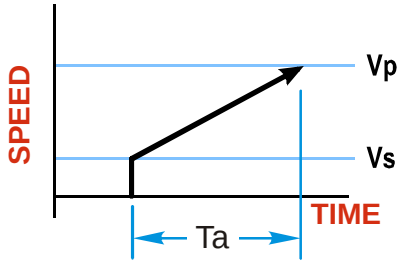


Figure R3.1 Constant Acceleration Curves

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 R3.1 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 R3.1 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 R3.2 for a move profile.

Name	Value	SMD23A2 or SMD24A2 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 R3.2 Sample Values

From figure R3.1:

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 SMD17E2 will generate a Trapezoidal profile. If the move is equal to 449,984 steps, the unit will generate a Triangular profile and the it will output one pulse at the programmed speed. If the move is less than 449,984 steps, the unit 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 R3.2, assume a total travel distance of 300,000 steps.

$$D_A + D_D = \frac{V_M^2 - V_S^2}{2a} + \frac{V_M^2 - V_S^2}{2d}$$

$$300{,}000 \text{ steps} = \frac{V_M^2 - 141^2}{2(20{,}000)} + \frac{V_M^2 - 141^2}{2(25{,}000)}$$

$$300{,}000 \text{ steps} = \frac{V_M^2 - 20{,}000}{40{,}000} + \frac{V_M^2 - 20{,}000}{50{,}000}$$

$$300{,}000 \text{ steps} = \frac{5}{5}\left(\frac{V_M^2 - 20{,}000}{40{,}000}\right) + \frac{4}{4}\left(\frac{V_M^2 - 20{,}000}{50{,}000}\right)$$

$$300{,}000 \text{ steps} = \frac{5V_M^2 - 100{,}000}{200{,}000} + \frac{4V_M^2 - 80{,}000}{200{,}000}$$

$$300{,}000\ (200{,}000) = 9V_M^2 - 180{,}000$$

$$\frac{60{,}000.18 \times 10^6}{9} = V_M^2$$

$$V_M = 81{,}650 \text{ steps/sec}$$

Once you have calculated the maximum speed, you can substitute this value into the time and distance formulas in table R3.1 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_P below), from the second equation.

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

$T_P = T_A + T_D + D_P/V_P$

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

$T_P = T_A + T_D$

S-Curve Acceleration Equations

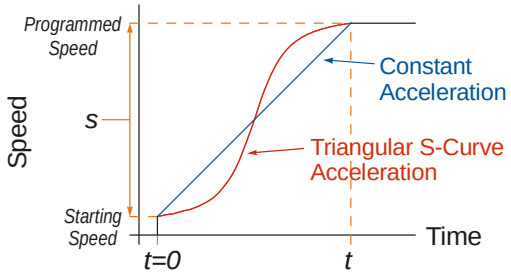
When the Acceleration Jerk parameter value is in the range of 1 to 5,000, the SMD17E2 uses this value to smoothly change the acceleration value applied during the move. In this case, the speed of the move does not increase linearly, but exponentially, resulting in an "S" shaped curve. This limits mechanical 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 SMD17E2 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 R3.2 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.



$$s = \text{Programmed Speed} - \text{Starting Speed}$$

$$\text{Acceleration} = \frac{\text{speed}}{\text{time}} \qquad a = \frac{s}{t} \qquad at = s$$

$$\text{jerk} = \frac{\text{acceleration}}{\text{time}} \qquad j = \frac{a}{t} \qquad jt = a$$

$$\text{SMD Acceleration Jerk Parameter(J)} = \frac{100j}{a} \qquad j = \frac{Ja}{100}$$

Figure R3.2 Move Profile Example

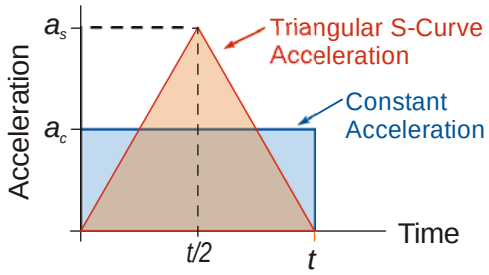


Figure R3.3 Triangular Acceleration

Speed is equal to acceleration multiplied by the time it is applied. This is shown graphically in figure R3.3 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.

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

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

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

$$Ja_s t = 200a_s$$

$$J = \frac{200}{t}$$ 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 $a_S = a_C$

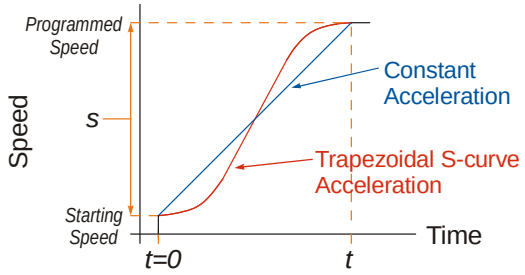
The above examples assume that you can increase the programmed acceleration value to keep the acceleration time the same. If your constant acceleration value is the maximum your system will allow, then using S-curve accelerations will lengthen the time needed to accelerate to your desired speed.

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 R3.4 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.



$$S = \text{Programmed Speed} - \text{Starting Speed}$$

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

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

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

Figure R3.4 Move Profile Example

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

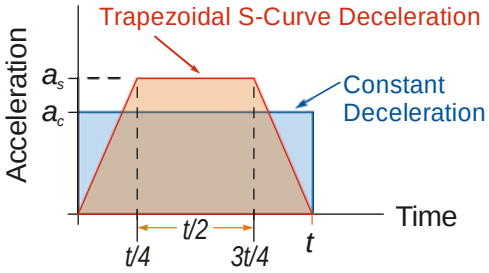


Figure R3.5 Trapezoidal Acceleration

Speed is equal to acceleration multiplied by the time it is applied. This is shown graphically in figure R3.5 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.

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

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

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

$$Ja_s t = 400a_s$$

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

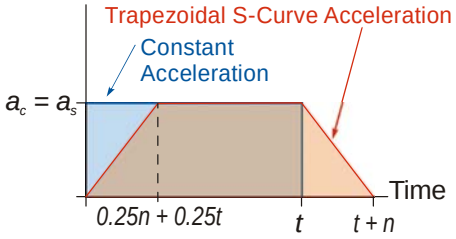
This value represents the ideal Acceleration Jerk parameter value for a trapezoidal S-curve acceleration with a constant acceleration for half of the phase. Setting the value lower than this will result in a shorter constant period, while setting the value greater than this will result in a longer constant 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 R3.6. 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 R3.6 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

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

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

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

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

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

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 R3.7, 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

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

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

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

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

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

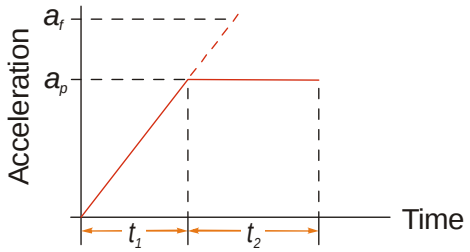


Figure R3.7 Calculating Trapezoidal S-Curve

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 4

HOMING AN SMD17E2

This chapter explains the various ways of homing an SMD17E2 unit. 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 an SMD17E2 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 SMD17E2 for speed control, don't require position data at all.

With the exception of Absolute Moves, the SMD17E2 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 over the network. On units with integral encoders, both the motor position and the encoder position can be preset separately, and the motor position can also be preset to the encoder position. The motor and encoder position values can be preset anywhere in the range of –8,388,607 to +8,388,607.

NOTE When presetting the motor position to the encoder position, the programmed Steps per Turn and Counts per Turn parameter values are used to scale the encoder position before the motor position is set to it. For example, assume that the Encoder Counts per Turn is programmed to 4,096, and the Motor Steps per Turn is programmed to 2,000. If the encoder position is 4,096 when the preset command is issued, the motor position will be set to 2,000.

CW/CCW Find Home Commands

The other choice is to use the driver's Find Home commands to order the SMD17E2 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 clockwise direction and ends when the home sensor triggers while the SMD17E2 is rotating in the clockwise direction *at the starting speed*. The CCW Find Home command operates in the same way but starts and ends with motion in the counter-clockwise direction.

Homing Inputs

Both of the physical DC inputs can be used when homing 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 is triggered multiple times in a machine cycle, and the system need 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.

Network Data Input

- **Backplane_Proximity_Bit:** An SMD17E2 can be configured to ignore changes on the physical homing input until the Backplane_Proximity_Bit makes a 0→1 transition. The unit 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 SMD17E2 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 SMD17E2 will never act on the physical Home Input.

Homing Configurations

A SMD17E2 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 unit this way, then the SMD17E2 has no way to automatically prevent over travel during a homing operation. In linear applications, you must prevent over travel 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 Backplane_Proximity_Bit) as a home proximity input. Using this bit is completely optional and prevents the Home Input from being acted upon until the Backplane_Proximity_Bit makes a 0→1 transition.

Homing Profiles

NOTE 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.

Home Input Only Profile

Figure R4.1 below shows the move profile generated by a CW Find Home command when you use the Home Input without the Backplane_Proximity_Bit.

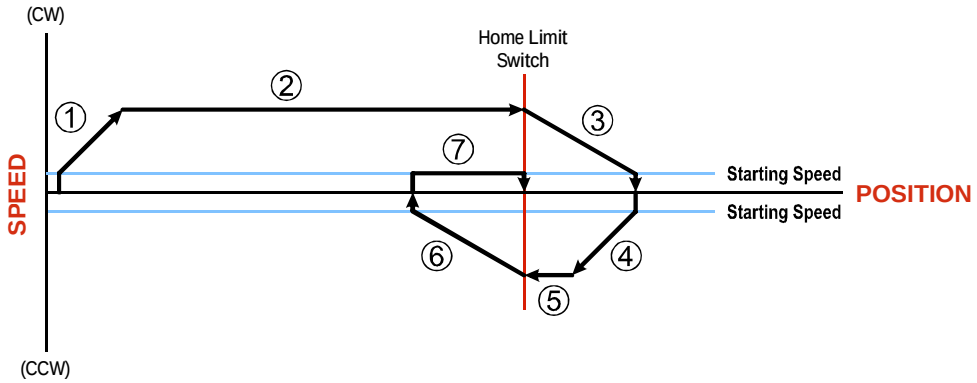


Figure R4.1 Home Input Profile

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 Backplane_Proximity_Bit

Figure R4.2 below shows the move profile generated by a CW Find Home command when you use the Home Input with Backplane_Proximity_Bit.

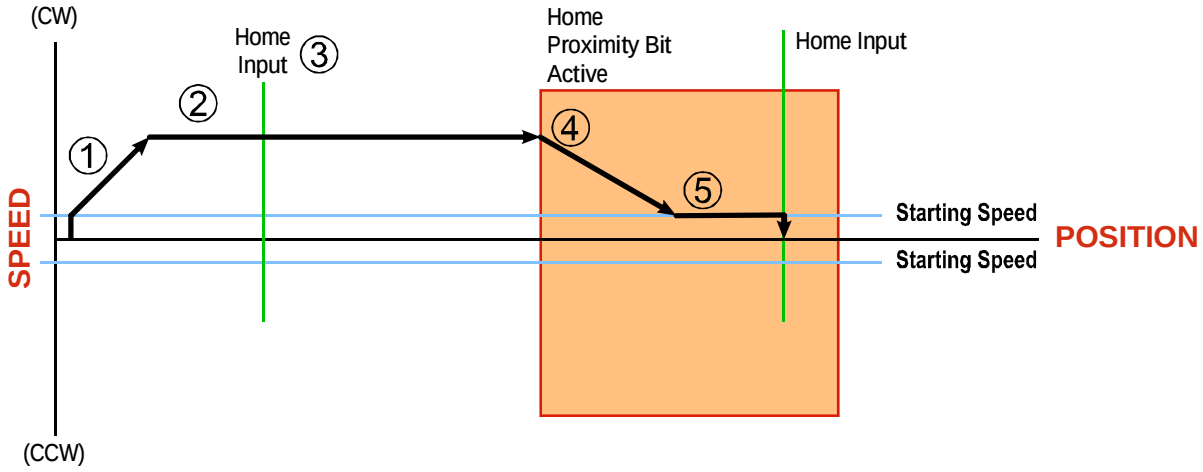


Figure R4.2 Homing with Proximity

1) Acceleration from the configured Starting Speed to the Programmed Speed
2) Run at the Programmed Speed
3) Ignores the Home Input because Backplane_Proximity_Bit has not made a 0→1 transition.
4) Deceleration towards the Starting Speed when the Backplane_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 R4.2 shows the Backplane_Proximity_Bit staying active until the SMD17E2 reaches its home position. This is valid, but does not have to occur. As stated in step 4, the unit starts to hunt for the home position as soon as the Backplane_Proximity_Bit makes a 0→1 transition.

Homing Profiles (continued)

Profile with Overtravel Limit

Figure R4.3 below shows the move profile generated by a CW Find Home command when you use:

- CW Overtravel Limit
- Home Input without the Backplane_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 SMD17E2 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 unit correctly, or not configured correctly when the unit was configured.

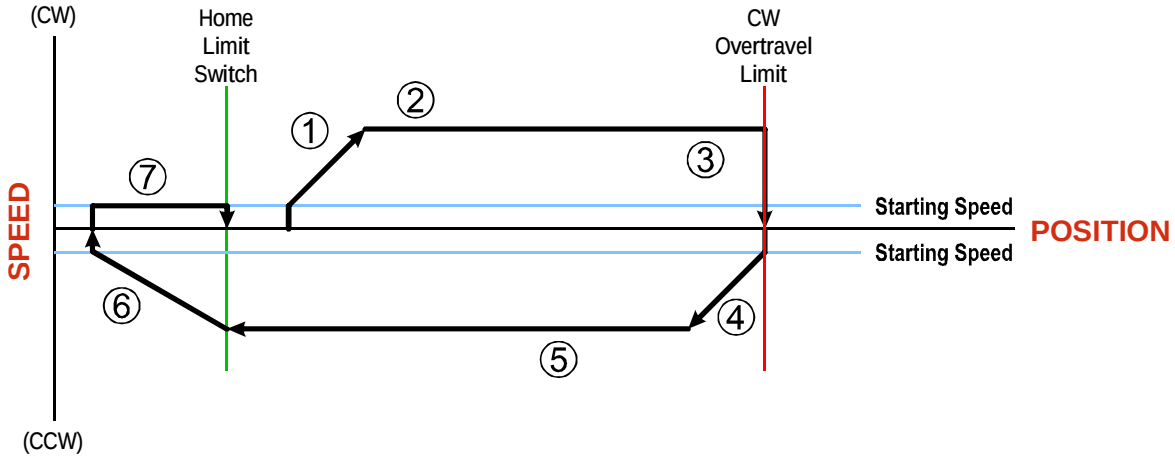


Figure R4.3 Profile with Overtravel Limit

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 issued, the profile will begin at step 4.

Controlling Find Home Commands In Progress

Controlled Stop Conditions

- 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 Stop Conditions

- The Immediate Stop bit makes a 0→1 transition in the Network Input 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 SMD17E2 correctly, or not configured correctly when the unit was configured.

REFERENCE 5

CONFIGURATION MODE DATA FORMAT

This chapter covers the formats of the Network Output Data used to configure an SMD17E2 as well as the formats of the Network Input Data that contains the responses from the device.

Modes of Operation

An SMD17E2 has two operating modes, Configuration Mode and Command Mode. You switch between these modes by changing the state of a single bit in the Network Output Data.

Configuration Mode

Configuration Mode gives you the ability to select the proper configuration for your application without having to set any switches. The ladder logic needed to configure a unit is included in the sample programs available from AMCI. This method simplifies change over if the unit ever needs to be replaced.

A valid configuration can be saved to the unit's Flash memory and the unit will use this as a default configuration on every power up. If you use this method, you can still write down a new configuration to the unit at any time. The new configuration is stored in RAM and is lost on power down unless you issue a command to store the new configuration in Flash.

Command Mode

This mode gives you the ability to program and execute stepper moves, and reset errors when they occur. The SMD17E2 units will always power up in this mode. The command data formats are described in the following chapter.

Power Up Behavior

An SMD17E2 will always power up in Command Mode. If available, the unit will use its stored configuration data to configure itself. The SMD17E2 will then check for valid network command data and will only enable the motor driver section if the Enable_Driver bit is set.

Configuration Mode Data Format

An SMD17E2 requires twenty bytes of Output Data as well as twenty bytes of Input Data. Many of the hosts that can be used with the SMD17E2 only support sixteen bit integers. Sixteen bit integers support a range of values from -32,768 to 32,767 or 0 to 65,535. The Starting Speed parameter, which is programmed as part of the configuration data, can exceed this range. This parameter is transmitted in two separate words. The table below shows how values are split.

Value	First Word	Second Word
12	0	12
12,345	12	345

Table R5.1 Multi-Word Format Examples

Command Mode Data Formats

When issuing commands to the SMD17E2, there are several parameters that are larger than sixteen bits. These parameters are:

- Target Position
- Programmed Speed
- Stopping Distance
- Minimum Registration Move Distance
- Position Preset Value
- Encoder Preset Value

Command Mode Data Formats (continued)

Likewise, when reading data back from a unit while it is in Command Mode, there are values that are larger than sixteen bits. These data values are:

- Motor Position
- Encoder Position
- Captured Encoder Position

By default, these thirty-two bit parameters and data values are written to and read from the SMD17E2 using the multi-word format described above. When configuring the unit, it is possible to program it to use a 32-bit double integer format instead of the custom format shown above.

There are two separate programming bits. The *Binary_Output_Format* Bit, controls the format of the programmable parameters written to the unit when issuing commands. The *Binary_Input_Format* Bit, controls the format of the data values written to the host controller by the SMD17E2.

When using the signed thirty-two bit format, there is an additional parameter named *Binary_Endian*. Siemens processors typically use big endian format, but you should refer to your PLC's documentation to verify the format used by your processor.

Examples of the formats are given below.

	Multi-Word Format		32 bit Signed Integer Little Endian Format		32 bit Signed Integer Big Endian Format	
Value	First Word	Second Word	First Word	Second Word	First Word	Second Word
12	0	12	16#000C	16#0000	16#0000	16#000C
-12	0	-12	16#FFF4	16#FFFF	16#FFFF	16#FFF4
1,234,567	1,234	567	16#D687	16#0012	16#0012	16#D687
-7,654,321	-7,654	-321	16#344F	16#FF8B	16#FF8B	16#344F

Table R5.2 Position Data Format Examples

NOTE The range of values when using the multi-word format is -32,768,000 to 32,767,999. When used in continuous rotation applications, such as control of a conveyor belt, it is possible to overflow these values. When any of the three position values overflow, the value of the associated data words will become indeterminate. AMCI strongly suggests using the signed 32-bit integer format for continuous rotation applications.

Output Data Format

The correct format for the Network Output Data when the SMD17E2 is in Configuration Mode is shown below.

Network Word	Configuration Data	Range
0	Configuration Word 0	See below
1	Configuration Word 1	See below
2	Starting Speed: Upper Word	Combined value between 1 and 1,999,999 steps/sec.
3	Starting Speed: Lower Word	
4	Motor Steps/Turn	200 to 32,767
5	Reserved	Set to zero
6	Encoder _Resolution	Set to 1,024, 2,048, or 4,096 for incremental encoder. Set to 2,048 for absolute encoder.
7	Idle Current Percentage	0 to 100%
8	Motor Current (X10)	1 to 20, Represents 0.1 to 2.0 Arms
9	Reserved	Set to zero

Table R5.3 Network Output Data Format: Configuration Mode

Configuration Word 0 Format

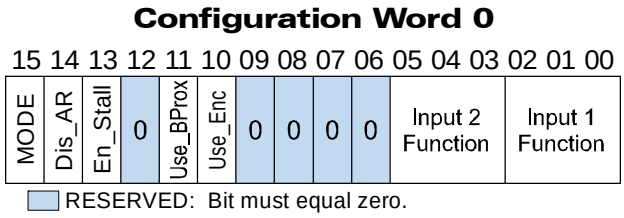


Figure R5.1 Configuration Mode: Control Word Format

Bit 15: Mode – "1" for Configuration Mode Programming, "0" for Command Mode Programming.

Bit 14: Disable_Antiresonance – "1" disables the unit's antiresonance feature. "0" enables the unit's anti-resonance feature. 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 12: Reserved – Must equal zero.

Bit 13: Enable_Stall_Detection – "0" disables motor stall detection. "1" enables motor stall detection. Only valid on SMD17E2 units with built in encoders. The *Use_Encoder* bit, which is bit 10 of this word, must be also be set to "1". You must also program the *Encoder_Resolution* parameter in configuration word 6.

Bit 11: Use_Backplane_Proximity – "0" when the Backplane_Proximity_Bit is not used when homing the SMD17E2. "1" when the Backplane_Proximity_Bit is used when homing the unit. Note that this bit is not the Backplane_Proximity_Bit, but enables or disables its operation. Do not use the Backplane_Proximity_Bit if you only want to home to a Home Limit Switch. (Leave this bit equal to "0".) If you enable this bit and then never turn on the Backplane_Proximity_Bit, the SMD17E2 unit will ignore all transitions of the home limit switch and you will not be able to home the device.

Output Data Format (continued)

Configuration Word 0 Format (continued)

Bit 10: Use_Encoder – "0" when the built-in encoder is not used or not available. "1" to enable the built-in absolute or quadrature encoder. You must also program the Encoder_Resolution parameter in configuration word 6.

Bits 9-6: Reserved – Must equal "0".

Bits 5-3: Input 2 Function – See the table below.

Bits 2-0: Input 1 Function – See the table below.

Bits				
5	4	3		
2	1	0	Function	Available On
0	0	0	General Purpose Input	The input is not used in any of the functions of the SMD17E2, but it's status is reported in the Network Data. This allows the input to be used as a discrete DC input to the host controller.
0	0	1	CW Limit	Input defines the mechanical end point for CW motion.
0	1	0	CCW Limit	Input defines the mechanical end point for CCW motion.
0	1	1	Start Indexed Move	Starts the move that is currently located in the output registers.
0	1	1	Start Indexed Move / Capture Encoder Value	When the encoder is enabled on an SMD17E2, the encoder position value is captured whenever this input transitions. An inactive-to-active state transition will also trigger an Indexed Move if one is pending in the unit.
1	0	0	Stop Jog or Registration Move	Brings a Jog or Registration Move to a controlled stop.
1	0	0	Stop Jog or Registration Move & Capture Encoder Value	When the encoder is enabled on an SMD17E2, the encoder position value is captured when the input triggers a controlled stop to a Jog or Registration move.
1	0	1	Emergency Stop	All motion is immediately stopped when this input makes an inactive-to-active transition.
1	1	0	Home	Used to define the home position of the machine.
1	1	1	Invalid Combination	This bit combination is reserved.

Table R5.4 Configuration Data: Input Function Selections

Output Data Format (continued)

Configuration Word 1 Format

Configuration Word 1

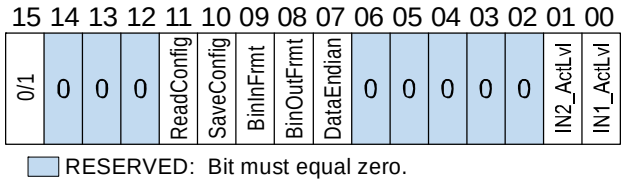


Figure R5.2 Configuration Mode: Config Word Format

Bit 15: Reserved – State ignored.

Bits 14 - 12: Reserved – Must equal zero.

Bit 11: Read_Present_Configuration – If this bit is set when you enter Configuration Mode, the unit responds by placing the present configuration data in the Network Input Data. You cannot write new configuration data to the unit while this bit is set. The format of the Configuration Data is given in the ***Input Data Format*** section of this chapter, starting on page 64.

Bit 10: Save_Present_Configuration – An SMD17E2 will store its configuration data to flash memory when this bit makes a 0→1 transition. The validity of the configuration data is checked before being written. If the data is not correct, the transition on this bit is ignored. If the write to flash completes successfully, the unit will write 16#AAAA into the last word of the Network Input Data and the Module Status LED will start flashing green. If the write is unsuccessful, the unit will write 16#EEEE into the last word of the Network Input Data and the Module Status LED will start flashing red. Once the unit issues its response to the Save_Present_Configuration command, it stops responding to commands and you must cycle power to the unit. This design decision prevents the SMD17E2 from responding to constant save commands from the host controller.

NOTE

1) This feature was added to support users whose host controllers have very limited functionality. Consider the consequences of using this feature. Adding the code necessary to write down the configuration to an SMD17E2 on power up or network connection is fairly straight forward on most PLC based hosts. Adding this code allows you to easily change the configuration in the host and easily configure a new drive if you ever need to swap one out on the machine.

2) The endurance of the flash memory is a minimum of 10,000 write cycles.

Bit 9: Binary_Input_Format – Set to “0” to have the Motor Position, Encoder Position, and Trapped Encoder Position reported in the multi-word format. Set to “1” to have the Motor Position, Encoder Position, and Trapped Encoder Position reported in signed 32-bit integer format. When this parameter is set to “1”, the Binary_Endian parameter in bit 7 sets the word order of the 32-bit value. See ***Position Data Format Examples*** found on page 60 for examples of the different formats.

Bit 8: Binary_Output_Format – Set to “0” to program the multi-word parameters in the multi-word format. Set to “1” to program the multi-word parameters in signed 32-bit integer format. When this parameter is set to “1”, the Binary_Endian parameter in bit 7 sets the word order of the 32-bit value. See ***Position Data Format Examples*** found on page 60 for examples of the different formats.

Bit 7: Binary_Endian – Only used when bits 8 and/or 9 above are set to “1”, set to “0” to the 32-bit values in little endian format. Set to “1” to program the 32-bit values in big endian format. Siemens processors typically use big endian format, but you should refer to your PLC’s documentation to verify the format used by your processor. See ***Position Data Format Examples*** found on page 60 for examples of the different formats.

Bits 6 - 2: Reserved – Must equal zero.

Output Data Format (continued)

Configuration Word 1 Format (continued)

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 the input, sets its Active_Level bit to "1". The input will always report as inactive in the network data.

Notes on Other Configuration Words

- Information on the ***Configuration Mode Data Format*** used when programming the Starting Speed can be found on page 59.
- Changes to the Idle Current only take effect at the *end of the first move after re-configuration.*

Input Data Format

The format for the Network Input Data when an SMD17E2 is in Configuration Mode is shown below.

PROFINET Word	Configuration Data
0	Mirror of Output Configuration Word 0
1	Mirror of Output Configuration Word 1
2	Mirror of Starting Speed: Upper Word
3	Mirror of Starting Speed: Lower Word
4	Mirror of Motor Steps/Turn
5	0000
6	Mirror of Encoder_Resolution
7	Mirror of Idle Current Percentage
8	Mirror of Motor Current (X10)
9	0000 or Status message when writing Configuration data to flash memory.

Table R5.5 Network Input Data Format: Configuration Mode

Configuration Word 0 Format (Word 0)

When the Configuration data is valid and accepted, this word mirrors the value of the Configuration Word 0 written to the SMD17E2. When the data written to it is invalid, the unit remains in Command Mode and sets the Configuration Error bit in the first word written back to the host controller. The format of this word is explained in the ***Status Word 0 Format*** section starting on page 81.

Starting Speed Format

The Starting Speed parameter is always programmed using the ***Configuration Mode Data Format*** as described on page 59.

Input Data Format (continued)

Stall Detect Enable

When in Configuration Mode, bit 13 of word 0 is set to "1" when stall detection is enabled. When in Command Mode, bit 13 of word 0 is set to "1" when there is a configuration error. When using the state of bit 13 of word 0 in your logic, always include the state of bit 15 of word 0 to assure that you are only acting on the bit based on the mode of the SMD17E2.

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. This includes setting the Lower Word of the Starting Speed to a value greater than 999.
3) You configure two or more 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 *Stall Detection Enable Bit* without configuring the SMD17E2 to use its built in encoder.
5) Setting the Input Configuration bits for any input to "111". See table R5.4 on page 62 for more information.

Notes

REFERENCE 6

COMMAND MODE DATA FORMAT

This chapter covers the formats of the Network Output Data used to command the SMD17E2 and the formats of the Network Input Data that contains the responses from the device. The SMD17E2 units require twenty bytes of Output Data as well as twenty bytes for Input Data.

Data Format

An SMD17E2 requires twenty bytes of Output Data as well as twenty bytes of Input Data. In most cases the data is represented as ten 16-bit (single) integers. Sixteen bit integers support a range of values from -32,768 to 32,767 or 0 to 65,535. When issuing commands to the SMD17E2, there are several parameters that are larger than sixteen bits. These parameters are:

- Target Position
- Programmed Speed
- Stopping Distance
- Minimum Registration Move Distance
- Position Preset Value
- Encoder Preset Value

Likewise, when reading data back from a unit while it is in Command Mode, there are values that are larger than sixteen bits. These data values are:

- Motor Position
- Encoder Position
- Captured Encoder Position

By default, these thirty-two bit parameters and data values are written to and read from the SMD17E2 using the multi-word format described below. When configuring the SMD17E2, it is possible to program it to use a 32-bit double integer format instead of the custom format shown above.

There are three configuration bits that control the data format when the SMD17E2 is in command mode. The *Binary_Output_Format* Bit, controls the format of the programmable parameters written to the SMD17E2 when issuing commands. The *Binary_Input_Format* bit controls the format of the data values written to the host controller by the SMD17E2. When either of these parameters are set to their 32-bit signed integer format settings, the *Binary_Endian* bit determines if the 32-bit values are stored and transmitted least significant bits first or most significant bits first. Examples of the formats are given below.

	Multi-Word Format		32 bit Signed Integer Little Endian Format		32 bit Signed Integer Big Endian Format	
Value	First Word	Second Word	First Word	Second Word	First Word	Second Word
12	0	12	16#000C	16#0000	16#0000	16#000C
-12	0	-12	16#FFF4	16#FFFF	16#FFFF	16#FFF4
1,234,567	1,234	567	16#D687	16#0012	16#0012	16#D687
-7,654,321	-7,654	-321	16#344F	16#FF8B	16#FF8B	16#344F

Table R6.1 Position Data Format Examples

NOTE The range of values when using the multi-word format is -32,768,000 to 32,767,999. When used in continuous rotation applications, such as control of a conveyor belt, it is possible to overflow these values. When any of the three position values overflow, the value of the associated data words will become indeterminate. AMCI strongly suggests using the signed 32-bit integer format for continuous rotation applications.

Command Bits Must Transition

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

This condition also applies when switching from Configuration Mode to Command Mode. If a bit is set in Configuration Word 0 while in Configuration Mode and you switch to Command Mode with the same bit set, the command will not occur because the bit must transition between writes to the unit.

The command bits are split between two, 16 bit words, Command Word 0 and Command Word 1. Only one bit in Command Word 0 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 SMD17E2.

PROFINET Word	Function
0	Command Word 0
1	Command Word 1
2	Command Parameters Word meaning depends on the command set to the SMD17E2
3	
4	
5	
6	
7	
8	
9	

Figure R6.1 Command Data Format

Command Word 0

Command Word 0

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MODE	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 R6.2 Command Word 0 Format

Bit 15: Mode_Select – "1" for Configuration Mode Programming "0" for Command Mode Programming.

Bit 14: Preset_Encoder – When this bit makes a 0→1 transition, the unit will preset the Encoder Position to the value stored in Output Words 2 and 3.

Bit 13: Run_Assembled_Move – When this bit makes a 0→1 transition, the unit will run the Assembled Move already stored in memory.

- **Assembled_Move_Type – Command Word 1, 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 in word 9 of the Command Data. The value is programmed in milliseconds and can range from 0 to 65,536.
- **Reverse_Blend_Direction – Command Word 1, 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 12 & 11: Read_Assembled_Data & Program_Assembled – 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 38.

Bit 10: Reset_Errors – When this bit makes a 0→1 transition, the unit will clear all existing command errors and reset the *Move_Complete* bit in the Network Input Data. This bit does not clear a configuration error or the *Position_Invalid* status bit.

Bit 9: Preset_Position – When this bit makes a 0→1 transition, the unit will preset the Motor Position. The value depends on the state of the *Preset_To_Encoder* bit (Command Word 1, bit 13). If the *Preset_To_Encoder* bit equals "0", the Motor Position is preset to the value stored in Output Words 2 and 3. If the *Preset_To_Encoder* bit equals "1", the Motor Position is set to:

$$\text{Motor Position} = \text{Encoder Position} \times \frac{\text{Motor Programmed Steps per Turn}}{\text{Encoder Programmed Pulses per Turn}}$$

In either case, the *Move_Complete* and *Position_Invalid* bits in the Network Input Data are reset to "0".

Bit 8: Jog_CCW – When this bit makes a 0→1 transition, the unit 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 32.

- **Registration_Move – Command Word 1, 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.

Command Word 0 (continued)

Bit 7: **Jog_CW –** When this bit makes a 0→1 transition, the unit will run a Jog Move in the clockwise direction. The full explanation of a ***CW/CCW Jog Move*** can be found starting on page 32.

- **Registration_Move – Command Word 1, 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.

Bit 6: **Find_Home_CCW –** When this bit makes a 0→1 transition, the unit 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 SMD17E2*** reference chapter starting on page 53.

Bit 5: **Find_Home_CW –** When this bit makes a 0→1 transition, the unit 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 SMD17E2*** reference chapter starting on page 53.

Bit 4: **Immediate_Stop –** When this bit makes a 0→1 transition, the unit 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 –** When this bit makes a 0→1 transition, the unit will 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 41. Note that a move in its hold state need not be resumed. The move is automatically cancelled if another move is started in its place.

Bit 2: **Hold_Move –** When this bit makes a 0→1 transition, the unit will place a move in its hold state. 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 41.

Bit 1: **Relative_Move –** When this bit makes a 0→1 transition, the unit will 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 30.

Bit 0: **Absolute_Move –** When this bit makes a 0→1 transition, the unit will 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 31.

Command Word 1

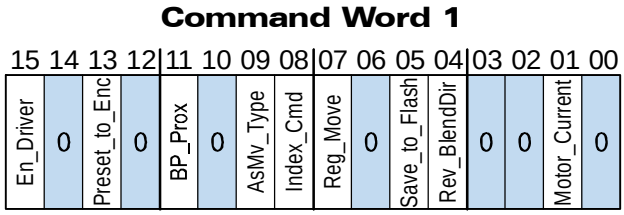


Figure R6.3 Command Word 1 Format

Bit 15: Enable_Driver – "0" to disable the motor current, "1" to enable motor current. A valid configuration must be written to the SMD17E2 before the driver can be enabled.

Bit 14: Reserved – Must equal "0".

Bit 13: Preset_to_Encoder – Only used when the Preset Motor Position bit (Command Word 0, Bit 9) is set to "1". If this bit equals "0" when the Preset Motor Position bit equals "1", the Motor Position is set to the value contained in words 2 and 3 of the command block. If this bit equals "1" when the Preset Motor Position bit equals "1", the Motor Position is set to:

$$\text{Motor Position} = \text{Encoder Position} \times \frac{\text{Motor Programmed Steps per Turn}}{\text{Encoder Programmed Pulses per Turn}}$$

In either case, the *Move_Complete* and *Position_Invalid* bits in the Network Input Data are reset to "0".

Bit 12: Reserved – Must equal "0".

Bit 11: Backplane_Proximity_Bit – When the SMD17E2 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 Backplane_Proximity_Bit*** section found on page 56.

Bit 10: Reserved – Must equal "0".

Bit 9: Assembled_Move_Type – When this bit equals "0", a Blend Move is started when the Run Assembled Move bit, (Command Word 1, 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, (Command Word 1, Bit 4). In a Dwell Move, the Dwell Time between segments is programmed in Word 9 of the command data.

Bit 8: Indexed_Command – If this bit is set when a move command is issued, the SMD17E2 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. The move command data, including this bit, must remain in the Network Output Registers while performing an Indexed Move.

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: Reserved – Must equal "0".

Command Word 1 (continued)

Bit 5: **Save_to_Flash -** This bit can be used to save a programmed Assembled Move to flash memory or to store the absolute encoder position offset to flash. (The absolute encoder position offset is generated by the Encoder Preset command.)

- When using this bit to save the programmed Assembled Move to flash memory, this bit must be set when the Program_Assembled bit (Command Word 0, bit 11) makes a 1 → 0 transition at the end of the programming cycle. The unit responds by flashing the Status LED when the writing is complete. If the LED is flashing green, the write to flash memory was successful. If it flashes red, then there was an error in writing the data. In either case, power must be cycled to the unit before you can continue. This design decision is to protect the flash memory from constant write commands. The flash memory has a minimum of 10,000 write cycles.
- When using this bit to save the calculated absolute encoder offset value to flash memory, this bit must be set when the Preset Encoder command is issued. (Bit 14 of ***Command Word 0*** is set to "1", see page 69.) If the offset is stored without error, the unit will respond by setting the Acknowledge bit. (Bit 13 of ***Status Word 1 Format***, see page page 83.)

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.

Bits 3-2: **Reserved –** Must equal "0".

Bit 1: **Motor Current –** If reset to "0", the motor current will be the value specified when the SMD17E2 was configured. Set to "1" to program the motor current to the value in word 8 of the command block. Motor current can set as a separate command or as part of a move command.

Bit 0: **Reserved –** Must equal "0".

Command Blocks

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

Absolute Move

PROFINET Word	Function	Units	Range
0	***Command Word 0***		16#0001
1	***Command Word 1***		See pg. 71
2	Abs. Target Position: Upper Word	Steps	Combined value between –8,388,608 and +8,388,607
3	Abs. Target Position: Lower Word		
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Motor Current	0.1 amps	1 to 20. Ignored if bit 1 of Command Word 1 is not set.
9	Acceleration Jerk		0 to 5000

Table R6.2 Absolute Move Command Block

Command Blocks (continued)

Relative Move

PROFINET Word	Function	Units	Range
0	***Command Word 0***		16#0002
1	***Command Word 1***		See pg. 71
2	Rel. Target Position: Upper Word	Steps	Combined value between –8,388,608 and +8,388,607
3	Rel. Target Position: Lower Word		
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Motor Current	0.1 amps	1 to 20. Ignored if bit 1 of Command Word 1 is not set.
9	Acceleration Jerk		0 to 5000

Table R6.3 Relative Move Command Block

Hold Move

PROFINET Word	Function	Units	Range
0	***Command Word 0***		16#0004
1	***Command Word 1***		See pg. 71
2	Unused		See Note Below
3	Unused		See Note Below
4	Unused		See Note Below
5	Unused		See Note Below
6	Unused		See Note Below
7	Unused		See Note Below
8	Unused		See Note Below
9	Unused		See Note Below

Table R6.4 Hold Move Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Command Blocks (continued)

Resume Move

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#0008
1	*Command Word 1*		See pg. 71
2	Unused		See Note Below
3	Unused		See Note Below
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Motor Current	0.1 amps	1 to 20. Ignored if bit 1 of Command Word 1 is not set.
9	Acceleration Jerk		0 to 5000

Table R6.5 Resume Move Command Block

Unused words are ignored by the SMD17E2 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.

Immediate Stop

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#0010
1	*Command Word 1*		See pg. 71
2	Unused		See Note Below
3	Unused		See Note Below
4	Unused		See Note Below
5	Unused		See Note Below
6	Unused		See Note Below
7	Unused		See Note Below
8	Unused		See Note Below
9	Unused		See Note Below

Table R6.6 Immediate Stop Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Command Blocks (continued)

Find Home CW

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#0020
1	*Command Word 1*		See pg. 71
2	Unused		See Note Below
3	Unused		See Note Below
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Motor Current	0.1 amps	1 to 20. Ignored if bit 1 of Command Word 1 is not set.
9	Acceleration Jerk		0 to 5000

Table R6.7 Find Home CW Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Find Home CCW

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#0040
1	*Command Word 1*		See pg. 71
2	Unused		See Note Below
3	Unused		See Note Below
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Motor Current	0.1 amps	1 to 20. Ignored if bit 1 of Command Word 1 is not set.
9	Acceleration Jerk		0 to 5000

Table R6.8 Find Home CCW Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Command Blocks (continued)

Jog CW

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#0080
1	*Command Word 1*		See pg. 71 Bits 7 & 6 must equal "00"
2	Unused		See Note Below
3	Unused		See Note Below
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Motor Current	0.1 amps	1 to 20. Ignored if bit 1 of Command Word 1 is not set.
9	Acceleration Jerk		0 to 5000

Table R6.9 Jog Move CW Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Registration Move CW

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#0080
1	*Command Word 1*		See pg. 71 Bits 7 & 6 must equal "10"
2	Stopping Distance: Upper Word	Steps	Combined value between 0 and +8,388,607
3	Stopping Distance: Lower Word		
4	Programmed Speed: Upper Word	Steps per Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Min. Reg. Move Distance: Upper Word	Steps	Combined value between 0 and +8,388,607
9	Min. Reg. Move Distance: Lower Word		

Table R6.10 Registration Move CW Command Block

Command Blocks (continued)

Jog CCW

PROFINET Word	Function	Units	Range
0	***Command Word 0***		16#0100
1	***Command Word 1***		See pg. 71 Bits 7 & 6 must equal "00"
2	Unused		See Note Below
3	Unused		See Note Below
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Motor Current	0.1 amps	1 to 20. Ignored if bit 1 of Command Word 1 is not set.
9	Acceleration Jerk		0 to 5000

Table R6.11 Jog CCW Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Registration Move CCW

PROFINET Word	Function	Units	Range
0	***Command Word 0***		16#0100
1	***Command Word 1***		See pg. 71 Bits 7 & 6 must equal "10"
2	Stopping Distance: Upper Word	Steps	Combined value between 0 and +8,388,607
3	Stopping Distance: Lower Word		
4	Programmed Speed: Upper Word	Steps per Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	Min. Reg. Move Distance: Upper Word	Steps	Combined value between 0 and +8,388,607
9	Min. Reg. Move Distance: Lower Word		

Table R6.12 Registration Move CCW Command Block

Command Blocks (continued)

Preset Position

PROFINET Word	Function	Units	Range
0	***Command Word 0***		16#0200
1	***Command Word 1***		See pg. 71
2	Position Preset Value: Upper Word	Steps	Combined value between –8,388,608 and +8,388,607
3	Position Preset Value: Lower Word		
4	Unused		See Note Below
5	Unused		See Note Below
6	Unused		See Note Below
7	Unused		See Note Below
8	Unused		See Note Below
9	Unused		See Note Below

Table R6.13 Preset Position Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Presetting the position resets the *Position_Invalid* and *Move_Complete* status bits in the Network Input Data.

Reset Errors

PROFINET Word	Function	Units	Range
0	***Command Word 0***		16#0400
1	***Command Word 1***		See pg. 71
2	Unused		See Note Below
3	Unused		See Note Below
4	Unused		See Note Below
5	Unused		See Note Below
6	Unused		See Note Below
7	Unused		See Note Below
8	Unused		See Note Below
9	Unused		See Note Below

Table R6.14 Reset Errors Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Resetting errors will also reset the *Move_Complete* status bit in the Network Input Data. Resetting errors will not reset the *Position_Invalid* or *Configuration_Error* bits.

Command Blocks (continued)

Run Assembled Move

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#2000
1	*Command Word 1*		See pg. 71 Blend Move: Bit 9 = "0" Dwell Move: Bit 9 = "1" Reverse_Blend_Direction is set by Bit 4.
2	Unused		See Note Below
3	Unused		See Note Below
4	Unused		See Note Below
5	Unused		See Note Below
6	Unused		See Note Below
7	Unused		See Note Below
8	Unused		See Note Below
9	Unused with Blend Move Dwell Time with Dwell Move	milliseconds	0 to 65,535

Table R6.15 Run Assembled Move Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Preset Encoder Position

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#4000
1	*Command Word 1*		See pg. 71
2	Encoder Preset Value: Upper Word	Steps	Combined value between –8,388,608 and +8,388,607
3	Encoder Preset Value: Lower Word		
4	Unused		See Note Below
5	Unused		See Note Below
6	Unused		See Note Below
7	Unused		See Note Below
8	Unused		See Note Below
9	Unused		See Note Below

Table R6.16 Preset Encoder Position Command Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Programming Blocks

The following blocks are used to program an Assembled Move. Both of the move types, 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

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#0800
1	*Command Word 1*		See pg. 71
2	Unused		See Note Below
3	Unused		See Note Below
4	Unused		See Note Below
5	Unused		See Note Below
6	Unused		See Note Below
7	Unused		See Note Below
8	Unused		See Note Below
9	Unused		See Note Below

Table R6.17 Assembled Move First Programming Block

Unused words are ignored by the SMD17E2 and can be any value, including parameter values from the previous command.

Once the first block is transmitted, the SMD17E2 responds by setting bits 8 and 9 in Status Word 0. (See *Status Word 0 Format* starting on page 81.) Once these are set, you can then start transmitting Segment Blocks.

Segment Block

PROFINET Word	Function	Units	Range
0	*Command Word 0*		16#1800
1	*Command Word 1*		See pg. 71
2	Rel. Target Position: Upper Word	Steps	Combined value between –8,388,608 and +8,388,607
3	Rel. Target Position: Lower Word		
4	Programmed Speed: Upper Word	Steps/Second	Combined value between the Configured Starting Speed and 2,999,999
5	Programmed Speed: Lower Word		
6	Acceleration	Steps/sec/ms	1 to 5000
7	Deceleration	Steps/sec/ms	1 to 5000
8	*Reserved*		Must equal zero for compatibility with future releases.
9	Acceleration Jerk		0 to 5000

Table R6.18 Assembled Move Segment Programming Block

Note that each Segment Block starts with bits 11 and 12 in Command Word 0 set to “1” (16#1800). When the unit sees bit 12 of Command Word 0 set, it will accept the block and reset bit 9 in Status Word 0. When your program sees this bit reset, it must respond by resetting bit 12 of Command Word 0. The SMD17E2 will respond to this by setting bit 9 in Status Word 0 and the next Segment Block can be written to the unit. 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 SMD17E2 is in Command Mode is shown below.

EtherNet/IP Word	Modbus/TCP Register	Command Mode Input Data
0	0	Status Word 0
1	1	Status Word 1
2	2	Motor Position: Upper Word
3	3	Motor Position: Lower Word
4	4	Encoder Position: Upper Word
5	5	Encoder Position: Lower Word
6	6	Trapped Encoder Position: Upper Word
7	7	Trapped Encoder Position: Lower Word
8	8	Programmed Motor Current (X10)
9	9	Value of Acceleration Jerk Parameter

Table R6.19 Network Input Data Format: Command Mode

Format of Position Data Values

The format of the Motor Position, Encoder Position, and Trapped Encoder Position values is controlled by the *Binary_Input_Format* bit in the configuration data written to the Networked Driver. (See ***Configuration Word 1 Format***, bit 9 starting on page 63.) When the *Binary_Input_Format* bit equals "0", the position values are reported with the bottom three digits of the value in the lower word (000 - 999) and the remaining digits in the upper word. See ***Data Format*** on page 67 for an explanation of this format. When the *Binary_Input_Format* bit equals "1", the position values are reported as 32-bit signed integers, with the location of the least significant bit determined by the *Binary_Endian* bit in the Configuration data.

NOTE The range of values when using the multi-word format is -32,768,000 to 32,767,999. When used in continuous rotation applications, such as control of a conveyor belt, it is possible to overflow these values. When any of the three position values overflow, the value of the associated data words will become indeterminate. AMCI strongly suggests using the signed 32-bit integer format for continuous rotation applications.

Status Word 0 Format

Status Word 0

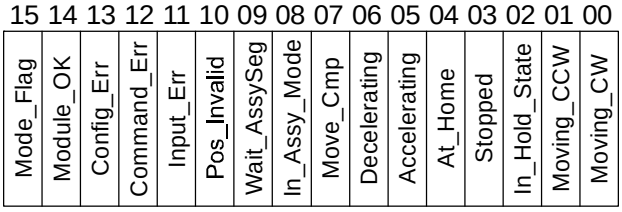


Figure R6.4 Command Mode: Status Word 0 Format

Bit 15: Mode_Flag – Set to "1" if in Configuration Mode, and set to "0" if in Command Mode.

Bit 14: Module_OK – "1" when the SMD17E2 is operating without a fault, "0" when an internal fault condition exists.

Input Data Format (continued)

Status Word 0 Format (continued)

Bit 13: **Configuration_Error –** "1" on power up before a valid configuration has been written to the SMD17E2, or after any invalid configuration has been written to the unit. "0" when the unit has a valid configuration in memory.

NOTE When in Command Mode, bit 13 of word 0 is set to "1" when there is a configuration error. When in Configuration Mode, bit 13 of word 0 is set to "1" when stall detection is enabled. When using the state of bit 13 of word 0 in your logic, always include the state of bit 15 of word 0 to assure that you are only acting on the bit when in the proper mode.

Bit 12: **Command_Error –** "1" when an invalid command has been written to the SMD17E2. This bit can only be reset by the *Reset_Errors* bit, Command Word 0, Bit 10.

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.

This bit is reset by a ***Reset Errors*** command. The format of the command is given on page 78.

Bit 10: **Position_Invalid –** "1" when:

- ➤A configuration is written to the SMD17E2
- ➤The motor position has not been preset or the machine has not been homed
- ➤The network connection has been lost and re-established
- ➤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. The state of this bit is ignored for all other move types.

Bit 9: **Waiting_For_Assembled_Segment –** The SMD17E2 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 38.

Bit 8: **In_Assembled_Mode –** The SMD17E2 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 38.

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 SMD17E2, 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.

Input Data Format (continued)

Status Word 0 Format (continued)

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.

Status Word 1 Format

Status Word 1

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Drive_Enabled	Stall_Detected	Cmd_Ack	Abs_Enc_Err	Heartbeat_Bit	Limit_Condition	Inv_Jog_Cng	0	Driver_Fault	Connect_Lost	0	Temp_90°C	0	0	IN2_Active	IN1_Active

Figure R6.5 Command Mode: Status Word 1 Format

Bit 15: **Drive_Is_Enabled –** Set to "1" when the motor driver section of the SMD17E2 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. Motor current is removed if there is a Driver_Fault (Bit 7 below) regardless of the state of this bit. Motor current is also removed if the motor is idle and Idle Current Reduction is programmed to its *To 0%* setting.

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

Bit 13: **Command_Acknowledge –** Normally "0". This bit is set to "1" when one of the following commands completes successfully:

- Preset Position
- Preset Encoder Position
- Reset Errors

This bit resets to "0" when the command bit is reset to "0" by the host controller.

Bit 12: **Absolute Encoder Error –** Only available on units with the absolute encoder, this bit is set to "1" under the following conditions:

- The shaft was subject to acceleration in excess of 160,000°/sec^2 (444.4 rev/sec^2) while power was removed from the unit
- The internal battery is fully discharged or damaged
- The unit itself is damaged

If this bit is set, cycle power to the unit. If the bit remains set, contact AMCI technical support for assistance.

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. Note that this bit is only available while the unit is in Command Mode.

Input Data Format (continued)

Status Word 1 Format (continued)

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.

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

Bit 7: **Driver_Fault –** If the driver section of the SMD17E2 is enabled, this bit will be a "1" during a Over-temperature Fault. Even though the driver is enabled, it will not supply current to the motor until the motor's temperature decreases to a safe value. At this point the fault will clear itself.

Bit 6: **Connection_Was_Lost –** If the *physical* network connection is lost at any time, this bit will be set when the connection is re-established. The *Input_Error* bit will also be set. Note that this bit is not set if the communication loss is on the protocol level, not the physical level.

Bit 5: **Reserved –** Will always equal zero.

Bit 4: **Temperature_Above_90°C –** This bit is set to "1" when the processor internal temperature exceeds 90°C. At this point, the heatsink temperature is typically near 83°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 over the device.

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

Bit 2: **Reserved –** Will always equal zero.

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 ***Configuration Word 1 Format*** section starting on page 63.

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 ***Configuration Word 1 Format*** section starting on page 63.

Notes on Clearing a Driver Fault

A Driver Fault occurs when there is an over temperature condition. When a Driver Fault occurs, the SMD17E2 will set bit 7 of the Status Word 1 word in the Network Input Data. Even though the driver is enabled, it will not supply current to the motor until the motor's temperature decreases to a safe value. At this point the fault will clear itself.

TASK 1
INSTALLING THE SMD17E2

1.1 Location

1.1.1 IP50 Rated Units (SMD17E2-M12)

SMD17E2 units that are IP50 rated are suitable for use in an industrial environment 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.

These criteria are equivalent to the *Pollution Degree 2* and *Over Voltage Category II* designations of the International Electrotechnical Commission (IEC).

1.1.2 IP64 Rated Units (SMD17E2-M12S)

SMD17E2 units that are IP64 rated are suitable for use in an industrial environment that meet the following criteria:

- Conductive pollution occurs, or dry, non-conductive pollution occurs which becomes conductive due to condensation is to be expected.
- Transient voltages are controlled and do not exceed the impulse voltage capability of the product's insulation.

These criteria are equivalent to the *Pollution Degree 3* and *Over Voltage Category II* designations of the International Electrotechnical Commission (IEC).

Safe Handling Guidelines

1.1.3 Prevent Electrostatic Damage

CAUTION

Electrostatic discharge can damage the SMD17E2 units. Follow these guidelines when handling the unit.

1) Touch a grounded object to discharge static potential before handling the unit.
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 network connectors or I/O connector.
5) Do not disassemble the unit
6) Store the unit in its shipping box when it is not in use.

1.1.4 Prevent Debris From Entering the Unit

WARNING

While mounting devices, be sure that all debris (metal chips, wire strands, tapping liquids, etc.) is prevented from falling into the unit, specifically into the M12 connectors. Debris may cause damage to the unit or unintended machine operation with possible personal injury.

1.1.5 Remove Power Before Servicing

WARNING

Remove power before removing or installing any SMD17E2 units.

1.2 Operating Temperature Guidelines

Due to the onboard electronics, the maximum operating temperature of the SMD17E2 is limited to 203°F/95°C. The motor by itself has a maximum operating temperature of 266°F/130°C. Depending on the operating current setting, move profiles, idle time, and the idle current reduction setting, it is possible to exceed these temperatures in a thermally isolated environment. As explained in the mounting section, mounting the SMD17E2 to a large metal heatsink is the best way to limit the operating temperature of the device. Operating temperature should be monitored during system startup to verify that the maximum motor temperature remains below its 203°F/95°C specification. SMD17E2 devices have an onboard thermistor and will remove motor current if the operating temperature exceeds this limit. Ovetemperature faults are also reported in the Network Input Data.

1.3 Mounting

All AMCI motors 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 unit'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 unit.

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 #14 gauge stranded wire or 1/4" wire braid as the grounding wire
2) Do not disassemble *any* stepper motor. A significant reduction in motor performance will result.

1.3.1 SMD17E2-M12 Mounting

The SMD17E2-M12 units are not water tight. Their IP50 rating makes them acceptable for use in dusty environments with occasional condensation. These units should be mounted in such a way that condensation will naturally drain off of the unit instead of pooling at the motor shaft, where the motor wires exit the motor, or on the motor laminations.

1.3.2 SMD17E2-M12S Mounting

The SMD17E2-M12S units are not water tight. Their IP64 rating makes them acceptable for use in dusty environments, environments with condensation, and environments where the unit may be exposed to splashing water. SMD17E2-M12S units should be mounted in such a way that condensation and liquids will naturally drain off of the unit instead of pooling on the motor laminations.

1.3 Mounting (continued)

1.3.3 SMD17E2-60 Outline Drawing

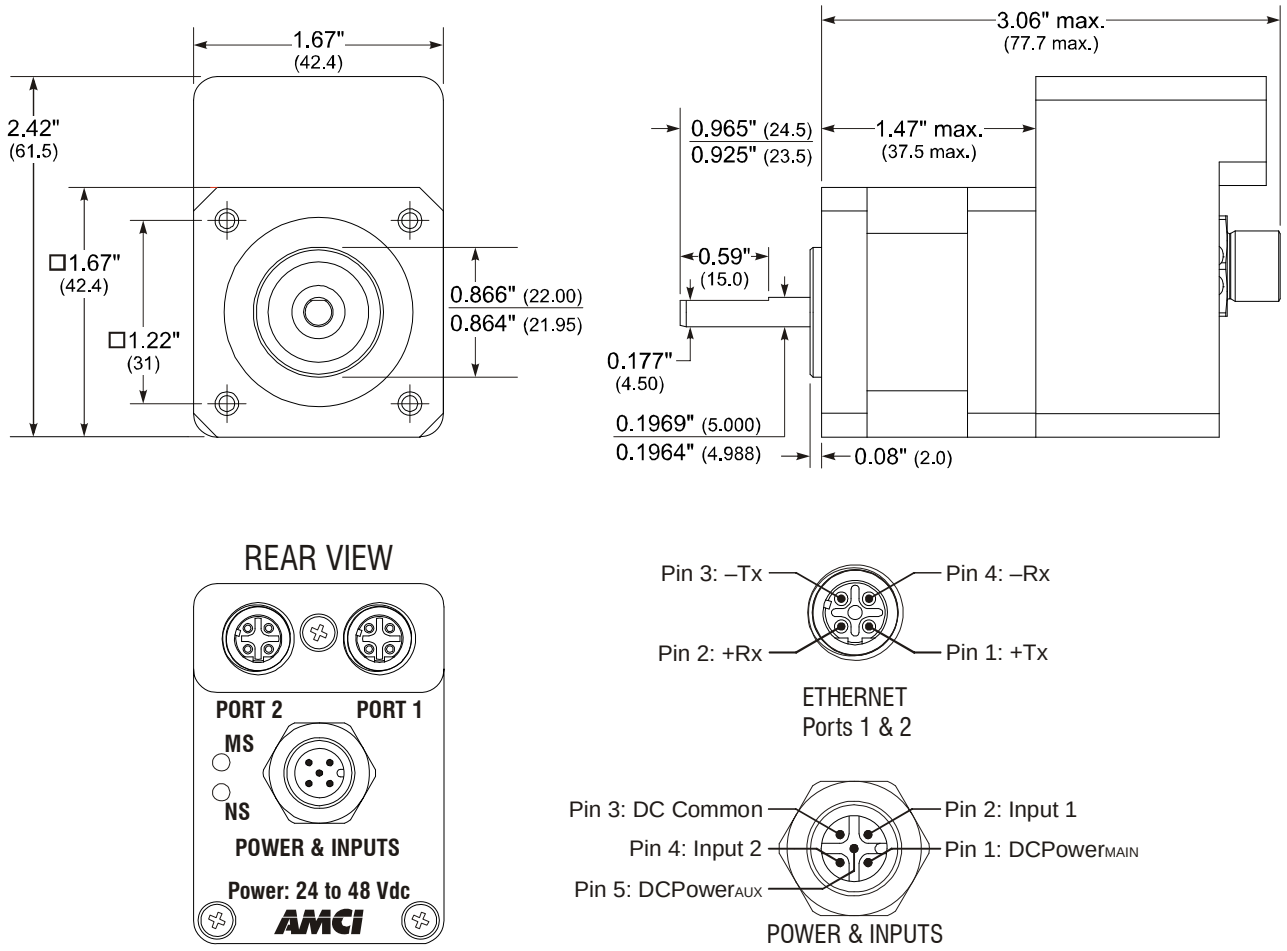


Figure T1.1 SMD17E2-60 Outline Drawing

1.3 Mounting (continued)

1.3.4 SMD17E2-80 Outline Drawing

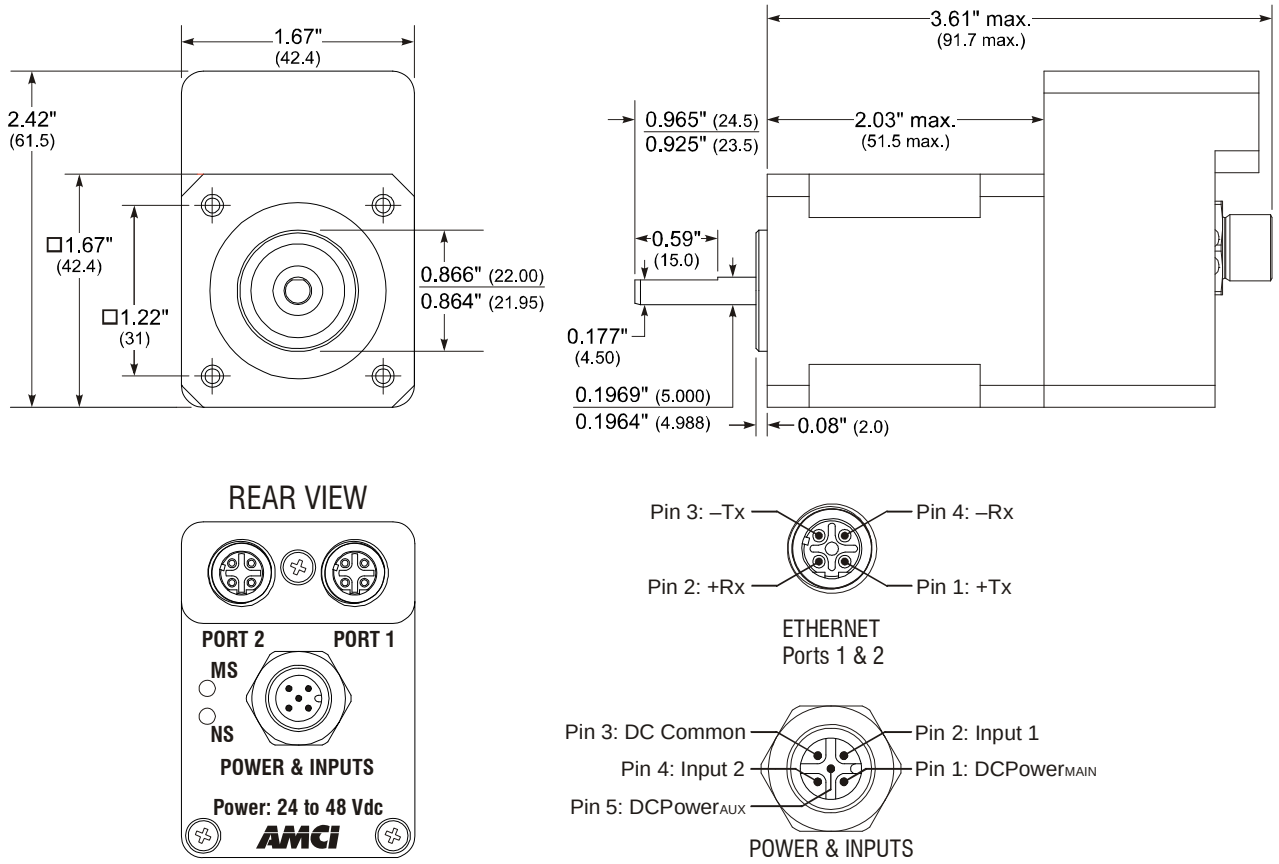


Figure T1.2 SMD17E2-80 Outline Drawing

1.3.5 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.

- Maximum radial load is 6.5 lbs. (29 N) at the center of the flat on the shaft.
- Maximum axial load is 5.6 lbs. (25 N)

NOTE Internal encoders are mounted on the end of the motor shaft that is internal to the unit. Excessive axial load may cause encoder mis-alignment and damage to the unit. This type of damage is not covered under warranty.

1.4 Power and Input Connector

The Power and Input Connector is located on the back of the SMD17E2 unit below the Ethernet connectors. Figure T1.3 shows the pinout for the Input Connector when viewed from the back of the unit.

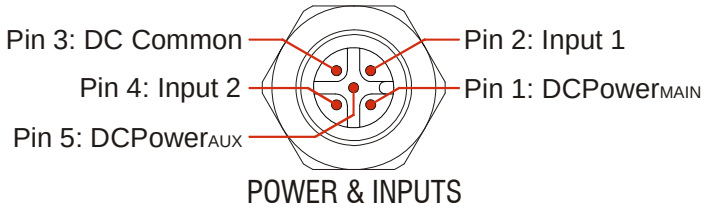


Figure T1.3 Power and Input Connector

Digital inputs on the SMD17E2 units are single ended and referenced to the DC Common pin. There are two power pins.

- $DCPower_{MAIN}$ powers both the control electronics and the motor.
- $DCPower_{AUX}$ powers only the control electronics, which includes the encoder if it is available.

Using the $DCPower_{AUX}$ pin is optional. If your application requires you to cut power to your motor under some conditions, using the $DCPower_{AUX}$ pin allows you to cut power to your motor without losing your network connection. The connector is a standard five pin A-coded M12 connector that is rated to IP67 when the mate is properly attached.

1.4.1 Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17E2 units provided that the manufacturer follows the A-coded M12 standards. The following connector and cordset is available from AMCI.

Connector

AMCI #	Binder #	Description
MS-31	99-0436-12-05	Mating connector for Power Connector. Female, 5 pin A-coded. Screw terminal connections. 6 to 8 mm dia. cable. Straight, IP67 rated when properly installed.

Table T1.1 Compatible Connectors

Power Cordset

AMCI Part #	Description
CNPL-2M	5-position, 18 AWG. Connector: Straight M12, A-coded, Female to 2 inch flying leads, 0.28” stripped. Cable length: 2 meter
CNPL-5M	5-position, 18 AWG. Connector: Straight M12, A-coded, Female to 2 inch flying leads, 0.28” stripped. Cable length: 5 meter

Table T1.2 Power Cordset

Note that the CNPL-2M and CNPL-5M cordsets are manufactured with 18AWG wires. This is the suggested minimum gauge wire when using the SMD17E2. If you use a different cordset, verify the gauge of the wire before making your purchase.

1.5 Power Wiring

The SMD17E2 accepts 24 to 48Vdc as its input power. AMCI strongly suggests using 18 AWG wire for the power connections to minimize power loss in the cable. The CNPL-2M and the CNPL-5M cables are made with 18 gauge wire, and the MS-31 connector will accept up to 18 gauge wire.

CAUTION Do not apply 120 Vac to any pins of an SMD17E2. If this occurs, the unit will be damaged and you will void the unit's warranty.

WARNING The SMD17E2 does not have a circuit to limit inrush current when power is applied to the unit. If the power supply voltage is applied through the switching of contacts, damage to the contacts or contact welding may occur.

- Use a power supply that limits the peak output current to a value below the contact switching limit.
- Switch the power input to the power supply instead of the power supply's output.

Figure T1.4 below shows how to wire power to the SMD17E2 units. Note that Pin 5, $DCPower_{AUX}$, is only used when you introduce a circuit for removing power from the motor. Colors in parentheses are the appropriate wire color of the CNPL-5M cable.

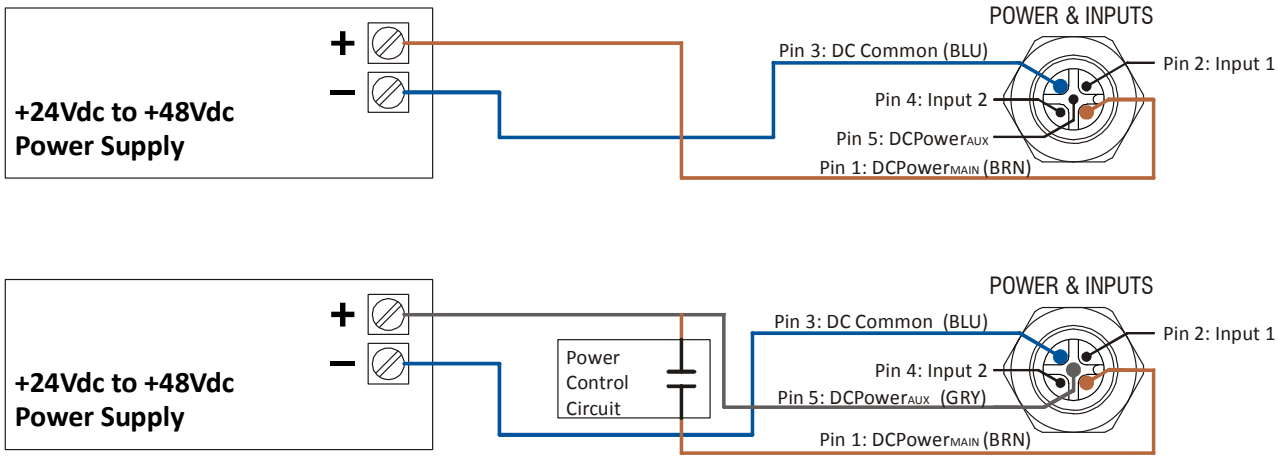


Figure T1.4 M12 Power Wiring

1.6 Input Wiring

Inputs 1 and 2 are single ended inputs that share the DC Common return pin. They accept 3.5 to 27 Vdc without the need for an external current limiting resistor. Figure T1.5 below shows how to wire discrete DC sourcing and sinking sensors to inputs 1 and 2 of the SMD17E2. Colors in parentheses are the appropriate wire color of the CNPL-5M cable.

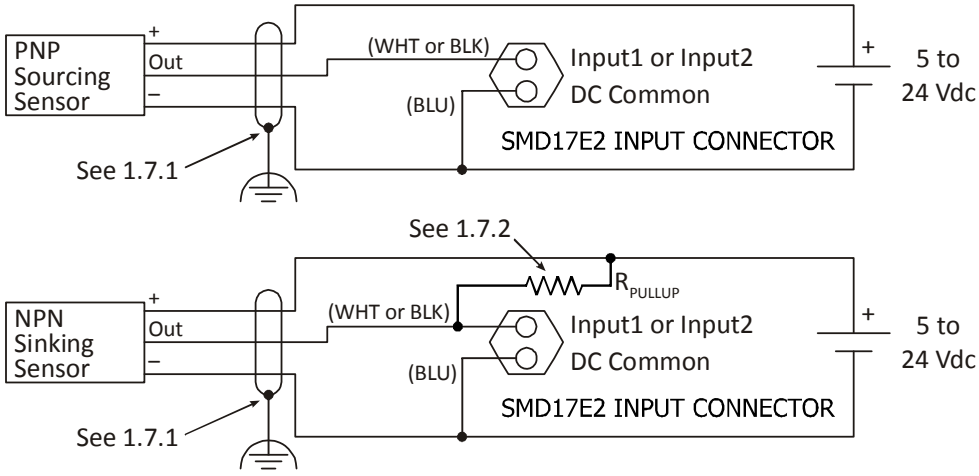


Figure T1.5 Input Wiring

1.6.1 Cable Shields

Because they are low power signals, cabling from the sensor to the SMD17E2 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 SMD17E2.

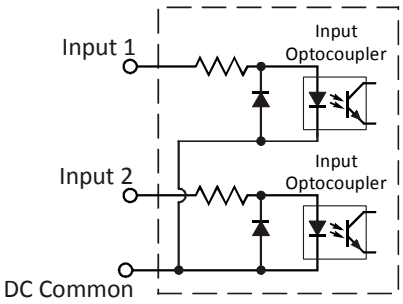


Figure T1.6 Input Schematic

1.6.2 Sinking Sensors Require a Pull Up Resistor

Sinking output sensors require an external pull up resistor because inputs 1 and 2 of the SMD17E2 units also sink current. Table T1.1 below shows the values of pull up resistors that will allow the unit's input to activate along with the current that the sensor must be able to sink when it is active.

Input Voltage	Pull Up Resistor	Sensor Current When Active
5 Vdc	300 ohm	16.7 mA
12 Vdc	1.4 kilohm	8.6 mA
24 Vdc	3.8 kilohm	6.3 mA

Table T1.1 TIA/EIA Color Codes

The logical states of the sensor and SMD17E2 input will be reversed. The SMD17E2 input is off when the sensor is active. You can set the logic state of the input when you configure the unit.

1.7 Network Connectors

Figure T1.7 shows the Ethernet connector pinout when viewed from the back of the SMD17E2. The Ethernet ports on the units are an "auto-sense" ports that will automatically switch between 10baseT and 100baseT depending on the network equipment they are attached to. The ports also have "auto switch" capability. This means that a standard cable can be used when connecting the SMD17E2 to any device, including a personal computer.

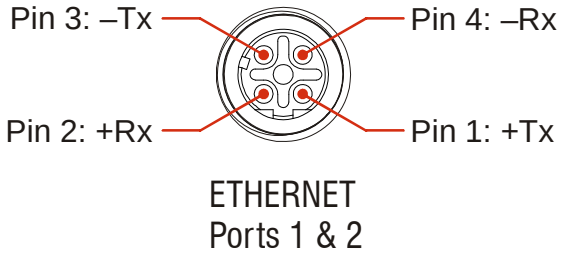


Figure T1.7 M12 Ethernet Connector Pinout

The connector is a standard female four pin D-coded M12 connector that is rated to IP67 when the mate is properly attached.

The SMD17E2 units have two ethernet ports. In non-redundant applications, either port can be used to attach the unit to the network.

1.7.1 Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17E2 provided that the manufacturer follows the connector and Ethernet standards. AMCI offers the following mating connector and cordsets that mate with the Ethernet port connectors.

AMCI #	Description
MS-28	Mating connector for Ethernet port connector. Screw terminal connections. 6 to 8 mm dia. cable. Straight, IP67 rated when properly installed.
CNER-5M	Molded cordset for Ethernet connector. 5 meters in length. Straight M12 4 pin D-coded to RJ-45 connector. Wired to TIA/EIA-568B. IP67 rated when properly installed.

Table T1.2 Compatible Ethernet Connectors and Cordsets

1.7.2 TIA/EIA-568 Color Codes

There are two color codes in common use when wiring Ethernet connections with twisted pairs. Either one of these standards is acceptable. The CNER-5M cable available from AMCI follows the 568B standard. Note that accidently reversing the Tx/Rx pairs will not affect the operation of the SMD17E2. Each unit has an "auto-sense" port that will automatically adjust for swapped pairs.

Signal	568A Color	568B Color
+Transmit (+Tx)	White/Green Tracer	White/Orange tracer
–Transmit (–Tx)	Solid Green	Solid Orange
+Receive (+Rx)	White/Orange Tracer	White/Green Tracer
–Receive (–Rx)	Solid Orange	Solid Green

Table T1.3 TIA/EIA Color Codes

1.8 EtherNet/IP Connections

1.8.1 Non-DLR Applications

The SMD17E2 units have two Ethernet ports with a built-in Ethernet switch connecting the two. In non-DLR applications, either port can be used to attach the unit to the network. The remaining port can be used to extend the network to another device if this would reduce wiring costs.

1.8.2 DLR Applications

In Device Level Ring applications, the SMD17E2 units function as Beacon-Based Ring Nodes. In these applications, both ports are used when wiring the ring, daisy chaining from one unit in the ring to the next.

1.9 Modbus TCP Connections

The SMD17E2 units have two Ethernet ports with a built-in Ethernet switch connecting the two. Either port can be used to attach the unit to the network. The remaining port can be used to extend the network to another device if this would reduce wiring costs.

1.10 PROFINET Connections

1.10.1 Non-MRP Applications

The SMD17E2 units have two Ethernet ports with a built-in Ethernet switch connecting the two. In non-MRP applications, either port can be used to attach the unit to the network. The remaining port can be used to extend the network to another device if this would reduce wiring costs.

1.10.2 MRP Applications

In Media Redundancy Protocol applications, the SMD17E2 units function as a Media Redundancy Client (MRC). In these applications, both ports are used when wiring the ring, daisy chaining from one unit in the ring to the next.

Notes

TASK 2

SET THE IP ADDRESS AND PROTOCOL

This section is intended for the engineer or technician responsible for setting the IP address of an AMCI SMD17E2.

2.1 Determine the Best Method for Setting the IP Address

There are three methods for setting the IP address on an SMD17E2. Table T2.1 below outlines the available methods and when you can use them.

Method	Restrictions	Section
Use Factory Default Settings	1) The machine must use 192.168.0.xxx subnet. 2) The 192.168.0.50 address must be available.	2.2a
Use the Embedded Web Server	No restrictions on use. This is the preferred method. The internal webserver can be used to set the SMD17E2 to any IPv4 address. The IP address and protocol will be stored in nonvolatile memory and used on subsequent power-ups.	2.2b
Use the AMCI NET Configurator Utility	No restrictions on use. The software can be used to set the SMD17E2 to any IPv4 address. The IP address and protocol choice will be stored in non-volatile memory and used on subsequent power-ups. The software utility can also be used to issue commands to the SMD17E2.	2.2c

Table T2.1 Methods for Setting the IP Address

NOTE There is a MAC address label on each SMD17E2 which has a writable surface. There is room on the label for writing the programmed IP address of the unit. It is a best practice to use this label to document the IP address of the unit in case it is ever repurposed.

NOTE In order to conform to the ODVA specification for EtherNet/IP, the SMD17E2 also supports the DHCP protocol. You will need an EtherNet/IP DHCP server, such as the one available from Rockwell Automation, in order to use this protocol. The AMCI Net Configurator utility offers the same functionality and should be used unless your company policy prevents you from installing third party utilities.

2.2a Use Factory Default Settings

The factory default address for the SMD17E2 is 192.168.0.50 with a subnet mask of 255.255.255.0. The easiest way to verify this address is with the ping command as described in steps A.3 and A.4 of the Optional Task *Configure Your Network Interfaces* which starts on page 129.

If the driver does not respond to this address then it may take some effort to determine the correct address. There is a label on the driver that lists the MAC address of the device. There is space on the label for noting the IP address of the device if it is changed. If the address was not documented, a program called Wireshark (*https://www.wireshark.org/*) can be used to determine the address of the driver.

Task Complete

2.2b Use the Embedded Web Server

PREREQUISITE: You must know the present IP address of the SMD17E2. The factory default address is 192.168.0.50.

PREREQUISITE: Task 1.5: ***Power Wiring*** found on page 90. You must be able to power the SMD17E2.

PREREQUISITE: Tasks: 1.7 and 1.8, 1.9, or 1.10, starting on page 92. You must attach your SMD17E2 to your computer.

PREREQUISITE: Optional Task B: ***Configure Your Network Interfaces***. (page 129) The network interfaces on your computer must be on the same subnet before you can communicate with an SMD17E2.

2.2b.1 Disconect the SMD17E2 from the host controller and cycle power to the SMD17E2.

This ensures that the unit does not have any open connections to the host controller.

2.2b.2 Start your web browser and connect to the SMD17E2

The internal HTML pages should work with any browser. Once your web browser is running, enter the present IP address of the SMD17E2 into the address bar. The default address is 192.168.0.50. The unit will respond with the following page.

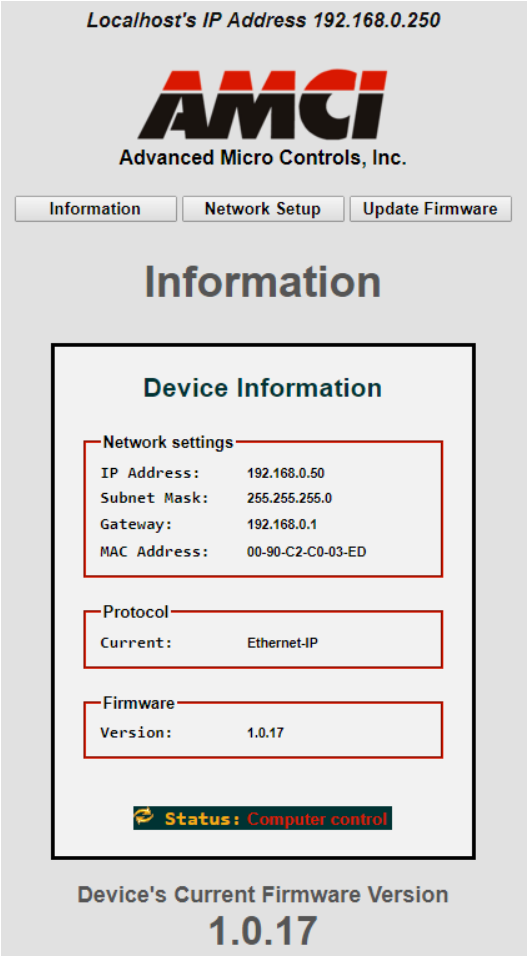


Figure T2.1 SMD17E2 Information Webpage

2.2b Use the Embedded Web Server (continued)

2.2b.3 Network Setup Page

1) Click on the [Network Setup] button to switch to the Network Setup page shown below. This page shows the current IP address settings, as well as the configured protocol.

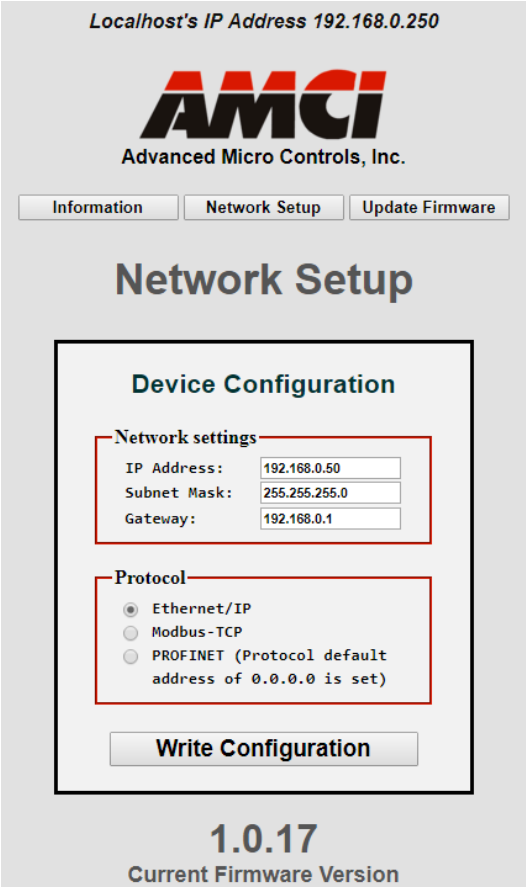


Figure T2.2 SMD17E2 Network Setup Web Page

2) Enter your desired values into the IP Address, Subnet Mask, and Default Gateway fields.

NOTE The Default Gateway setting is not optional! It must be set to a valid address on the chosen subnet. Because the Default Gateway is often not used in device level networks, if you do not have a required value for it, AMCI suggests setting the Default Gateway to the IP address of your host controller.

3) If need be, click on the proper radio button to select the required protocol.

4) Click on the [Write Configuration] button to write the new configuration to the unit. If there are any errors with the data, the unit will display a warning message instead of accepting the new values.

2.2b Use the Embedded Web Server (continued)

2.2b.3 Network Setup Page (continued)

5) If the values are accepted, the following pages will be displayed while the data is being written to the unit.

NOTE Wait for the pop up window to appear before cycling power to the SMD17E2. Cycling power before this window appears may corrupt the non-volitile memory of the SMD17E2. The SMD17E2 will also flash the Network Status LED red to indicate that power must be cycled.

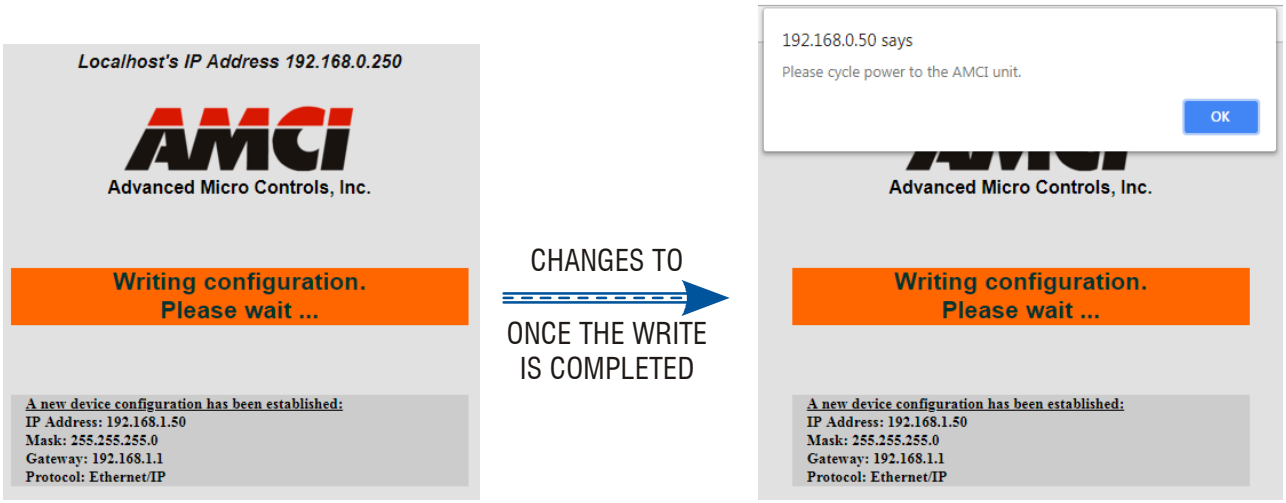


Figure R6.6 Write Configuration to Flash Memory Pages

6) Once instructed to, cycle power to the unit. You can now enter the new IP address into the address bar of your web browser to reconnect with the SMD17E2.

Task Complete

2.2c Use the AMCI NET Configurator Utility

PREREQUISITE: You must know the present IP address. The factory default address is 192.168.0.50.

PREREQUISITE: Task 1.5: ***Power Wiring*** found on page 90. You must be able to power the SMD17E2.

PREREQUISITE: Tasks: 1.7 and 1.8, 1.9, or 1.10, starting on page 92. You must attach your SMD17E2 to your computer.

PREREQUISITE: Optional Task B: ***Configure Your Network Interfaces***. (page 129) The network interfaces on your computer must be correctly configured before you can communicate with an SMD17E2.

2.2c.1 Download the AMCI Net Configurator Utility

The AMCI Net Configurator utility is available on our website, ***www.amci.com***. The latest version available should be used. It can be found in our *Support* section under *Software*. The program exists as a ZIP file, and at the time of this writing, the link was "AMCI Configuration software for all networked products...".

2.2c.2 Install the AMCI Net Configurator Utility

Once downloaded, simply extract the program from the ZIP file and run the program to install the AMCI Net Configurator utility on your computer. The software installs as most products do, giving you the option to change the file locations before installing the utility. Once the install is complete, a link to the utility is available on the Start Menu.

The install process only copies the utility to the designated location and creates links to the Start Menu. No changes are made to your registry settings.

2.2c Use the AMCI Net Configurator Utility (continued)

2.2c.3 Verify that Your Host Controller is Disconnected from the SMD17E2

EtherNet/IP is not a multi-master protocol. There can be only one bus master on the network at a time. In order to program the SMD17E2, the AMCI Net Configurator utility must act as a bus master. Therefore, physically disconnect your host controller from the SMD17E2 before starting the Net Configurator utility.

2.2c.4 Apply or Cycle Power to the SMD17E2

Cycling power to the SMD17E2 will reset any connections it may have with the host controller.

2.2c.5 Start the AMCI Net Configurator Utility

Double click on the utility's icon. A welcome screen similar to the one in figure T2.3 below will appear.

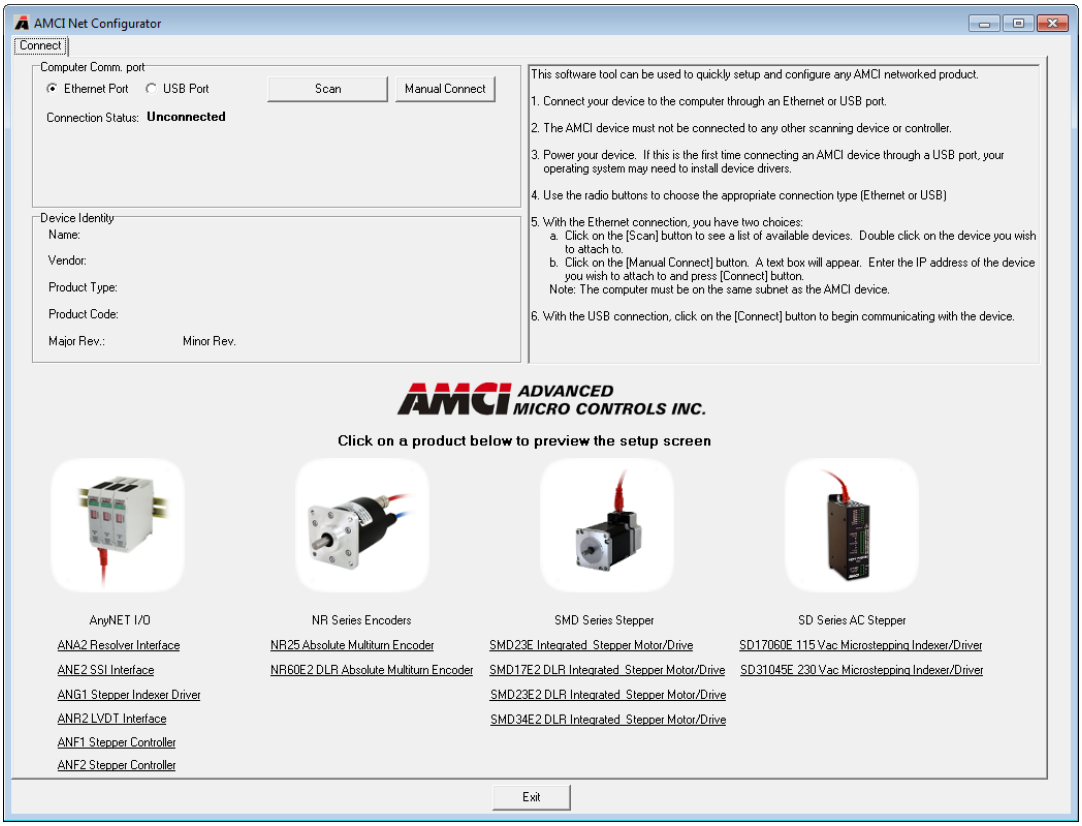


Figure T2.3 Net Configurator Welcome Screen

2.2c.6 Press the [SCAN] button and Connect to the SMD17E2

Pressing the [Scan] button will open the window shown in figure T2.4. The SMD17E2 will appear in the scan list only if the unit and your network interface are on the same subnet. Optionally, you can press the [Manual Connect] button and enter the IP address of the SMD17E2.

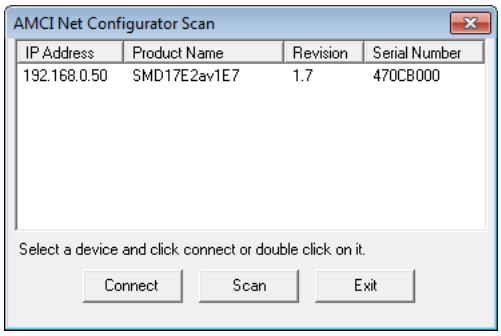


Figure T2.4 Scan for SMD17E2

If scanning for the SMD17E2, click on the IP Address of the SMD17E2 and click on the [Connect] button. The Net Configurator utility will connect to the unit.

2.2c.7 Click on the "Allow IP..." Checkbox to Access the IP Settings

Figure T2.5 below shows the screen that results when you are connected to the SMD17E2. In order to change the IP Address of the unit, you must first click on the checkbox next to the text “Allow IP configuration changes. You will need to restart the device.” Once the checkbox is selected, the [Set IP Address] button will be enabled.

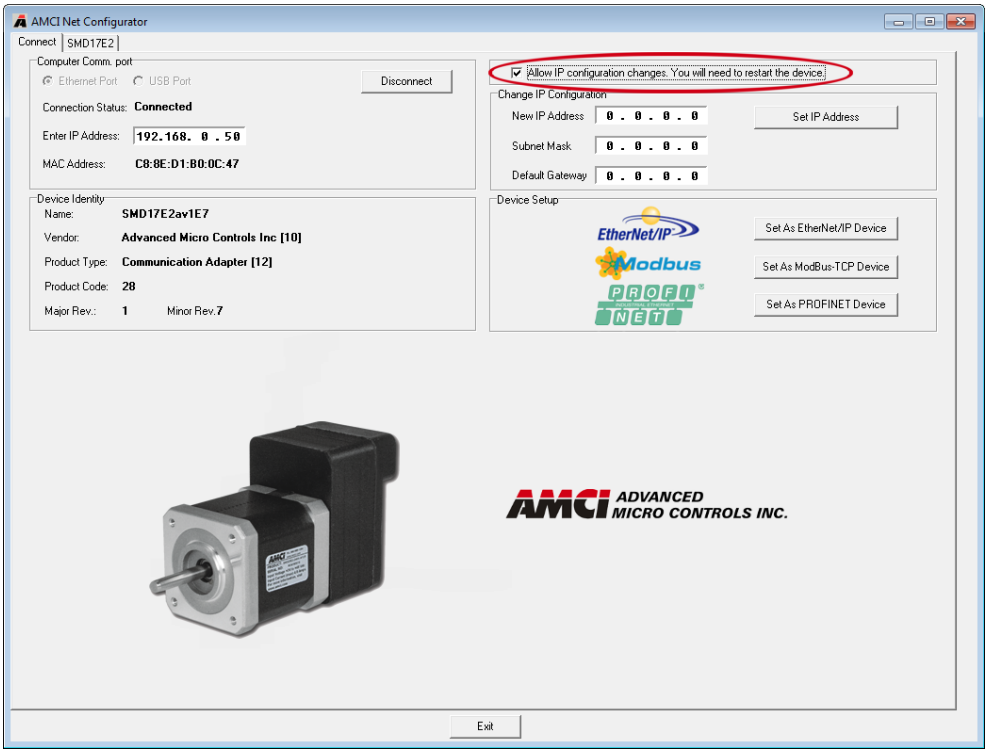


Figure T2.5 Enable IP Address Changes

2.2c.8 Set the IP Address, Subnet Mask, and Default Gateway

Enter your desired values into the IP Address, Subnet Mask, and Default Gateway fields.

NOTE The Default Gateway setting is not optional! In order to comply with the ODVA specification, it must be set to a valid address on the chosen subnet. Because the Default Gateway is often not used in device level networks, if you do not have a required value for it, AMCI suggests setting the Default Gateway to the IP address of your host controller.

2.2c.9 Set the Communications Protocol

The factory default protocol for the SMD17E2 is EtherNet/IP. In order to use the Modbus TCP or PROFINET protocols, simply click on the appropriate button.

2.2c.10 Write the New IP Address to the SMD17E2

Click on the [Set IP Address] button. If there is an error in the settings, the utility will tell you what is wrong. Once they are all correct, the utility will write the new IP address settings to the unit. These settings are automatically saved to nonvolatile memory.

2.2c.11 Remove Power from the SMD17E2

The new IP address will not be used until power to the SMD17E2 has been cycled.

Task Complete

Notes

Many EtherNet/IP platforms support the use of EDS files to simplify the addition and configuration of devices. This chapter covers the installation and use of the EDS file for systems that are programmed with Rockwell Automation Studio 5000 version 20 and above. Other systems will follow a similar pattern. Consult your controller's documentation if you need additional information.

Note: Use of an EDS file is completely optional. The SMD17E2 can always be added to a system as a generic module. If you are using a Rockwell Automation PLC, adding the unit as a generic module is the only option available if you are using RSLogix 5000 version 19 and below or RSLogix 500.

3.1 Obtain the EDS file

All AMCI EDS 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 to its own directory. The ZIP file contains the EDS text file and a custom icon file for the device.

3.2 Install the EDS file

3.2.1 Start the EDS Hardware Installation Tool

1) Once Studio 5000 is running, in the menu bar select Tools → EDS Hardware Installation Tool. This will open the EDS Wizard.

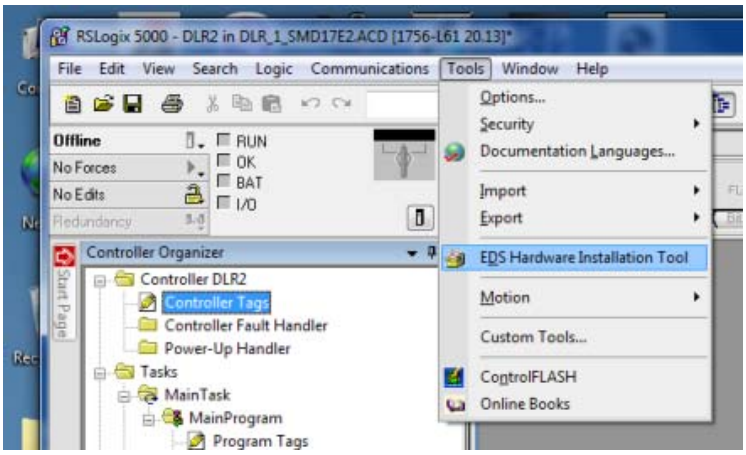


Figure T3.1 Opening the EDS Wizard

2) Click on [Next >] to advance to the Options screen.

3.2 Install the EDS file (continued)

3.2.2 Install the EDS File

1) On the Options screen, select the Register an EDS file(s) radio button and press [Next >].

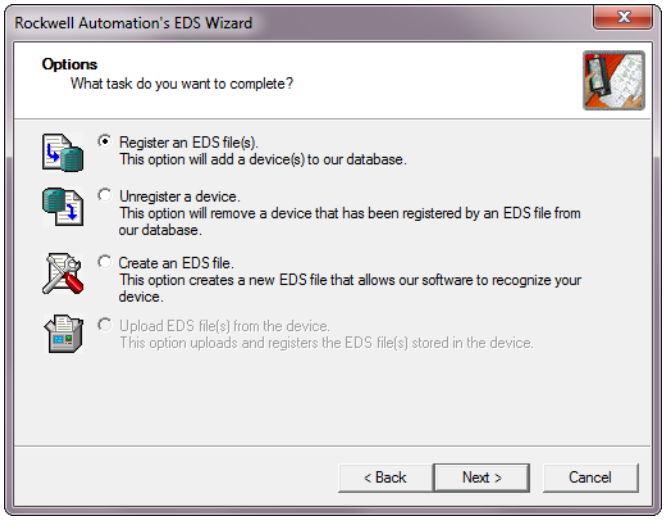


Figure T3.2 EDS Options Screen

2) The registration screen will open. Select the Register a single file radio button.

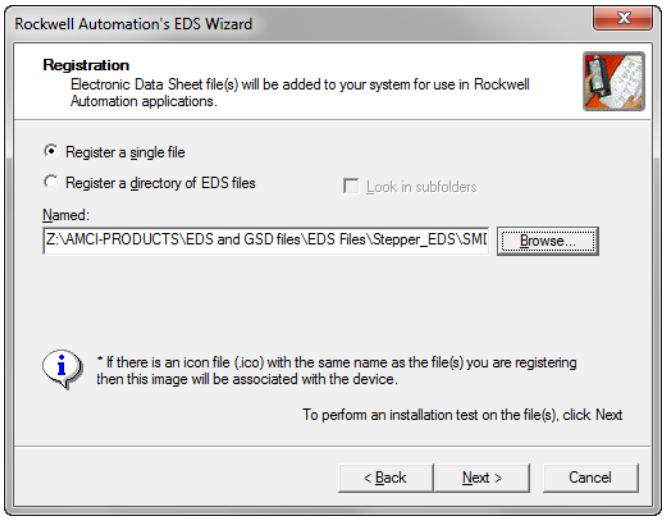


Figure T3.3 EDS Registration Screen

3) Click on the [Browse...] button and browse to the folder that contains the extracted EDS file you downloaded from the AMCI website. Select the EDS file and click on the [Open] button to return to the registration screen. Click on the [Next >] button to advance to the EDS file test screen.

3.2 Install the EDS file (continued)

3.2.2 Install the EDS File (continued)

4) Once at the EDS File Installation Test Results screen, expand the tree as needed to view the results of the installation test for the EDS file. You should see a green check mark next to the file name indicating that the EDS file is correct.

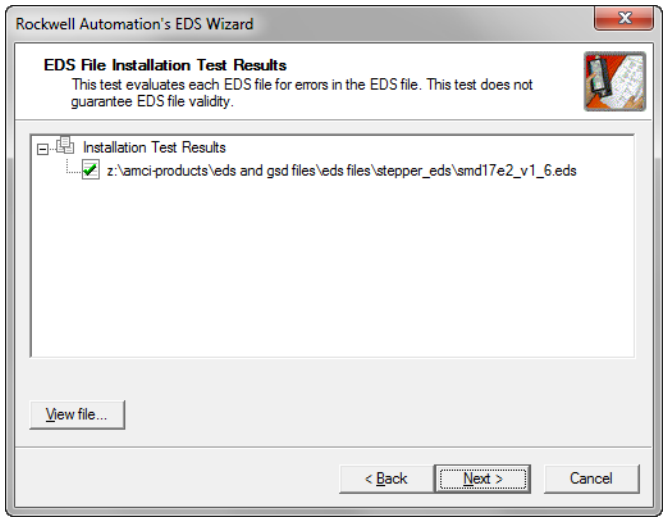


Figure T3.4 EDS Test Screen

5) Press on the [Next >] button to advance to the Change Graphic Image screen. This screen gives you the ability to change the icon associated with the device.

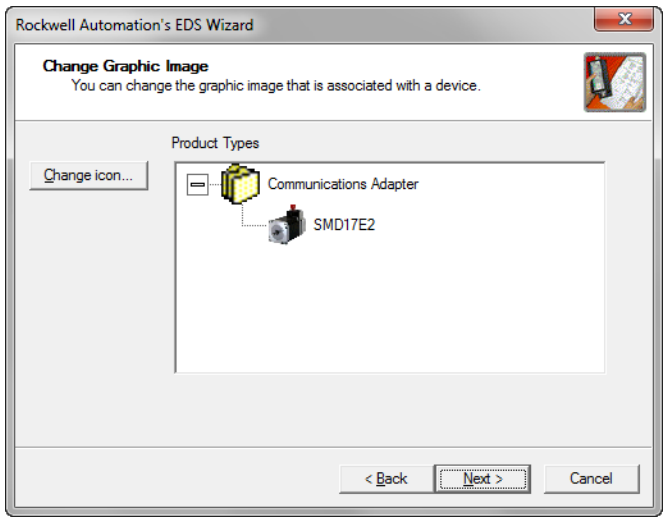


Figure T3.5 Change ECS Icon Screen

6) Click on the [Change icon...] button. In the window that opens, click on [Browse...] and browse to the folder that contains the extracted EDS and icon files you downloaded from the AMCI website.
7) Select the icon file (*.ico) associated with the device. Click on the [Open] button and then on [OK] to return to the Change Graphic Image screen.
8) Click on the [Next...] button to advance to the completion screen. The Completion screen tells you that you have successfully completed the wizard.
9) Click on the [Finish] button to exit the EDS wizard.

3.3 Host System Configuration

Studio 5000 is used to configure both the ControlLogix and CompactLogix platforms. When using these platforms, you have the option of using a separate Ethernet Bridge module or an Ethernet port built into the processor.

If the Ethernet port is built into processor, the only step you have to take before adding an AMCI SMD17E2 is to create a new project with the correct processor or modify an existing project. Once this is done, the Ethernet port will automatically appear in the Project Tree. If you are using an Ethernet bridge module, you will have to add it to the I/O Configuration tree before adding the unit to your project.

Refer to your Rockwell Automation documentation if you need instructions for configuring the ethernet port.

3.4 Add the SMD17E2 to Your Project

You can add an AMCI SMD17E2 to the project once the Ethernet port (built-in or bridge module) is configured. As shown in figure T3.6 below, the Ethernet port will be listed under the I/O Configuration tree.

1) Right click on the Ethernet port and then click on "New Module..." in the pop-up menu.

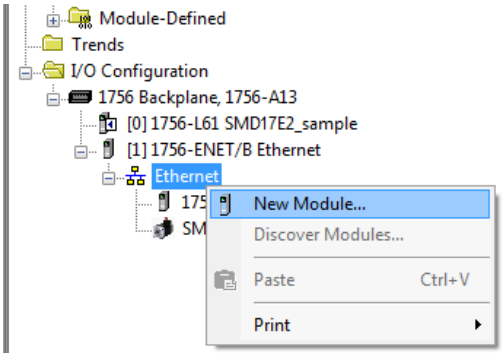


Figure T3.6 Adding an AMCI Ethernet Driver

2) In the resulting Select Module Type screen, select "Advanced Micro Controls In. (AMCI)" in the Vendor Filters. This will limit the results to catalog numbers from AMCI.
3) Select "SMD17E2" in the resulting list.
4) Click on the [Create] button to create the module.
5) Click on [Close] if necessary to close the Select Module Type screen.

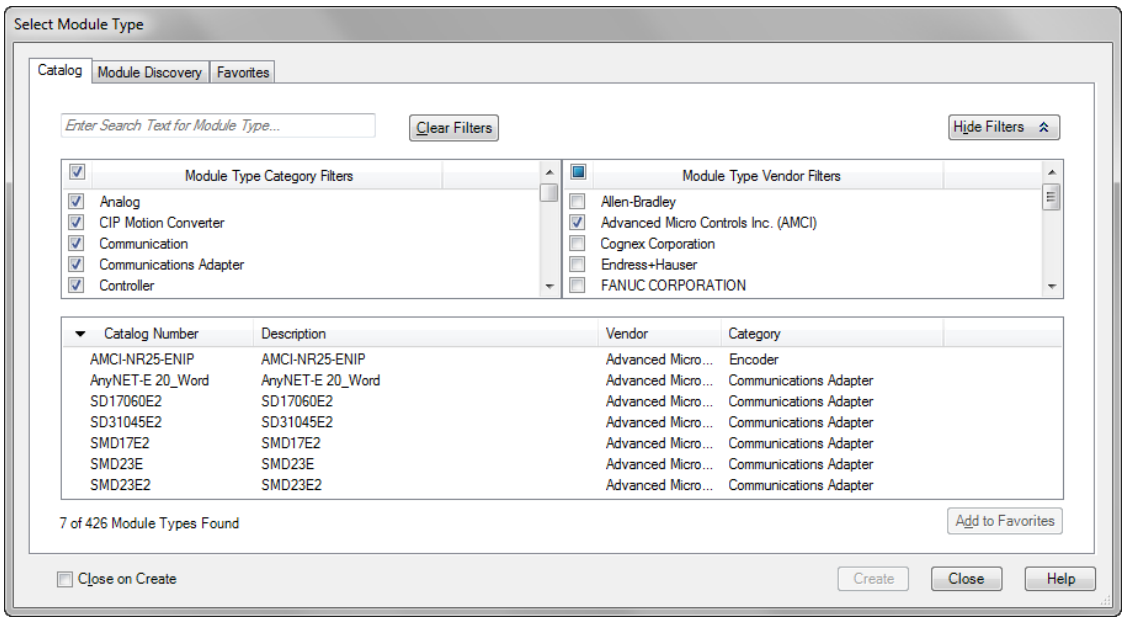


Figure T3.7 Selecting the Networked Driver

3.5 Configure the SMD17E2 Driver

If you are continuing from step 3.4, the resulting New Module screen is used to configure the network connection between the SMD17E2 and your controller. If you need to open the screen to perform this task at a later time, right click on the SMD17E2 in the project tree and then select "Properties" from the drop-down menu

NOTE Tabs that are not listed in the steps below are filled with reasonable defaults by the EDS file.

3.5.1 General Tab

The Name, Description, and IP address of the device must be specified here. The [Change...] button allows you to change the Module Definition if needed.

3.5.2 Connection Tab

The default RPI time is eight milliseconds. This value can be changed in this tab.

3.5.3 Configuration Tab

The Configuration tab is used to define the configuration data that is written down to the SMD17E2 when the device connects to the network. You can also click on the [Apply] button to write down the configuration data to the SMD17E2 at any time.

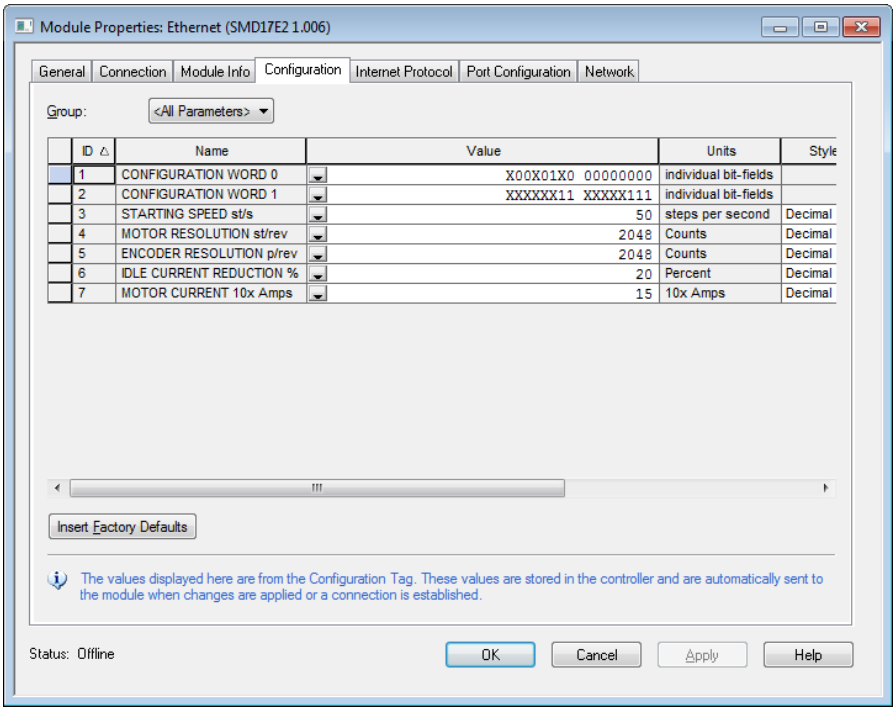


Figure T3.8 Networked Driver Configuration with EDS File

The EDS file defines tags that are used to configure the SMD17E2. These tags follow the format of the Configuration Data given in reference chapter 6, ***Configuration Mode Data Format***, starting on page 59.

NOTE Bits 8 and 9 of Configuration Word 1, *Binary_Output_Format* and *Binary_Input_Format*, should both be set to "1" when using and EDS setup so that command and response data is sent as 32-bit binary values.

3.5 Configure the SMD17E2 Driver (continued)

3.5.3 Configuration Tab (continued)

NOTE When using the EDS file, the Starting Speed is transmitted as a double integer value. The parameter does not use the multi-word format outlined in Configuration Mode Data Format reference chapter.

3.6 Buffering the I/O Data

Input and output data is transferred asynchronously to the program scan. These data tags should be buffered with Synchronous Copy File instructions to guarantee stable input data during the program scan and guarantee that complete command data is delivered to the device.

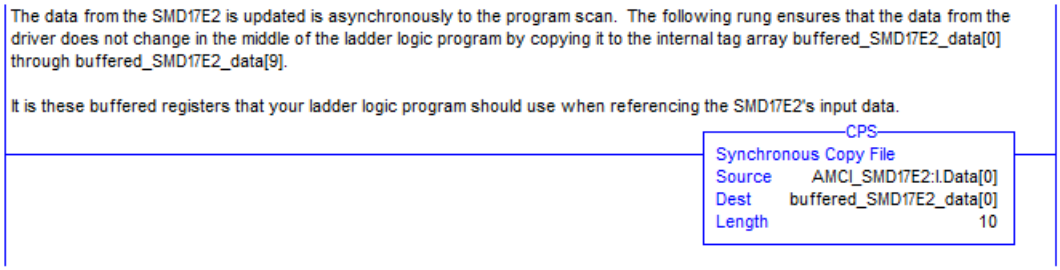


Figure T3.9 Buffer I/O Data

- When copying input data, the data can be converted from byte to integer format by specifying an integer array as the destination for the instruction. The array must contain at least ten integer elements. The length of the copy should be ten.
- When copying output data, the data can be converted from integer to byte format by specifying an integer array as the source for the instruction. The source array must contain at least ten elements and the length of the copy must be twenty. (The destination determines the length of the transfer, not the source.)

IMPLICIT COMMUNICATIONS WITHOUT AN EDS

An AMCI SMD17E2 requires a host controller to issue configuration and motion commands to it. This chapter tells you how to configure implicit connections in EtherNet/IP systems that do not use EDS files. If you instead wish to use explicit messaging, refer to the next chapter for information on using message instructions.

Rockwell Automation's RSLogix 5000 version 20 software is used for the example installation in this chapter.

4.1 Host System Configuration

RSLogix 5000 is used to configure both the ControlLogix and CompactLogix platforms. When using these platforms, you have the option of using a separate Ethernet Bridge module or an Ethernet port built into the processor.

If the Ethernet port is built into processor, the only step you have to take before adding the SMD17E2 is to create a new project with the correct processor or modify an existing project. Once this is done, the Ethernet port will automatically appear in the Project Tree. If you are using an Ethernet bridge module, you will have to add it to the I/O Configuration tree before adding the driver to your project.

Refer to your Rockwell Automation documentation if you need instructions for configuring the ethernet port.

4.2 Add the SMD17E2

You can add the SMD17E2 to the project once the Ethernet port (built-in or bridge module) is configured.

1) Right click on the Ethernet port and then click on "New Module..." in the pop-up menu.

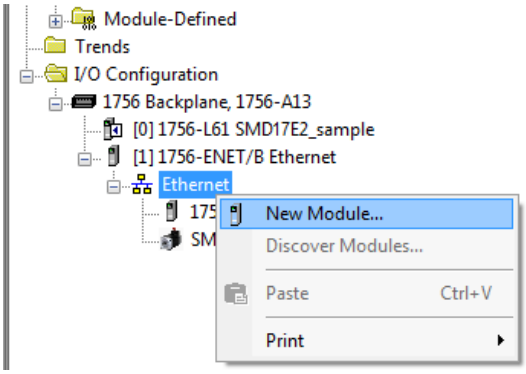


Figure T4.1 Adding an AMCI SMD17E2

4.2 Add the SMD17E2 (continued)

2) In the resulting Select Module Type screen, type “generic” into the filter as shown in figure T4.2. This will limit the results in the Catalog Number list.
3) Select the Catalog Number “ETHERNET-MODULE” in the list.
4) Click on the [Create] button to create the module.
5) Click on [Close] if necessary to close the Select Module Type screen.

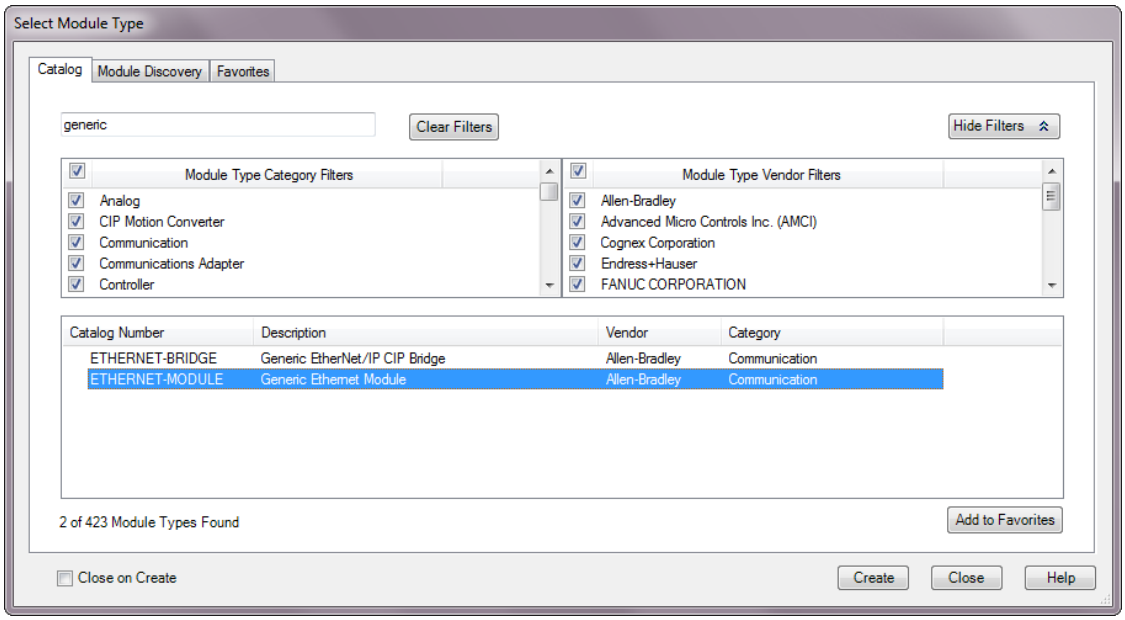


Figure T4.2 Selecting a Generic Device

4.2 Add the SMD17E2 (continued)

6) Set the following parameters in the Module Properties window. All parameters not listed here are optional. Figure T4.3 shows a completed screen.

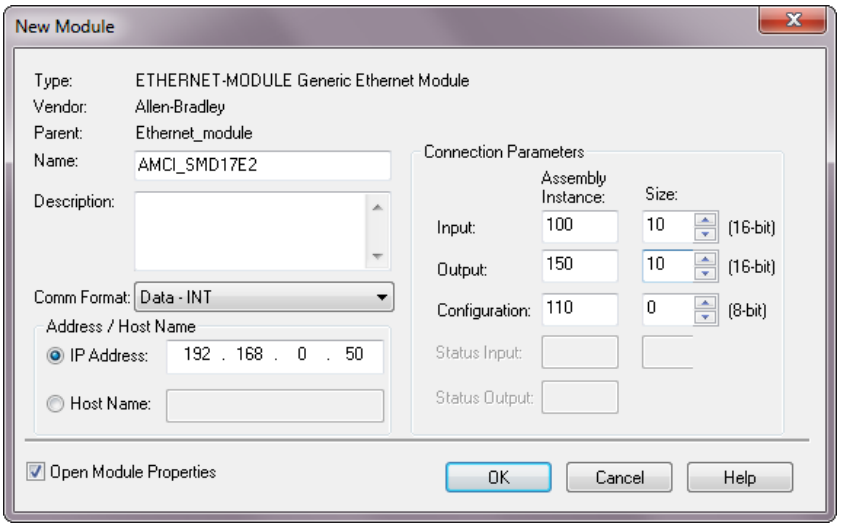


Figure T4.3 Configuration Screen - Generic Device

- **Name:** A descriptive name for the SMD17E2
- **Comm Format:** Data - INT

CAUTION

The Comm Format defaults to Data - DINT. The SMD17E2 will not be able to communicate with the host controller if this format is not changed when the device is added to the system. Once added, the Comm Format cannot be changed. The device must be deleted and added to the project if the Comm Format is incorrect.

- **IP Address:** Must be the address you set for the SMD17E2. Refer to the ***Set the IP Address and Protocol*** task chapter starting on page 95 for information on setting the IP Address of the unit.
- **Input:** Assembly Instance = 100, Size = 10 words.
- **Output:** Assembly Instance = 150, Size = 10 words.
- **Configuration:** Assembly Instance = 110, Size = 0

7) Verify that the "Open Module Properties" check box is selected and click on [OK]. The Module Properties window will open. You can set the RPI time as required for your system in this window. The minimum RPI time for an SMD17E2 is 2 milliseconds. When done, click on [OK] to complete the setup.

Error Code 16#0109

The PLC will generate an Error Code 16#0109 when the Comm Format parameter is not changed from its default of "Data-DINT" to "Data-INT". This is the most common cause of communication failures with the SMD17E2.

4.3 Configure the SMD17E2

With the configuration assembly instance size set the zero, the device will join the EtherNet/IP network as soon as the request is made to it. If the SMD17E2 has a configuration stored in flash memory, it will be to configure the unit on power up. You can also configure the unit at anytime and store this new configuration to flash. Configuration is accomplished by writing a block of data to the device that is formatted according to the specifications in the ***Configuration Mode Data Format***, reference chapter, which starts on page 59.

It is possible to store configuration data in the flash memory of the SMD17E2 and this configuration will be used on power up to configure the device. However, writing the configuration data to the driver on power up may simplify system maintenance if the device ever has to be swapped out.

4.4 Buffer I/O Data

Data to and from the SMD17E2 should be buffered once per scan using Synchronous Copy instructions. This is to insure stable input data during the program scan and guarantee that complete command data is delivered to the device. Ten word integer arrays can be used for this purpose.

These data tags should be buffered with Synchronous Copy File instructions to guarantee stable input data during the program scan and guarantee that complete command data is delivered to the device.

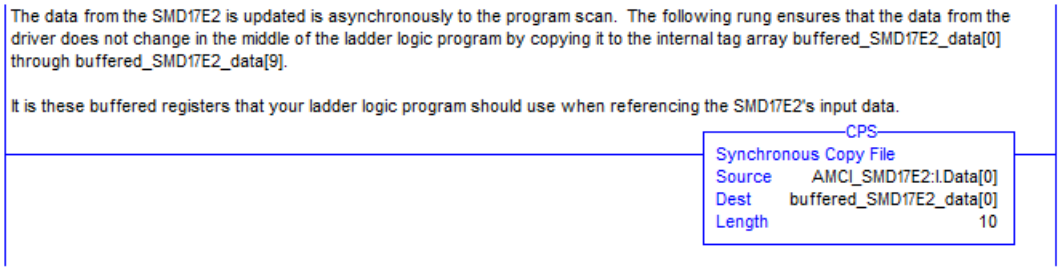


Figure T4.4 Buffer I/O Data

EtherNet/IP Explicit Messaging

All controllers that support EtherNet/IP support explicit messaging. When using explicit messaging, Message Instructions must be added to your program to communicate with the SMD17E2. Explicit messaging can be use on platforms that also support implicit messaging.

Rockwell Automation controllers which are programmed with the RSLogix 500 software only support explicit messaging. A MicroLogix 1100 will be used as an example in this chapter.

5.1 Required Message Instructions

Only two instructions are required to transfer data between the PLC and the SMD17E2. One instruction reads data from the unit and the other writes data to it. The sample programs available from AMCI use this style of programming. The two instructions are alternately triggered using the instruction's ENABLE bits. The remainder of the program controls when data in the source tags of the write instruction changes. The following table gives the required attributes for the instructions.

	Read Instruction	Write Instruction
Service Type	Read Assembly	Write Assembly
Service Code	E (hex)	10 (hex)
Class	4 (hex)	4 (hex)
Instance	100 (decimal)	150 (decimal)
Attribute	3 (hex)	3 (hex)
Length	20 bytes	20 bytes

Table T5.1 Message Instruction Attributes

NOTE Only RSLogix 500 version 8.0 or above can be used to configure Message Instructions to communicate with an EtherNet/IP device. Message Instructions do not work correctly in version 10 of RSLogix 500.

5.2 Create Four New Data Files.

- An Integer file to contain the data read from the SMD17E2. This file must be at least 10 words in length.
- An Integer file to contain the data written to the SMD17E2. This file must be at least 10 words in length.
- A Message (MG) data file. This file must have at least two elements, one to control the Read Operation and one to control the Write Operation.
- An Extended Routing Information (RIX) data file. This file is used to store information used by the Message Instructions. This file must have at least two elements, one for the Read Operation and one for the Write Operation.

5.3 Add the Message Instructions to your Ladder Logic

The following rungs show how you can alternately read data from and write data to your SMD17E2.

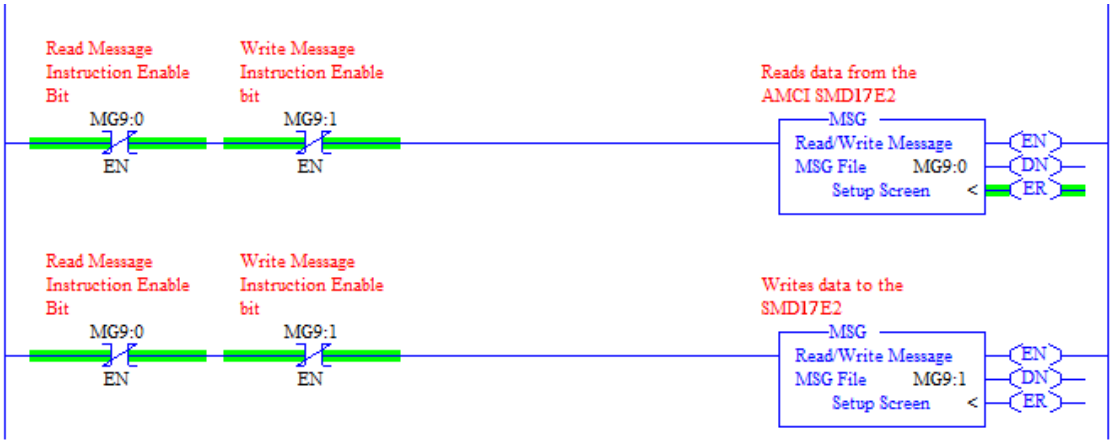


Figure T5.1 Message Instruction Example

1) Double click on *Setup Screen* text inside the Message Instruction. The following window will open. Note that this is the default window and its appearance will change considerably as you progress through these steps.

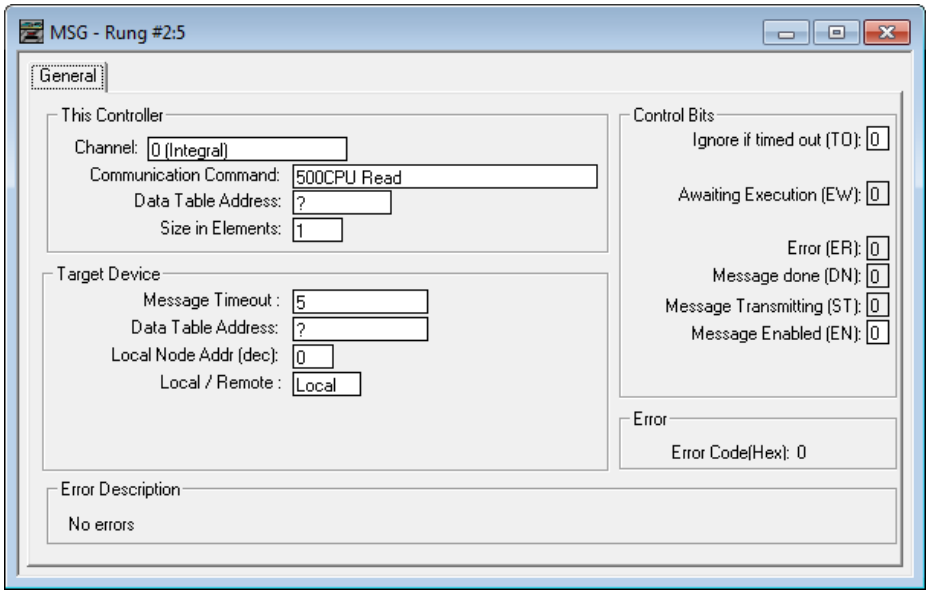


Figure T5.2 Message Instruction Setup Screen

2) Double click in the *Channel* field, click on the ▼, select "1 (Integral)", and press Enter.
3) Double click in the *Communication Command* field, click on the ▼, select "CIP Generic" and press Enter.
4) If the Message Instruction is being used to read data from the SMD17E2, enter the integer file where the data will be placed in the *Data Table Address (Received)* field and press enter.
5) If the Message Instruction is being used to write data to the SMD17E2, enter the integer file where the source data will be located in the *Data Table Address (Send)* field and press Enter.
6) Enter "20" as the number of bytes needed in either the *Size In Bytes (Receive)* or *Size In Bytes (Send)* fields. The SMD17E2 requires 20 bytes for both Receive and Send.
7) Enter a RIX address in the *Extended Routing* Info field. Please note that each Message Instruction must have its own RIX address.

5.3 Add the Message Instructions to your Ladder Logic (continued)

8) Double click in the *Service* field and select "Read Assembly" for a Message Instruction that is being used to read data from the SMD17E2, or "Write Assemble" for a Message Instruction that is being used to send data to the SMD17E2, and press Enter.

9) For *Read* operations, the *Service Code* field will change to "E" (hex). For *Write* operations, the *Service Code* field will change to "10" (hex). For both read and write operations, the *Class* field will change to "4" (hex), and the *Attribute* field will change to "3" (hex).

10) For Read operations, enter a value of 100 decimal (64 hex) in the *Instance* field.
For Write operations, enter a value of 150 decimal (96 hex) in the *Instance* field.

The figure below show a typical configuration for Message Instructions being used to read data from the SMD17E2. Please note that the Data Table Address (Receive) field may be different in your application.

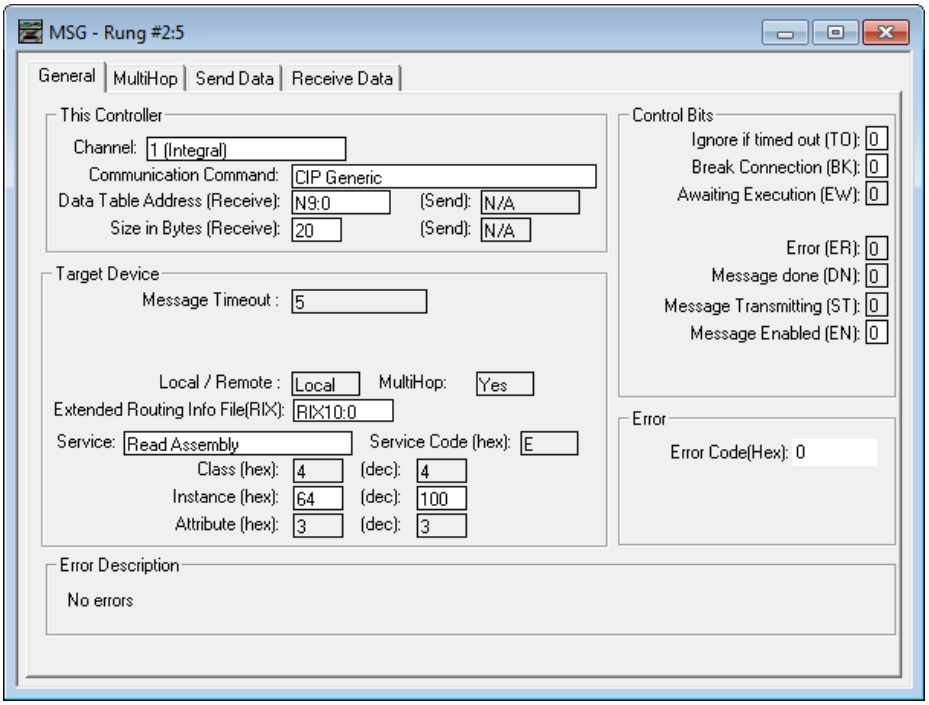


Figure T5.3 Read Message Instruction Setup Screen

5.3 Add the Message Instructions to your Ladder Logic (continued)

The figure below show a typical configuration for Message Instructions being used to write data to the SMD17E2. Please note that the Data Table Address (Send) field may be different in your application.

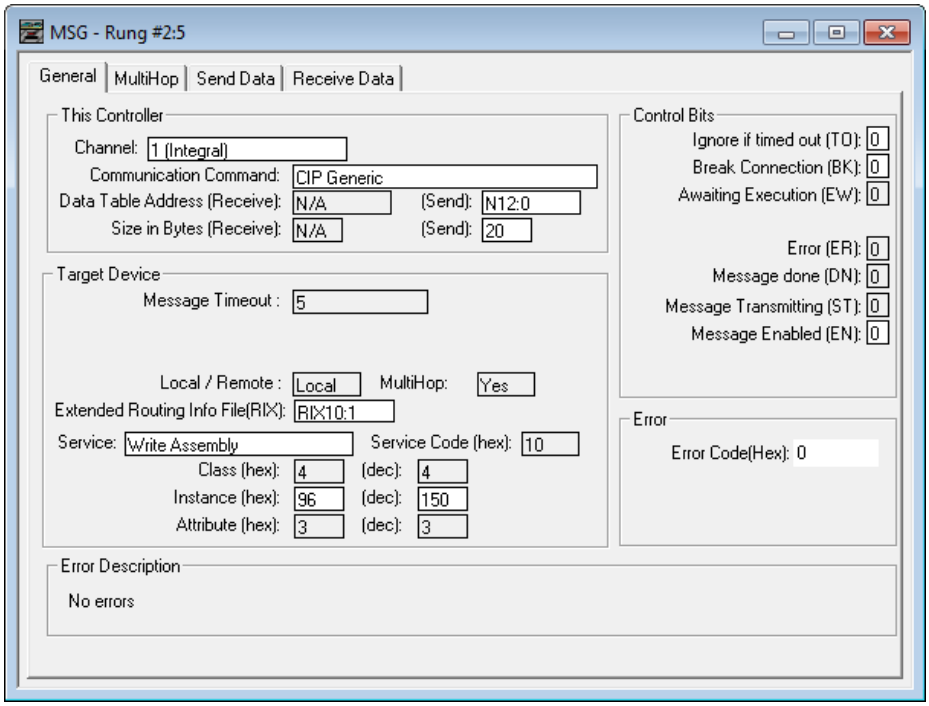


Figure T5.4 Write Message Instruction Setup Screen

Click on the MultiHop tab on the top of the window. As shown in figure T5.5, enter the IP address of the SMD17E2 and press Enter.

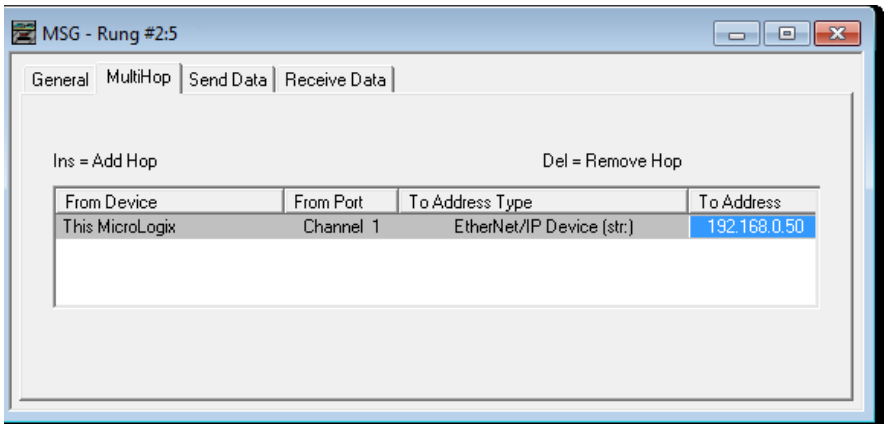


Figure T5.5 Message Instruction MultiHop Settings

After you are finished adding both the read and write message instructions to your program, save and download the program to the PLC.

5.4 Troubleshooting

If you are unable to communicate with the SMD17E2, the problem may be that the Ethernet port of your MicroLogix 1100 has not been configured. To check this:

1) Double click on Channel Configuration in the Project Tree and then select the Channel 1 tab. The following window will open.

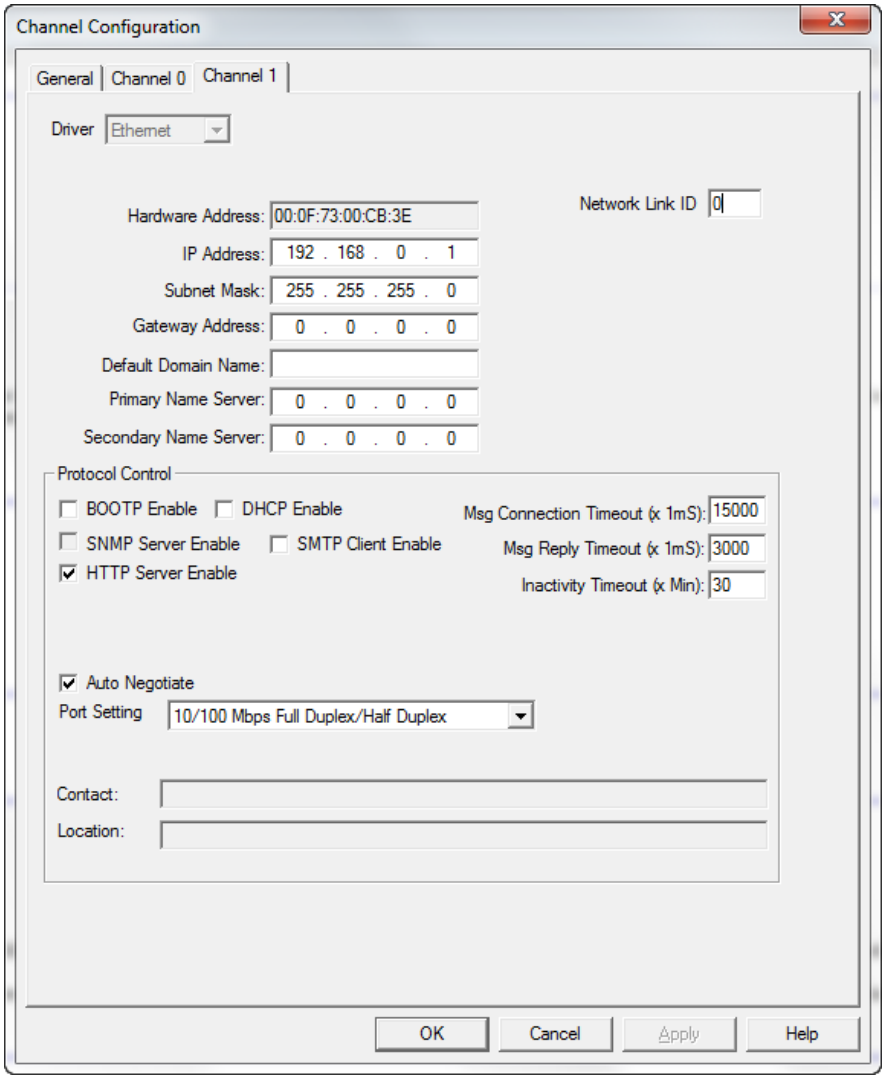


Figure R6.7 MicroLogix Ethernet Configuration Screen

2) Enter the IP address and Subnet Mask of your MicroLogix 1100, (not the address of the SMD17E2) and click on [Apply]. The Ethernet Port should now be working.

NOTE AMCI is aware of an issue with the RIX data type in version 10 of RSLogix 500. If you are experiencing communications errors and are running version 10, please contact Rockwell Automation for support.

Notes

MODBUS TCP CONFIGURATION

An AMCI SMD17E2 that has been configured for the Modbus TCP protocol requires a host controller to issue configuration data and motion commands to the unit. This chapter tell you how the I/O words used by an AMCI SMD17E2 are mapped to the Modbus I/O registers.

6.1 Enable Modbus TCP Protocol

The AMCI Net Configurator utility can be used to change the communications protocol used by the SMD17E2. This is typically done while setting the IP address. Specifically, follow the steps in section 2.2c, ***Use the AMCI NET Configurator Utility*** which starts on page 98.

6.2 Modbus Addressing

The register addresses used in this manual are the *Modbus logical reference numbers*†, which are unsigned integers starting at zero. This is often called *zero based* addressing. In this scheme, the first register is given an address of zero. This is the actual addressing scheme used in the Modbus packets.

Another common addressing scheme is *one based* or *data model* addressing. In this scheme, the register's number is used as its address, so the first register, Register 1 in the data model, has an address of 1.

6.2.1 Modbus Table Mapping

The Discrete Input and Input Register tables in the Modbus data model map to the same physical memory locations in the SMD17E2 units.

- These registers hold data that is reported from the driver to the host controller. This data is typically command responses and status data.
- Addresses for these registers and inputs start at 0 in zero based addressing.

As examples:

- Discrete Input 0 is the same memory location as bit 0 of the first Input Register.
- Register address 3, the fourth register, contains Discrete Inputs 48 through 63.

The Coil and Holding Register tables in the Modbus data model map to the same physical memory locations in the SMD17E2 units.

- These registers hold data that is from the host controller to the unit. This data is typically commands.
- Addresses for these registers start at 1024 in zero based addressing. Coil addresses start at 16,384 in zero based addressing (1024*16).

As examples:

- Coil 16384 is the same memory location as bit 0 of the first Holding Register.
- Register address 1025, the address of the second Holding Register, contains Discrete Inputs 16,400 through 16,415 in zero based addressing.

6.2.2 Host Addressing

Your host controller may not use these basic addressing schemes for communicating over a Modbus connection. For example, Modicon controllers use addresses starting at 30000 for Input Registers and addresses starting at 40000 for Holding Registers. GE hosts internally use their %R memory for Holding Registers and %AI memory for Input Registers.

If this is the case, you will define a mapping between your host controller's addressing scheme and the zero based Modbus TCP addresses when you add the SMD17E2 to your host controller. Refer to your host controller's documentation for information on how to accomplish this.

† MODBUS Application Protocol Specification V1.1b3, section 4.3: MODBUS Data model. ***www.modbus.org***

6.3 AMCI Modbus TCP Memory Layout

The SMD17E2 has a starting Input Register address of 0 and a starting Output Register address of 1024. Input Registers hold the data from the driver while Output Registers hold the data to be written to the unit. Figure T6.1 shows how an SMD17E2 is mapped to the Modbus data reference. The complete specification for the Modbus protocol can be downloaded at http://www.modbus.org/specs.php.

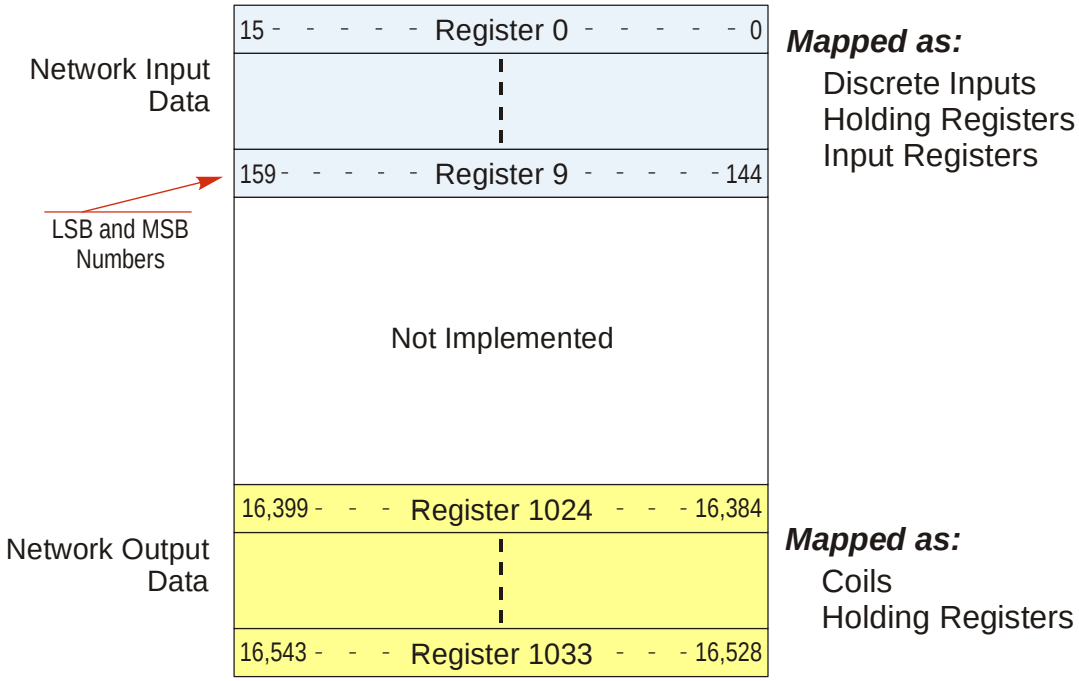


Figure T6.1 Modbus Data Reference Map

6.4 Supported Modbus Functions

Function Code	Function Name	SMD17E2 Register	Addressing method
1	Read Coils	OUTPUT	Bit: Addresses starting at 16,384
2	Read Discrete Inputs	INPUT	Bit: Addresses starting at 0
3	Read Holding Registers	OUTPUT & INPUT	Word: Out Regs. Starting at 1024 In Regs. Starting at 0
4	Read Input Registers	INPUT	Word: Addresses starting at 0.
5	Write Single Coil	OUTPUT	Bit: Addresses starting at 16,384
6	Write Single Register	OUTPUT	Word: Addresses starting at 1024
15	Write Multiple Coils	OUTPUT	Bit: Addresses starting at 16,384
16	Write Multiple Registers	OUTPUT	Word: Addresses starting at 1024
22	Mask Write Register	OUTPUT	Word: Addresses starting at 1024
23	Read/Write Registers	INPUT/OUTPUT	Word: Out Regs. Starting at 1024 In Regs. Starting at 0

Table T6.1 Supported Modbus Functions

Table T6.1 above lists all of the Modbus functions supported by an SMD17E2. AMCI supports all of these functions so that you can control the unit as you see fit. However, if you are looking for the easiest way to interface with your unit, then you only need to use the *Read/Write Registers* function, which is function code 23.

NOTE Each SMD17E2 buffers the data that is sent to it over the network. If you use the *Read/Write Registers* function to write configuration data to the unit, then the data read with that command will not contain the response to the new configuration data. The response to the new data will be sent with the next data read.

6.5 Supported Modbus Exceptions

Code	Name	Description
01	Illegal function	The SMD17E2 does not support the function code in the query
02	Illegal data address	The data address received in the query is outside the initialized memory area
03	Illegal data value	The data in the request is illegal

Table T6.2 Supported Modbus Exceptions

Notes

PROFINET NETWORK CONFIGURATION

This chapter outlines the steps commonly needed to get an SMD17E2 communicating with the PROFINET master. A Siemens SIMATIC S7-1212C controller is used as an example.

Basic Steps

Configuring a PROFINET host requires a few basic steps.

1) Download the ZIP archive that contains the GSDML files for the SMD17E2 from the www.amci.com website.
2) Install the GSDML file into the configuration software for your host controller.
3) Add the SMD17E2 to the PROFINET Network.
4) Set the I/O word addresses used to communicate with the SMD17E2.

7.1 Download the GSDML files

The GSDML files are available on the AMCI website on the ***http://www.amci.com/industrial-automation-support/configuration-files/*** web page. The file is a ZIP archive that has to be extracted to a folder on your computer. Extracting the ZIP file will leave you with multiple files. One is the GSDML file and the others are icon files for the various devices.

7.2 GSDML File Installation

1) Open or create a new project that will include the SMD17E2 and open the Project View of the project.
2) In the menu, select *Options -> Manage general station description files(GSD).*
3) In the window that opens, click on the [...] button and navigate to the folder that contains the extracted GSDML file you downloaded from the AMCI website. Once at the folder, click on the [OK] button.
4) Click on the check box next to the name of the GSD file and click on the [Install] button. The system will install the GSD file.
5) Click the [Close] button and wait for the software to finish installing the file and updating the Hardware Catalog.

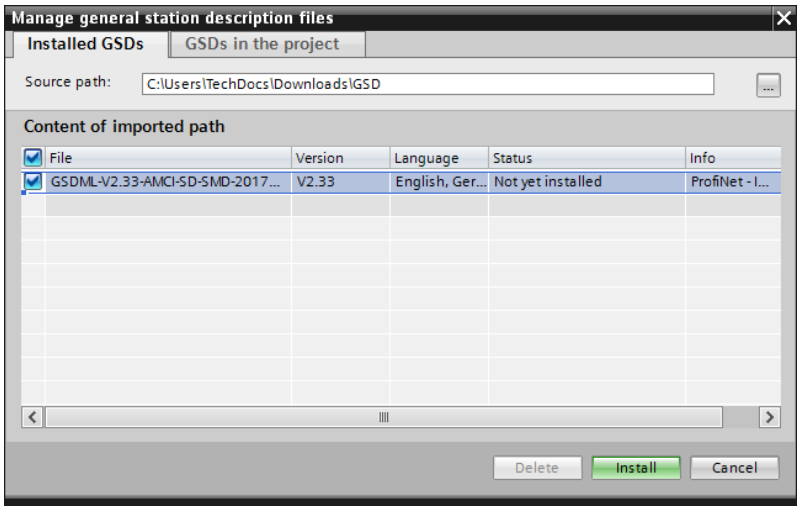


Figure T7.1 GSD File Installation

7.3 Configure the PROFINET Network

A CPU must be added to the project and the PROFINET network must be configured before a Networked Stepper Driver can be added to the system.

Refer to Siemens documentation for information on configuring the PROFINET network to suit your application.

7.4 Add the SMD17E2 to the PROFINET Network

1) With the project open in Project View, double click on “Device & Networks” in the project tree.
2) If need be, click on the “Hardware Catalog” vertical tab to open the Hardware Catalog.
3) You can search for “SMD17”, or browse to the SMD17x2 icon by clicking through *Other field devices +> PROFINET IO +> IO +> Advanced Micro Controls Inc. +> AMCI_Products +> AMCI_Drives*. Drag and drop the appropriate icon onto the PROFINET network.
4) Drag the green square on the SMD17x2 icon onto the PROFINET network line to connect the device to the network.

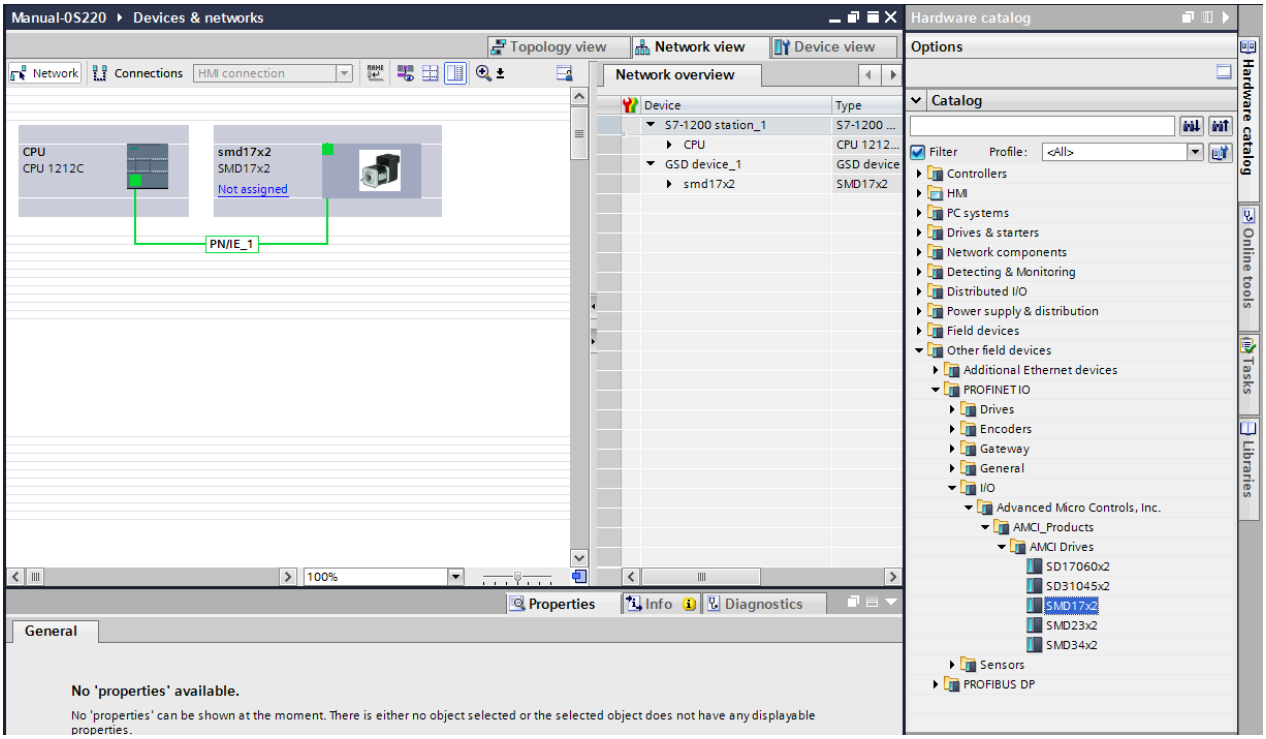


Figure T7.2 Networked Driver Added to PROFINET Network

7.4 Add the SMD17E2 to the PROFINET Network (continued)

5) Right click on the SMD17 icon and select "Properties" from the pop up menu. The Inspector window will open at the bottom of the screen. Under the "General" tab, select the "▶General" heading. You can rename the SMD17 by changing the Name: field.

6) Under the "▶PROFINET interface [x1]" heading, select "Ethernet addressing". Under the IP protocol section, set the desired IP address and subnet mask for the SMD17.

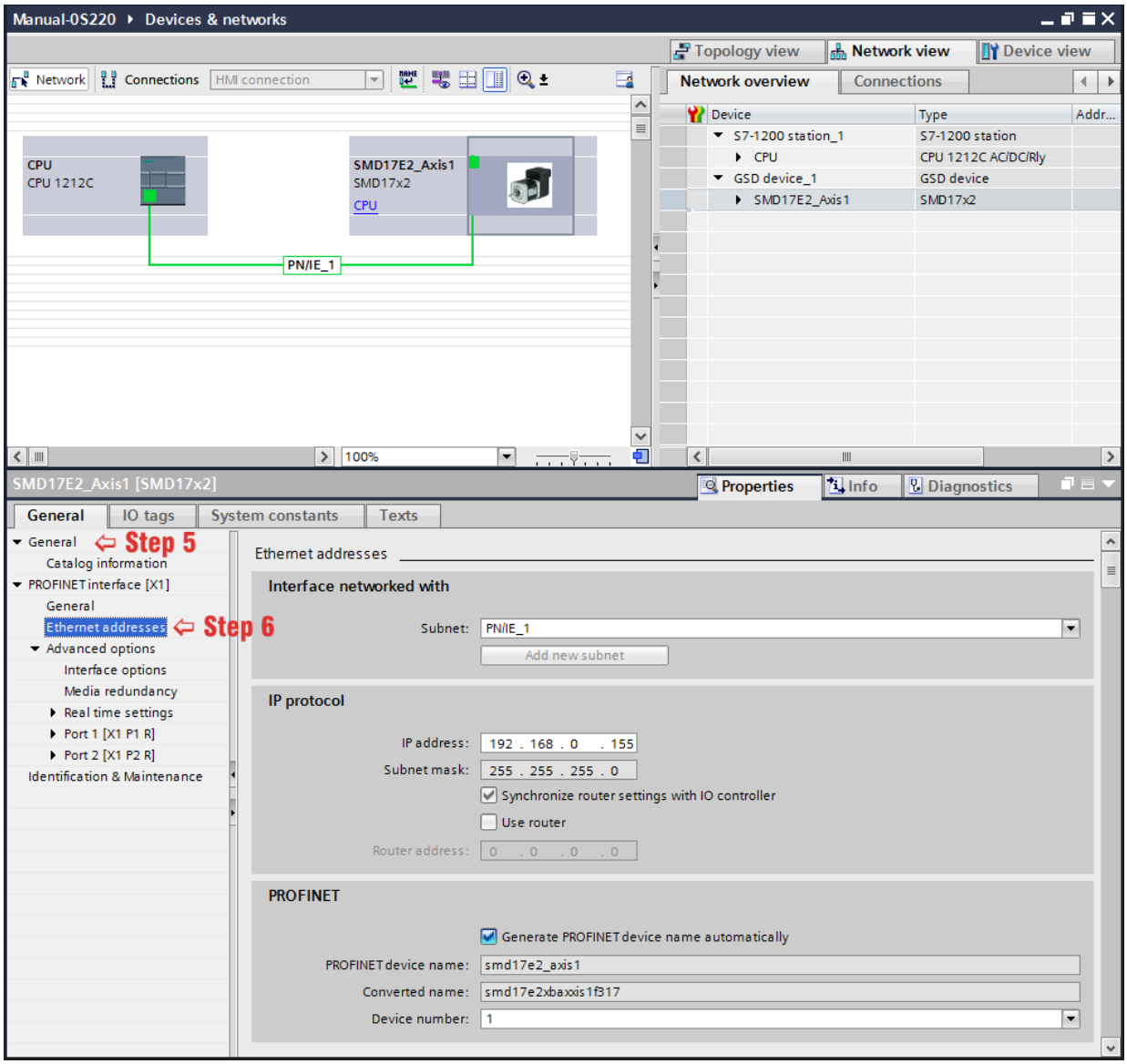


Figure T7.3 Networked Driver IP Addressing

7.5 Set the I/O Configuration

The SMD17E2 units require 10 Input Words (20 Input Bytes) and 10 Output Words (20 Output Bytes). All required Input and Output Bytes are defined by the GSDML file and divided into suitable modules. These settings are shown in the Table T7.1.

Input / Output Bytes of an SMD17E2	Input / Output Modules of an SMD17E2
20 Input Bytes	Input Module - Slot 1: 20 bytes
20 Output Bytes	Output Module - Slot 2: 20 bytes

Table T7.1 PROFINET I/O Configuration

1) With the SMD17E2 icon selected on the PROFINET bus, click on the “Device view” tab. The view in the Hardware Catalog will change. Expand the Module tree to show both the Input and Output modules.

2) To map the I/O bytes to the CPU, double click on the “20 bytes IN” and “20 bytes OUT” icons in the Hardware Catalog. The system will automatically assign the next I and Q addresses to the data table.

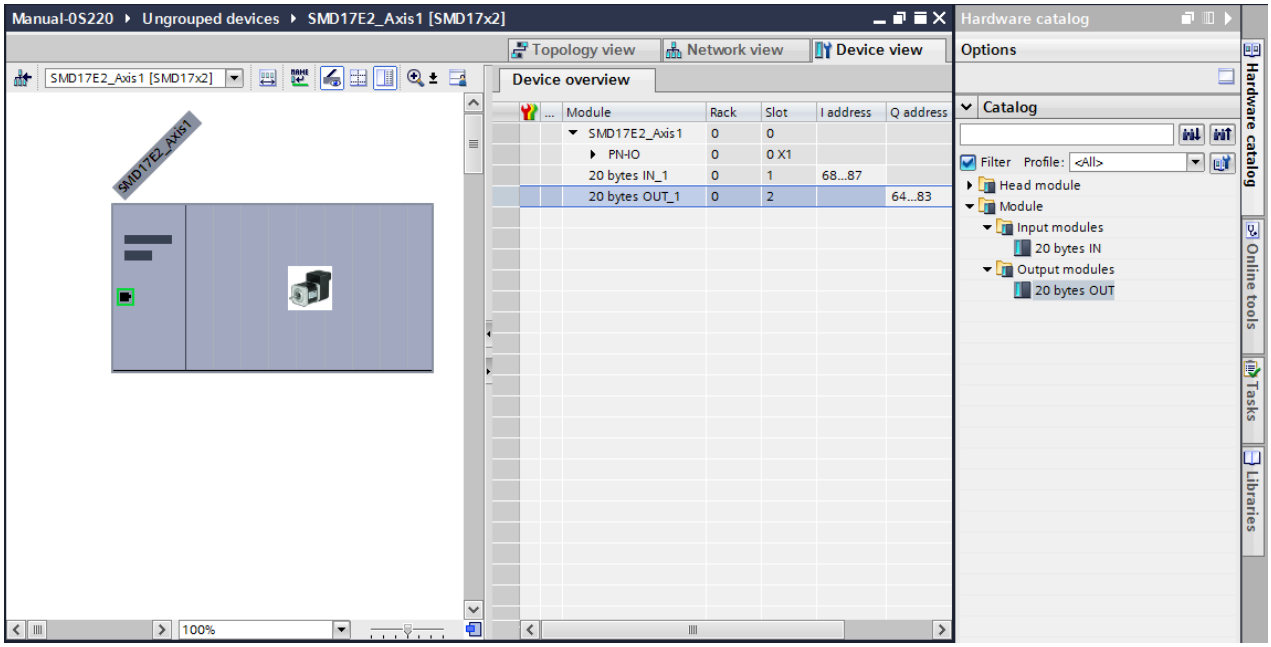


Figure T7.4 I/O Byte Mapping

7.6 Verify and Download the New Configuration

1) Continue by adding any remaining devices to your PROFINET network.
2) Compile and download the project to the CPU.

MRP Installations

At this point, the SMD17E2 is configured and ready to use. If you are using the unit in a redundant, ring based, network that uses the Media Redundancy Protocol (MRP), continue with the following instructions.

Media Redundancy Protocol (MRP) installations require that the SMD17E2 be installed in a ring topology. In these applications, both Ethernet ports are used when wiring the ring, daisy chaining from one unit in the ring to the next. The steps below covers typical software configuration that must also be completed.

7.7 Configure the SMD17E2 as an MRC

The SMD17E2 functions as a Media Redundancy Client (MRC) in an MRP network.

1) Switch to Topology view and drag the additional connections between the appropriate ports.
2) Click on the SMD17E2 icon to select it. In the Inspector window, select *Advanced options +> Media redundancy.* Use the "MRP domain:" drop down menu to select the appropriate domain. Use the "Media redundnacy role:" drop down menu to select "Client".
3) Continuing in the Inspector window, select *Advanced options +> Port 1 +> Port interconnection*. Under "Partner port:", the partner port you assigned to the port when you drew the topology is shown. If you do not know which port will be the partner port in the actual installation, you can use the drop down menu to select "Any partner".
4) If need be, repeat step 3 for Port 2 of the SMD17E2.

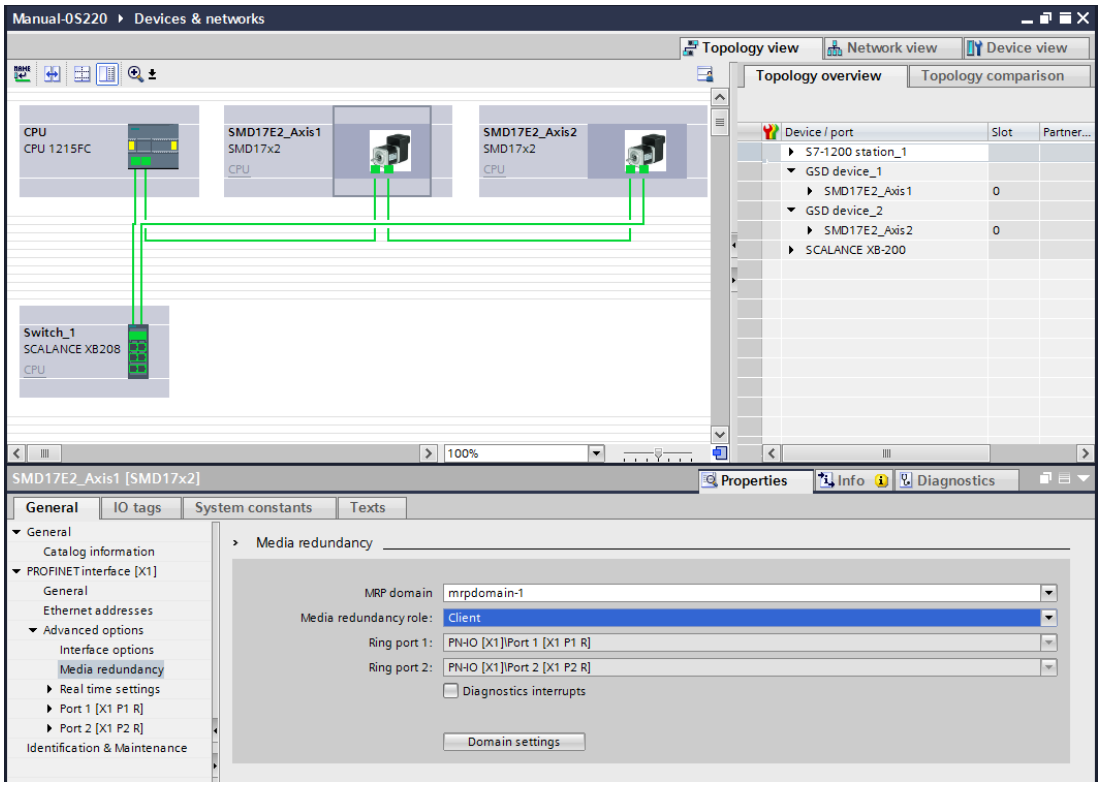


Figure T7.5 MRP Topology and Client settings

7.7 Configure the SMD17E2 as an MRC (continued)

5) Continue configuring the rest of the devices on the network before compiling the project and downloading it to the CPU.

OPTIONAL TASK A

CONFIGURE YOUR NETWORK INTERFACES

A.1 Firewall Settings

Firewalls are hardware devices or software that prevent unwanted network connections from occurring. Firewall software is present in Windows XP and above and it may prevent your computer from communicating with the SMD17E2. Configuring your firewall to allow communication with the SMD17E2 is beyond the scope of this manual.

AMCI strongly suggests temporarily disabling any firewall software while using the Net Configurator utility. You should enable the firewall once you have finished using the utility.

A.2 Disable All Unused Network Interfaces

Routing and default gateway setting on your computer can interfere with the proper operation of the Net Configurator software. The Net Configurator software uses broadcast packets to locate devices on the network, and sometimes these packets are sent out through the default gateway instead of the interface attached to the AMCI product. The easiest way to avoid this problem is to temporarily disable all network interfaces that are not attached to the SMD17E2.

NOTE This includes all wireless interfaces as well as all Bluetooth interfaces.

A.3 Configure Your Network Interface

Before you can communicate with the SMD17E2, your network interface must be on the same subnet as the driver.

NOTE The rest of this procedure assumes you are using the 192.168.0.xxx subnet. If you are not, you will have to adjust the given network addresses accordingly.

The easiest way to check the current settings for your NIC is with the 'ipconfig' command.

- For Windows 7, click on the [Start] button, and type "cmd" in the "*Search programs and files*" text box. Press [Enter] on the keyboard.
- For Windows 8 and 10, press the [Win+X] keys and select "Command Prompt" from the resulting popup. There is no need to run the command prompt as the administrator, so do not select "Command Prompt (Admin)".

A DOS like terminal will open. Type in 'ipconfig', press [Enter] on the keyboard and the computer will return the present Address, Subnet Mask, and Default Gateway for all of your network interfaces. If your present address is 192.168.0.xxx, where 'xxx' does not equal 50, and your subnet mask is 255.255.255.0, then you are ready to configure your SMD17E2. Figure A.1 shows the output of an ipconfig command that shows the "Local Area Connection 2" interface on the 192.168.0.xxx subnet.

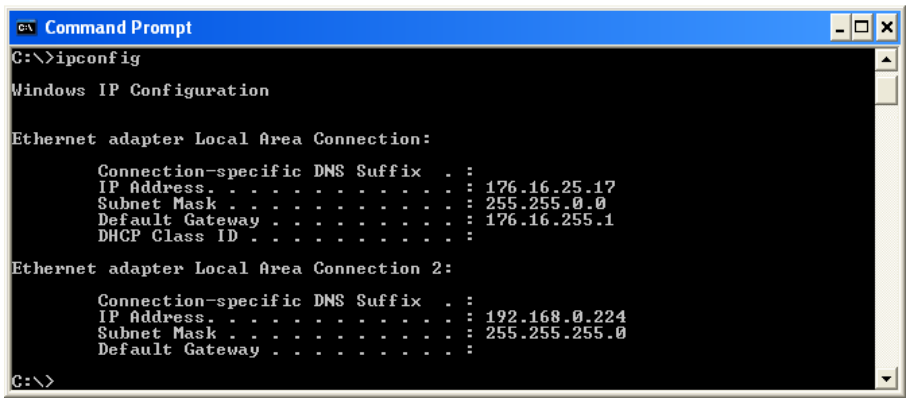


Figure A.1 ipconfig Command

A.3 Configure Your Network Interface (continued)

If your present address in not in the 192.168.0.xxx range, type in 'ncpa.cpl' at the command prompt and hit [Enter] on the keyboard.

- For Windows 7, this open the *Network Connections* window. Double click on the appropriate interface. In the window that opens, select "Internet Protocol Version 4 (TCP/IP v4)" from the list and then click on the [Properties] button.
- For Windows 8 and 10, this open the Network Connections window. Double click on the appropriate interface. In the window that opens, select "Internet Protocol Version 4 (TCP/IP v4)" from the list and then click on the [Properties] button.

Set the address and subnet mask to appropriate values. (192.168.0.1 and 255.255.255.0 will work for an SMD17E2 that has factory default settings.) The default gateway and DNS server settings can be ignored.

A.4 Test Your Network Interface

Going back to the terminal you opened in the last step, type in 'ping aaa.bbb.ccc.ddd' where 'aaa.bbb.ccc.ddd' in the IP address of the SMD17E2. The computer will ping the unit and the message "Reply from aaa.bbb.ccc.ddd: bytes=32 time<10ms TTL=128" should appear four times.

If the message "Request timed out." or "Destination host unreachable" appears, then one of four things has occurred:

- You set a new IP address, but have not yet cycled power to the SMD17E2
- You did not enter the correct address in the ping command.
- The IP address of the SMD17E2 is not set correctly.
- The SMD17E2 and the computer are not on the same subnet.

Notes

*

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