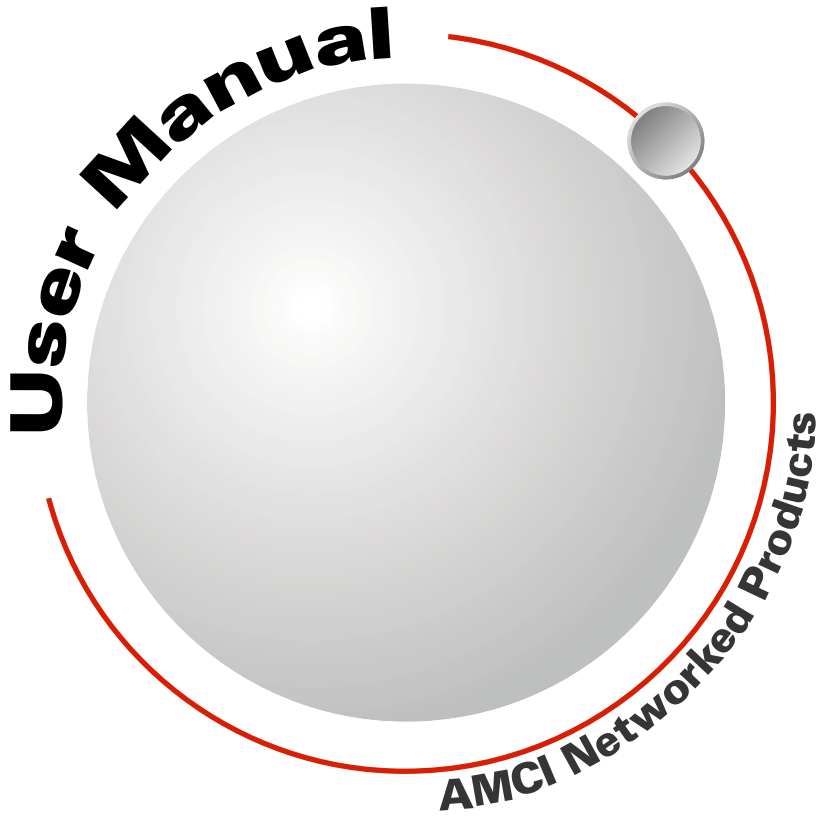
ADVANCED MICRO CONTROLS INC.

Manual #: 940-0S370

SMD17L

Integrated Stepper Indexer/Driver/Motor

with Integral IO-Link Interface



IO-Link

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) 585-1254. 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 contact an engineer on call. Please have your product model number and a description of the problem ready before you call.

Waste Electrical and Electronic Equipment (WEEE)

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

Table of Contents

Reference 2: Motion Control (continued)

Reference 3: Calculating Move Profiles

Reference 4: Homing an SMD17L

Task 1: Install the SMD17L

Task 2: IODD Configuration (RA)

Task 2: IODD Configuration (RA)

Task 3: Controlling the SMD17L (RA)

Reference 5: AOI Reference (RA)

Reference 6: Input Status Bits

Notes

About this Manual

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

Audience

This manual explains the installation and operation of the SMD17L 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 IO-Link® interface. IO-Link is a cost effective interface for incorporating sensors and actuators into an industrial controls system. Each SMD17L unit contains an M12 5-pin port that can be configured as an IO-Link Class A or Class B port. A Class A port allows the SMD17L to work with any IO-Link master device. When the SMD17L is configured for a Class B port, power for the motor can be drawn from the IO-Link master as long as the master's port is also a Class B port.

Applicable Units

This manual applies to all of the units in the SMD17L family.

Model Number	Description
SMD17L-80A-M	Size 17 motor, 80 oz-in holding torque with an integrated absolute multi-turn encoder and sealed connectors for an IP50 rating.
SMD17L-80E-M	Size 17 motor, 80 oz-in holding torque with an integrated incremental multi-turn encoder and sealed connectors for an IP50 rating.
SMD17L-80A-MS	Same as the SMD17L-80A-M with a shaft seal for an IP64 rating.
SMD17L-80E-MS	Same as the SMD17L-80E-M with a shaft seal for an IP64 rating.
SMD17L-80A-MP	Same as the SMD17L-80A-M with a shaft seal and epoxy coated motor laminations for an IP65/67 rating.
SMD17L-80E-MP	Same as the SMD17L-80E-M with a shaft seal and epoxy coated motor laminations for an IP65/67 rating.

Part Number Description

Trademark Notices

The AMCI logo is a trademark of Advanced Micro Controls Inc. "IO-Link" is a registered trademark of PROFIBUS Nutzerorganisation e.V. (PNO). "Adobe" and "Acrobat" are registered trademarks of Adobe Systems Incorporated.

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

Revision Record

This manual, 940-0S370, is the first release of this manual. It was first released September 14th, 2026.

Revision History

940-0S370: September 14th, 2026 - Initial Release

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, ***SMD17L 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	Page #	Section Description
SMD17L Specifications	11	Complete specifications for the SMD17L products.
Motion Control	25	Reference information on how the SMD17L can be used to control motion in your application.
Calculating Move Profiles	47	Reference information on calculating detailed move profiles.
Homing an SMD17L	57	Reference information on how to set the home position of the SMD17L.
Install the SMD17L	63	Task instructions covering how to install an SMD17L on a machine. Includes information on mounting, grounding, and wiring specific to the units.
IODD Configuration (RA)	73	Task instructions that cover how to add and configure an SMD17L to a Rockwell Automation host that supports the use of an IODD file.
Controlling the SMD17L (RA)	81	Task instructions for adding AMCI Add-On Instructions to your program and how to use them to control the SMD17L.
AOI Reference (RA)	95	Reference information that lists all of the AMCI Add-On Instructions, their parameters, and their enumerations.
Input Status Bits	109	Reference information that lists all of the input status bits from the SMD17L and the events that control their states.

Manual Sections

Notes

REFERENCE 1

SMD17L SPECIFICATIONS

This manual is designed to get you up and running quickly with an SMD17L product from AMCI. As such, it assumes you have some basic knowledge of stepper systems. If you are new to stepper systems, 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 SMD17L Family

The SMD17L 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. The SMD17L product line is designed to interface with an IO-Link master device that connects IO-Link to a fieldbus.

Like all IO-Link devices, the unit is configured using an IODD file. Most systems that support IO-Link devices use the IODD to create named tags and easy to use configuration screens that simplify device integration.

The IO-Link port always supplies power to the IO-Link communications subsystem. Power to the stepper indexer and driver subsystems can come from one of two sources. The source is jumper selectable. The factory default is to use an external supply. The external supply can be up to 48Vdc. The SMD17L is compatible with Class A and Class B IO-Link ports in this configuration. By changing the jumper settings, you can power the entire SMD17L from an IO-Link Class B port on your IO-Link Master. This can reduce cabling cost and complexity, but you are limited to 24Vdc.

Figure R1.1 IP50 Rated SMD17L

Each unit has a single five pin M12 port for connections to your IO-Link master, and a four pin M8 connector that supplies two DC discrete inputs and additional power connections if you do not draw indexer and driver power from the IO-Link port.

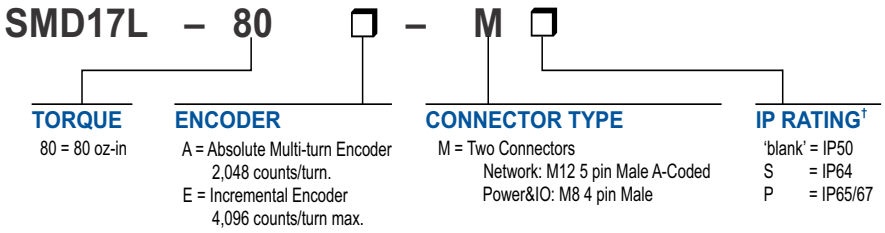
Each unit must be ordered with an incremental or absolute multi-turn encoder. This encoder gives you the additional functionality of position verification and stall detection. The absolute encoder also allows you to track machine position with power removed, eliminating the need to home the machine after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

Three different IP ratings are available in the SMD17L product line.

- IP50: Units have sealed M12 connectors for their IO-Link, power, and input connections.
- IP64: Units have a shaft seal and sealed M12 connectors for their IO-Link, power, and input connections.
- IP65/67: Units have a shaft seal and M12 connectors for their IO-Link, power, and input connections. Additionally, the units are sealed with an FDA approved epoxy.

The SMD17L Family (continued)

Part Numbering System



† All connectors must be properly installed to achieve the listed IP rating.

Figure R1.2 Part Numbering System

General Functionality

Each member of the SMD17L family has four integrated parts:

- An indexer that accepts commands over an IO-Link connection
- 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
- An incremental or absolute multi-turn encoder

The incremental or absolute multi-turn encoder allows for stall detection as well as position feedback or verification.

The availability of the IO-Link interface makes the SMD17L 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 SMD17L unit
- All motion logic is programmed in the host, eliminating the need to learn a separate motion control language
- The elimination of the separate indexer and driver lowers total system cost.

An SMD17L can be powered from a Class B IO-Link port or from an external supply. When powered by an external supply, it accepts 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 SMD17L suitable to a wide range of applications. In addition to the Motor Current setting, the Anti-Resonance, Motor Steps per Turn, and Idle Current Reduction features are also fully programmable. If you have used other stepper indexer products from AMCI, you will find programming an SMD17L to be very similar to these products.

The SMD17L 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 SMD17L 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
- Emergency Stop Input
- Start Indexer Move
- Capture Encoder Position Input
- Home Limit Switch
- Start Indexer Move
- Stop Jog or Registration Move Input
- General Purpose Input

The SMD17L Family (continued)

Motion Axis Functionality (*SynchroStep* Technology)

On host platforms that offer motion control, such as the RA ControlLogix platform, a programmed axis can be used to supply data to the SMD17L by copying position and velocity data from the axis to the registers assigned to the unit. This allows the unit to perform complex motion profiles that are programmed using your host's motion control instructions.

Encoder Functionality

All SMD17L units must be ordered with an internal incremental or absolute multi-turn encoder. The encoder is typically used for position verification and stall detection. Additionally, an input can be configured to capture the encoder value when the input makes a transition. This captured value is written to the host controller.

The motor position can be preset to the encoder position with a single command. All SMD17L units with absolute encoders allow you to preset the encoder position and save the resulting offset in Flash memory.

If you enable stall detection, the Stall_Detected bit is set in the network input data whenever the motor position and the encoder position differ by more than 45 degrees. This is most commonly used to detect if a motor stalls during a move. However, the feature can also be used to detect motion when the motor should be stopped. For example, if a mechanical break is used to hold a load against gravity, the stall detection will trigger if the motor rotates more than 45° while the mechanical break is engaged.

Incremental Encoder Option

The incremental encoder can be programmed to 1,024, 2,048, or 4,096 counts per turn. The SMD17L has an internal thirty-two bit counter associated with the encoder. The position value is reset to zero after a power cycle.

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 SMD17L uses a battery backed circuit to count zero crossings while power is removed from the rest of the device. The battery life is 10 years in the absence of power. 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.

Data Endian

The IO-Link specification requires thirty-two bit data to be transmitted in big endian format. This means that the upper sixteen bits of a thirty-two bit value are transmitted before the lower sixteen bits. This is the format used by Siemens controllers.

Rockwell Automation controllers internally use *little endian* format, which transmits the lower sixteen bits before the upper sixteen bits. However, in AMCI's experience, Rockwell Automation IO-Link masters automatically swap word and byte order, so standard IO-Link devices will work with Rockwell Automation controllers automatically.

In order to support any other controller that may not automatically swap word and byte order, a configuration parameter named "Binary Endian" is available on the SMD17L. This parameter will allow you to swap word order in the SMD17L instead of in your PLC code.

NOTE For Rockwell Automation and Siemens controllers, leave the Binary Endian parameter at its default *Big Endian* value.

Torque and Power Curves

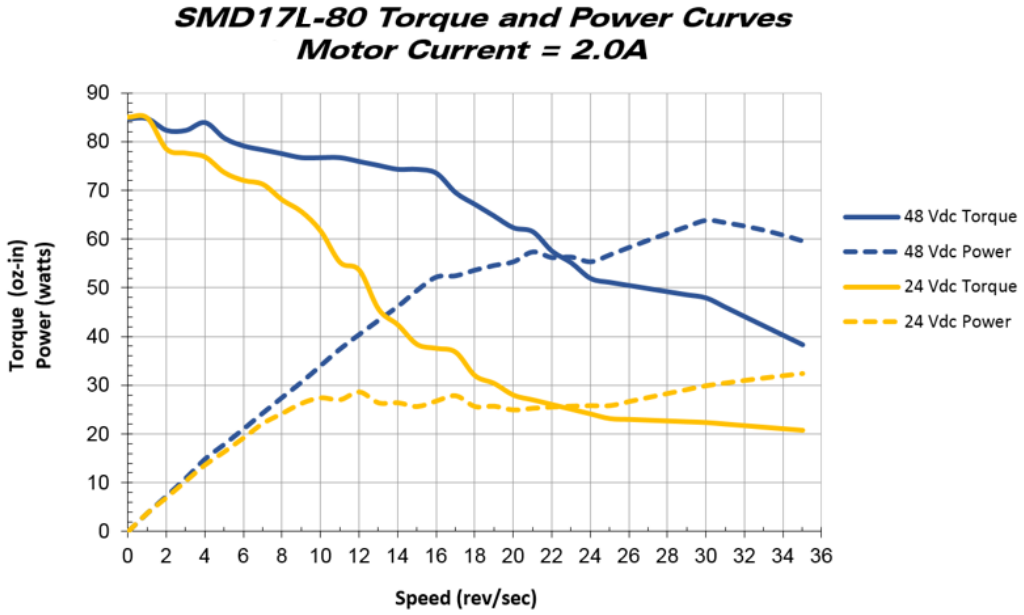


Figure R1.3 SMD17L-80 Torque and Power Curves

Conformance Markings

The SMD17L family meets the requirements for the following conformance markings when installed and operated in accordance with the instructions contained in the product documentation.

CE

- Directive 2014/30/EU of the European Parliament and of the Council (EMC), of 26 February 2014 on the harmonization of the laws of the Member States relating to electromagnetic compatibility; per EN 61800-3:2023.
- Directive 2014/35/EU of the European Parliament and of the Council (Low Voltage), of 26 February 2014 on the harmonization of the laws of the Member States relating to the making available on the market of electrical equipment designed for use within certain voltage limits; per EN 61010-1:2010.

RoHS

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

Specifications

Communication Interface

IO-Link. Single M12, A-coded, 5-pin connector. Factory default is a Class A port. Can be changed to a Class B port by changing two user accessible jumpers.

Driver Type

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

Physical Dimensions

See Outline Drawing, page 67

Weight

SMD17L-80-M12 ..... 1.1 lb. (0.50 kg)
Weight is without mating connectors

Rotor Inertia

82 g-cm^2

Maximum Shaft Loads

Radial: 6.5 lb (29 N) at center of flat on shaft
Axial: 5.6 lb (25 N)

Maximum Operating Temperature†

203°F /95°C Note that this is the internal temperature of the driver, not maximum ambient temperature. An Over Temperature fault occurs at this point and current is removed from the motor. The temperature of the motor is not directly measured.

Over Temperature Fault

Over temperature faults are detected and reported.

Inputs

Electrical Characteristics:
IN1 and IN2: Single ended sinking.

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

Opto-isolated, 1500 Vac/dc isolation

Motor Current

Programmable from 0.1 to 2.0 Arms in 0.1 A steps.

Motor Counts per Turn

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

Internal Encoder

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†

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

Class B Power .... 24 Vdc per IO-Link Specification

Ambient Operating Temperature
..... -4° to 122°F (-20° to 50°C)

Storage Temperature
..... -40° to 185°F (-40° to 85°C)

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

IP Rating ..... IP50 standard
IP64 with shaft seal
IP65/67 with epoxy coating.

Status LED's

See ***Status LED's*** section starting on page 21.

Connectors and Cables

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

Connector	AMCI Part #	Wire	Strip Length	Connection Type
IO-Link	MS-31	18 AWG max.	0.197 inches	Screw Terminals

Cable	AMCI Part #	Length
IO-Link	CNPL-5M	5 meter
Power & I/O	CNSL-5M	5 meter

† A properly designed system requires a heatsink sufficient to keep the operating temperature within prescribed limits based on ambient conditions and required power levels.

Indexer Functionality

The table below lists the functionality offered by the indexer built into the SMD17L.

Feature	Description
IO-Link	Allows easy setup and communication with a wide range of host controllers such as the latest PLC's from Allen-Bradley and Siemens.
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 SMD17L 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.
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.
SynchroStep Move	Allows the SMD17L to follow a virtual motion axis of a PLC.
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 and perform 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	The integrated encoder can be used to: ➤ verify motion when a move command is issued ➤ verify that the shaft does not rotate when it should be holding its position.

Table R1.1 Indexer Functionality

Indexer Functionality (continued)

Stall Detection with SMD17L Units

Stall Detection is one of the additional features available on an SMD17L. When Stall Detection is enabled, the SMD17L 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 SMD17L responds in the following manner:

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

Note that a move does not have to be in progress for stall detection to be useful. By enabling stall detection, the SMD17L can notify the system if the motor shaft moves more than forty-five degrees when it should be holding a position.

Driver Functionality

This table summarizes the features of the stepper motor driver portion of an SMD17L.

Feature	Benefits
RMS Current Control	RMS current control give an SMD17L 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 SMD17L 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 SMD17L sets a warning bit in the network data when the temperature of the unit approaches its safe operating threshold.
Over Temperature Protection	Protects your SMD17L 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 SMD17L always switches from RMS to peak current control when the motor is idle to prevent motor damage due to excessive heating.)

Control Inputs

The SMD17L uses a combination of network I/O bits and discrete DC inputs to control the motor.

Network I/O Bits

These bits are available in the registers assigned to the unit when it is configured in the host controller.

Position Status Bit

The SMD17L unit has a single bit in the input registers that reports the status of the motor position value. The name of this bit is Position_Invalid, and it is set the "1" when the reported motor position may not match the corresponding position on the machine. The Position_Invalid bit is always set on power up. A successful Find_Home command or a Preset_Position command will reset this bit to "0". Absolute Moves cannot be run while The Position_Invalid bit is set. Attempting one will result in a command error response from the module. All other move types can be run while this bit is set.

Motion Status Bits

The SMD17L unit has five bits in the input registers that report on the state of a move in progress. These bits are Stopped, Moving_CW, Moving_CCW, Accelerating, and Decelerating and are fully described in the following chapter.

Command Status Bits

The SMD17L unit has four bits in the input registers that report on the state of the last command issued to the module. These bits are Move_Complete, In_Hold_State, At_Home, and Command_Error. They are fully described in the following chapter.

Module Status Bits

The SMD17L unit has sixteen bits in the input registers that report on the state of the module. These bits are fully described in the following chapter.

Control Inputs (continued)

Network I/O Bits (continued)

Motion Command Bits

The following motion command bits are mutually exclusive. This means that only one of these bits can be at a logic "1" state at a time. The module will issue a command error if two or more of these bits are a logic "1" at the same time.

- Absolute Move
- Hold Move
- Find Home CW
- Jog CW†
- Run Blend CW
- Run Dwell Move
- Preset Encoder Position
- Reset Errors
- Relative Move
- Resume Move
- Find Home CCW
- Jog CCW†
- Run Blend CCW
- Preset Current Position
- Immediate Stop

† The two Jog bits are also used to trigger Registration Moves and SychroStep moves.

Blend Move Programming Bits

There are two bits in the network output data and two bits in the network input data that are used to program a Blend Move. The two output bits are named Program_Assembled and Read_Assembled_Data. The two input bits are named In_Assembled_Mode and Wait_For_Assembled_Segment. Their use is described in the *Assembled Move Programming* section of the following chapter, starting on page 43.

Available Discrete DC Inputs

The SMD17L unit has two discrete DC inputs that accept 3.5 to 27Vdc signals. (5 to 24Vdc nominal) How your SMD17L 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. These inputs are open collector sinking and share their common connection.

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. The SMD17L units allow 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 SMD17L that will cause the unit to seek this sensor. How the unit actually finds the home sensor is described in the reference chapter *Homing an SMD17L* starting on page 57.

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.

Control Inputs (continued)

Available Discrete DC Inputs (continued)

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 an inactive-to-active state transition. This allows an SMD17L to run critically timed moves that cannot be reliably started from the network due to data transfer lags.

If the integrated encoder is enabled, an inactive-to-active transition on a Start Indexer Move Input will capture the present encoder position and place it in the Trapped Encoder Position register of the network input data.

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.

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 integrated encoder is enabled, the encoder position data will be captured when the DC input makes an inactive-to-active transition. 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 inactive-to-active transition of a DC input, configure the input 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, the SMD17L units 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 more of the inputs, you can configure the unused inputs as General Purpose Inputs. The inputs are not used by the SMD17L units, but their on/off state is reported in the network data and is available to your host controller.

Status LED's

Each SMD17L has two status LED's that show module and network status. As shown in figure R1.4, these LED's are located on the rear cover.

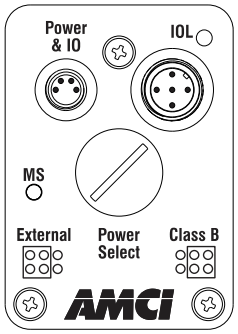


Figure R1.4 Rear Cover Status LED's

Module Status (MS) LED

The Module Status LED is a bi-color red/green LED. The state of the LED depends on the state of the IO-Link adapter module.

LED State	IO-Link Definition
Off	No Power
Alternating Red/Green	Initializing: Power up Self-Test
	Communications failure. There is a communications error between the main processor and the communications co-processor within the unit. You must cycle power to the SMD17L to attempt to clear this fault.
Flashing Green	Initializing: Waiting for valid physical connection.
	Successful write to flash memory. Power must be cycled to the unit before additional commands can be written to it.
Steady Green	Drive and Network are operational.
Flashing Red	Failed write to flash memory. Cycle power to the unit to clear this fault.
Steady Red	Overtemperature fault. Remove power from the unit and allow it to cool to clear the fault.

Table R1.3 Module Status LED States

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.

Status LED's (continued)

IO-Link Status (IOL) LED

The Network Status LED is a bi-color red/green LED.

LED State	IO-Link Definition
Off	**No Power, or Startup/Inactive:** Power is not applied to the device or the device is in its startup sequence, waiting for the wake pulse, or is inactive.
Steady Green	**Normal Operation:** The device is fully initialized and operates normally.
Flashing Green (1 Hz)	**Configuration Mode:** The device is communicating with the host but not yet in its normal operating state. The device is waiting for, or applying, configuration data.)
Steady Red	**Critical Alarm:** A critical device event or alarm is currently active. The most common cause is when the SMD17L is not receiving main power from an external supply or the Class B port.
Flashing Red (>1Hz)	**Error/Timeout:** Communication with the master has timed out, or the device is reporting a failure state.

Table R1.4 Network Status LED States

SMD17L Connectors

Motor Power Source Selection Jumpers

Figure R1.6 below shows the PG9 cover removed and the location of the motor power source selection jumpers. The figure shows the jumpers in the factory default location. This selects an external supply that is brought in on the M8 Power & IO connector. If you change the jumpers to select Class B, then motor power must be supplied from a Class B IO-Link port.

NOTE Both jumpers must be moved when changing the motor power source selection. The graphic on the back of the unit and in figure R1.6 below shows the proper location of the jumpers for each selection.

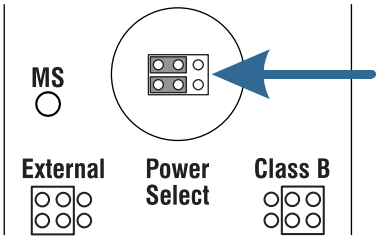


Figure R1.6 Power Source Selection Jumpers

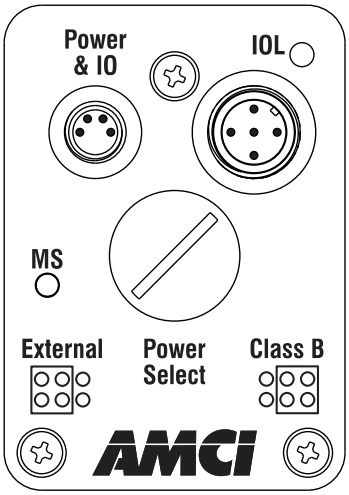
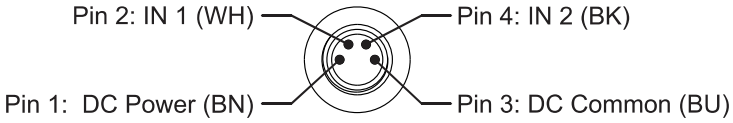


Figure R1.5 SMD17L Connector Locations

SMD17L Connectors (continued)

Power & Input Connector

As shown in figure R1.5, the Power and IO Connector is located on the back of the unit. All digital input and external power supply connections are made at this connector. The connector is a standard four pin A-coded M8 connector that is rated to IP67 when the mate is properly attached. Figure R1.7 shows the pinout of the connector when viewed from the back of the SMD17L.



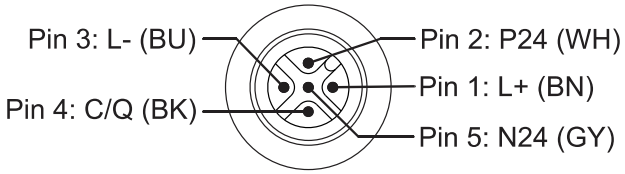
Power and IO
M8 A-Coded
Figure R1.7 M8 Input Connector

Digital inputs are single ended and referenced to the DC Common pin. (Sinking inputs.) They accept a nominal 5 Vdc to 24 Vdc signal without the need of an external current limiting resistor.

The DC Power pin is only used if the power source selection jumpers are set to “External”. The external supply can be up to 48Vdc.

IO-Link Connector

Figure R1.5 also show the placement of the sealed M12 IO-Link connector, while figure R1.8 shows the connector pinout when viewed from the back of the SMD17L. The connector is a standard five pin A-coded male M12 connector. The connection is rated to IP67 when the mate is properly attached.



IOL
M12 A-Coded
Figure R1.8 IO-Link Connector Pinout

The connector pinout is specified by the IO-Link Specification, and the figure uses specified names and wire colors. The L± pins are always used to supply power to the IO-Link communication subsystem. The P24 and N24 pins are used to supply the SMD17L motor and driver subsystem when the power source select jumpers are set to “Class B” and the unit is connected to a Class B IO-Link port.

Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17L, provided that the manufacturer follows the connector and IO-Link standards. AMCI has reviewed the following connectors and cordsets for compatibility with the SMD17L.

Connectors

AMCI #	Binder #	Description
MS-31	99-0436-12-05	Mating connector for IO-Link 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

IO-Link Cordset

AMCI Part #	Description
CNPL-5M	5-position, 22 AWG. Connector: Straight M12, A-coded, Female to 2 inch flying leads, 0.28" stripped. Cable length: 5 m

Table R1.6 Ethernet Cordset

Power Cordset

AMCI Part #	Description
CNSL-5M	4-position, 24 AWG. Connector: Straight M8, A-coded, Female to 2 inch flying leads, 0.28" stripped. Cable length: 5 m

Table R1.7 Power Cordset

When a move command is sent to an SMD17L, 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 SMD17L, 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 range is -2,147,483,648 to +2,147,483,647, (or 0 to 4,294,967,295 if the integer is unsigned). Note that in continuous rotation applications, if the position overflows, the value will go from one end of the range directly to the other and continue to count from there. For example, for increasing counts, the value will jump from +2,147,483,647 directly to -2,147,483,648 and the count will continue to increase from there.

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 SMD17L 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 SMD17L moves use relative coordinates.

- The range of values for the Target Position when it is treated as an offset is ±8,388,607 counts. Positive offsets will result in clockwise moves, while negative offsets result in counter-clockwise moves.
- The Motor Position value reported back to the host can exceed ±8,388,607 counts. 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 counter-clockwise if the Target Position is less than the Current Position.
- The Motor Position value reported back to the host can exceed ±8,388,607 counts. However, you cannot move beyond ±8,388,607 counts with an Absolute Move. The only way to move beyond ±8,388,607 counts is with relative moves or jog commands.

Position Status Bit

The SMD17L units have one position status bit, the Position_Invalid bit. This bit is set when the reported motor position may not correspond to the actual machine position. The reported motor position is a bi-directional counter that tracks the number, and direction, of motor steps. The actual machine position is the physical position of the axis on the machine.

Absolute Moves cannot be run while the Position_Invalid bit is set. All other move types can be run while the Position_Invalid bit is set.

There are two ways to reset the Position_Invalid bit.

1) Use a Find_Home command. If the command completes successfully, the motor position will be reset to zero and the Position_Invalid bit will be reset to zero.
2) Use a Preset_Position command. Once the command completes, the motor position will be set to the commanded value and the Position_Invalid bit will be reset to zero.

Five operations will always set the Position_Invalid bit.

1) Cycle power to the SMD17L.
2) Write Configuration data to the SMD17L.
3) Issue an Immediate_Stop command from the host. (Even when motion is not occurring.)
4) Activate an Emergency Stop input.
5) Activate a CW or CCW end limit switch while motion is occurring.

Motion Status Bits

The SMD17L units have five motion status bits:

- **Moving_CW:** Set while the SMD17L unit is moving clockwise.
- **Moving_CCW:** Set while the SMD17L unit is moving counter-clockwise.
- **Accelerating:** Set while the move is in its acceleration phase.
- **Decelerating:** Set while the move is in its deceleration phase.
- **Stopped:** Set when motion is not occurring.

Command Status Bits

The SMD17L units have four command status bits:

- **Move_Complete:** Set at the end of any Absolute, Relative, Jog, Registration, or Blend move.
- **In_Hold_State:** Absolute, Relative and Jog Moves can be interrupted after they have started. The move decelerates to a stop and is placed in a "Hold State". This bit is on while the move is in its Hold State. Once it its Hold State, a move can be restarted or a new move can be issued.
- **At_Home:** This bit is set to "1" when a homing command completes successfully. It is reset to "0" when the next command is accepted.
- **Command_Error:** This bit is set to "1" when there is an error in a command.

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 SMD17L 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 47.

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.

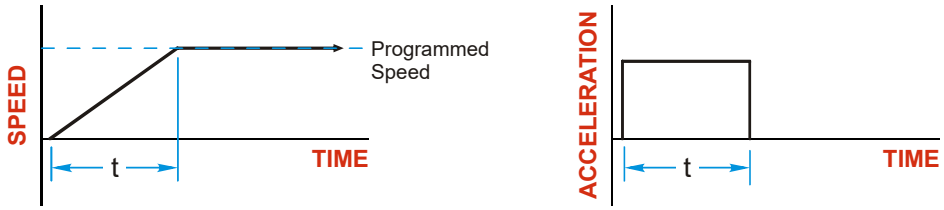


Figure R2.1 Linear Acceleration

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.

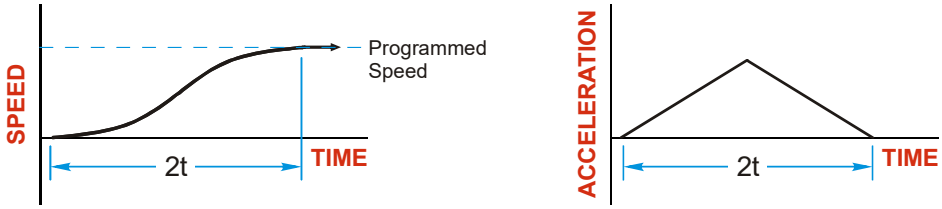


Figure R2.2 Triangular S-Curve Acceleration

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.

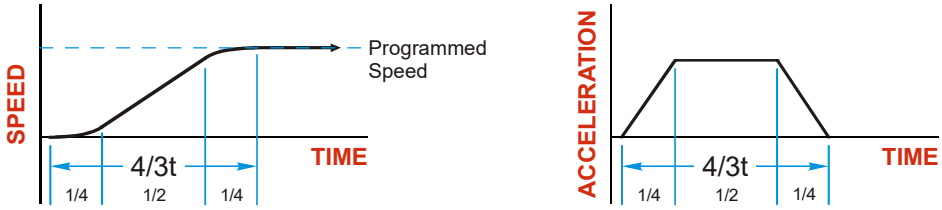


Figure R2.3 Trapezoidal S-Curve Acceleration

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 50, for a methods of calculating the Acceleration Jerk parameter.

How to Control Moves in Progress

There are two ways of stopping a move in progress.

- **Controlled Stops:** The motor decelerates at the move's programmed deceleration value until it reaches the configured Starting Speed. The move stops at this point. If the Position_Invalid bit is reset to "0" at the end of the move, the motor position value is still considered valid. The machine does not need to be homed again before Absolute Moves can be run.
- **Immediate Stops:** The motor stops immediately without deceleration. Because it is possible for the inertia of the load to pull the motor beyond the stopping point, the motor position value is considered invalid after an Immediate Stop. The Position_Invalid bit will be set to "1" and the machine must be homed again before Absolute Moves can be run.

The SMD17L units also have the ability to "hold" a relative and absolute move. The move is brought to a controlled stop, and the move is placed in a Hold State. The move can later be resumed and it will run to its completion. One example of the use of the Hold Move feature is on an axis that runs until there is an out of material condition. After the material is replenished, the axis is restarted.

Note that a held move does not have to be resumed. If a new move is written to the unit while the previous move is in its held state, the previous move is canceled and the new move begins from the current position.

Available Inputs

The table below shows the available DC input types and available network bits that affect the available moves. The complete table is shown here for easy reference. The appropriate rows are repeated in the sections below that explain each move in detail.

Note that the SMD17L units have two physical digital inputs, each of which can be programmed for one of seven different functions. One of these input functions is General Purpose Input. An input with this function type has its state reported over the network, but has no effect on moves.

	DC Digital Input Types							Backplane Bits			
	General Purpose	CW L.S.	CCW L.S.	Home L.S.	E-Stop	Stop Jog or Reg	Start Indexed	Indexed_Cmd	Hold Move	Resume Move	Immediate Stop
Absolute Move		1	1		2		3,4	3	5	6	2
Relative Move		1	1		2		3,4	3	5	6	2
CW Find Home		7	8	9	2			3	10		2
CCW Find Home		8	7	9	2			3	10		2
Jog_CW Move		11	12		2	13	3,4	3	5	6	2
Jog_CCW Move		12	11		2	13	3,4	3	5	6	2
CW Registration Move		11	12		2	14	3,4	3	10		2
CCW Registration Move		12	11		2	14	3,4	3	10		2
CW Blend Move		1	1		2		3,4	3	10		2
CCW Blend Move		1	1		2		3,4	3	10		2
Dwell Move		1	1		2		3,4	3	10		2
SynchroStep Move		1	1		2				10		2

Table R2.1 Control Inputs

See the numbered notes on the following page.
A blank cell means that the state of the input as no effect on the move.

How to Control Moves in Progress (continued)

Available Inputs (continued)

1) An active limit switch input will immediately stop all motion, and prevent further motion in that direction.
2) An inactive-to-active transition on an Emergency Stop Input, or an active Immediate_Stop command will immediately stop all motion. The Position_Invalid bit will be set if motion was occurring when either of these conditions became true.
3) If the Indexed_Command bit is set when the motion command is written to the unit, the move will not start until there is an inactive-to-active transition on the Start Indexed input.
4) If the encoder is enabled, a transition on the Start Indexed input will capture the encoder position and copy it to the Captured Encoder Position registers.
5) An active Hold command will start to decelerate the move. The move will stop when the configured Starting Speed is reached and the move will enter its Hold State.
6) An active Resume command will resume a move if it is in its hold state. Activating the Resume command when a move is not in its hold state will generate a Command Error response from the unit. If the move is running at this time, the move will continue.
7) An inactive-to-active transition on this limit switch input will immediately stop all motion. The unit will wait for two seconds, and then begin motion in the opposite direction while searching for the Home Input.
8) An inactive-to-active transition on this limit switch input will immediately stop all motion. The module will set the Position_Invalid and Input_Error bits and abort the Find_Home command.
9) Transitions on a Home Limit Switch input type will cause the controller to finish its homing sequence. Detailed explanations of the homing sequences can be found in the ***Homing an SMD17L*** chapter, starting on page 57.
10) Find_Home, Registration, Blend, Dwell, and SynchroStep moves cannot be held. If the Hold command is triggered during one of these moves, the unit will immediately begin to decelerate the move and bring the move to a controlled stop. The move is not put into a hold state.
11) Activating a CW or CCW limit during a Jog_CW or a CW Registration Move will bring the move to an immediately stop. If a CW limit is active when a Jog_CW or a CW Registration Move is started, the unit will set the Input_Error bit. If a CCW limit is active when a Jog_CW or a CW Registration Move is started, the move runs without an error.
12) Activating a CW or CCW limit during a Jog_CCW or a CCW Registration Move will bring the move to an immediately stop. If a CCW limit is active when a Jog_CCW or a CCW Registration Move is started, the unit will set the Input_Error bit. If a CW limit is active when a Jog_CCW or a CCW Registration Move is started, the move runs without an error.
13) An inactive-to-active transition on a Stop Jog or Registration Move input will start to decelerate the move. The move will stop when the configured starting speed is reached. If the encoder is enabled, the transition will also capture the encoder position and copy it to the Captured Encoder Position registers.
14) An inactive-to-active transition on a Stop Jog or Registration Move input will cause the unit to start to output the programmed number of steps at the end move. If the encoder is enabled, the transition will also capture the encoder position and copy it to the Captured Encoder Position registers. Note that this input is ignored until the programmed Minimum Registration Move Distance is exceeded.

A Simple Move

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

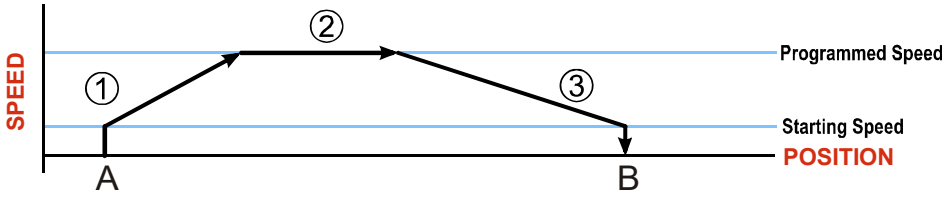


Figure R2.4 A Trapezoidal Profile

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 SMD17L.
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.

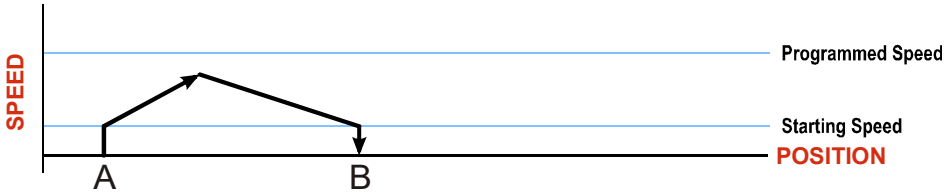


Figure R2.5 A Triangular Profile

Basic Move Types

Absolute Move

Absolute Moves move from the Current Position (A) to a given position (B). (The SMD17L 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.

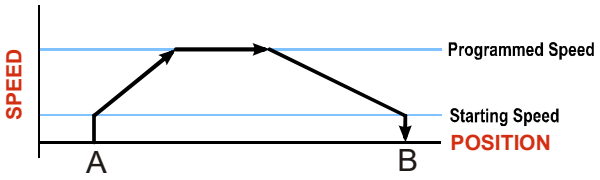


Figure R2.6 Absolute Move

NOTE

1) The *Home Position* of the machine must be set before running an Absolute Move. See the reference section, ***Homing an SMD17L***, which starts on page 57, for information on homing the machine.
2) 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.

Motion Status Bits

The motion status bits function as described in the ***Motion Status Bits*** section on page 27.

Control Inputs

	DC Digital Input Types							Backplane Bits			
	General Purpose	CW L.S.	CCW L.S.	Home L.S.	E-Stop	Stop Jog or Reg	Start Indexed	Indexed_Cmd	Hold Move	Resume Move	Immediate Stop
Absolute Move		1	1		2		3,4	3	5	6	2

Table R2.2 Absolute Move Control Inputs

1) An active limit switch input will immediately stop all motion, and prevent further motion in that direction.
2) An inactive-to-active transition on an Emergency Stop Input, or an active Immediate_Stop command will immediately stop all motion. The Position_Invalid bit will be set if motion was occurring when either of these conditions became true.
3) If the Indexed_Command bit is set when the motion command is written to the unit, the move will not start until there is an inactive-to-active transition on the Start Indexed input.
4) If the encoder is enabled, a transition on the Start Indexed input will capture the encoder position and copy it to the Captured Encoder Position registers.
5) An active Hold command will start to decelerate the move. The move will stop when the configured Starting Speed is reached and the move will enter its Hold State.
6) An active Resume command will resume an absolute or relative move if it is in its hold state. Activating the Resume command when a move is not in its hold state will generate a Command Error response from the unit. If the move is running at this time, the move will continue.

Basic Move Types (continued)

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.

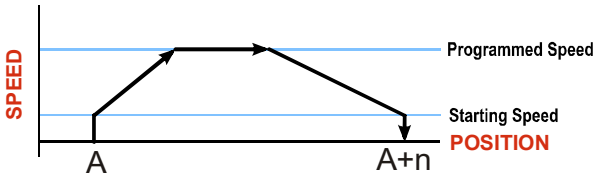


Figure R2.7 Relative Move

NOTE

1) You do not have to preset the position or home the machine before you can use a Relative Move. 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.

Motion Status Bits

The motion status bits function as described in the *Motion Status Bits* section on page 27.

Control Inputs

	DC Digital Input Types							Backplane Bits			
	General Purpose	CW L.S.	CCW L.S.	Home L.S.	E-Stop	Stop Jog or Reg	Start Indexed	Indexed_Cmd	Hold Move	Resume Move	Immediate Stop
Relative Move		1	1		2		3,4	3	5	6	2

Table R2.3 Relative Move Control Inputs

1) An active limit switch input will immediately stop all motion, and prevent further motion in that direction.
2) An inactive-to-active transition on an Emergency Stop Input, or an active Immediate_Stop command will immediately stop all motion. The Position_Invalid bit will be set if motion was occurring when either of these conditions became true.
3) If the Indexed_Command bit is set when the motion command is written to the unit, the move will not start until there is an inactive-to-active transition on the Start Indexed input.
4) If the encoder is enabled, a transition on the Start Indexed input will capture the encoder position and copy it to the Captured Encoder Position registers.
5) An active Hold command will start to decelerate the move. The move will stop when the configured Starting Speed is reached and the move will enter its Hold State.
6) An active Resume command will resume an absolute or relative move if it is in its hold state. Activating the Resume command when a move is not in its hold state will generate a Command Error response from the unit. If the move is running at this time, the move will continue.

Basic Move Types (continued)

Jog Moves

Jog Moves move in the programmed direction as long as the command is active. Two commands are available. The Jog_CW move will rotate the shaft in the clockwise direction and increase the motor position count while the Jog_CCW move will rotate the shaft in the counter-clockwise direction and 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 Jog Moves and Registration Moves than all other move types. Like all moves, activating a CW or CCW limit *while a Jog move is running* will bring the move to an Immediate Stop. Unlike other moves, *a Jog 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 Jog_CW 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 SMD17L 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 and the Invalid_Jog_Change bit will be set.

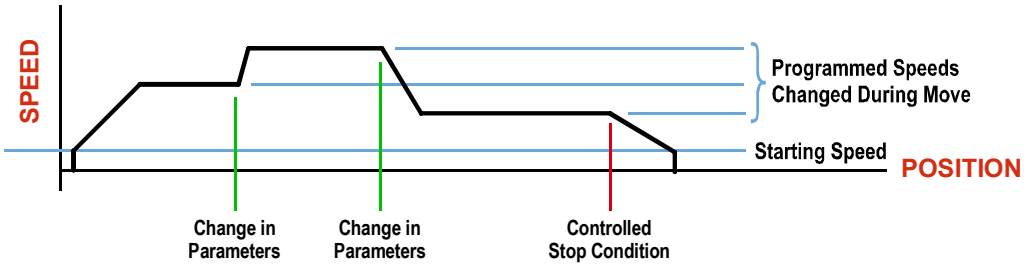


Figure R2.8 Jog Move

Basic Move Types (continued)

Jog Moves (continued)

Motion Status Bits

The motion status bits function as described in the *Motion Status Bits* section on page 27.

Control Inputs

	DC Digital Input Types							Backplane Bits			
	General Purpose	CW L.S.	CCW L.S.	Home L.S.	E-Stop	Stop Jog or Reg	Start Indexed	Indexed_Cmd	Hold Move	Resume Move	Immediate Stop
Jog_CW Move		11	12		2	13	3,4	3	5	6	2
Jog_CCW Move		12	11		2	13	3,4	3	5	6	2

Table R2.4 Jog Move Control Inputs

A blank cell means that the state of the input as no effect on the move.

2) An inactive-to-active transition on an Emergency Stop Input, or an active Immediate_Stop command will immediately stop all motion. The Position_Invalid bit will be set if motion was occurring when either of these conditions became true.

3) If the Indexed_Command bit is set when the motion command is written to the unit, the move will not start until there is an inactive-to-active transition on the Start Indexed input.

4) If the encoder is enabled, a transition on the Start Indexed input will capture the encoder position and copy it to the Captured Encoder Position registers.

5) An active Hold command will start to decelerate the move. The move will stop when the configured Starting Speed is reached and the move will enter its Hold State.

6) An active Resume command will resume a move if it is in its hold state. Activating the Resume command when a move is not in its hold state will generate a Command Error response from the unit. If the move is running at this time, the move will continue.

11) Activating a CW or CCW limit during a Jog_CW or a CW Registration Move will bring the move to an immediately stop. If a CW limit is active when a Jog_CW or a CW Registration Move is started, the unit will set the Input_Error bit. If a CCW limit is active when a Jog_CW or a CW Registration Move is started, the move runs without an error.

12) Activating a CW or CCW limit during a Jog_CCW or a CCW Registration Move will bring the move to an immediately stop. If a CCW limit is active when a Jog_CCW or a CCW Registration Move is started, the unit will set the Input_Error bit. If a CW limit is active when a Jog_CCW or a CCW Registration Move is started, the move runs without an error.

13) An inactive-to-active transition on a Stop Jog or Registration Move input will start to decelerate the move. The move will stop when the configured starting speed is reached. If the encoder is enabled, the transition will also capture the encoder position and copy it to the Captured Encoder Position registers.

Basic Move Types (continued)

Registration Moves

Similar to a Jog Move, a Registration Move will travel in the programmed direction as long as the command is active. When the command terminates under Controlled Stop conditions, the SMD17L 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.

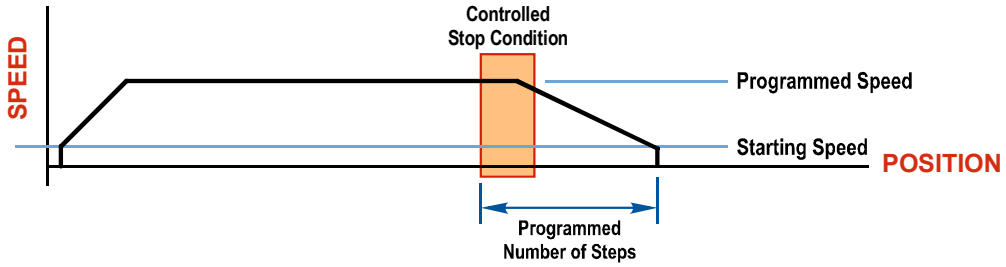


Figure R2.9 Registration Move

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 SMD17L 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.

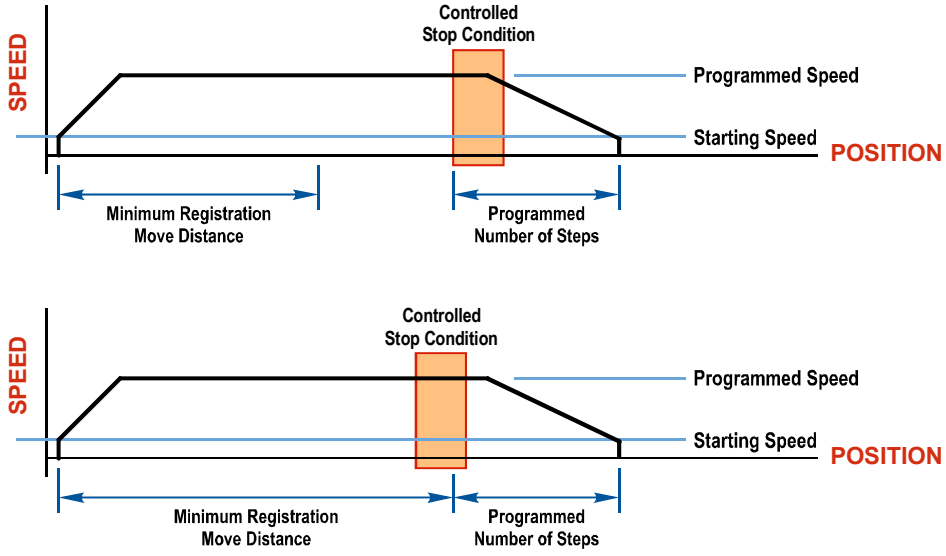


Figure R2.10 Min. Registration Move Distance

Basic Move Types (continued)

Registration Moves (continued)

Motion Status Bits

The motion status bits function as described in the *Motion Status Bits* section on page 27.

Control Inputs

	DC Digital Input Types							Backplane Bits			
	General Purpose	CW L.S.	CCW L.S.	Home L.S.	E-Stop	Stop Jog or Reg	Start Indexed	Indexed_Cmd	Hold Move	Resume Move	Immediate Stop
CW Registration Move		11	12		2	14	3,4	3	10		2
CCW Registration Move		12	11		2	14	3,4	3	10		2

Table R2.5 Registration Move Control Inputs

A blank cell means that the state of the input as no effect on the move.

2) An inactive-to-active transition on an Emergency Stop Input, or an active Immediate_Stop command will immediately stop all motion. The Position_Invalid bit will be set if motion was occurring when either of these conditions became true.

3) If the Indexed_Command bit is set when the motion command is written to the unit, the move will not start until there is an inactive-to-active transition on the Start Indexed input.

4) If the encoder is enabled, a transition on the Start Indexed input will capture the encoder position and copy it to the Captured Encoder Position registers.

10) Find_Home, Registration, Blend, Dwell, and SynchroStep moves cannot be held. If the Hold command is triggered during one of these moves, the unit will immediately begin to decelerate the move and bring the move to a controlled stop. The move is not put into a hold state.

11) Activating a CW or CCW limit during a Jog_CW or a CW Registration Move will bring the move to an immediately stop. If a CW limit is active when a Jog_CW or a CW Registration Move is started, the unit will set the Input_Error bit. If a CCW limit is active when a Jog_CW or a CW Registration Move is started, the move runs without an error.

12) Activating a CW or CCW limit during a Jog_CCW or a CCW Registration Move will bring the move to an immediately stop. If a CCW limit is active when a Jog_CCW or a CCW Registration Move is started, the unit will set the Input_Error bit. If a CW limit is active when a Jog_CCW or a CCW Registration Move is started, the move runs without an error.

14) An inactive-to-active transition on a Stop Jog or Registration Move input will cause the unit to start to output the programmed number of steps at the end move. If the encoder is enabled, the transition will also capture the encoder position and copy it to the Captured Encoder Position registers. Note that this input is ignored until the programmed Minimum Registration Move Distance is exceeded.

Basic Move Types (continued)

SynchroStep (Virtual Axis) Moves

On controllers that support motion axis programming, such as the Rockwell Automation ControlLogix platforms, an SMD17L 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 SMD17L 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 is closed with respect to the internal motor position.
- A proportional constant is available to control the sharpness of the control. AMCI suggests a starting value of 1 or 2 to control the loop.
- A network delay value that can be used to offset some of the network delays incurred when communicating with the SMD17L. 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 motor axis position rolls over and returns to zero. On the SMD23L units, this parameter has a range of 21 to 65,535.

NOTE When using the SMD17L as an axis follower, it is best to run the virtual axis as a high priority event driven periodic task.

NOTE When using the SMD17L 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 SMD17L 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 SMD17L 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. 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.

NOTE Blended Moves offer you an alternative to using a linear motion axis 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 an SMD17L 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 40.

Basic Move Types (continued)

SynchroStep (Virtual Axis) Moves (continued)

Motion Status Bits

The motion status bits function as described in the *Motion Status Bits* section on page 27.

Control Inputs

	DC Digital Input Types							Backplane Bits			
	General Purpose	CW L.S.	CCW L.S.	Home L.S.	E-Stop	Stop Jog or Reg	Start Indexed	Indexed_Cmd	Hold Move	Resume Move	Immediate Stop
SynchroStep Move		1	1		2				10		2

Table R2.6 SynchroStep Move Control Inputs

A blank cell means that the state of the input as no effect on the move.

1) An active limit switch input will immediately stop all motion, and prevent further motion in that direction.
2) An inactive-to-active transition on an Emergency Stop Input, or an active Immediate_Stop command will immediately stop all motion. The Position_Invalid bit will be set if motion was occurring when either of these conditions became true.

10) Find_Home, Registration, Blend, Dwell, and SynchroStep moves cannot be held. If the Hold command is triggered during one of these moves, the unit will immediately begin to decelerate the move and bring the move to a controlled stop. The move is not put into a hold state.

Assembled Moves

Assembled Moves are an option when the move profile is well defined before the move begins. Consider using this functionality for simple profiles. An Assembled Move is programmed into the SMD17L before it is run and the unit precisely controls position and velocity throughout the entire move.

All of the moves explained so far must be run individually to their completion or must be stopped before another move can begin. The SMD17L also gives you the ability to pre-assemble a more complex profile from a series of relative moves. This profile is then run with a single command. Each Assembled Move can consist of 2 to 16 segments.

Two types of Assembled Moves exist in an SMD17L:

- **Blend Move -** A Blend Move gives you the ability to string multiple relative moves together that all run in the same direction. (All programmed number of steps in each relative move are positive.) 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 during a Blend Move.
- **Dwell Move -** A Dwell Move gives you the ability to string multiple relative moves together, and the SMD17L unit 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. (The programmed number of steps in each relative move can be positive or negative.)

Assembled Moves (continued)

Blend Move

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

1) Each segment of the Blend Move must be written to the unit before the move can be initiated.
   - ➤ The SMD17L units support 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 Moves*** which is explained starting on page 41.
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.

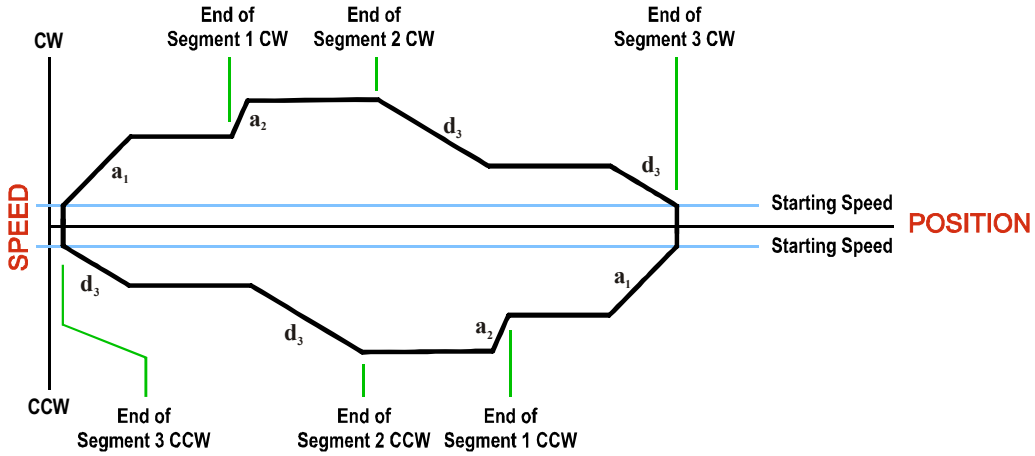


Figure R2.11 Blend Move

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 SMD17L 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 44, 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 the SMD17L is communicating with the host PLC.

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.

Dwell Moves

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 SMD17L 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.

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 can accomplish this Dwell Move with a series of six relative moves that are sent down to the SMD17L 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.

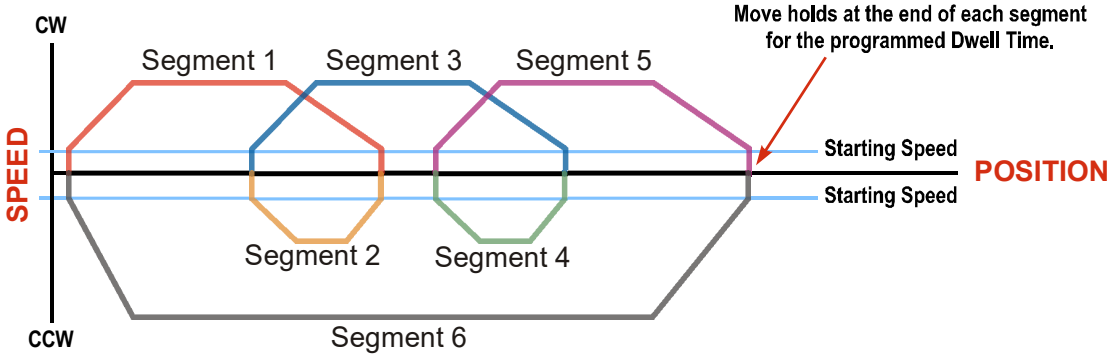


Figure R2.12 Dwell Move

Assembled Moves (continued)

Dwell Move (continued)

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 SMD17L 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 44, 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 the SMD17L is communicating with the host PLC.

Motion Status Bits

The motion status bits function as described in the ***Motion Status Bits*** section on page 27.

Control Inputs

	DC Digital Input Types							Backplane Bits			
	General Purpose	CW L.S.	CCW L.S.	Home L.S.	E-Stop	Stop Jog or Reg	Start Indexed	Indexed_Cmd	Hold Move	Resume Move	Immediate Stop
CW Blend Move		1	1		2		3,4	3	10		2
CCW Blend Move		1	1		2		3,4	3	10		2
Dwell Move		1	1		2		3,4	3	10		2

Table R2.7 Blend and Dwell Control Inputs

See the numbered notes on the following page.
A blank cell means that the state of the input as no effect on the move.

1) An active limit switch input will immediately stop all motion, and prevent further motion in that direction.
2) An inactive-to-active transition on an Emergency Stop Input, or an active Immediate_Stop command will immediately stop all motion. The Position_Invalid bit will be set if motion was occurring when either of these conditions became true.
3) If the Indexed_Command bit is set when the motion command is written to the unit, the move will not start until there is an inactive-to-active transition on the Start Indexed input.
4) If the encoder is enabled, a transition on the Start Indexed input will capture the encoder position and copy it to the Captured Encoder Position registers.
10) Find_Home, Registration, Blend, Dwell, and SynchroStep moves cannot be held. If the Hold command is triggered during one of these moves, the unit will immediately begin to decelerate the move and bring the move to a controlled stop. The move is not put into a hold state.

Assembled Moves (continued)

Assembled Move Programming

All of the segments in a Blend or Dwell Move must be written to the SMD17L before the move can be run. Segment programming is controlled with two bits in the Network Output Data and two bits in the Network Input Data. Blend and Dwell Moves are programmed in exactly the same way. When you start the move, a bit in the command data determines which type of Assembled Move is run. In the case of a Blend Move, the signs of the segment's Target Positions are ignored and all segments are run in the same direction. In the case of a Dwell Move, the signs of the segment's Target Positions determine the direction of the segment. For Dwell Moves, the Dwell Time is sent to the SMD17L as part of the command.

Control Bits – Output Data

- **Program_Assembled bit –** Set this bit to tell the SMD17L 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 SMD17L will also set the *Wait_For_Assembled_Segment* bit to signify that it is ready for the first segment.
- **Read_Assembled_Data bit –** Set this bit to tell the SMD17L that the data for the next segment is available in the remaining data words.

Control Bits – Input Data

- **In_Assembled_Mode bit –** The SMD17L 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 *Read_Assembled_Data* and *Wait_For_Assembled_Segment* bits.
- **Wait_For_Assembled_Segment bit –** The SMD17L will set this bit to signal the host that it 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 SMD17L responds by setting both the *In_Assembled_Mode* and *Wait_For_Assembled_Segment* bits in the Network Input Data.
3) When the host detects that the *Wait_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 SMD17L checks the data, and when finished, resets the *Wait_For_Assembled_Segment* bit. If an error is detected, it also sets the *Command_Error* bit.
5) When the host detects that the *Wait_For_Assembled_Segment* bit is reset, it resets the *Read_Assembled_Data* bit.
6) The SMD17L detects that the *Read_Assembled_Data* bit is reset, and sets the *Wait_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 SMD17L resets the *In_Assembled_Mode* bit and the *Wait_For_Assembled_Segment* bit.

Assembled Moves (continued)

Assembled Move Programming (continued)

Saving an Assembled Move in Flash

The SMD17L also contains the *Save_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_To_Flash* bit set. When you reach step 9 in the sequence, the SMD17L responds by resetting the *In_Assembled_Mode* and *Wait_For_Assembled_Segment* bits as usual and then it will flash the Status LED on the back of the unit. 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 SMD17L before you can continue. With a limit of 10,000 write cycles, the design decision that requires you to cycle power to the SMD17L 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 SMD17L 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 SMD17L 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.

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 2Arms and the current reduction to 25% will result in an idle current of 0.5Apk. (The SMD17L will always switch from RMS to peak current control when the motor is idle to prevent motor damage due to excessive heating.)

Stall Detection

When Stall Detection is enabled, the SMD17L will monitor 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 unit responds in the following manner:

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

Note that a move does not have to be in progress for stall detection to be useful. By enabling stall detection, the unit can notify the system if the motor shaft moves more than forty-five degrees while the motor is idle.

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 SMD17L 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 SMD17L, 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 SMD17L.
- 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 SMD17L 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. ($141^2 \approx 20{,}000$)

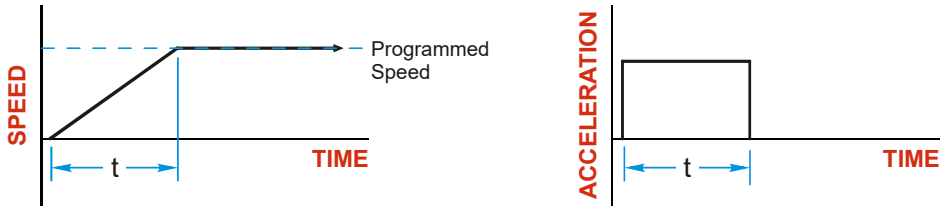
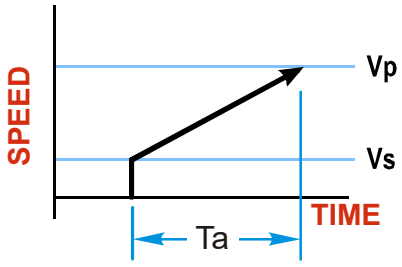


Figure R3.1 Constant Acceleration Curves

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	SMD17L 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 SMD17L 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

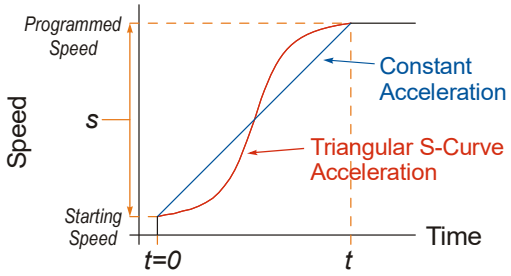
When the Acceleration Jerk parameter value is in the range of 1 to 5,000, the SMD17L uses this value to smoothly change the acceleration value applied during the move. In this case, the speed of the move does not increase linearly, but exponentially, resulting in an "S" shaped curve. This limits mechanical 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 SMD17L 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 equaling ({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

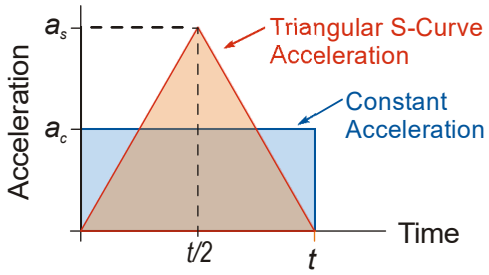


Figure R3.3 Triangular Acceleration

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:

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

$$a_s = 2a_c$$

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 $a_s = a_c$

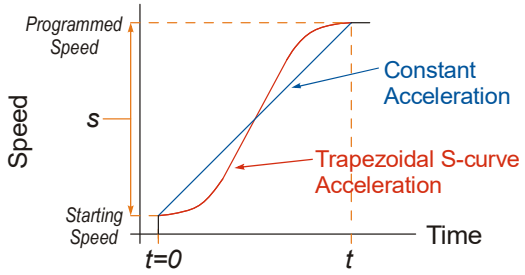
The above examples assume that you can increase the programmed acceleration value to keep the acceleration time the same. If your constant acceleration value is the maximum your system will allow, then using S-curve accelerations will lengthen the time needed to accelerate to your desired speed.

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.

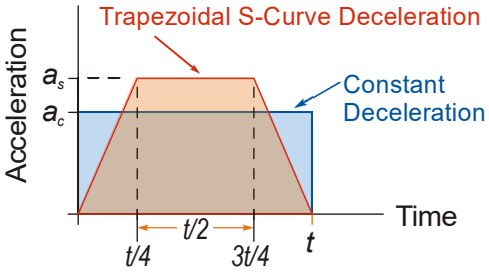


Figure R3.5 Trapezoidal Acceleration

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.

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

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

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

$$Ja_s t = 400a_s$$

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

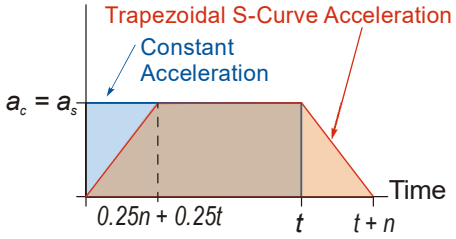
This value represents the ideal Acceleration Jerk parameter value for a trapezoidal S-curve acceleration with a constant acceleration for half of the phase. Setting the value lower than this will result in a shorter constant period, while setting the value greater than this will result in a longer constant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

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

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

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

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

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

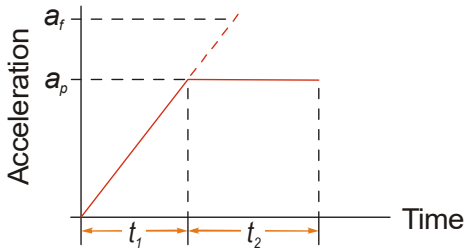


Figure R3.7 Calculating Trapezoidal S-Curve

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 SMD17L

This chapter explains the various ways of homing an SMD17L 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 SMD17L 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 SMD17L for speed control, don't require position data at all.

With the exception of Absolute Moves, the SMD17L 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. Both the motor position and the encoder position can be preset separately, and the motor position can also be preset to the encoder position.

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.

Home to a Hard Stop

Some machines can only be homed to a hard stop at one end of travel. The SMD17L can be homed on these machines as long as the unit has an encoder. The following steps can be used to home to a hard stop.

1) Enable the Encoder in the Configuration data.
2) Enable the Stall Detect in the Configuration data.
3) If possible, reduce the motor's current setting to the lowest value that achieves motion. The motor current can be set within the Jog command below.
4) Jog at low speed towards the hard stop.
5) Monitor the Stall_Detected status bit. Remain at step five until the Stall_Detected bit is set to "1".
6) Issue the Reset Error command and wait for the Stall_Detected status bit to be reset to "0"
7) Issue the Preset Position command to set the motor position to your desired value. This will reset the Position_Invalid status bit.
8) Once the motor position has been preset, issue the Preset Encoder command if you want the motor and encoder positions to be the same.

Find_Home Commands

The other choice is to use the driver's Find Home commands to order the SMD17L 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. It ends when the home sensor triggers while the SMD17Lis 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 needs to control which trigger is acted upon.

Physical Inputs

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

Network Data Input

- **Backplane_Proximity_Bit:** An SMD17L 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 SMD17L 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 SMD17L will never act on the physical Home Input.

Homing Configurations

A SMD17L must have one of its DC inputs configured as the home input before a homing command can be issued.

NOTE You do not have to configure and use CW or CCW Limits. If you choose to configure the unit this way, then the SMD17L has no way to automatically prevent over travel during a homing operation. In this case you must do one of the following:

- You must prevent over travel by some external means
- Ensure that the homing command is issued in the direction that will result in reaching the homing input directly
- Enable the encoder and the stall detection option.

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.

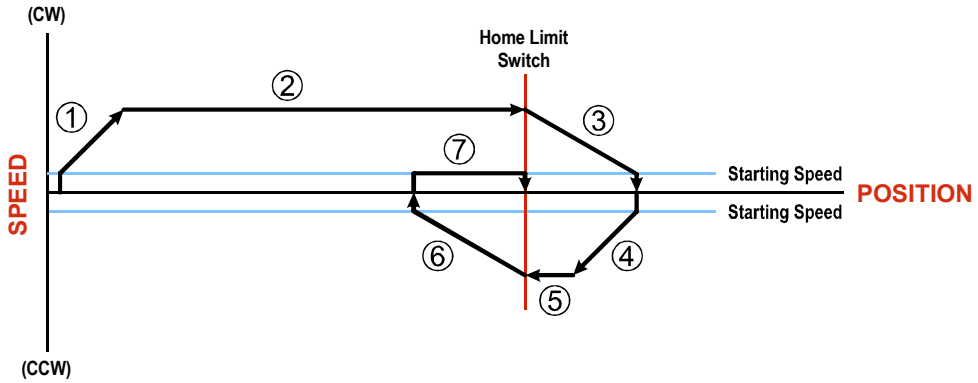


Figure R4.1 Home Input Profile

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.

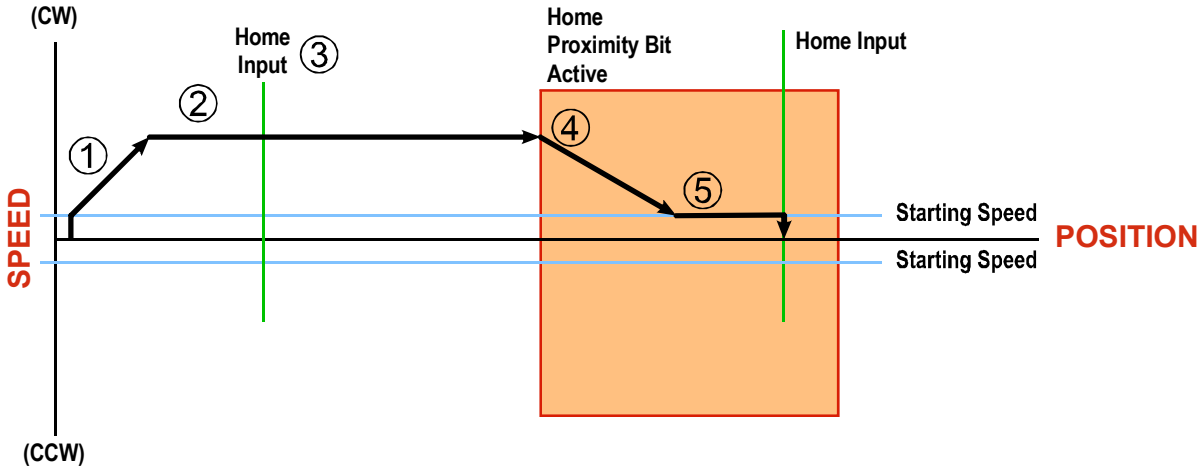


Figure R4.2 Homing with Proximity

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 SMD17L 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 SMD17L will stop and the *Input Error* bit will be sent 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.

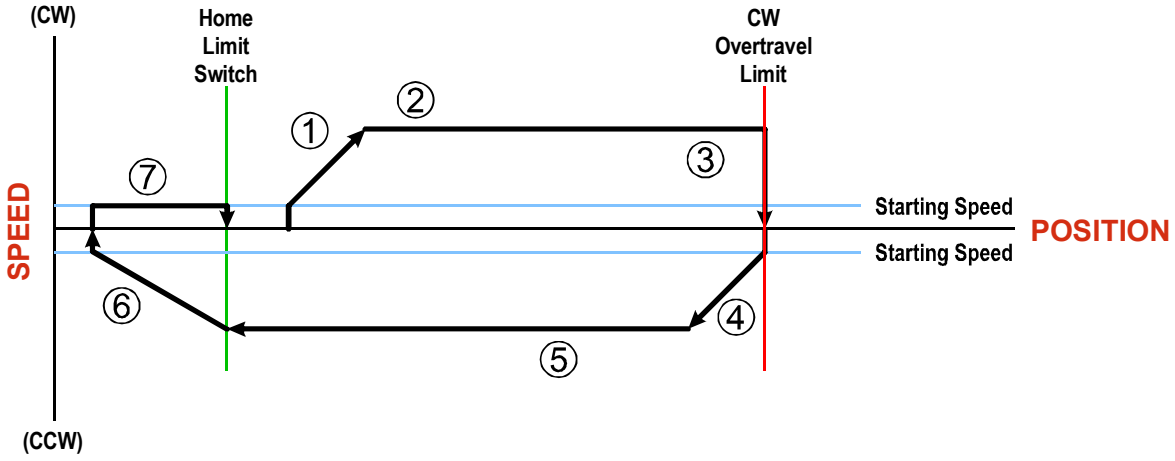


Figure R4.3 Profile with Overtravel Limit

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.

Notes

TASK 1
INSTALL THE SMD17L

1.1 Location

1.1.1 IP50 Rated Units (SMD17L-M)

SMD17L 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 (SMD17L-MS)

SMD17L 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).

1.1.3 IP65/7 Rated Units (SMD17L-MP)

SMD17L units that are IP65/7 rated are suitable for use in an industrial environment that meet the following criteria:

- Continuous conductivity occurs due to conductive dust, rain, or other wet conditions.
- 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 4* and *Over Voltage Category II* designations of the International Electrotechnical Commission (IEC).

1.2 Safe Handling Guidelines

1.2.1 Prevent Electrostatic Damage

CAUTION Electrostatic discharge can damage the SMD17L 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.

1.2 Safe Handling Guidelines (continued)

1.2.2 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 connectors. Debris may cause damage to the unit or unintended machine operation with possible personal injury.

1.2.3 Remove Power Before Servicing

WARNING Remove power before removing or installing any SMD17L units.

1.3 Operating Temperature Guidelines

Due to the onboard electronics, the maximum operating temperature of the SMD17L 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 SMD17L 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 by external means to verify that the maximum unit temperature remains below this 203°F/95°C specification. SMD17L devices have an onboard thermistor to monitor the electronics temperature and will remove motor current if the operating temperature exceeds this limit. Ovetemperature faults are also reported in the Network Input Data.

1.4 Power Supply Sizing

1.4.1 Motor Power Output

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 ***Torque and Power Curves*** on page 14 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 SMD17L-80 will be running at a maximum of 12 RPS and a 24 Vdc supply will be used. Based on the power curve in figure R1.3 on page 14, the combinations will generate a maximum of 29 Watts. Therefore a 24 Vdc supply with a power range of 44 W to 51 W can be used in the application.

Example 2: An SMD17L-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 R1.3 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.

Table T1.1 below shows the suggested power supply sizes based on the maximum power the motor can generate over its entire speed range.

		SMD17L-80		
		Motor Power	150% Supply	175% Supply
Supply Voltage	24 Vdc	33 W	50 W	58 W
	48 Vdc	64 W	96 W	112 W

Table T1.1 Suggested Power Supply Ratings

1.4 Power Supply Sizing (continued)

1.4.2 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 SMD17L 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 a large enough voltage spike 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 not included in the SMD products because of the limited ability to dissipate the heat generated by the resistor. An external resistor and control circuitry can be added to the system if needed.

1.5 Select Power Input

The SMD17L has the option of receiving power from an external power supply or from an IO-Link master Class B port. Two jumpers on the back of the SMD17L are used to select the power source. The SMD17L ships from the factory configured to use an external power supply. In this configuration it can be used with any Class A or Class B port available on an IO-Link master. Follow the procedure below to configure the IO-Link port on the SMD17L as a Class B port.

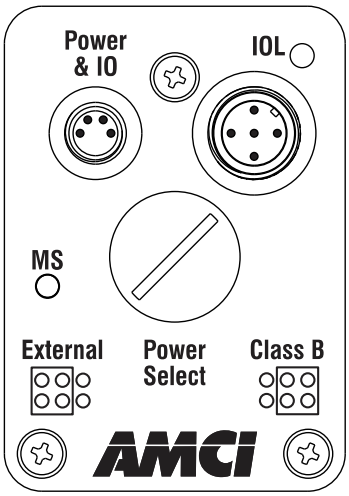


Figure R4.4
SMD17L Connector Locations

NOTE

1) The IO-Link power supply common is located on pin 3 of the M12 IO-Link connector. This common is always isolated from the rest of the system.
2) The Input common is located on pin 3 of the M8 connector.
3) The common used by the motor driver depends on the location of the two motor power source selection jumpers.
   - When the jumpers are set to their "External" positions, the common of the motor driver is attached to the Input common on pin 3 of the M8 Power & IO connector.
   - When the jumpers are set to their "Class B" positions, the common of the motor driver is attached to the "N24" pin of the IO-Link M12 connector.

1.5.1 Remove the PG9 Plug

WARNING

Follow the Safe Handling Guidelines in section 1.2 above. The PC board of the SMD17L is exposed when the PG9 plug is removed. This procedure must be completed in a clean environment. The SMD17L can be damaged if this procedure is not followed. This damage is not covered under warranty.

With power removed, use a flat head screwdriver to remove the PG9 plug. Set it aside on a clean surface.

1.5 Select Power Input (continued)

1.5.2 Switch the Location of the Two Jumpers

There are two motor power source selection jumpers that must be moved. Figure T1.1 shows the default jumper location. This selects an external supply that is brought in on the M8 Power & IO connector. If you change the jumpers to select Class B, then motor power must be supplied from a Class B IO-Link port.

NOTE Both jumpers must be moved when changing the motor power source selection. The graphic on the back of the unit and in figure R1.6 below shows the proper location of the jumpers for each selection.

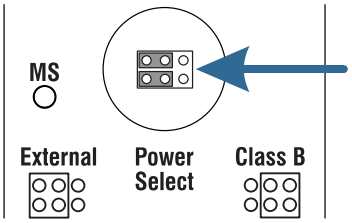


Figure T1.1 Power Source Selection Jumpers

1.5.3 Install the PG9 Plug

The plug must be tighten securely in order to maintain the IP rating of the SMD17L. Inspect the o-ring on the plug for damage before installing the plug. Use care when installing the plug so that the o-ring is not damaged.

1.6 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. This also allows higher current setting in environments with elevated ambient temperatures. 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 hardware with a minimum strength rating of 8.8 whenever 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.

1.6.1 SMD17L-M Mounting

The SMD17L-80(A,E)-M 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.

1.6 Mounting (continued)

1.6.2 SMD17L-MS Mounting

The SMD17L-80(A,E)-MS 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. SMD17L-80(A,E)-MS 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.

1.6.3 SMD17L-MP Mounting

The SMD17L-80(A,E)-MP units are water tight. Their IP65/7 rating makes them acceptable for use in washdown environments and can be exposed to low pressure / low volume water sprays as well as temporary immersions.

1.6.4 SMD17L-80 Outline Drawing

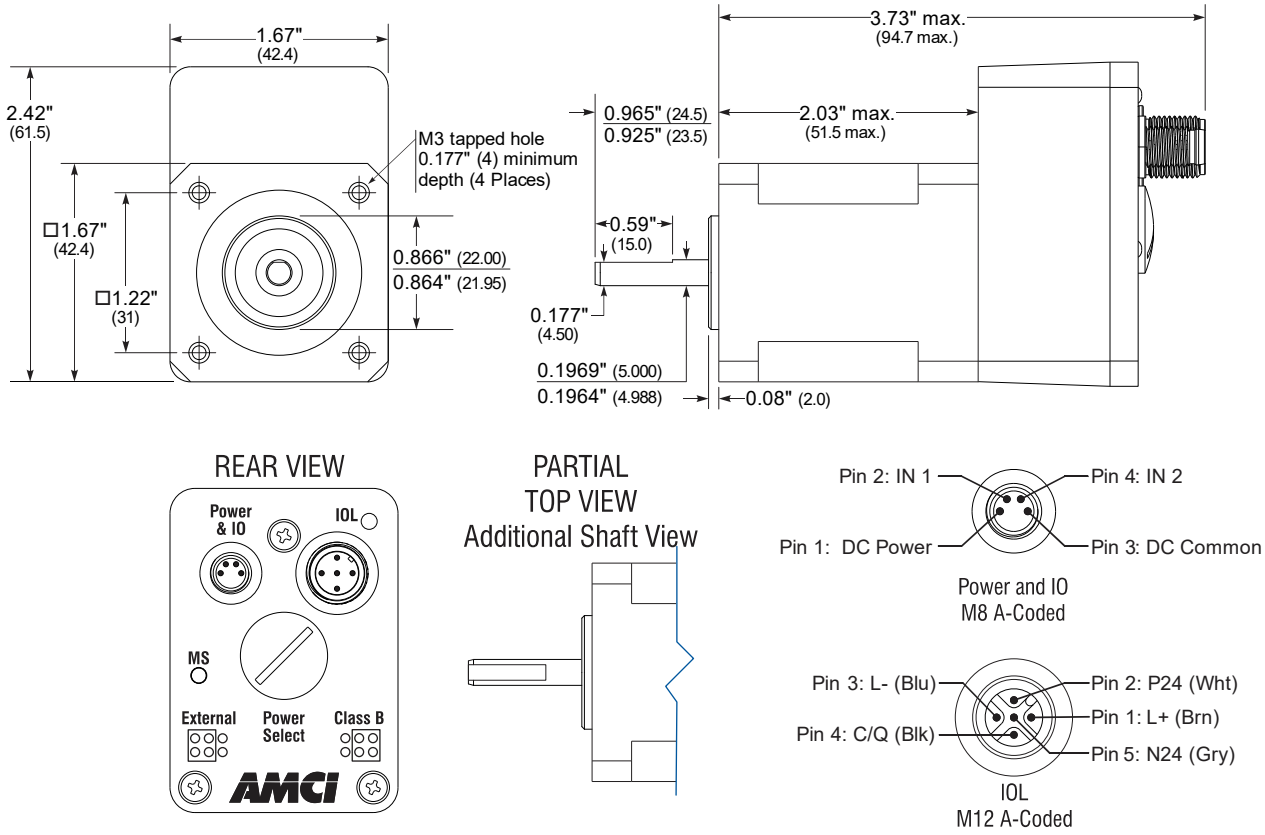


Figure T1.2 SMD17L-80 Outline Drawing

CAUTION

The 0.177" (4mm) minimum depth for the mounting screws refers to the depth of full thread. When you determine the length of your mounting screws, this length must not be exceeded. For example, if your mounting plate is 0.125" aluminum, the length of the mounting screw cannot exceed 0.302" (7.67mm). If this length is exceeded, the screw will bottom out against the motor laminations and separate the front flange from the rest of the motor.

1.6 Mounting (continued)

1.6.5 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 lb. (29 N) at the center of the flat on the shaft.
- Maximum axial load is 5.6 lb. (25 N)

NOTE Each SMD17L motor is a dual shaft motors. The internal encoder is mounted on the end of the second motor shaft. Excessive axial load on the motor shaft may cause encoder mis-alignment and damage to the unit. This type of damage is not covered under warranty.

1.7 Power and Input Connector

The Power and Input Connector is located on the back of the SMD17L. The connector is labeled "Power & IO". The connector is a standard four pin A-coded M8 connector that is rated to IP67 when the mate is properly attached. Figure T1.3 shows the pinout of the Power and Input Connector when viewed from the back of the unit.

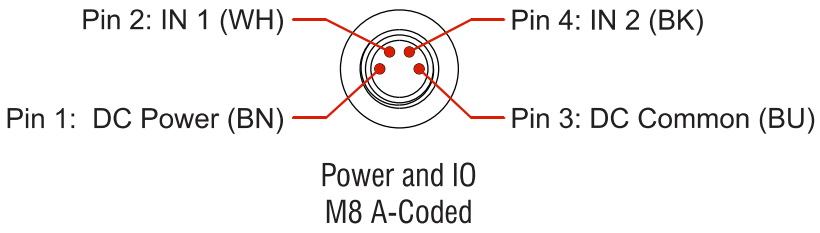


Figure T1.3 Power and Input Connector

The DC Power and DC Common pins supply power to the driver and motor when the motor power source selection jumpers are in their default "External" position. See section 1.5, ***Select Power Input*** on page 65 for more information.

Digital inputs IN1 and IN2 are single ended, and referenced to the DC Common pin.

1.7.1 Compatible Cordset

Many different connectors and cordsets are available on the market, all of which will work with the SMD17L units provided that the manufacturer follows the A-coded M8 standards. The following cordset is available from AMCI.

AMCI Part #	Description
CNSL-5M	4-position, 24 AWG. Connector: Straight M8, A-coded, Female to 2 inch flying leads, 0.28" stripped. Cable length: 5 m

Table T1.2 Power Cordset

1.8 Power Wiring

CAUTION Do not apply 120 Vac to any pins of an SMD17L. If this occurs, the unit will be damaged and you will void the unit's warranty.

WARNING The SMD17L 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.

1.8.1 External Power Supply

NOTE This section only applies when the motor power source selection jumpers on the SMD17L are set to their "External" setting. See 1.5, ***Select Power Input*** starting on page 65, for information on changing the jumper settings. If the motor power source selection jumpers are set to their "Class B" settings, see section 1.8.2, ***Power from Class B Port*** below.

When using an external supply, the SMD17L accepts 24 to 48Vdc as its input power. This should be derived from a power limited power supply. The power and input connector is a standard four pin A-coded M8 connector.

1.8.1.1 CNSL-5M Cable

The CNSL cable is made with 24 AWG wire. They are rated to 4 amps at 40°C. Based on the attached power supply and expected ambient temperatures, verify that these cable satisfy the requirements of the electrical code enforced in your jurisdiction before use.

1.8.1.2 Extending Power Leads

AMCI strongly suggests using 18 AWG or larger wire when the power leads to the SMD17L unit must be extended. This is to minimize power loss in the cable.

1.8.1.3 External Power Wiring

Figure T1.4 below shows how to wire power to an SMD17L unit when using an external supply. Colors in parentheses are the wire colors of the CNSL-5M cable.

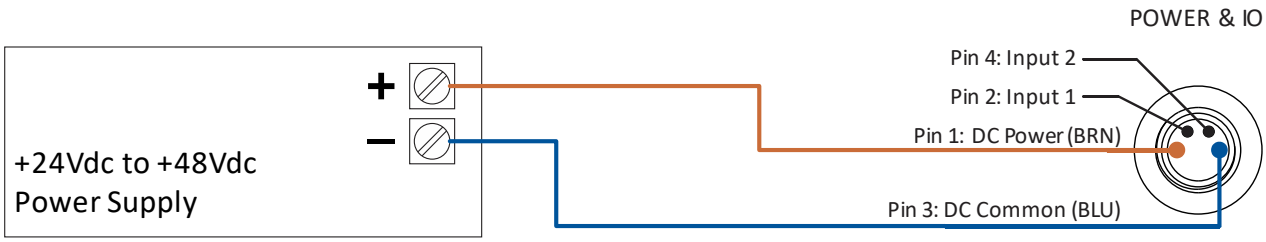


Figure T1.4 SMD17L Main Power Wiring

1.8 Power Wiring (continued)

1.8.2 Power from Class B Port

NOTE This section only applies when the motor power source selection jumpers on the SMD17L are set to their "External" setting. See 1.5, ***Select Power Input*** starting on page 65, for information on changing the jumper settings. If the motor power source selection jumpers are set to their "Class B: settings, see section 1.8.1, ***External Power Supply*** above.

When using a Class B port a 24Vdc power supply must be used. The IO-Link specification does not allow a 48Vdc supply. This should be derived from a power limited power supply.

1.8.2.1 CNPL-5M Cable

The CNPL cable is made with 22 AWG wire. They are rated to 4 amps at 40°C. Based on the attached power supply and expected ambient temperatures, verify that these cable satisfy the requirements of the electrical code enforced in your jurisdiction before use.

1.8.2.2 Extending Power Leads

AMCI suggests using 18 AWG or larger wire when the power leads to the IO-Link master must be extended. This is to minimize power loss in the cable.

1.8.2.3 Class B Power Wiring

When using a Class B port, the external power supply is wired directly into the IO-Link master. Refer to the documentation of your IO-Link master for proper wiring instructions.

1.9 Input Wiring

Inputs 1 and 2 are single ended, optoisolated 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 SMD17L. Colors in parentheses are the appropriate wire color of the CNSL-5M cable.

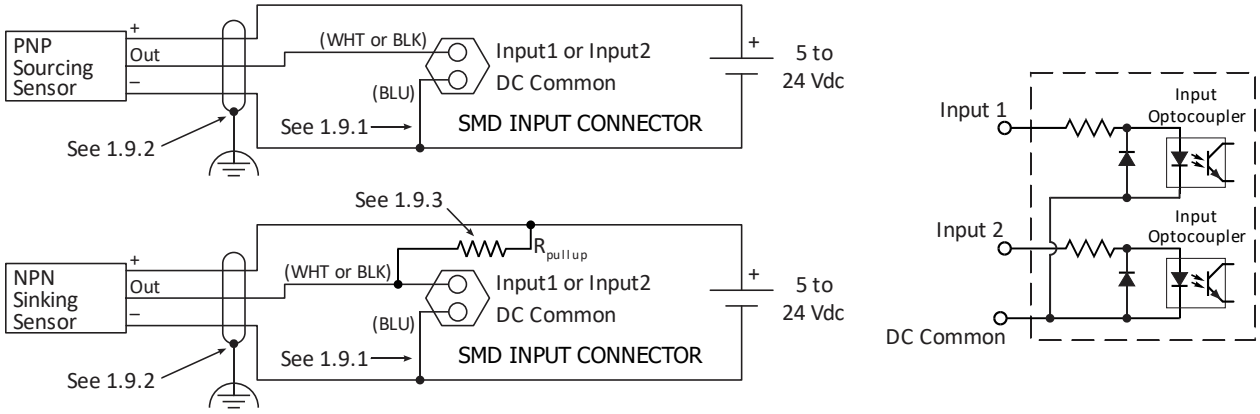


Figure T1.5 Input Wiring

1.9.1 DC Common

When the motor power source selection jumpers are set to the factory default position of "External", the motor driver and inputs share the DC Common pin. Therefore, the inputs are not completely isolated. The power supply that powers the motor driver (Pin 1 of the M8 connector) does not have to be the same supply that powers the inputs, but the commons of the two supplies must be tied together.

When the motor power source selection jumpers are set to the position of "Class B", the motor driver and inputs are completely isolated. Power for the motor driver comes from pins 2 and 5 of the IO-Link M12 connector. The commons of the two supplies do not need to be tied together.

1.9 Input Wiring (continued)

1.9.2 Cable Shields

Because they are low power signals, cabling from the sensor to the SMD17L 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 SMD17L.

1.9.3 Sinking Sensors Require a Pull Up Resistor

Sinking output sensors require an external pull up resistor because inputs 1 and 2 of the SMD17L 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 T1.3 Input Pull Up Resistors

The logical states of the sensor and SMD17L input will be reversed. The SMD17L input is off when the sensor is active. You can set the logic state of the input when you configure the unit.

1.10 IO-Link Connector

Figure T1.6 shows the IO-Link connector pinout when viewed from the back of the SMD17L. Pins 1, 3, and 4 are the IO-Link power and communication pins. The IO-Link power supply common pin (L-) is always isolated from the rest of the SMD17L system. Pins 2 and 5 are used by Class B ports to power the SMD17L.

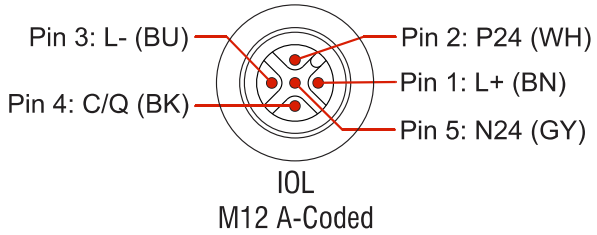


Figure T1.6 M12 IO-Link Connector Pinout

The connector is a standard male, five pin, A-coded M12 connector that is rated to IP67 when the mate is properly attached.

1.10.1 Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17L provided that the manufacturer follows the connector and IO-Link standards. AMCI offers the following mating connector and cordset to mate with the IO-Link port connector.

AMCI #	Description
MS-31	Mating connector for IO-Link connector. Female, 5 pin A-coded. Screw terminal connections. 6 to 8 mm dia. cable. Straight, IP67 rated when properly installed.
CNPL-5M	5-position, 22 AWG. Connector: Straight M12, A-coded, Female to 2 inch flying leads, 0.28" stripped. Cable length: 5 m

Table T1.4 Compatible IO-Link Connector and Cordset

Notes

TASK 2

IODD CONFIGURATION (RA)

This chapter explains how to add the SMD17L to your Rockwell Automation PLC project and configure the unit for your application.

2.1 Sample System

The screen captures in this chapter use Studio 5000 v37.0, a 5069-L310ERM processor, and a 5034-AENTR/A PointMax-IO base. This is the same setup used in the sample program which is available from AMCI. Your screens may be different if you are using a different system.

2.2 Obtain the IODD file

All AMCI IODD 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 IODD text file for the device.

2.3 Install the IODD file

2.3.1 Start the Device Description Installation Tool

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

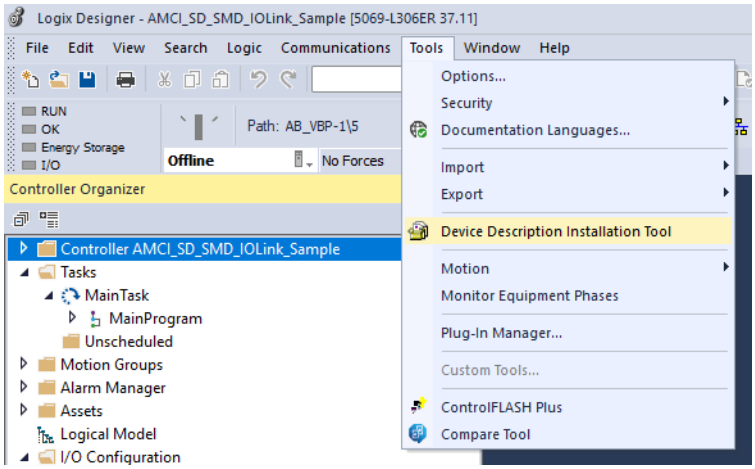


Figure T2.1 Opening the Device Description File Installation Tool

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

2.3 Install the IODD file (continued)

2.3.2 Install the IODD File

1) In the Options screen, select the Register a Device Description file(s) radio button and press [Next >].

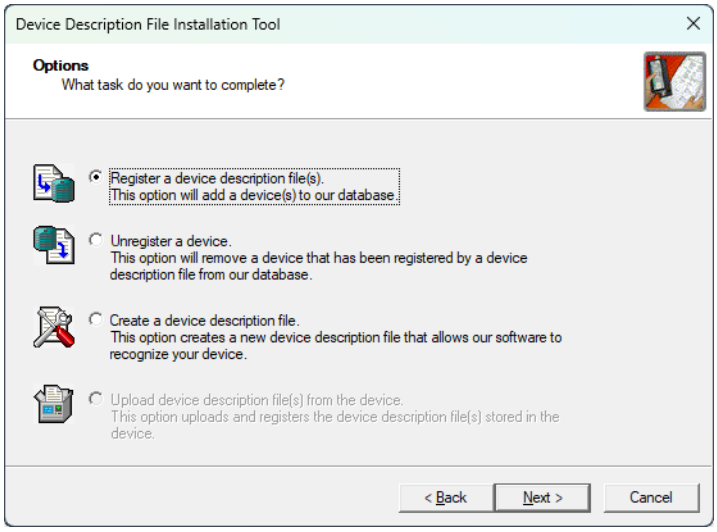


Figure T2.2 IODD Options Screen

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

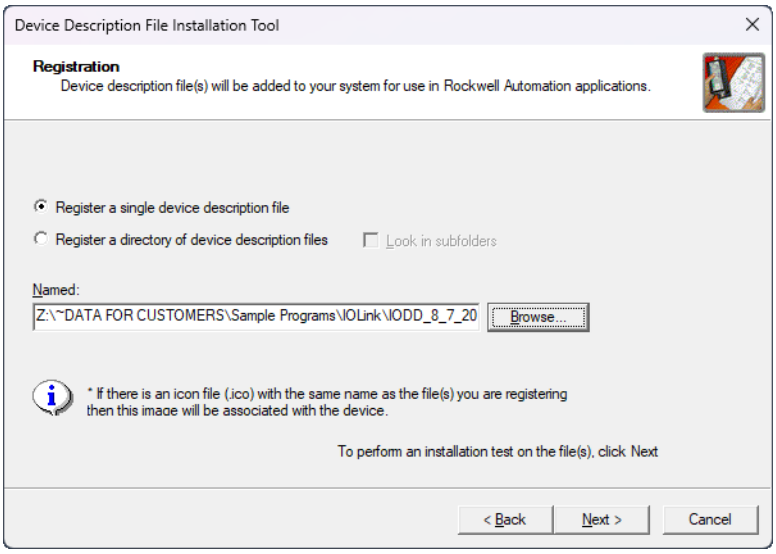


Figure T2.3 IODD Registration Screen

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

2.3 Install the IODD file (continued)

2.3.2 Install the IODD File (continued)

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

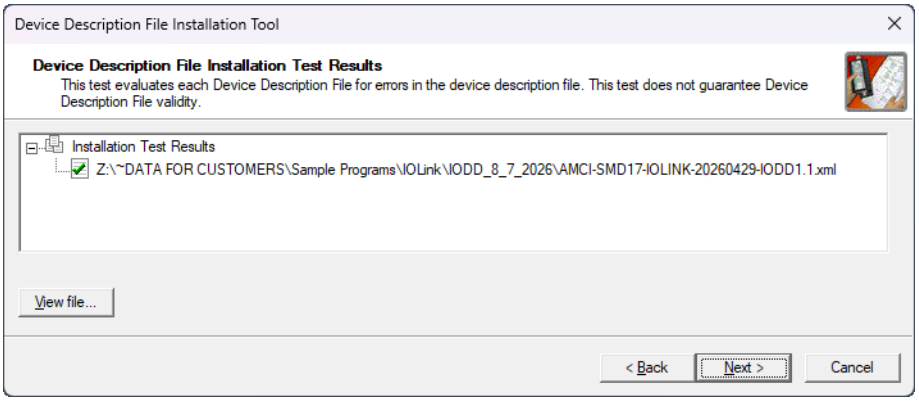


Figure T2.4 IODD Test Screen

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.

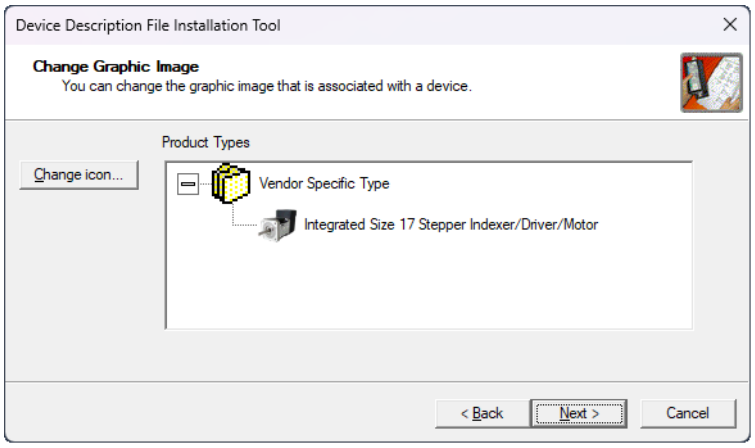


Figure T2.5 Change ECS Icon Screen

6) Click on the [Change icon...] button. In the window that opens, click on [Browse...] and browse to the folder that contains the extracted IODD 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 Device Description File Installation wizard.

2.4 Host System Configuration

Before you can add an SMD17L to your project, you must add an IO-Link adapter. The AMCI sample project makes use of Rockwell PointMax-IO 5034-IOL/4 IO-Link adapters. Refer to your Rockwell Automation documentation if you need instructions for configuring the IO-Link adapter, including how to configure the port for IO-Link.

2.5 Add the SMD17L to Your Project

You can add an AMCI SMD17L to the project once the IO-Link adapter is configured. As shown in figure T2.6 below, the IO-Link adapter will be listed under the I/O Configuration tree.

1) Right click on the IO-Link adapter and then click on "New Module..." in the pop-up menu.

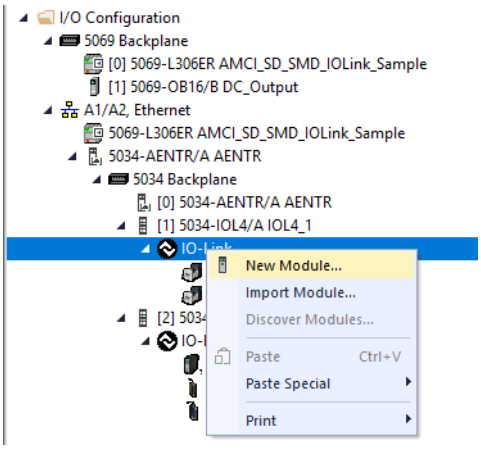


Figure T2.6 Adding an AMCI IO-Link Device

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 "SMD17L" in the resulting list.
4) Click on the [Create] button to create the module.

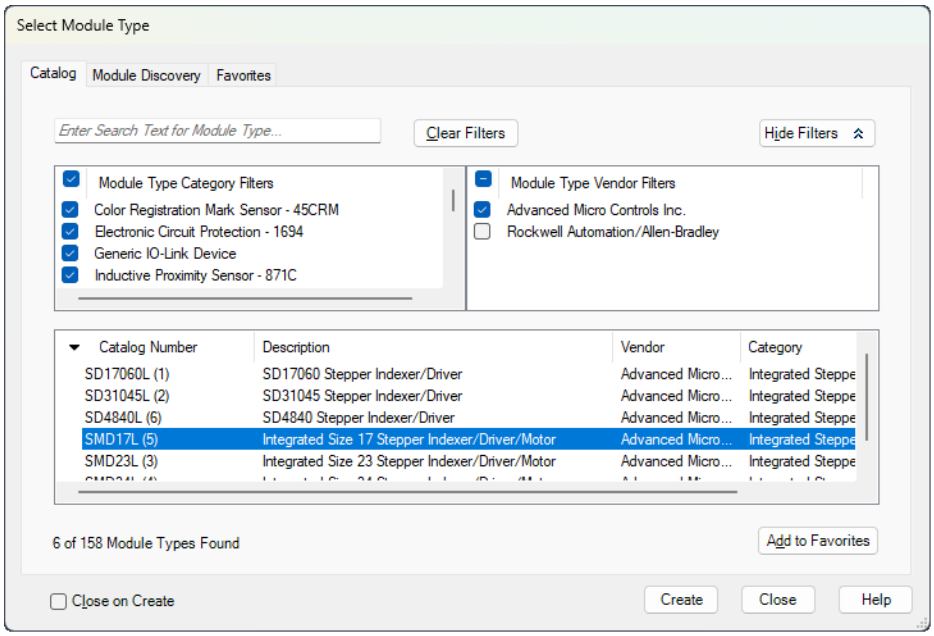


Figure T2.7 Selecting the IO-Link Device

2.5 Add the SMD17L to Your Project (continued)

5) Enter the appropriate information for the SMD17L unit.
   - Name: This is a required field. In the sample program, the module is named "AMCI_SMD17L"
   - Description: This is an optional field.
   - Slot: The slot number occupied by the module.
   - Revision: Leave at its default value.
   - Electronic keying: Set to "Compatible Module".
6) Click on the [OK] button to accept the definition of the SMD17L module.
7) Click on [Close] if necessary to close the Select Module Type screen.

2.6 Configure the SMD17L

If you are continuing from step 2.5, the resulting New Module screen is used to configure the network connection between the SMD17L and your controller. If you need to open the screen to perform this task at a later time, right click on the SMD17L 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 IODD file.

2.6.1 General Tab

The Name of the unit must be specified here. You can add a description here if required. The [Change...] button allows you to change the Module Definition if needed.

2.6.2 Connection Tab

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

2.6.3 Configuration Tab

The Configuration tab is used to define the configuration data that is written down to the SMD17L when the device connects to the network. You can also click on the [Apply] button to write down the configuration data to the SMD17L at any time.

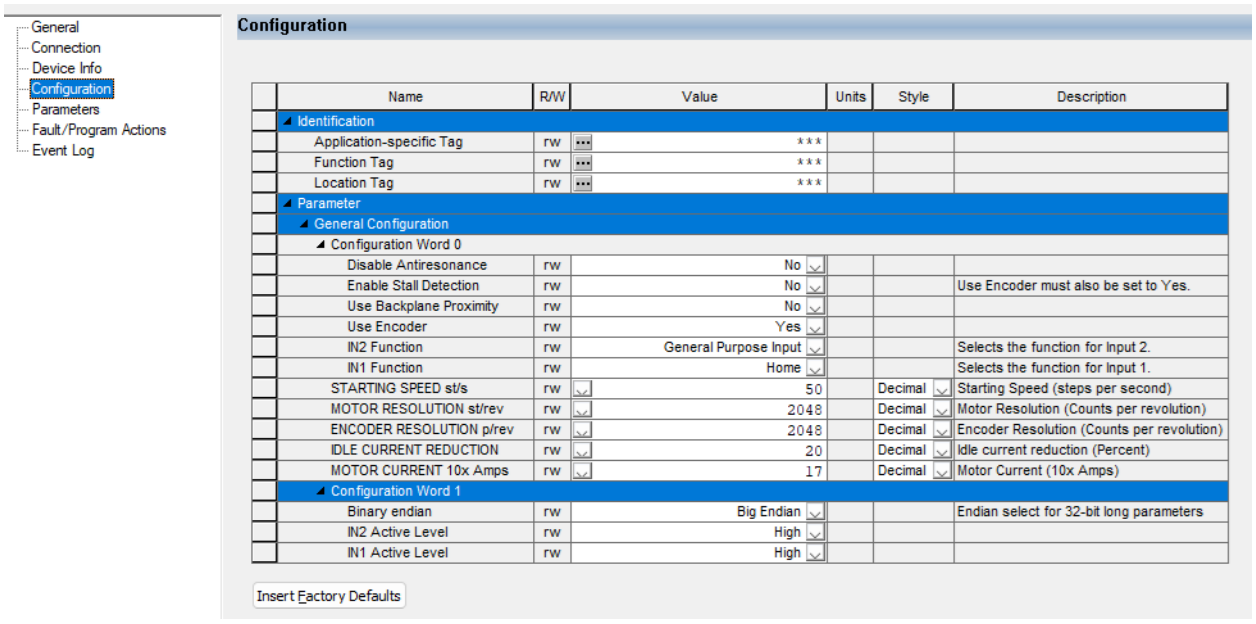


Figure T2.8 SMD17L Configuration

2.6 Configure the SMD17L Driver (continued)

2.6.3 Configuration Tab (continued)

Disable Antiresonance: The Anti-resonance feature will provide smoother operation in most cases. If you are experiencing resonance problems with this feature enabled, set the parameter's value to "Yes" to disable this feature and test the move again.

Enable Stall Detection: Set to "No" to disable motor stall detection. Set to "Yes" to enable motor stall detection. The **Use Encoder** parameter must also be set to "Yes". You must also program the **Encoder_Resolution** parameter.

Use Backplane Proximity: Set to "No" when the *Backplane_Proximity_Bit* is not used when homing the SMD17L. Set to "Yes" 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 parameter equal to "No".) If you enable this bit and then never turn on the *Backplane_Proximity_Bit*, the SMD17L unit will ignore all transitions of the home limit switch and you will not be able to home the device.

Use Encoder: Set to "Yes" to enable the built-in encoder. Set to "No" to disable the built-in encoder. If this parameter is set to "No", then the reported Encoder Position will always be zero. This parameter must be set to "Yes" when the **Enable Stall Detection** parameter is set to "Yes".

IN1 and IN2 Function: Use the drop down menu to select the functions of inputs one and two. A full description of the input functions can be found in the ***Available Discrete DC Inputs*** section starting on page 19. The default General Purpose Input setting means that the input has no effect on the unit but its state is reported in the network input data.

Binary Endian: The IO-Link specification requires thirty-two bit data to be transmitted in big endian format. When using Rockwell Automation IO-Link controllers leave this parameter set to its default *big endian* setting.

IN1 and IN2 Active Level: Use the drop down menu to select which input voltage level is considered active for the input. The selection depends on whether the sensor is sinking or sourcing, and whether the sensor acts as a normally open or normally closed contact.

Starting Speed: This is the speed that all moves begin and end at. Fifty steps/second is a reasonable default. This value can be tuned based on motor loading and the **Motor Resolution** setting. The valid range of values is 1 to 32,767. Note that the inverse of the starting speed is equal to the time in seconds needed to output the first pulse. See ***A Simple Move*** on page 31 for an example of how the Starting Speed is used.

Motor Resolution: This parameter programs the number of steps needed to complete one rotation of the motor shaft. The default value of 2,048 equals the default value for the encoder resolution and offers smooth micro-step movement. The range of values is 200 to 32,767. Note that the motor resolution and encoder resolution do not have to be equal.

2.6 Configure the SMD17L Driver (continued)

2.6.3 Configuration Tab (continued)

Encoder Resolution: Sets the number of counts per turn of the encoder. For SMD17L units with an absolute encoder, this parameter must be left at its default setting of 2048 counts/turn. For SMD17L units with an incremental encoder, the this parameter can be set to 1,024, 2,048 or 4,096 counts per turn.

Idle Current Reduction: Sets the percentage of motor current that the current is reduced to when the motor is idle. For example, if the motor current is set to 1.7 amps and the Idle Current Percentage is set to 20%, the current is reduced to (1.7 * 0.20) = 0.34 amps when the motor is idle for more than one second. Note that reducing idle current has an exponential effect (I^2R) on reducing motor heating when idle.

Motor Current: Sets the initial motor current used for each move. The range of values is 0.1 to 2.0 amps. Note that this parameter is programmed in steps of 0.1 amp, so the input value ranges from 1 to 20. The SMD17L gives you the ability to change the motor current used by subsequent moves at any time.

Notes

TASK 3

CONTROLLING THE SMD17L (RA)

This chapter explains how to add the AMCI Add-On Instructions (AOI) to your ladder logic program and the format of the IO tags assigned to the SMD17L.

3.1 Download the Sample Program

The AMCI sample program is available as a ZIP file on our website www.amci.com. At the time this manual was written, the direct link to the sample program page is ***https://www.amci.com/industrial-automation-support/sample-programs/***. The ZIP file contains the Add-On Instructions and the ladder logic sample program.

Once the ZIP file has downloaded, extract the files to a separate directory.

3.2 Import the Add-On Instructions

AMCI has written Add-On Instructions that cover all of the functionality offered by the SMD17L. If needed, complete step 3.1 above to download the AMCI sample program for the SMD17L unit. Add-On Instructions are located in the "AOI" sub-folder of the directory where you un-zipped the AMCI sample program.

The following table lists all of the instructions and gives descriptions of their uses.

AOI Name	Description
AMCI_IOLink_Input_conversion	*This AOI must be added to your project when using any of the other AMCI AOIs.* This instruction converts the IO-Link data to a format usable by the AOIs.
AMCI_IOLink_Output_conversion	*This AOI must be added to your project when using any of the other AMCI AOIs.* This instruction converts the data format used by the AOIs into the proper format for the IO-Link interface.
AMCI_MAFR	(Motion Axis Fault Reset) Use this instruction to recover from command faults. This instruction can also be run when the module is not in a fault state to clear the MoveComplete bit.
AMCI_MAH	(Motion Axis Home) Use this instruction to start a Find Home CW or Find Home CCW move.
AMCI_MAJ	(Motion Axis Jog) Use this instruction to start a Jog CW or Jog CCW move to a physical home input.
AMCI_MAM	(Motion Axis Move) Use this instruction to start an Absolute or Relative Move.
AMCI_MAS	(Motion Axis Stop) Use this instruction to start a Hold Move command. This command brings the axis to a controlled stop.
AMCI_MRP	(Motion Redefine Position) Use this instruction to initiate a Motor Position Preset command.
AMCI_MSO_Enable	Use this instruction to enable the motor of the SMD17L.
AMCI_MSF_Disable	Use this instruction to disable the motor of the SMD17L.
AMCI_MSO_and_Preset_to_Absolute_Encoder	Use this instruction to enable the motor and preset the motor position to the present value of the absolute encoder.
AMCI_MA_Preset_Encoder	Use this instruction to preset the encoder position.
AMCI_MA_Registration_Move	Use this instruction to run a Registration Move.
AMCI_MA_Reset_Driver_Fault	Use this instruction reset a driver fault on the SMD17L.
AMCI_MA_Resume	Use this instruction to resume a move that was stopped with an AMCI_MAS AOI.

(continued on next page)

3.2 Import the Add-On Instructions (continued)

AOI Name	Description
AMCI_MA_SD_SMD_Circular_Follower	Use this instruction to associate an SMD17L with a virtual circular motion axis and follow its motion.
AMCI_MA_SD_SMD_Linear_Follower	Use this instruction to associate an SMD17L with a virtual linear motion axis and follow its motion.
AMCI_MA_Program_Assembled_Move	Use this instruction to program the parameters of a future Assembled Move.
AMCI_MA_Run_Blend_Move	Use this instruction to command a Blend Move CW or Blend Move CCW move.
AMCI_MA_Run_Dwell_Move	Use this instruction to command a Dwell Move.

Table T3.1 SMD17L Add-On Instruction List

The Add-On Instructions are completely independent of each other. You can import only the AOIs you need for your program instead of importing all of them.

NOTE You must import the AMCI_IOLink_Input_conversion and AMCI_IOLink_Output_conversion instructions. These instructions are responsible for converting between the formats used by the IO-Link data stream and the formats used by the other AOIs.

To import an AOI:

1) From the main menu of Studio 5000, select “File” → “Import Component”→ “Add-On Instruction...”
2) In the window that opens, navigate to the folder that contains the unzipped AMCI sample program.
3) Navigate to the “AOI” folder.
4) Select the AOI you need to import and click on the [Open] button.
5) The “Import Configuration” window opens. If needed, change the name of the AOI in your project. Use the “Final Name:” text field for this purpose.

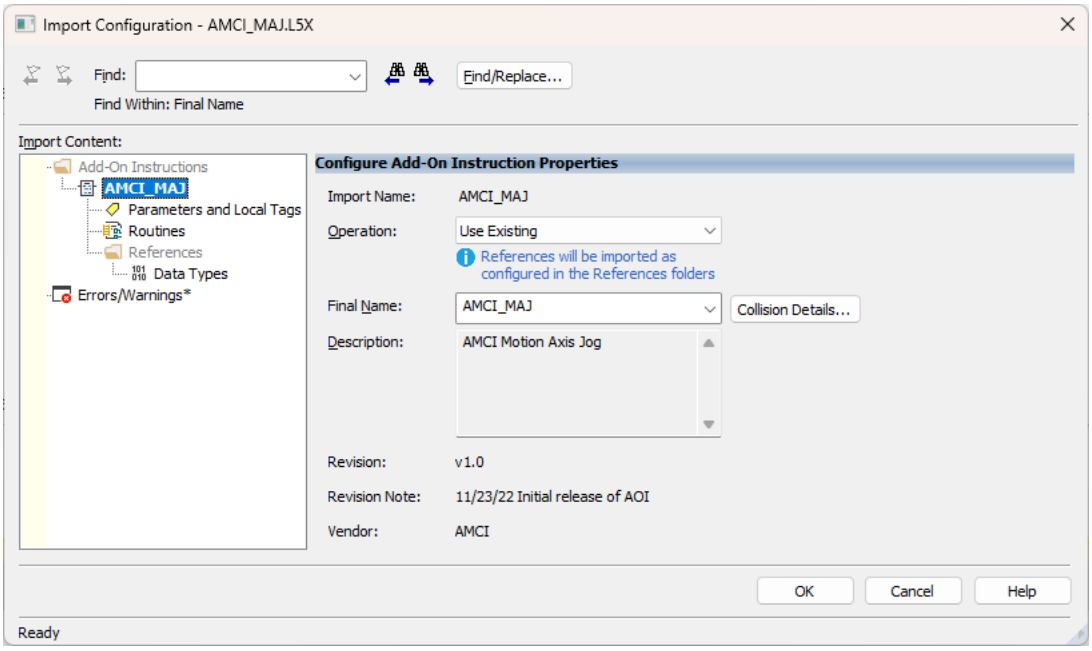


Figure T3.1 Import Add-On Instructions

6) Click on the [OK] button to import the AOI.
7) Repeat this section to import the remaining AOIs needed for your program.

3.3 Import the AMCI Data Types (If Needed)

AMCI has created User-Defined Data Types for use with the SMD17L Add-On Instructions. These data types are used to create buffers for the unit's I/O data. The latest versions of Studio 500 automatically import these data types when you import an AOI that uses these data types.

If these data types were not automatically imported, use the following steps to import them:

1) From the main menu of Studio 5000, select "File" → "Import Component"→ "Data Type..."
2) In the window that opens, navigate to the folder that contains the unzipped AMCI sample program.
3) Navigate to the "Data_Types" folder.
4) Select the "AMCI_Motion_Axis_Input_Data.L5X" and click on the [Open] button.
5) The "Import Configuration" window opens. If needed, change the name of the data type in your project. Use the "Final Name:" text field for this purpose.

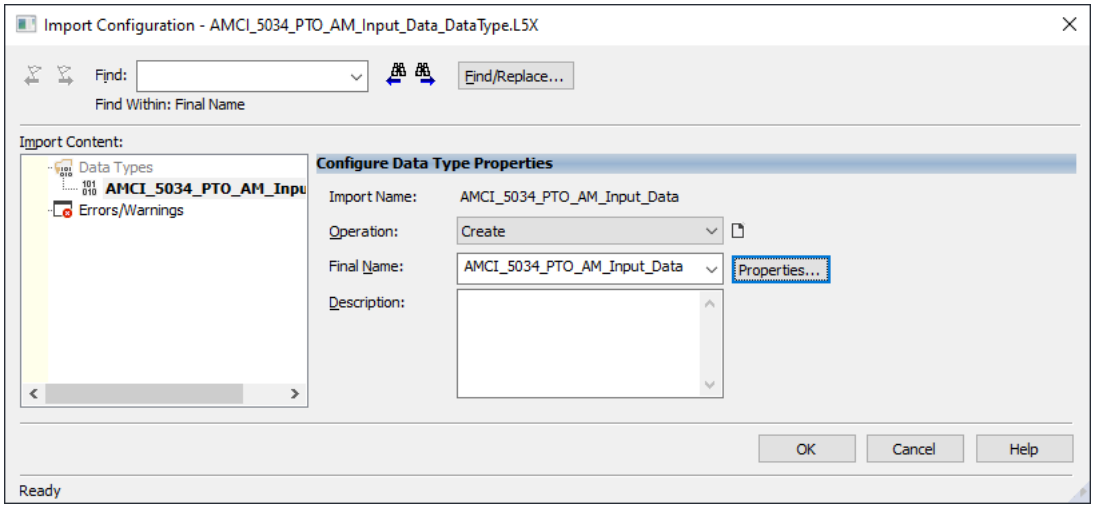


Figure T3.2 Import Data Types

6) Click on the [OK] button to import the data type.
7) Repeat this section to import the Output Data Type. The name of the file used in step 4 is "AMCI_Motion_Axis_Output_Data.L5X".

3.4 Create the Input and Output Buffers

The add-on instructions require that the input data remain stable while the instructions are scanned. Systems that are programmed with the Studio 5000 software update I/O asynchronously to the program scan. Therefore, the input data from the SMD17L must be buffered before any AMCI add-on instructions are scanned.

Two SINT arrays that have a length of 48 values are required to buffer the data to and from the SMD17L. These arrays must be created before using the AMCI AOIs. In the sample program, these buffers are called "SMD17L_Buffered_Input_Data" and "SMD17L_Buffered_Output_Data".

To create a SINT array:

1) In the main menu of the Studio 5000 program, click on "Logic" → "Edit Tags".
2) Scroll to the bottom of the list and enter the name for the input buffer tags in the line with the yellow tag.
3) Click in the Data Type field and then click on the [...] button that appears.
4) Select "SINT" as the data type and set the "Dim 0" value to "48."
5) Click on the [OK] button to set the Data Type.
6) Click inside the next row to accept the new data tag.

3.5 Create the Input and Output User Defined Tags

NOTE You must verify that the AMCI custom data types exist in your project before creating the I/O tags. Follow section 3.3 above if the custom data types were not automatically created when the AOIs were imported.

These user defined tags hold the data that is converted from the SINT buffers into the format used by the AMCI AOIs.

1) In the main menu of the Studio 5000 program, click on "Logic" → "Edit Tags".
2) Scroll to the bottom of the list and enter the name for the input buffer tags in the line with the yellow tag.
3) Click in the Data Type field and then click on the [...] button that appears.
4) Select the AMCI_Motion_Axis_Input_Data data type as shown in the figure below.

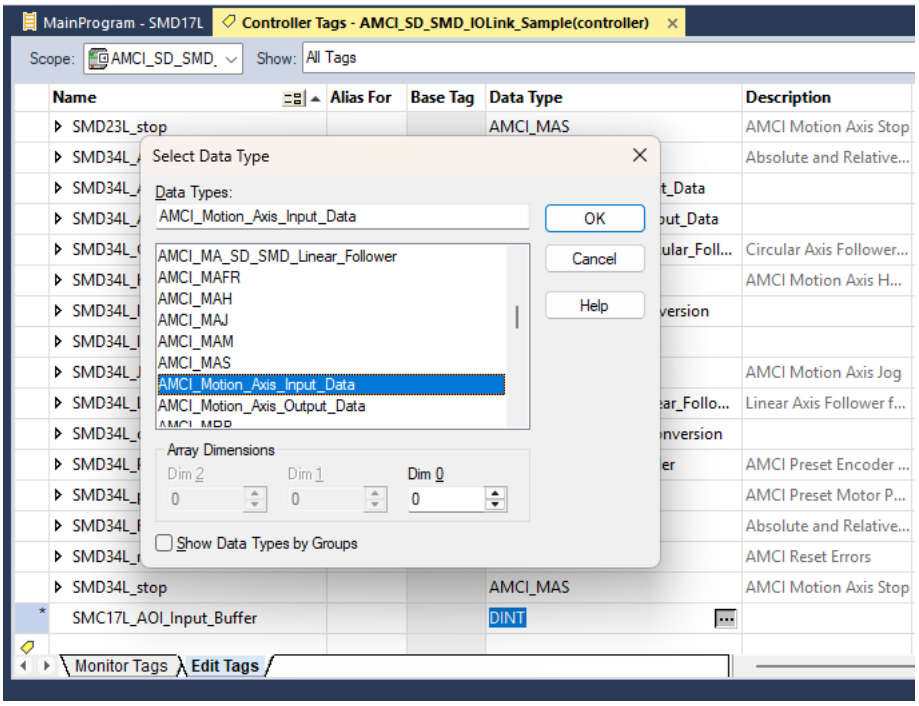


Figure T3.3 Create Input Data Buffer

5) Click on the [OK] button to set the Data Type.
6) Click inside the next row to accept the new data tags.

The SMD17L requires that all output data be updated at the same time. Therefore, the program must write to an output buffer during the program scan. This buffer must be copied to the module's process data after all AMCI Add-On Instructions are scanned.

7) Enter a name for the output buffer tags.
8) Click in the Data Type field and then click on the [...] button.
9) In the "Select Data Type" window that opens, select the AMCI_Motion_Axis_Output_Data data type.
10) Click on the [OK] button to set the Data Type.
11) Click inside the next row to accept the new data tags.

3.6 Add Code to Update Buffers and Convert Data

The input and output buffers should be updated in your code using unconditional CPS (Copy Synchronous) instructions. This ensures that the module is not updated during the copy. The input data conversion AOI should occur on an unconditional rung right after the input data CPS instruction. The output data conversion AOI should occur on an unconditional rung right before the output data CPS instruction.

Input data from the SMD17L is buffered and converted before any of the SMD17L control AOIs. The output buffer is converted and written to the SMD17L tags after all of the SMD17L control AOIs.

NOTE The CPS and conversion instructions do not have to be the first and last rungs of your ladder logic, but all logic associated with the AMCI add-on instructions must occur between the CPS and conversion rungs.

The figure below shows how to buffer and convert the data. A single AMCI_MAM AOI between them starts a relative move.

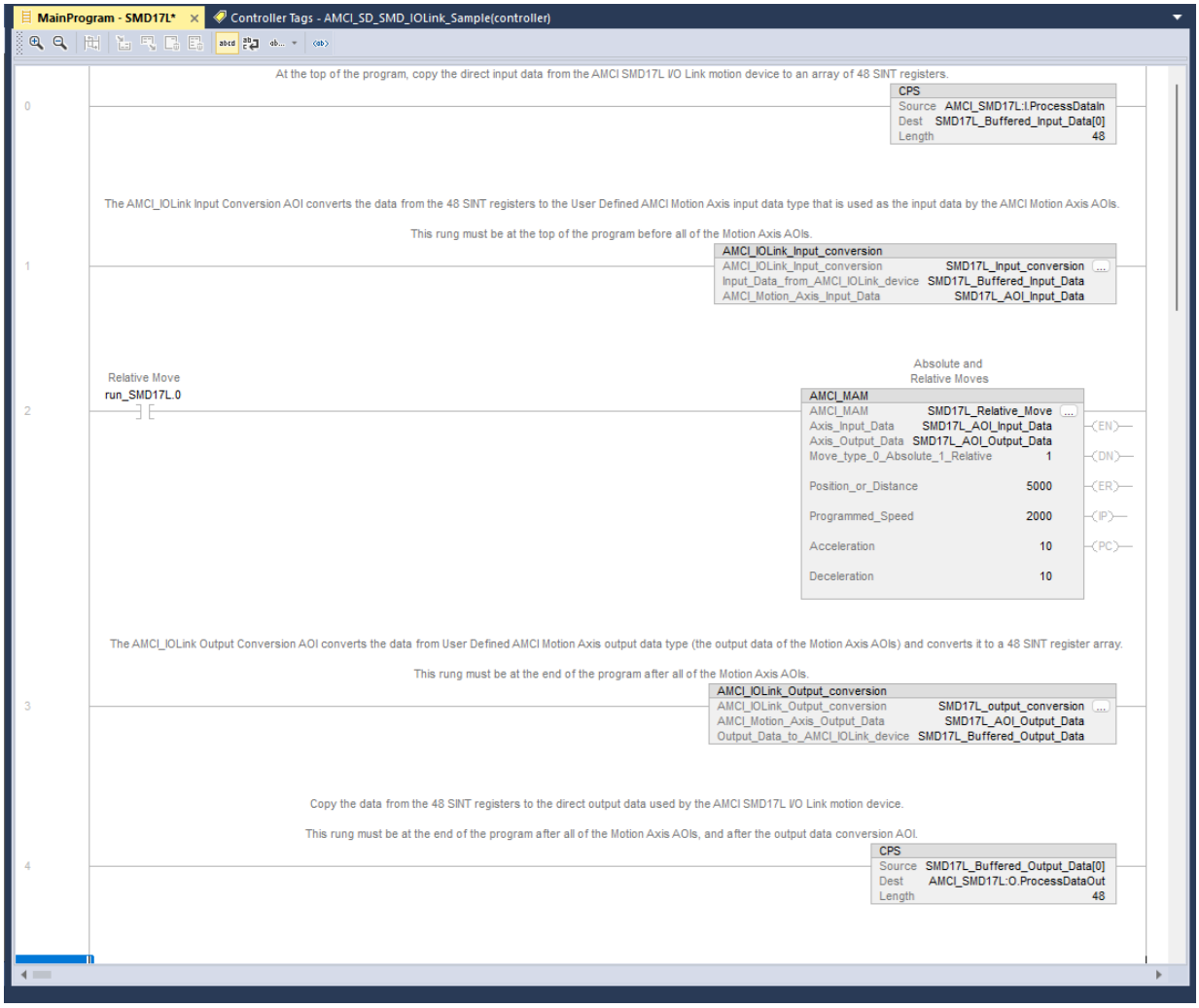


Figure T3.4 Sample Ladder Logic

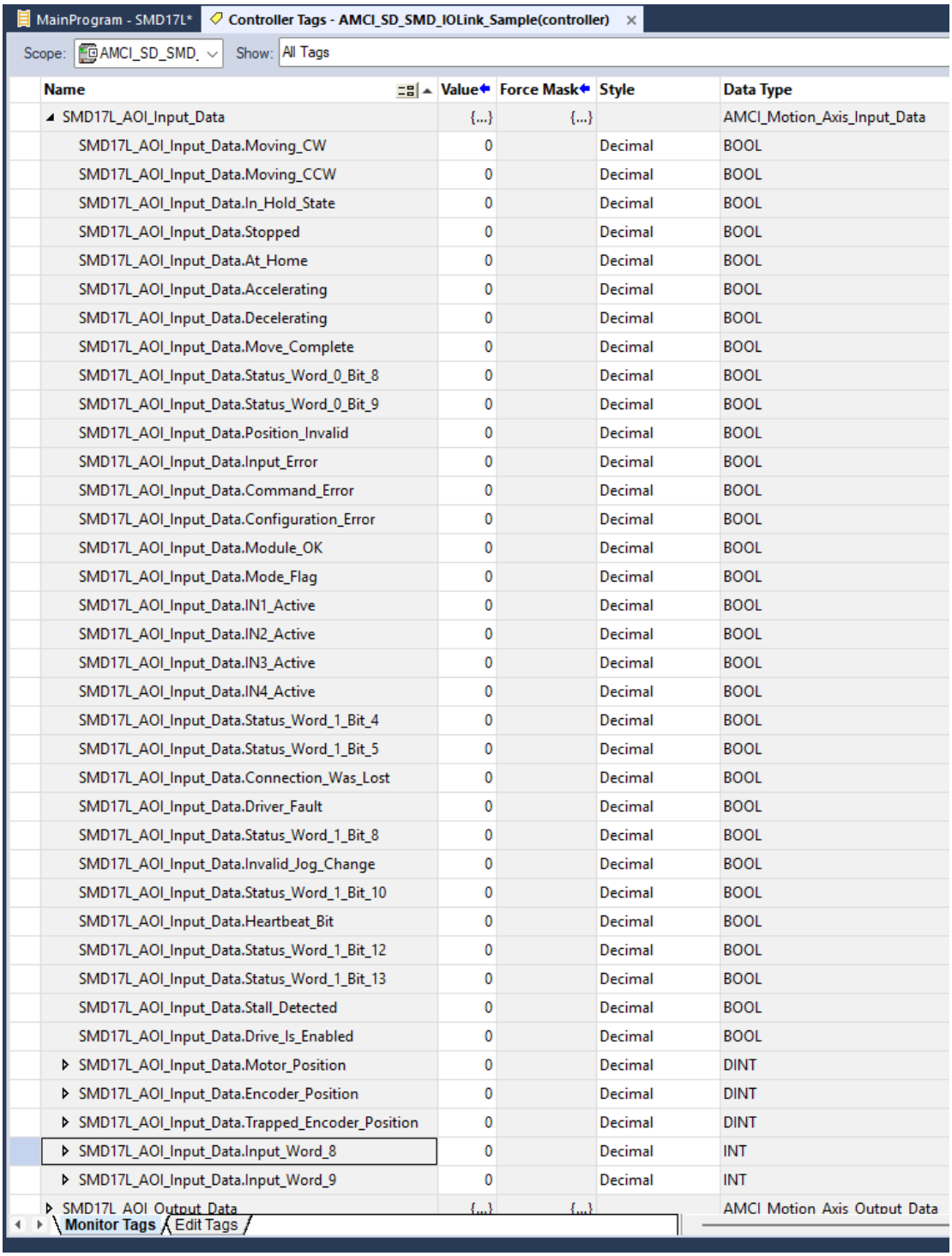
3.7 Use the Add-On Instructions

Each AMCI add-on instruction requires the rung to make a false-to-true transition before it operates. Please refer to the sample program for information on using the AMCI AOIs.

The *AOI Reference (RA)* chapter, which starts on page 95, lists the each AOI, its parameters, and its enumerations.

Format of Input Data Tags

The format of the buffered input data available from the SMD17L is shown below. To access this data, start from the Studio 5000 software's main menu, click on "Logic" and then click on "Monitor Tags". The data is listed under the name of the buffer you created in step 3.5 above. In the image below the name of the buffer is "SMD17L_AOI_Input_Data".



Name	Value	Force Mask	Style	Data Type
SMD17L_AOI_Input_Data	{...}	{...}		AMCI_Motion_Axis_Input_Data
SMD17L_AOI_Input_Data.Moving_CW	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Moving_CCW	0		Decimal	BOOL
SMD17L_AOI_Input_Data.In_Hold_State	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Stopped	0		Decimal	BOOL
SMD17L_AOI_Input_Data.At_Home	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Accelerating	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Decelerating	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Move_Complete	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_0_Bit_8	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_0_Bit_9	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Position_Invalid	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Input_Error	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Command_Error	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Configuration_Error	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Module_OK	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Mode_Flag	0		Decimal	BOOL
SMD17L_AOI_Input_Data.IN1_Active	0		Decimal	BOOL
SMD17L_AOI_Input_Data.IN2_Active	0		Decimal	BOOL
SMD17L_AOI_Input_Data.IN3_Active	0		Decimal	BOOL
SMD17L_AOI_Input_Data.IN4_Active	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_1_Bit_4	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_1_Bit_5	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Connection_Was_Lost	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Driver_Fault	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_1_Bit_8	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Invalid_Jog_Change	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_1_Bit_10	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Heartbeat_Bit	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_1_Bit_12	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Status_Word_1_Bit_13	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Stall_Detected	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Drive_Is_Enabled	0		Decimal	BOOL
SMD17L_AOI_Input_Data.Motor_Position	0		Decimal	DINT
SMD17L_AOI_Input_Data.Encoder_Position	0		Decimal	DINT
SMD17L_AOI_Input_Data.Trapped_Encoder_Position	0		Decimal	DINT
SMD17L_AOI_Input_Data.Input_Word_8	0		Decimal	INT
SMD17L_AOI_Input_Data.Input_Word_9	0		Decimal	INT
SMD17L_AOI_Output_Data	{...}	{...}		AMCI_Motion_Axis_Output_Data

Figure T3.5 Buffered Input Tag Layout

Format of Input Data Tags (continued)

Boolean Input Bits

All of the boolean bits shown in the figure above are located in two 16 bit words of the buffered input data. The figure below shows the layout of the bits within the words. Note that some of the bits in the Data Type have generic names, such as "Status_Word_0_Bit_8". Units that use this Date Type use these bits in different ways. The following descriptions explains how these bits are used by the SMD17L.

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Status Word 0	Mode_Flag	Module_OK	Config_Err	Command_Err	Input_Err	Pos_Invalid	Wait_AssySeg	In_Assy_Mode	Move_Cmp	Decelerating	Accelerating	At_Home	Stopped	In_Hold_State	Moving_CCW	Moving_CW
Status Word 1	Drive_Enabled	Stall_Detected	Cmd_Ack	Abs_Enc_Err	Heartbeat_Bit	Limit_Condition	Inv_Jog_Cng	0	Driver_Fault	0	0	Temp_90°C	0	0	IN2_Active	IN1_Active

Figure T3.6 Status Word Input Registers

Bit 0.0: Moving_CW – Set to "1" when the motor is rotating in a clockwise direction.

Bit 0.1: Moving_CCW – Set to "1" when the motor is rotating in a counter-clockwise direction.

Bit 0.2: In_Hold_State – Set to "1" when an Absolute, Relative, or Jog move has been successfully brought to a controlled stop and placed in its Hold State. The AMCI_MAS AOI is used to perform this function.

Bit 0.3: 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 0.4: At_Home – Set to "1" when a homing command has completed successfully, "0" at all other times.

Bit 0.5: Accelerating – Set to "1" when the present move is accelerating. Set to "0" at all other times.

Bit 0.6: Decelerating – Set to "1" when the present move is decelerating. Set to "0" at all other times.

Bit 0.7: 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 SMD17L or when the Reset Errors (MAFR) command is issued. 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 0.8: In_Assembled_Mode – (Status_Word_0_Bit_8 in the Data Type.) The SMD17L sets this bit to signal the host that it is ready to accept assembled move profile programming data. You will see this bit turn on and off when the AMCI_MA_Program_Assembled_Move AOI is enabled and active. The In_Assembled_Mode bit is also set when the Assembed Move is being performed.

Bit 0.9: Waiting_For_Assembled_Segment – (Status_Word_0_Bit_9 in the Data Type.) The SMD17L sets this bit to tell the host that it is ready to accept the data for the next segment of your assembled move profile. You will see this bit turn on and off when the AMCI_MA_Program_Assembled_Move AOI is enabled and active.

Format of Input Data Tags (continued)

Boolean Input Bits (continued)

Bit 0.10: Position_Invalid – "1" on every power up or when:

- Configuration data is written to the SMD17L
- 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.

This bit is reset to "0" after the motor position has been set with a MAH or a MRP AOI. Absolute moves cannot be performed while this bit is set to "1". The state of this bit is ignored for all other move types.

Bit 0.11: 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 the AMCI_MAFR AOI.

Bit 0:12: Command_Error – "1" when an invalid command has been written to the SMD17L. This bit is reset with the AMCI_MAFR AOI.

Bit 0:13: Configuration_Error – "1" on power up before a valid configuration has been written to the SMD17L, or after any invalid configuration has been written to the unit. "0" when the unit has a valid configuration in memory. Configuration is set from the Module Properties screen. For additional information, see section 2.6, ***Configure the SMD17L*** starting on page 77.

Bit 0:14: Module_OK – "1" when the SMD17L is operating without a fault, "0" when an internal fault condition exists.

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

Bit 1:0: IN1_Active – "1" when Input 1 is in its active state. The active state of the input, "High" or "Low", is programmed as part of the unit's configuration.

Bit 1:1: IN2_Active – "1" when Input 2 is in its active state. The active state of the input, "High" or "Low", is programmed as part of the unit's configuration.

Bit 1:2: Reserved – (IN3_Active in the Data Type.) Will always equal zero.

Bit 1:3: Reserved – (IN4_Active in the Data Type.) Will always equal zero.

Bit 1:4: Temperature_Above_90°C – (Status_Word_1_Bit_4 in the Data Type.) 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 1.5: Reserved – (Status_Word_1_Bit_5 in the Data Type.) Will always equal zero.

Bit 1.6: Reserved – (Connection_Was_Lost in the Data Type.) Will always equal zero.

Bit 1.7: Driver_Fault – If the driver section of the SMD17L 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 1.8: Reserved – (Status_Word_1_Bit_8 in the Data Type.) Will always equal zero.

Format of Input Data Tags (continued)

Boolean Input Bits (continued)

Bit 1.9: 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. Enabling a AMCI_MAFR AOI will not reset this bit. This bit will reset when valid jog data is sent from the host controller.

Bit 1.10: Limit_Condition – (Status_Word_1_Bit_10 in the Data Type.) 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 1.11: 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 not available if there is a configuration error.

Bit 1.12: Absolute Encoder Error – (Status_Word_1_Bit_12 in the Data Type.) 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 1.13: Command_Acknowledge – (Status_Word_1_Bit_13 in the Data Type.) 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 1.14: Stall_Detected – Set to "1" when a motor stall has been detected.

Bit 1.15: Drive_Is_Enabled – Set to "1" when the motor driver section of the SMD17L 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.

Other Input Data Tags

Motor_Position: Current motor position value.

Encoder_Position: Current encoder position value.

Trapped_Encoder_Position: Value captured on the last inactive-to-active transition on a Capture Input. If an encoder is enabled in the configuration, the captured value is the encoder position. If there is no encoder, the captured value is the motor position.

Programmed_Motor_Current: (Input_Word_8 in the Data Type.) Present value of the motor current used by moves. The default value is programmed as part of the Configuration Data.

Acceleration Jerk Parameter: (Input_Word_9 in the Data Type.) Last value of the Acceleration Jerk parameter set to the unit as part of an AOI that produces motion. The default value of this parameter is zero.

Format of Output Data Tags

Viewing output data values is usually only done during troubleshooting, and looking at the buffer instead of the actual output data assigned to the module offers no real advantage. Therefore, the figure below shows the tags of the output data assigned to the SMD17L instead of the output buffer tags.

To access this data, start from the Studio 5000 software's main menu, click on "Logic" and then click on "Monitor Tags". The data is listed under the name given to the SMD17L followed by ":O.ProcessDataOut". In the image below the name of the SMD17L is "AMCI_SMD17L.

All of the boolean bits shown in the figure below are located in words 0 and 1 of the output data. Most of these bits are directly controlled by the available AOI's. They are shown here to help with troubleshooting.

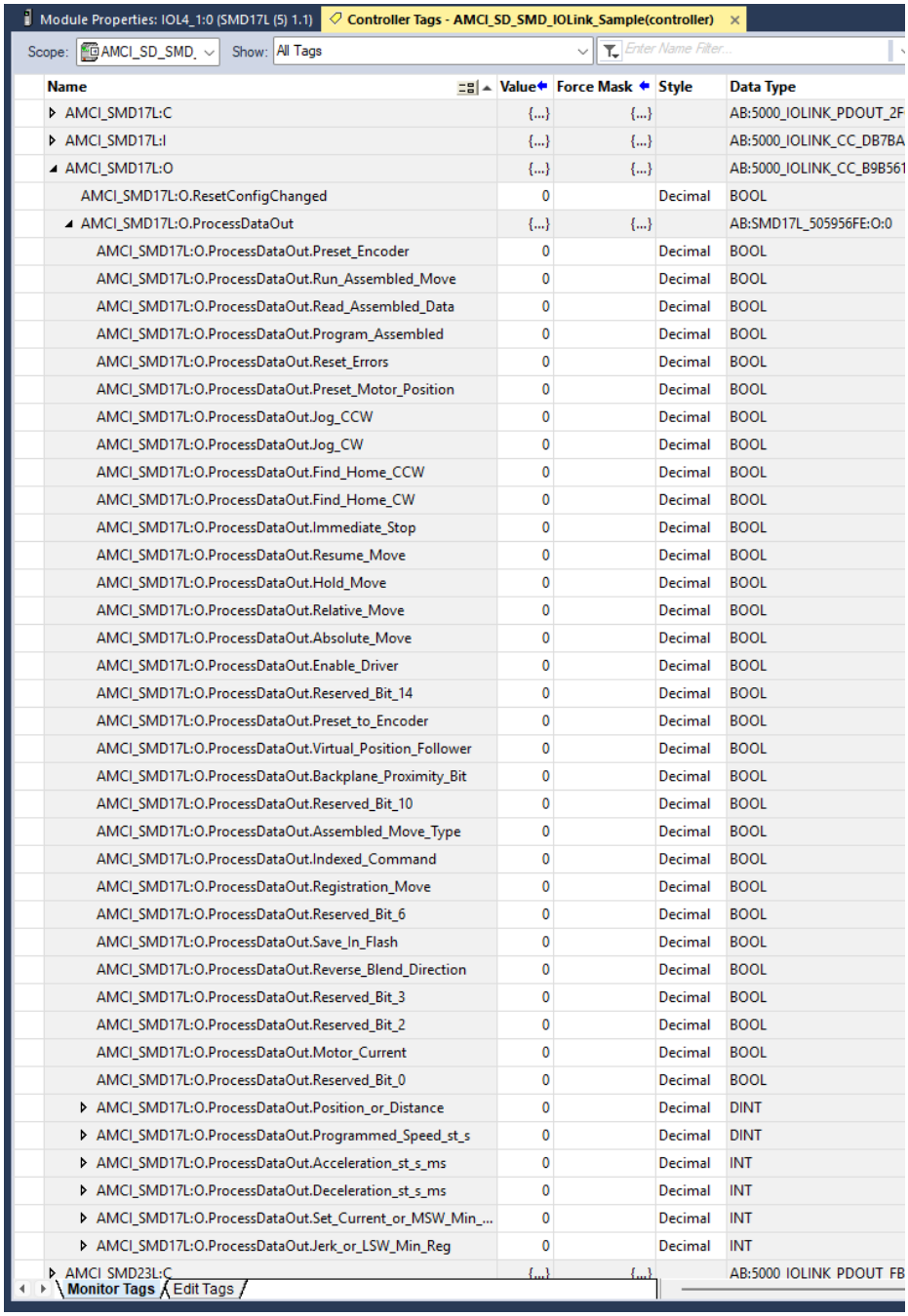


Figure T3.7 Output Tag Layout

Format Output Data Tags (continued)

Boolean Control Bits

The control bits in the output buffer are located in the first two 16 bit words of the data.

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Command Word 0	0	Prst_Enc	Run_AMov	Rd_AData	Prg_Assm	RSet_Err	Prst_Pos	Jog_CCW	Jog_CW	Home_CCW	Home_CW	I-Stop	Resm_Mv	Hold_Mv	Rel_Mv	Abs_Mv
Command Word 1	En_Driver	0	Preset_to_Enc	V_Pos_Follow	BP_Prox	0	AsMv_Type	Index_Cmd	Reg_Move	0	Save_to_Flash	Rev_BlendDir	0	0	Motor_Current	0

Figure T3.8 Command Word Output Bits

In the bit descriptions that follow, the term "Exclusive Bit" is used. When commanding the module, only one of the set of exclusive bits can be set at a time. If two or more of these bits are set at the same time, the "Command_Error" bit will be set in the module's input data tags.

Bit 0.15: Mode_Flag – *This bit is not defined in the IO-Link IODD file and it is important that this bit remains "0" at all times.* Setting this bit to "1" will not damage the unit, but it will switch to a mode of operation that is not documented in this manual. In this mode, configuration data can be changed with IO data. Rockwell Automation systems should always use the Properties screen to change configurations as described in the ***Configure the SMD17L*** section found on page 77.

Bit 0.14: Preset_Encoder – Exclusive Bit. Set by the AMCI_MA_Preset_Encoder AOI.

Bit 0.13: Run_Assembled_Move – Exclusive Bit. Set by the AMCI_MA_Run_Blend AOI and the AMCI_MA_Run_Dwell AOI.

- **Assembled_Move_Type – Command Word 1.9:** This bit will equal "0", while a Blend Move runs. This bit equals "1", while a Dwell Move is run.
- **Reverse_Blend_Direction – Command Word 1.4:** 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 0.12 & 0.11: Read_Assembled_Data & Program_Assembled – These bits are used by the AMCI_MA_Program_Assembled_Move AOI to program the segments of an Assembled Move before the move can be run.

Bit 0.10: Reset_Errors – Exclusive Bit. Set by the AMCI_MAFR AOI. When this bit makes a 0→1 transition, the unit will clear all existing command errors, reset the *Input_Error* bit, and reset the *Move_Complete* bit in the Network Input Data. This bit does not clear the *Position_Invalid* status bit.

Bit 0.9: Preset_Motor_Position – Exclusive Bit. Set by the AMCI_MRP. 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 1.13). If the *Preset_To_Encoder* bit equals "0", the Motor Position is preset to the value stored in Position_or_Distance tag. 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".

Format Output Data Tags (continued)

Boolean Control Bits (continued)

Bit 0.8: **Jog_CCW –** Exclusive Bit. Set by AMCI_MAJ, AMCI_Registration_Move, and motion axis follower AOIs. When this bit becomes set, the unit will run the move in the counter-clockwise direction. The full explanation of ***Jog Moves*** can be found starting on page 34.

- **Registration_Move – Command Word 1.7:** When this bit equals "0", and an AMCI_MAJ AOI is enabled, it will run as a standard Jog Move. When this bit equals "1" and an AMCI_MAJ AOI is enabled, , the move will run as a Registration Move.
- **Synchrostep Mode – Command Word 1.12:** If this bit equals "1" and the Registration_Move bit equals "0", the SMD17Lwill enter its Synchrostep Mode when the Jog Move bit makes a 0→1 transition.

Bit 0.7: **Jog_CW –** Exclusive Bit. Set by AMCI_MAJ, AMCI_Registration_Move, and motion axis follower AOIs. When this bit becomes set, the unit will run the move in the clockwise direction. The full explanation of ***Jog Moves*** can be found starting on page 34.

- **Registration_Move – Command Word 1.7:** When this bit equals "0", and an AMCI_MAJ AOI is enabled, it will run as a standard Jog Move. When this bit equals "1" and an AMCI_MAJ AOI is enabled, , the move will run as a Registration Move.
- **Synchrostep Mode – Command Word 1.12:** If this bit equals "1" and the Registration_Move bit equals "0", the SMD17Lwill enter its Synchrostep Mode when the Jog Move bit makes a 0→1 transition.

Bit 0.6: **Find_Home_CCW –** Exclusive Bit. Set by the AMCI_MAH AOI. 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 SMD17L*** reference chapter starting on page 57.

Bit 0.5: **Find_Home_CW –** Exclusive Bit. Set by the AMCI_MAH AOI. 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 SMD17L*** reference chapter starting on page 57.

Bit 0.4: **Immediate_Stop –** Exclusive Bit. 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.

Bit 0.3: **Resume_Move –** Exclusive Bit. When this bit makes a 0→1 transition, the unit will resume a move that you previously placed in a hold state. The AMCI_MAS AOI will place a move in its hold state. Use of the *Resume_Move* and *Hold_Move* bits can be found in the ***How to Control Moves in Progress*** section of this manual starting on page 29. Note that a move in its hold state need not be resumed. The move is automatically canceled if another move is started in its place.

Bit 0.2: **Hold_Move –** Exclusive Bit. Set by the AMCI_MAS AOI. 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 ***How to Control Moves in Progress*** section of this manual starting on page 29.

Bit 0.1: **Relative_Move –** Exclusive Bit. Set by the AMCI_MAM AOI. When this bit makes a 0→1 transition, the unit will perform a Relative Move using the data in the rest of output data. The full explanation of a ***Relative Move*** can be found starting on page 33.

Bit 0.0: **Absolute_Move –** Exclusive Bit. Set by the AMCI_MAM AOI. When this bit makes a 0→1 transition, the unit will perform an Absolute Move using the data in the rest of the output data. The full explanation of an ***Absolute Move*** can be found starting on page 32.

Format Output Data Tags (continued)

Boolean Control Bits (continued)

Bit 1.15: Enable_Driver – "0" to disable the motor current, A 0→1 transition enables motor current if a fault does not exits.

Bit 1.14: Reserved – Must equal "0".

Bit 1.13: Preset_to_Encoder – Only used when the Preset Motor Position bit (Command Word 0.9) is set to "1". See the Preset_Motor_Position bit description previously in this section for further information. The MSO_and_Preset_to_Absolute encoder performs this function.

Bit 1.12: Virtual_Position_Follower – Only used when the Jog_CW or Jog_CCW bits (Command Word 0.8 and 0.7) are set to "1". See the Jog_CW and Jog_CCW bit descriptions previously in this section for more information. This bit is set by the Linear and Circular Follower AOIs.

Bit 1.11: Backplane_Proximity_Bit – When the SMD17L 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* while homing the machine can be found in the ***Profile with Backplane_Proximity_Bit*** section found on page 60.

Bit 1.10: Reserved – Must equal "0".

Bit 1.9: Assembled_Move_Type – When this bit equals "0", a Blend Move is started when the Run_Assembled_Move bit, (Command Word 0.13) makes a 0→1 transition. When this bit equals "1", a Dwell Move is started on the transition. See the Run_Assembled_Move bit description previously in this section for more information.

Bit 1.8: Indexed_Command – If this bit is set when a move command is issued, the SMD17L 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. Most of the AMCI motion AOIs have this bit as a hidden parameter.

Bit 1.7: 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. This bit is automatically set by the AMCI_Registration_Move AOI.

Bit 1.6: Reserved – Must equal "0".

Bit 1.5: Save_to_Flash - This bit can be used to save the absolute encoder position offset to flash. (The absolute encoder position offset is generated by the AMCI_MA_Preset_Encoder AOI.)

- When using this bit to save the calculated absolute encoder offset value to flash memory, the proper bit must be set to its "1_yes" value when the AMCI_MA_Preset_Encoder AOI is issued. If the offset is calculated and stored without error, the AOI will respond by setting the Command_Acknowledge bit. (See page 89 for the description of the Command_Acknowledge bit.)

Bit 1.4: 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. This bit is an input parameter of the AMCI_MA_Run_Blend_Move AOI.

Bit 1.3: Reserved – Must equal "0".

Bit 1.2: Reserved – Must equal "0".

Bit 1.1: Motor Current – If reset to "0", the motor current will be the last value written to the unit. If set to "1" the motor current is set to the value specified in the Ouptut_Word_8 tag.

Bit 1.0: Reserved – Must equal "0".

Format Output Data Tags (continued)

Other Output Tags

Position_or_Distance: The target position or relative offset programmed into an AMCI_MAM AOI. Also used by other AOIs. One example is the AMCI_MRP AOI that uses this tag to define the position to preset to.

Programmed_Speed_st_s: The target velocity programmed into the AOI.

Acceleration_st_s_ms: The acceleration programmed into the AOI.

Deceleration_st_s_ms: The deceleration programmed into the AOI.

Set_Current_or_MSW_Min_Reg: This register allows you to change the motor current if output bit 1.1 is set. (See page 93). This tag is also used to specify the upper 16 bits of the Mininum Registration Move Distance parameter of a Registration Move.

Jerk_or_LSW_Min_Reg: Add-On Instructions that use the Jerk parameter use this tag to specify the new value. This tag is also used to specify the upper 16 bits of the Mininum Registration Move Distance parameter of a Registration Move.

AOI Reference (RA)

This chapter lists the AMCI add-on instructions, their parameters, and their enumerations.

AOI List

AMCI_IOL_Input_conversion

IO_Link Input Data Conversion. This AOI is required if you use the other AMCI AOIs. This AOI converts the IO-Link input data, which is a forty-eight SINT array, into the data structure used by the AMCI AOIs.

Parameter	Description
Input_Data_from_AMCI_IOLink_Device	Input data from the SMD17L. Use the buffered data created by the CPS instruction in step 3.5 on page 84 instead of the input tags directly associated with the unit.
AMCI_Motion_Axis_Input_Data	The destination User_Defined data tag for the conversion. This tag must have the AMCI_Motion_Axis_Input_Data data type.

AMCI_IOL_Output_conversion

IO_Link Output Data Conversion. This AOI is required if you use the other AMCI AOIs. This AOI converts the data structure used by the AMCI AOIs into the forty-eight SINT array used as the IO-Link output data.

Parameter	Description
AMCI_Motion_Axis_Output_Data	The source User_Defined data tag for the conversion. This tag must have the AMCI_Motion_Axis_Output_Data data type.
Output_Data_to_AMCI_IOLink_device	Output data to the SMD17L. The destination should be the output data buffer used by the output CPS instruction instead of the output tags directly associated with the unit. See step 3.5 on page 84 for additional information on buffering the SMD17L I/O data.

AMCI_MAFR

Motion Axis Fault Reset

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	A command or input error failed to clear.	Rung is false

AMCI_MAH

Motion Axis Home

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Direction	0 = Start Motion is Positive (CW) : 1 = Start Motion is Negative (CCW)
Programmed_Speed	Maximum speed of the move in steps/second. Range: Configured Starting Speed to 2,999,999
Acceleration	1 to 5000 steps/second/millisecond
Deceleration	1 to 5000 steps/second/millisecond

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false
PC (Process Complete)	The At_Home status bit is set.	The next time the rung containing the AOI transitions from false to true.

AMCI_MAJ

Motion Axis Jog

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Direction	0 = Positive (CW) Motion : 1 = Negative (CCW) Motion
Programmed_Speed	Maximum speed of the move in steps/second. Range: Configured Starting Speed to 2,999,999
Acceleration	1 to 5000 steps/second/millisecond
Deceleration	1 to 5000 steps/second/millisecond

AMCI_MAJ (continued)

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, Invalid Profile or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or the rung goes false

AMCI_MAM

Motion Axis Move

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Move_Type	0 = Absolute Move ： 1 = Relative Move
Position_or_Distance	Target position of Absolute move or distance to move with a Relative move. Range of –8,388,607 to +8,388,607
Programmed_Speed	Maximum speed of the move in steps/second. Range: Configured Starting Speed to 2,999,999
Acceleration	1 to 5000 steps/second/millisecond
Deceleration	1 to 5000 steps/second/millisecond

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false
PC (Process Complete)	Move is complete	The next time the rung containing the AOI transitions from false to true.

AMCI_MAS

Motion Axis Stop

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false
PC (Process Complete)	Move is complete	The next time the rung containing the AOI transitions from false to true.

AMCI_MRP

Motion Redefine Position

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Position	Range of -2,147,483,648 to +2,147,483,647. (Signed 32 bit integer)
Save_In_Flash	Only used by the SV family of Servo products, this parameter should always be set to “0” for the SMD17L.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false

AMCI_MSO_Enable

(Motion Servo On)

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false

AMCI_MSF_Disable

(Motion Servo Off)

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false

AMCI_MSO_and_Preset_to_Absolute_Encoder

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false

AMCI_MA_Preset_Encoder

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Position	Range of -2,147,483,648 to +2,147,483,647 (Signed 32 bit integer)
Save_In_Flash	Reset to “0” to store the calculated offset stored in RAM memory. Set to “1” to store the calculated offset in non-volatile Flash memory.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false

AMCI_MA_Registration_Move

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Direction	Reset to '0' to run the Registration Move in the clockwise direction. Set to '1' to run the Registration Move in the counter-clockwise direction.
Stopping_Distance	The number of steps to move after the registration input has become active. Range of 0 to +8,388,607.
Programmed_Speed	Maximum speed of the move in steps/second. Range of (Configured Starting Speed) to 2,999,999
Acceleration	1 to 5000 steps/second/millisecond
Deceleration	1 to 5000 steps/second/millisecond
Minimum_Registration_Distance	The number of steps that are output before the registration input is enabled. Range of 0 to +8,388,607.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false

AMCI_MA_Reset_Driver_Fault

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false

AMCI_MA_Resume

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
Not_In_Hold_State	This AOI is enabled, but a move is not in a Hold state.	Rung is false.
IP (In Process)	Motion is occurring	Motion stops or rung goes false

AMCI_MA_SD_SMD_Circular_Follower

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Follower_Position	Must point to the position tag of the virtual axis the unit is following.
Follower_Velocity	Must point to thevelocity tag of the virtual axis the unit is following.
Acceleration	Range of 1 to 5000. Higher values will force the unit to follow the axis more closely.
Deceleration	Range of 1 to 5000. Higher values will force the unit to follow the axis more closely.
Proportional_Coefficient	Range of 1 to 50. This parameter controls the sharpness of the control loop. Suggested starting values of 1 or 2.
Conversion_Constant	The virtual axis may not report values in terms of motor steps and motor steps per second. For example, the axis might output its values in revolutions and revolutions per second. The circular follower AOI multiplies the position and velocity from the virtual axis by this conversion constant to convert it to step and steps / sec units used by the SMD17L. In this example, if the Conversion_Constant is set to the configured Motor Resolution, one revolution of the virtual axis will cause one rotation of the motor shaft.
Position_Unwind	Range of 21 to 65,535. This parameter defines the point at which the motor axis position rolls over and returns to zero.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false

AMCI_MA_SD_SMD_Linear_Follower

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Follower_Position	Must point to the position tag of the virtual axis the unit is following.
Follower_Velocity	Must point to the velocity tag of the virtual axis the unit is following.
Acceleration	Range of 1 to 5000. This can be a constant or point to an INT tag. Higher values will force the unit to follow the axis more closely.
Deceleration	Range of 1 to 5000. This can be a constant or point to an INT tag. Higher values will force the unit to follow the axis more closely.
Proportional_Coefficient	Range of 1 to 50. This parameter controls the sharpness of the control loop. Suggested starting values of 1 or 2.
Network_Delay	Range of 0 to 20. Programmed in milliseconds, this value can be used to compensate for communication delays. A value between 0 and the configured network update time will work for most applications.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false

AMCI_MA_Program_Assembled_Move

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Number_of_Segments	Range of 2 to 16. The number of segments in the blend move.
Segment_'n'_Distance	Range of 1 to 8,388,607. The distance to travel in segment 'n'.
Segment_'n'_Programmed_Speed	Range of (Configured Starting Speed) to 2,999,999. The target speed of segment 'n'.
Segment_'n'_Acceleration	Range of 1 to 5,000. Acceleration that is used at the start of segment 'n' if the programmed speed of the previous segment was less than the programmed speed of the current segment.
Segment_'n'_Deceleration	Range of 1 to 5,000. Deceleration that is used at the start of segment 'n' if the programmed speed of the previous segment was greater than the programmed speed of the current segment. Also used to decelerate to the Starting Speed if the current segment is the last of the move.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
ER (Error)	There is an Input, Command, Invalid Profile or Configuration error	Rung is false
Assembled_Move_Programmed	All segments of the Assembled Move have been programmed	Rung is false

AMCI_MA_Run_Blend_Move

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Direction	Reset to '0' to run the Blend Move in the clockwise direction. Set to '1' to run the Blend Move in the counter-clockwise direction.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false

AMCI_MA_Run_Dwell_Move

Parameter	Description
Axis_Input_Data	Input data from the SMD17L. Use the AMCI_Motion_Axis_Input_Data User Defined tag that is the destination of the AMCI_IOL_Input_conversion AOI instead of the input tags associated with the unit.
Axis_Output_Data	Output data to the SMD17L. Use the AMCI_Motion_Axis_Output_Data User Defined tag that is the source of the AMCI_IOL_Output_conversion AOI instead of the input tags associated with the unit.
Dwell_Time	Range of 1 to 65,535. The time in milliseconds between move segments of the Dwell Move.

Enumeration	Set When...	Reset When...
EN (Enable)	Rung is true	Rung is false
DN (Done)	Command is sent to the module	Rung is false
ER (Error)	There is an Input, Command, or Configuration error	Rung is false
IP (In Process)	Motion is occurring	Motion stops or rung goes false

Notes

REFERENCE 6 (RA)

INPUT STATUS BITS

This reference lists the most commonly used status bits and the conditions under which they are set.

Stopped Bit

Set whenever the axis in not in motion for any reason.

At_Home Bit

Set whenever an AMCI_MAH AOI completes successfully.

Move_Complete Bit

Set after the completion of an Absolute Move, Relative Move, Jog Move, Registration Move, or Assembled Move. Also set at the end of a SynchroStep Move when the bits that started the SynchroStep Move are reset. Note that in this case the Move_Complete bit is not set if the virtual axis comes to a stop while the SMD17L follows the axis. The Move Complete bit will be reset when the next move operation is initiated or when an MAFR AOI is accepted.

Position_Invalid Bit

1) After a power cycle to the module.
2) After the module has been re-configured. This will be true even if the position was valid before the configuration operation occurred.
3) If the Emergency Stop input has been activated while motion is occurring.
4) If either of the End Limit Switches become active during any move.
5) If the same end limit is reached during a Jog or Registration move. For example, if the CW Limit Switch is reached during a Jog move in the clockwise direction.
6) If the opposite End Limit Switch is reached during a homing operation. For example, if the CCW Limit Switch is the first switch reached during a CW homing operation. Improper wiring or placement of the limit switches are the most likely causes of this error.
7) If both End Limit Switches are reached during a homing operation. This error will only occur when the motor has reversed direction after encountering the first End Limit switch. This could indicate a problem with the Home Limit Switch wiring because it stayed in its inactive state across the entire mechanical span of the machine.

Input_Error Bit

1) If an Emergency Stop input is activated while motion was occurring.
2) If either of the End Limit Switches become active during any move.
3) If the same end limit is reached during a Jog or Registration move. For example, if the CW Limit Switch is reached during a Jog move in the clockwise direction.
4) Performing a Jog move in the same direction as the active End Limit Switch. For example, starting a CW Jog when the CW Limit Switch is active.
5) If the opposite End Limit Switch is reached during a homing operation. For example, if the CCW Limit Switch is the first switch reached during a CW homing operation. Improper wiring or placement of the limit switches are the most likely causes of this error.
6) If both End Limit Switches are reached during a homing operation. This error will only occur when the motor has reversed direction after encountering the first End Limit switch. This could indicate a problem with the Home Limit Switch wiring because it stayed in its inactive state across the entire mechanical span of the machine.

Command_Error Bit

General Command Errors

1) If a move operation is started before the previous move operation is completed. The current move will run to completion.
2) If the Target Position, Target Speed, Acceleration, or Deceleration parameters are outside of their valid ranges.
3) If the AOI's Target Speed is less than the configured Starting Speed. This applies to Absolute Moves, Relative Moves, Homing operations, and Blend Moves.
4) Performing an Absolute Move when the Current Position is not valid. (The Position_Invalid bit equals "1")
5) Triggering an AMCI_MAS AOI when a move is not in progress.
6) Triggering an AMCI_MA_Resume AOI when the channel is not in a Hold state. The current move will run to completion and the Command_Error bit will be set at the end of the move.
7) Triggering an AMCI_MA_Preset_Encoder AOI when the encoder has been disabled in configuration.
8) If more than one Command bit in output word 0 (MSW) is set. This cause should not occur when using the AIO's.

Jog Command Errors

9) If invalid changes are made to the Target Velocity, Acceleration, or Deceleration parameters of a Jog move that is running. Please note that the Acceleration and Deceleration values are only checked when a change is made to the Target Velocity.

Assembled Move Command Errors

10) If an Blend or Dwell Move operation is triggered before a profile is programmed.

Registration Move Command Errors

11) If a Registration move is triggered and the Stopping_Distance parameter is negative.

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