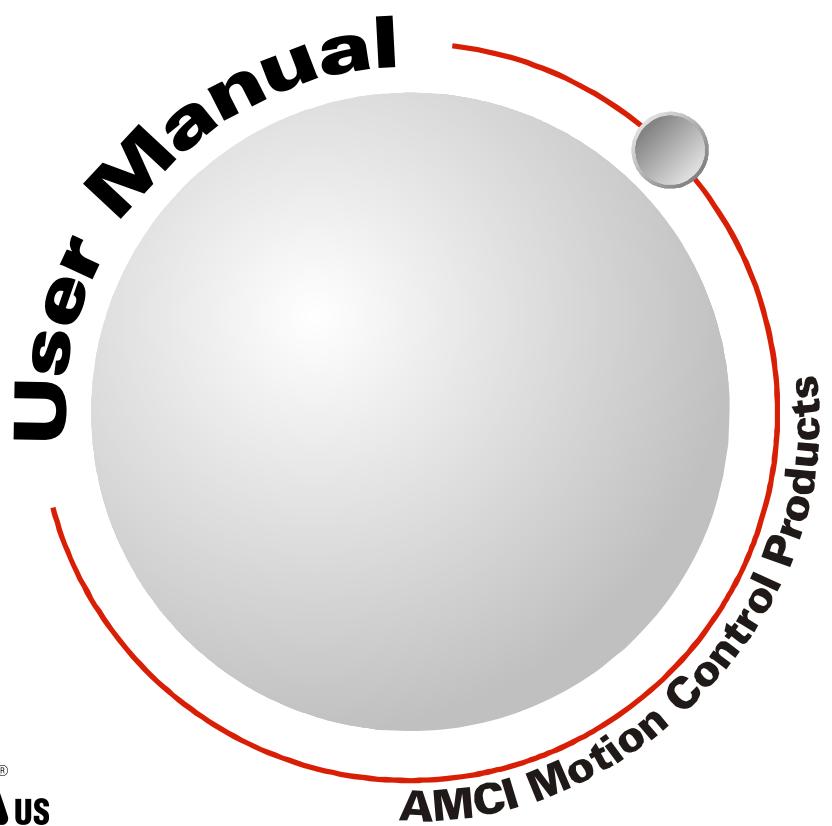


SMD17K

**Integrated Stepper
Indexer/Driver/Motor
with EtherCAT Interface**



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.

This manual is copyright 2020 by Advanced Micro Controls Inc. You may reproduce this manual, in whole or in part, for your personal use, provided that this copyright notice is included. You may distribute copies of this complete manual in electronic format provided that they are unaltered from the version posted by Advanced Micro Controls Inc. on our official website: www.amci.com. You may incorporate portions of this documents in other literature for your own personal use provided that you include the notice "Portions of this document copyright 2020 by Advanced Micro Controls Inc." You may not alter the contents of this document or charge a fee for reproducing or distributing it.

Standard Warranty

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

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

Returns Policy

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

24 Hour Technical Support Number

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

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

Waste Electrical and Electronic Equipment (WEEE)



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

TABLE OF CONTENTS

General Information

Important User Information	2
Standard Warranty	2
Returns Policy	2
24 Hour Technical Support Number	2
WEEE	2

About this Manual

Audience	7
Applicable Units	7
Navigating this Manual	7
Manual Conventions	7
Trademark Notices	7
Revision Record	8
Manual Layout	8

Reference: SMD17K Specifications

The SMD17K Family	9
Part Numbering System	10
General Functionality	10
Encoder Functionality	11
Conformance Markings	11
UL	11
CE	11
RoHS	11
Specifications	12
Indexer Functionality	13
Synchronizing Moves	14
Stall Detection with SMD17K Units	14
Driver Functionality	15
Idle Current Reduction	15
Available Discrete Inputs	15
Home Input	15
CW Limit Switch or CCW Limit Switch	16
Start Indexer Move Input	16
Emergency Stop Input	16
Stop Jog or Registration Move Input	16
Capture Encoder Position Input	16
General Purpose Input	16
Optional Encoder	17
Incremental Encoder	17
Absolute Multi-turn Encoder	17
Status LED's	17
Run LED	17
Error LED	18

Reference: SMD17K Specifications (continued)

SMD17K Connectors	19
Input Connector	19
Ethernet Connectors	19
Torque and Power Curves	20
Power Supply Sizing	20
Regeneration Effects	21
Compatible Connectors and Cordsets	22
Connectors	22
Ethernet Cordset	22
Power Cordsets	22

Reference: Motion Control

Definitions	23
Units of Measure	23
Motor Position	23
Home Position	23
Count Direction	23
Starting Speed	23
Target Position	24
Definition of Acceleration Types	24
Linear Acceleration	24
Triangular S-Curve Accel.	25
Trapezoidal S-Curve Accel.	25
A Simple Move	26
Controlled and Immediate Stops	27
Host Control	27
Hardware Control	27
Basic Move Types	28
Relative Move	28
Absolute Move	29
CW/CCW Jog Move	30
CW/CCW Registration Move	31
Assembled Moves	32
Blend Move	33
Dwell Move	34
Assembled Move Programming	36
Control Bits – Output Data	36
Control Bits – Input Data	36
Programming Routine	36
Saving an Assembled Move in Flash	36
Indexed Moves	37
Controlling Moves In Progress	38
Jog Moves	38
Registration Moves	38
Absolute and Relative Moves	38
Assembled Moves	38

Reference: Calculating Move Profiles

Constant Acceleration Equations	39
Variable Definitions	39
Total Time Equations	41
S-Curve Acceleration Equations	42
Triangular S-Curve Accel.	42
Trapezoidal S-Curve Accel.	44
Determining Waveforms by Values	46

Reference: Homing an SMD17K

Definition of Home Position	49
Position Preset	49
CW/CCW Find Home Commands	49
Homing Inputs	50
Physical Inputs	50
Network Data Input	50
Homing Configurations	50
Homing Profiles	51
Home Input Only Profile	51
Profile with Backplane_Proximity_Bit	52
Profile with Overtravel Limit	53
Controlling Find Home	
Commands In Progress	54
Controlled Stop Conditions	54
Immediate Stop Conditions	54

Reference: Configuration Data Format

CoE Registers	55
Data Format	55
CFG_Word_0 Format	55
CFG_Word_1 Format	58
Notes on Other Configuration Words	58
Invalid Configurations	58

Reference: Command Data Format

Command Bits Must Transition	59
Output Data Format	59
CMD_word0	60
CMD_word1	62
Command Blocks	63
Absolute Move	63
Relative Move	64
Hold Move	64
Resume Move	65

**Reference: Command Data Format
(continued)**

Immediate Stop	65
Find Home CW	66
Find Home CCW	66
Jog CW	67
Registration Move CW	67
Jog CCW	68
Registration Move CCW	68
Preset Position	69
Reset Errors	69
Run Assembled Move	70
Preset Encoder Position	70
Programming Blocks	71
First Block	71
Segment Block	71
Input Data Format	72
STATUS_word0 Format	72
STATUS_word1 Format	74
Notes on Clearing a Driver Fault	75

Task 1: Installing the SMD17K

Location	77
IP50 Rated Units	77
IP64 Rated Units	77
IP65/7 Rated Units	77
Safe Handling Guidelines	77
Prevent Electrostatic Damage	77
Prevent Debris From Entering the Unit	78
Remove Power Before Servicing ..	78
Operating Temperature Guidelines	78
Mounting	78
SMD17K-M12 Mounting	78
SMD17K-M12S Mounting	79
SMD17K-M12P Mounting	79
SMD17K-80 Outline Drawing	79
Connecting the Load	79
Power and Input Connector	80
Compatible Connectors and Cordsets	80
Power Wiring	81
MS-31 Connector	81
CNPL-2M and CMPL-5M Cables	81
Extending Power Leads	81
Main Power Wiring Only	81
Auxiliary Power, Single Supply	82
Auxiliary Power, Dual Supply	82
Power Supply Troubleshooting	82

**Task 1: Installing the SMD17K
(continued)**

Input Wiring	83
Cable Shields	83
Sinking Sensors Require a Pull Up Resistor	83
Network Connectors	84
Compatible Connectors and Cordsets	84
TIA/EIA-568 Color Codes	84

**Task 2: EtherCAT System
Configuration**

Install the ESI file	85
Obtain the ESI file	85
Install the ESI file	85
Restart the Programming System If Needed	85
Add the SMD17K to the Project	85
Scan for the SMD17K Device	85
Rename the Device	85
Configure the SMD17K	86
Create a DUT	87
Create Variables Based on the DUT	87
Link Variable Names to I/O Words	87

**Chapter 3: Distributed Clock -
SYNC0 Setup**

Verify Main PLC Task Timing	89
Create New PLC Task	90
Set Operational Mode	92
Set CFG_word1 Value to Enable SYNC0	93

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 SMD17K 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.

Applicable Units

This manual applies to all of the units in the SMD17K family.

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.
<i>Emphasis Font</i>	Font used the first time a new term is introduced.
<i>Cross Reference</i>	When viewing the PDF version of the manual, clicking on the cross reference text jumps you to referenced section.
<i>HTML Reference</i>	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.

Trademark Notices

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

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

Revision Record

This manual, 940-0S292, is the third release of this manual. It corrects typographical errors and adds some clarification notes to command and status bits. It was first released December 15th, 2020.

Revision History

940-0S291: November 20th, 2020 - Added UL Recognized Component information

940-0S290: October 28th, 2020 - Initial Release

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, **SMD17K 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
SMD17K Specifications	9	Complete specifications for the SMD17K products.
Motion Control	23	Reference information on how the SMD17K can be used to control motion in your application.
Calculating Move Profiles	39	Reference information on calculating detailed move profiles.
Homing an SMD17K	49	Reference information on how to set the home position of the SMD17K.
Configuration Data Format	55	Reference information on the format of the CANopen over EtherCAT (CoE) data that is used to configure the units.
Command Data Format	59	Reference information on the format of the network data to and from the SMD17K that is used to command it.
Installing the SMD17K	77	Task instructions covering how to install an SMD17K on a machine. Includes information on mounting, grounding, and wiring specific to the units.
EtherCAT System Configuration	85	Task instructions on how to add an SMD17K to an EtherCAT network.
Distributed Clock - SYNC0 Setup	89	Optional task instructions that covers how to configure an SMD17K to use the Distributed Clock functionality of the EtherCAT protocol.

Manual Sections

REFERENCE 1

SMD17K SPECIFICATIONS

This manual is designed to get you up and running quickly with an SMD17K 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 SMD17K Family

The SMD17K units are part of a growing product line from AMCI with a simple concept: a stepper indexer, driver, and motor that can be attached to any popular industrial network. Each SMD17K attaches to your Ethernet based network and communicates using the EtherCAT protocol.

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

An EtherCAT Slave Information (ESI) file is available for the SMD17K units. The ESI file is required to add the device to the network. Configuration of the SMD17K is accomplished using the CANopen PDO and SDO objects. EtherCAT supports CANopen through its CANopen over Ethernet (CoE) interface. Motion commands and status information is transmitted using the output and input variables assigned to the SMD17K when the EtherCAT system is configured.

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

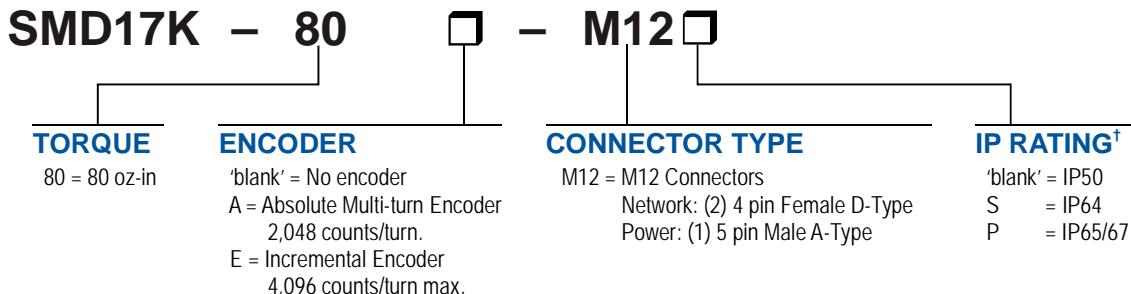
Each unit can be ordered with an optional incremental or absolute multi-turn encoder. This encoder gives you the additional functionality of position verification and stall detection. The absolute multi-turn encoder allows you to track machine position with power removed, eliminating the need to home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

Three different IP ratings are available in the SMD17K product lines.

- IP50: Units have sealed M12 D-coded connectors for their ethernet ports. A sealed M12 A-coded connector is used for power and two digital inputs.
- IP64: Units have a shaft seal and sealed M12 D-coded connectors for their ethernet ports. A sealed M12 A-coded connector is used for power and two digital inputs.
- IP65/67: Units have a shaft seal and sealed M12 D-coded connectors for their ethernet ports. A sealed M12 A-coded connector is used for power and two digital inputs. Additionally, the units are sealed with an FDA approved epoxy.



Figure R1.1 IP50 Rated SMD17K

The SMD17K 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 SMD17K family has three integrated parts:

- An indexer that accepts commands over an EtherCAT 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 is also available for applications that require position feedback or verification.

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 SMD17K 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 SMD17K is powered by a nominal 24 to 48 Vdc power source, and can accept surge voltages of up to 60 Vdc without damage. The output motor current is fully programmable from 0.1 Arms to 2.0 Arms which makes the SMD17K suitable to a wide range of applications. In addition to the Motor Current setting, the Motor Steps per Turn, Idle Current Reduction, and Anti-Resonance Circuit features are also fully programmable. If you have used other stepper indexer products from AMCI, you will find programming an SMD17K to be very similar to these products.

The SMD17K 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 SMD17K Family (continued)**General Functionality (continued)**

The SMD17K units have two DC inputs that are used by the indexer. Configuration data from the host sets the function of these inputs. Each input can be individually configured as a:

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

Encoder Functionality

All SMD17K units can be ordered with an internal incremental or absolute multi-turn encoder. Incremental encoders can be programmed to 1,024, 2,048, or 4,096 counts per turn. Absolute encoders have a fixed resolution of 2,048 counts per turn and encode a total of 2^{21} turns. (32 bits total.) Using an encoder gives you the ability to:

- Verify position during or after a move
- Detect motor stall conditions
- Maintain machine position when power is removed if using an absolute encoder.

The motor position can be preset to the encoder position with a single command. SMD17K units with absolute encoders allow you to preset the encoder position and save the resulting offset in Flash memory.

Conformance Markings

The SMD17K family meets the requirements for the following conformance markings when installed and operated in accordance with the instructions contained in the product documentation.

UL

Recognized component in the USA and Canada.

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

CE

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

RoHS

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

Specifications**Network Interface**

100Base-TX. Two M12 4 pin D-coded connectors.
Supports all standard EtherCAT topologies.

Physical Dimensions

See Outline Drawing, page 79

Weight

SMD17K-80-M12 1.07 lb. (0.49 kg)
Weight is without mating connectors

Maximum Shaft Loads

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

Rotor Inertia

82 g-cm²

Maximum Operating Temperature[†]

203°F / 95°C Note that this is the internal temperature of the driver electronics, not maximum ambient temperature. The temperature of the motor is not directly measured.

Over Temperature Fault

Over temperature faults are detected and reported.

Inputs

Electrical Characteristics:

Single ended sinking.
Accept 3.5 to 27Vdc without the need for an external current limiting resistor. Can accept surges up to 35 Vdc without damage.

Motor Current

Programmable from 0.1 to 2.0 Arms in 0.1 A steps.

DCPower_{AUX} Current

70 mA @ 24 Vdc, 40mA @48 Vdc

Motor Counts per Turn

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

Internal Encoder (Optional)

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

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

Idle Current Reduction

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

Environmental Specifications[†]

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

Ambient Operating Temperature -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 17.

Connectors and Cables

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

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

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

[†] 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 SMD17K.

Feature	Description
Synchronous Moves	Using the Distributed Clock functionality of the EtherCAT system, multiple devices can synchronize the start of their moves to the SYNC0 signal. Moves will begin within ±25 microseconds of each other.
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 SMD17K 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.
Indexer Move	Allows you to program a move that is located in the output registers. The move is run when one of the programmable inputs makes a transition. Note that an Indexer Move requires a connection to a host controller to program the move.
Hold Move	Allows you to suspend a move, and optionally restart it, without losing your position value.
Resume Move	Allows you to restart a previously held move operation.
Immediate Stop	Allows you to immediately stop all motion if an error condition is detected by your host controller.
Stall Detection	When an SMD17K is purchased with an encoder option, the encoder can be used to verify motion when a move command is issued.

Table R1.1 Indexer Functionality

Indexer Functionality (continued)**Synchronizing Moves**

By default, the SMD17K products use the SyncManager 2 event to control the transfer of data from the EtherCAT Slave Controller (ECS) to the microprocessor that controls motion. This allows the SMD17K to execute commands as soon as new data arrives. When more than one axis is updated with a single EtherCAT packet, the time difference between axis update is very short. Updating multiple axes with a single packet allows the EtherCAT network to out perform the update times of other industrial network protocols.

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

The EtherCAT Distributed Clock functionality is built into the ECS used by the SMD17K products. The SMD17K can act as the reference clock for the system if it is the first device in the EtherCAT network. The SYNC0 signal, which is based off of the Distributed Clock, can be used to synchronize the start of moves over multiple devices. The time between when the SYNC0 signal is received by the main processor of the SMD17K and when the driver begins to cause motion is 520 ± 25 microseconds. The 520 microseconds is the time required to read the data from the ECS once the SYNC0 signal becomes active. The ± 25 microseconds is caused by the 20 kHz update frequency of the PWM drivers.

For the SMD17K, the minimum update time on the SYNC0 signal is two milliseconds. If the task time is less than two milliseconds, the SYNC0 time must be a multiple of the task time.

Stall Detection with SMD17K Units

Stall Detection is one of the additional features available to you when you order an encoder option on an SMD17K. When Stall Detection is enabled, the SMD17K 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 SMD17K 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 SMD17K can notify the system if the motor shaft moves more than forty-five degrees while power is removed from the motor. This may occur when the motor must hold a load against gravity or other external forces affect the load when power is removed.

As described later in this chapter, there is an auxiliary power pin that powers the electronics of an SMD17K but does not power the motor. The primary use of this feature is to keep the unit on the network while power is removed from the motor. When using the DCPower_{AUX} pin, the SMD17K cannot sense when power has been removed from the DCPower_{MAIN} pin. Stall detect can be used in these cases as well. If motion is commanded and a stall detection event occurs immediately, it is likely that the main power has been removed from the motor and the SMD17K is running on DCPower_{AUX} power only.

Driver Functionality

This table summarizes the features of the stepper motor driver portion of an SMD17K.

Feature	Benefits
RMS Current Control	RMS current control give an SMD17K 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 SMD17K 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 SMD17K sets a warning bit in the network data when the temperature of the unit approaches its safe operating threshold.
Over Temperature Protection	Protects your SMD17K 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

Idle Current Reduction

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

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

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

Available Discrete Inputs

The SMD17K has two discrete, sinking, DC inputs that accept 3.5 to 27 Vdc signals. (5 to 24 Vdc nominal) How your SMD17K uses these inputs is fully programmable. The active state of each input is also programmable. Programming their active states allow them to act as Normally Open (NO) or Normally Closed (NC) contacts.

Home Input

Many applications require that the machine be brought to a known position before normal operations can begin. This is commonly called “homing” the machine or bringing the machine to its “home” position. An SMD17K allows you to define this starting position in two ways. The first is with a Position Preset command. The second is with a sensor mounted on the machine. When you define one of the inputs as the Home Input, you can issue commands to the SMD17K that will cause the unit to seek this sensor. How the SMD17K actually finds the Home sensor is described in the [Homing an SMD17K](#) chapter starting on page 49.

Available Discrete Inputs (continued)**CW Limit Switch or CCW Limit Switch**

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

Start Indexer Move Input

Indexer Moves are programmed through the Network Data like every other move. The only difference is that Indexer Moves are not run until a Start Indexer Move Input makes a inactive-to-active state transition. This allows an SMD17K to run critically timed moves that cannot be reliably started from the network due to data transfer lags.

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

Emergency Stop Input

When an input is defined as an Emergency Stop, or E-Stop Input, motion will immediately stop when this input becomes active. The driver remains enabled and power is supplied to the motor. Any type of move, including a Jog or Registration Move, cannot begin while this input is active.

Stop Jog or Registration Move Input

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

If the unit was order with an integral encoder, the encoder position data will be captured when the DC input makes an inactive-to-active transition if it is configured as a Stop Jog or Registration Move Input. The encoder position data is not captured if a Jog or Registration Move is not in progress. If you want to capture encoder position data on every transition of a DC input, configure it as a Start Indexer Move Input.

Capture Encoder Position Input

As described in the [**Start Indexer Move Input**](#) and [**Stop Jog or Registration Move Input**](#) sections above, an SMD17K can be configured to capture the encoder position value on a transition of a discrete DC input.

General Purpose Input

If your application does not require one or both of the inputs, you can configure the unused inputs as General Purpose Inputs. The inputs are not used by the SMD17K, but their on/off state is reported in the network data and is available to your host controller.

Optional Encoder

The SMD17K can be ordered with an integral encoder. The encoder can be used for position verification and stall detection. Additionally, an input can be configured to capture the encoder value when the input makes an inactive to active transition. This captured value is written to the host controller. Two encoder options are available:

Incremental Encoder

The incremental encoder can be programmed to 1,024, 2,048, or 4,096 counts per turn. The SMD17K has an internal thirty-two bit counter associated with the encoder. The incremental encoder are primarily used for position verification and stall detection.

Absolute Multi-turn Encoder

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

Like many intelligent absolute encoders on the market today, the absolute encoder in the SMD17K uses a battery backed circuit to count zero crossings while power is removed from the rest of the device. The circuit will accurately track position as long as the shaft acceleration is limited to 160,000 degrees/sec², (444.4 rev/sec²), or less. Battery life is a minimum of 10 years in the absence of power. The battery cannot be replaced in the field.

Status LED's

Each SMD17K has two status LED's that shows the status of the functional status of the device. As shown in figure R1.3, these LEDs are located on the rear cover. They are named "RUN" and "ERR". Two additional LEDs are located on the near cover between the two network connectors. These two LEDs are the LINK/ACT LEDs for the Ethernet ports. They are on when there is a physical connection between the device and the previous or next device in the network. They blink when data is being transmitted over the network.

Run LED

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

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

Table R1.3 Module RUN LED States

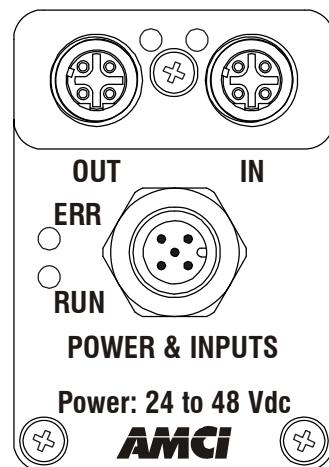


Figure R1.3 SMD17K Status LED's

Status LED's (continued)**Error LED**

The ERR LED location houses a red/green LED. The red LED indicates an error state in the EtherCAT protocol. The green LED indicates the operational state of the SMD17K itself. Both LEDs can be on simultaneously.

Red LED State	Description
Off	No errors in device operation
Single blink 200 ms ON / 1 s OFF	Problems with synchronization such as with the Distributed Clock (DC) PLL.
Double Blink 200 ms Pulses / 1 s between	SYNC manager watchdog timeout. The master has not updated the output data in the configured time interval. The SMD17K has switched to the Safe-Op state.
Slow Blink (1 Hz)	All other issues, such as cable disconnect. The SMD17K has switched to the Safe-Op state.

Table R1.4 Red ERR LED States

Green LED State	Description
Solid Green	Device in the EtherCAT Init state
Very Slow Blink Green [‡] (0.5 Hz)	Driver Fault (Overtemperature fault, etc.)
Slow Blink Green [‡] (1 Hz)	Unsuccessful write to flash memory
Fast Blink Green [‡] (4 Hz)	Successful write to flash memory

Table R1.5 Green ERR LED States

SMD17K Connectors

Input Connector

As shown in figure R1.4, the Input Connector is located on the back of the unit near its center. All digital input and power supply connections are made at this connector. The connector is a standard five pin A-coded M12 connector that is rated to IP67 when the mate is properly attached. Figure R1.5 shows the pinout of the connector when viewed from the back of the SMD17K.

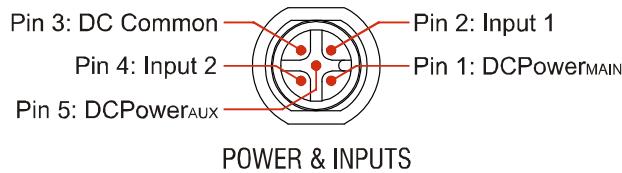


Figure R1.5 M12 Input Connector

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

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



If the unit was ordered with an encoder, the DCPower_{AUX} pin will also maintain power to the encoder. If the motor shaft is rotated while motor power is removed, the encoder position will update. (The motor position will not update.) Once power is restored to the motor, a Preset Position command can be issued to restore the correct motor position without having to go through a homing sequence. If Stall Detection is enabled on the SMD17K, it will also be able to tell the system if the motor shaft rotated more than forty-five degrees with power removed.

Ethernet Connectors

Figure R1.4 also show the placement of the sealed Ethernet Connector(s), while figure R1.6 shows the connector pinout when viewed from the back of the SMD17K. The Ethernet port on the SMD17K is an “auto-sense” port that will automatically switch between 10baseT and 100baseT depending on the network equipment it is attached to. The port also has “auto switch” capability. This means that a standard cable can be used when connecting the SMD17K to any device, including a personal computer.

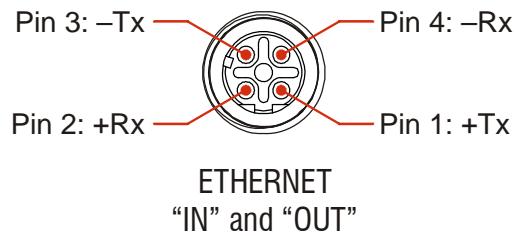


Figure R1.6 Ethernet Connector Pinout

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

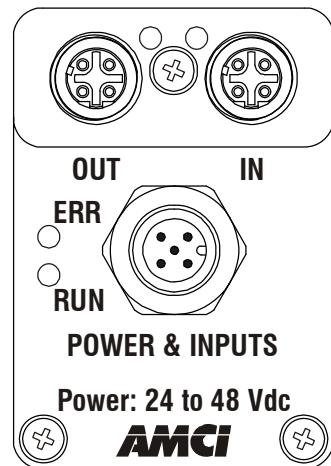


Figure R1.4 SMD17K Connector Locations

Torque and Power Curves

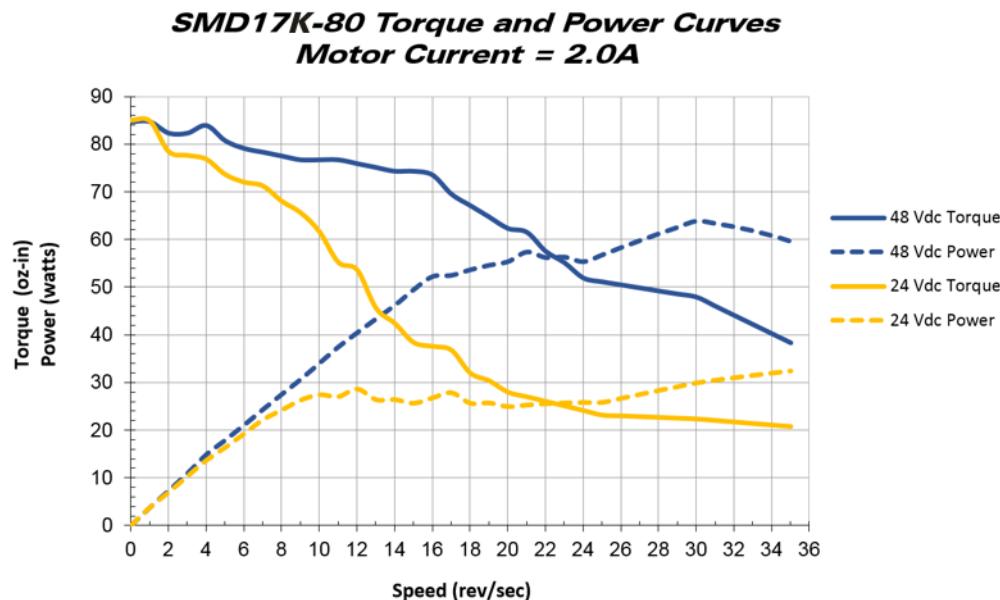


Figure R1.7 SMD17K-80 Torque and Power Curves

Power Supply Sizing

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

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

Example 1: An SMD17K-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 R1.7 on page 20, 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 SMD17K-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 R1.7 above, the power at this speed is 55 W, but the maximum power over the entire speed range is 58 W, which occurs at 21 RPS. Therefore, the 58 Watt value should be used, and the 48 Vdc supply should be able to generate 87 W to 102 W.

NOTE

- 1) SMD17K units have an additional pin that supplies power to the control electronics and encoder only. This allows you to remove power from the motor without losing the network connection.
- 2) Note that the unit is not powered from two isolated sources. There is only a single common in the system, and the electronics and motor sections are not electrically isolated.

Power Supply Sizing (continued)

Table R1.6 below shows the suggested power supply sizes based on the maximum power the motor can generate over its entire speed range.

		SMD17K-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 R1.6 Suggested Power Supply Ratings

Regeneration (Back EMF) Effects

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

The stepper motors used in the SMD17K units are all low inductance motors. Back EMF is typically not an issue unless there is a gearhead attached to the motor and it is driven by hand. In these instances, the motor acts as a generator. With the speed of the motor multiplied by the ratio of the gearhead, this can lead to large enough voltage spikes to damage the attached power supply.

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

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

Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17K provided that the manufacturer follows the connector and Ethernet standards. AMCI has reviewed the following connectors and ethernet cordsets for compatibility with the SMD17K.

Connectors

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

Table R1.7 Compatible Connectors

Ethernet Cordset

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

Table R1.8 Ethernet Cordset

Power Cordsets

AMCI Part #	Description
CNPL-2M	5-position, 22 AWG. Connector: Straight M12, A-coded, Female to 2 inch flying leads, 0.28" stripped. Cable length: 2 m
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 R1.9 Power Cordsets

REFERENCE 2

MOTION CONTROL

When a move command is sent to an SMD17K, 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/second². However, when programming an SMD17K, all acceleration and deceleration values must be programmed in the unit of measure of steps/second/millisecond.

- To convert from steps/second² 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/second², 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. The range is -2,147,483,648 to +2,147,483,647.

Home Position

The Home Position is any position on your machine that you can sense and stop at. There are two ways to defining the Home Position. The first is using the Preset Position command to set the Motor Position register to a known value. The second method is using one of the *Find Home* commands. If you use the unit's *Find Home* commands, the motor position and encoder position registers will automatically be set to zero once the home position is reached. Defining a Home Position is completely optional. Some applications, such as those that use the SMD17K 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 SMD17K moves use relative coordinates.

- The range of values for the Target Position when it is treated as an offset is $\pm 8,388,607$ 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 $\pm 8,388,607$ counts. If the starting position or the ending position is outside of the $\pm 8,388,607$ 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 $\pm 8,388,607$ 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 $\pm 8,388,607$ counts. However, you can not move beyond $\pm 8,388,607$ counts with an Absolute Move. The only way to move beyond $\pm 8,388,607$ counts is with relative moves or jog commands.

Definition of Acceleration Types

With the exception of Registration Moves, all move commands, including homing commands, allow you to define the acceleration type used during the move. The SMD17K 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 39.

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.

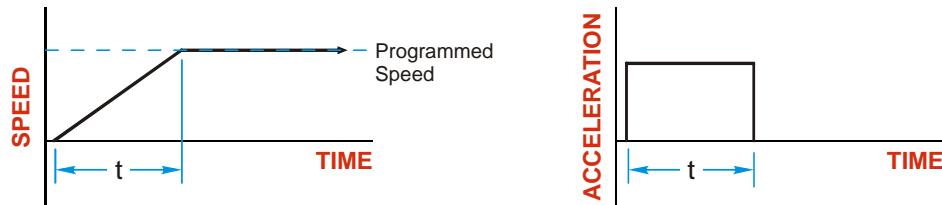


Figure R2.1 Linear Acceleration

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.

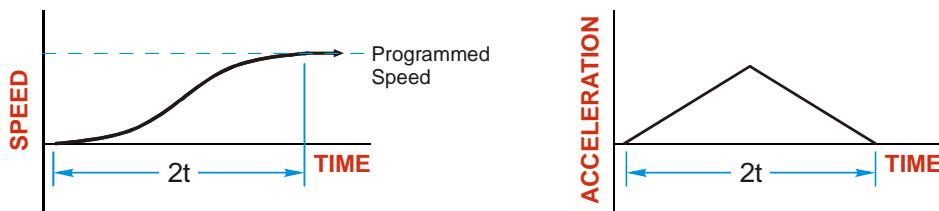


Figure R2.2 Triangular S-Curve Acceleration

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

A Simple Move

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

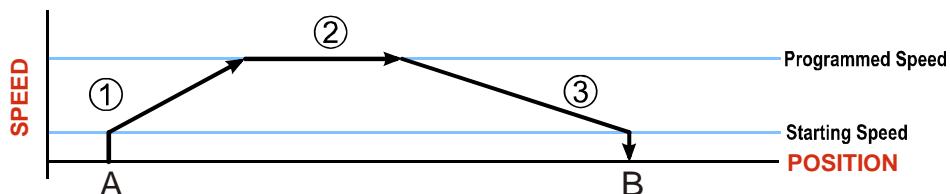


Figure R2.4 A Trapezoidal Profile

- 1)The move begins at point A, where the motor jumps from rest to the configured *Starting Speed*. The motor then accelerates at the programmed *Acceleration Value* until the speed of the motor reaches the *Programmed Speed*. Both the Acceleration Value and the Programmed Speed are programmed when the move command is sent to the SMD17K.
- 2)The motor continues to run at the Programmed Speed until it reaches the point where it must decelerate before reaching point B.
- 3)The motor decelerates at the *Deceleration Value*, which is also programmed by the move command, until the speed reaches the Starting Speed, which occurs at the Target Position (B). The motor stops at this point. Note that the acceleration and deceleration values can be different in the move.

Figure R2.4 above shows a Trapezoidal Profile. A Trapezoidal Profile occurs when the Programmed Speed is reached during the move. This occurs when the number of steps needed to accelerate and decelerate are less than the total number of steps in the move.

Figure R2.5 below shows a Triangular Profile. A Triangular Profile occurs when the number of steps needed to accelerate to the Programmed Speed and decelerate from the Programmed Speed are greater than the total number of steps in the move. In this case, the profile will accelerate as far as it can before it has to decelerate to reach the Starting Speed at the Target Position. The Programmed Speed is never reached.

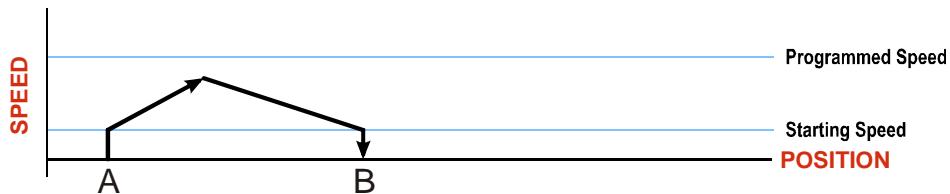


Figure R2.5 A Triangular Profile

Controlled and Immediate Stops

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

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

Host Control

Hold Move Command: This command can be used with some moves to bring the axis to a Controlled Stop. The move can be resumed and finished, or it can be aborted. Not all moves are affected by this command. The section *Basic Move Types*, starting on page 28, describes each move type in detail, including if the move is affected by this command.

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

Hardware Control

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

CW Limit and CCW Limit Inputs: In most cases, activating these inputs during a move will bring the axis to an Immediate Stop. The exceptions are the *CW/CCW Find Home* commands, the *CW/CCW Jog Move* commands, and the *CW/CCW Registration Move* commands. The *Find Home* commands are explained in the reference section, *Homing an SMD17K*, which starts on page 49. The *CW/CCW Jog Move* commands are fully explained on page 30, and the *CW/CCW Registration Move* commands are fully explained on page 31.

Emergency Stop Input: It is possible to configure an input as an Emergency Stop Input. When an Emergency Stop Input is activated, the axis will come to an Immediate Stop, regardless of the direction of travel. The move cannot be restarted and the machine must be homed again before Absolute Moves can be run. Note that power is not removed from the motor.

Basic Move Types

Relative Move

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

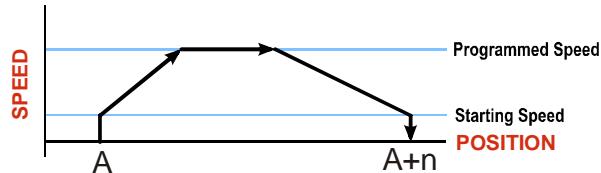


Figure R2.6 Relative Move

NOTE

- 1) You do not have to preset the position or home the machine before you can use a Relative Moves. That is, the *Position_Invalid* status bit can be set.
- 2) Relative Moves allow you to move your machine without having to calculate absolute positions. If you are indexing a rotary table, you can perform a relative move of 30° multiple times without recalculating new target positions in your controller. If you perform the same action with Absolute Moves, you would have to calculate your 30° position followed by your 60° position, followed by your 90° position, etc.

Relative Moves can be brought to a Controlled Stop by using the Hold Move Command from your host controller. When the command is accepted, the axis will immediately decelerate at the programmed rate and stop. When stopped successfully, the SMD17K will set an *In_Hold_State* bit in the input data table. The Relative Move can be restarted with the Resume Move command from the host controller or the move can be aborted by starting another move. The Resume Move command allows you to change the move's Programmed Speed, Acceleration Value and Type, and the Deceleration Value and Type. The Target Position cannot be changed with the Resume Move Command.

Controlled Stop Conditions

- The move completes without error.
- You issue a Hold Move command through the Network Output Data. Note that your holding position will most likely not be the final position you commanded. You can resume a held Relative Move by using the Resume Move command. The use of the Hold Move and Resume Move commands is further explained in the *Controlling Moves In Progress* section starting on page 38.

Immediate Stop Conditions

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

Basic Move Types (continued)

Absolute Move

Absolute Moves move from the Current Position (A) to a given position (B). (The SMD17K 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 $\pm 8,388,607$ counts. The move will be clockwise if the Target Position is greater than the Current Position and counter-clockwise if the Target Position is less than the Current Position.

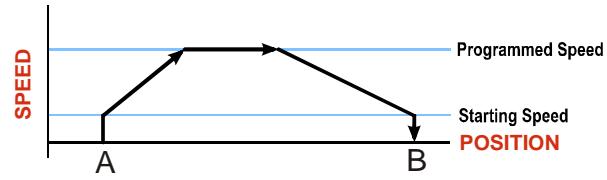


Figure R2.7 Absolute Move

NOTE

- 1) The *Home Position* of the machine must be set before running an Absolute Move. See the reference section, [Homing an SMD17K](#), which starts on page 49, for information on homing the machine.
- 2) The Motor Position must be valid before you can use an Absolute Move. The Motor Position becomes valid when you preset the position or home the machine.
- 3) Absolute Moves allow you to move your machine without having to calculate relative positions. If you are controlling a rotary table, you can drive the table to any angle without having to calculate the distance to travel. For example an Absolute Move to 180° will move the table to the correct position regardless of where the move starts from.

Controlled Stop Conditions

- The move completes without error.
- You issue a Hold Move command through the Network Output Data. Note that your holding position will most likely not be the final position you commanded. You can resume a held Absolute Move by using the *Resume_Move* bit or the move can be aborted by starting another move. The use of the Hold Move and Resume Move commands is explained in the [Controlling Moves In Progress](#) section starting on page 38.

Immediate Stop Conditions

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

Basic Move Types (continued)

CW/CCW Jog Move

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

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

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

As shown below, a Jog Move 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 SMD17K will decelerate the motor to the starting speed and stop without losing position. If it is an *Immediate Stop Condition*, the motion stops immediately and the position becomes invalid.

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

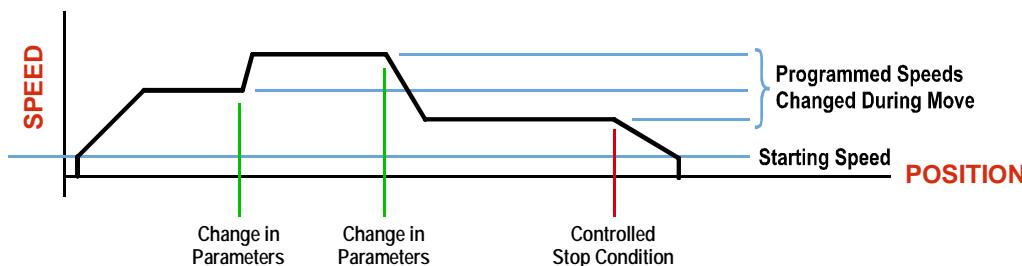


Figure R2.8 Jog Move

Controlled Stop Conditions

- The Jog Move Command bit is reset to “0”.
- An inactive-to-active transition on an input configured as a *Stop Jog or Registration Move Input*.
- You issue a Hold Move command in the Network Output Data. This brings the move to an end. The use of the Hold Move and Resume Move commands is explained in the *Controlling Moves In Progress* section starting on page 38.

Immediate Stop Conditions

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



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

Basic Move Types (continued)

CW/CCW Registration Move

Similar to a Jog Move, a Registration Move will travel in the programmed direction as long as the command is active. CW Registration Moves increase the motor position count while the CCW Registration Moves decrease the motor position count. When the command terminates under Controlled Stop conditions, the SMD17K 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.

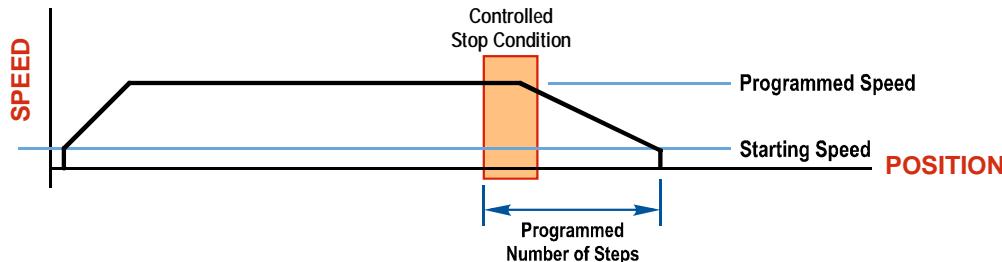


Figure R2.9 Registration Move



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 SMD17K will decelerate at the programmed Deceleration value until it has output the Programmed Number of Steps and then stop the move without further deceleration.

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

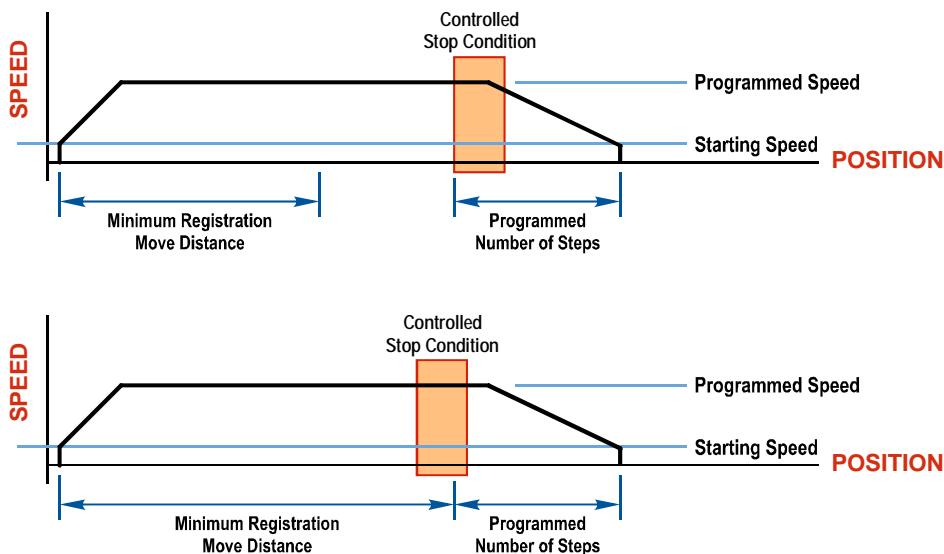


Figure R2.10 Min. Registration Move Distance

Basic Move Types (continued)**CW/CCW Registration Move (continued)****Controlled Stop Conditions**

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



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

- You issue a Hold Move command in the Network Output Data. The SMD17K responds by using the programmed Deceleration value to bring the move to a stop, without using the value of the Programmed Number of Steps parameter. A Registration Move does not go into the Hold State if the Hold Move command is used to stop the move and it cannot be restarted with the Resume Move command.

Immediate Stop Conditions

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



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

Assembled Moves

All of the moves explained so far must be run individually to their completion or must be stopped before another move can begin. The SMD17K also gives you the ability to pre-assemble more complex profiles from a series of relative moves that are then run with a single command. Each Assembled Move can consist of 2 to 16 segments. Assembled Moves are programmed through a hand shaking protocol that uses the input and output registers assigned to the unit. The protocol is fully described in the **Assembled Move Programming** section on page 36.

Two types of Assembled Moves exist in an SMD17K:

- **Blend Move** - A Blend Move gives you the ability to string multiple relative moves together and run all of them sequentially without stopping the shaft between moves. A Blend Move can be run in either direction, and the direction is set when the command is issued. The direction of motion cannot be reversed with a Blend Move.
- **Dwell Move** - A Dwell Move gives you the ability to string multiple relative moves together, and the SMD17K will stop between each move for a programmed *Dwell Time*. Because motion stops between each segment, a Dwell Move allows you to reverse direction during the move.

Assembled Moves (continued)

Blend Move

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

- 1) All segments of the Blend Move must be written to the unit before the move can be initiated.
 - The SMD17K supports Blend Moves with up to sixteen segments.
- 2) Each segment is programmed as a relative move. Blend Moves cannot be programmed with absolute coordinates.
- 3) All segments run in the same direction. The sign of the target position is ignored and only the magnitude of the target position is used. The move's direction is controlled by the bit pattern used to start the move. If you want to reverse direction during your move, consider using the **Dwell Move** which is explained starting on page 34.
- 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.

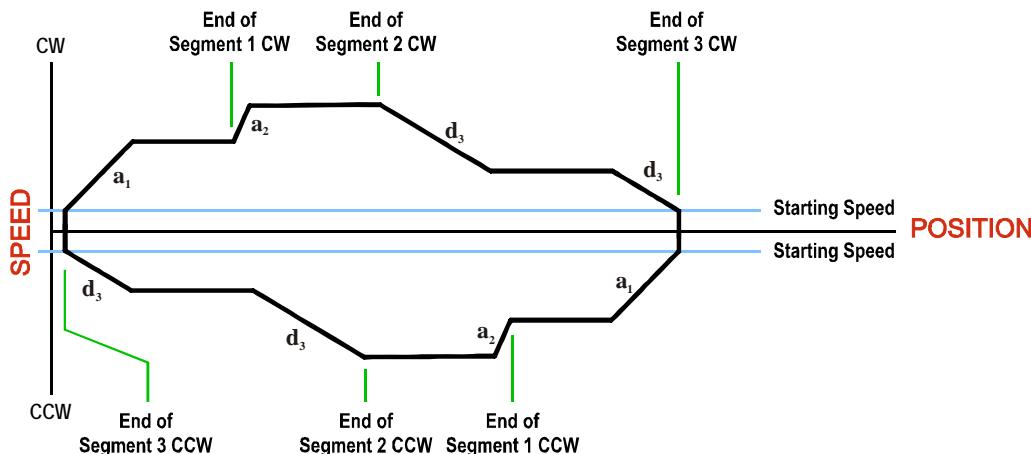


Figure R2.11 Blend Move

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 SMD17K and can be run multiple times once it is written to the unit. The Blend Move data stays in memory until power is removed, the unit is sent new Configuration Data, or a new Blend or Dwell Move is written to the unit. As described in *Saving an Assembled Move in Flash* on page 36, it is also possible to save a Blend Move to flash memory. This move is restored on power up and can be run as soon as you configure the SMD17K and enter Command Mode.
- 3) There are two control bits used to specify which direction the Blend Move is run in. This gives you the ability to run the Blend Move in either direction.

Controlled Stop Conditions

- The move completes without error.
- You issue a Hold Move command through the Network Output Data. When this occurs, the SMD17K decelerates the move at the deceleration rate of the present segment to the Starting Speed and ends the move. Note that your final position will most likely not be the one you commanded. A Blend Move that is brought to a controlled stop with the Hold Move command cannot be restarted. The use of the Hold Move command is explained in the *Controlling Moves In Progress* section starting on page 38.

Immediate Stop Conditions

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

Dwell Move

A Dwell Move gives you the ability to string multiple relative moves together and run all of them sequentially with a single start condition. Like a Blend Move, a Dwell Move is programmed into an SMD17K 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 programmed *Dwell Time*. The Dwell Time is programmed as part of the command that starts the move. The Dwell Time is the same for all segments. Because the motor is stopped between segments, the motor direction can be reversed during the move. The sign of the target position for the segment determines the direction of motion for that segment. Positive segments will result in clockwise shaft rotation while a negative segment will result in a counter-clockwise shaft rotation.

Assembled Moves (continued)

Dwell Move (continued)

The following figure shows a drilling profile that enters the part in stages and reverses direction during the drilling operation so chips can be relieved from the bit. You *could* accomplish this Dwell Move with a series of six relative moves that are sent down to the SMD17K sequentially. The two advantages of a Dwell Move in this case are that the unit will be more accurate with the Dwell Time then you can be in your control program, and Dwell Moves simplify your program's logic.

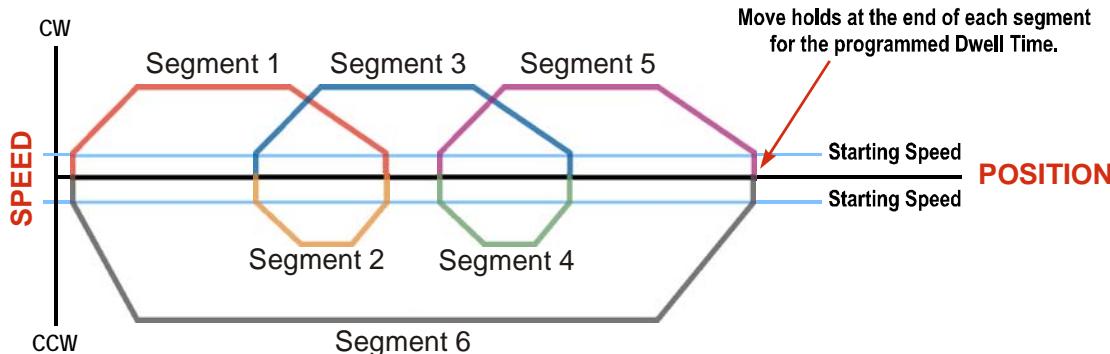


Figure R2.12 Dwell Move

NOTE

- 1) You do not have to preset the position or home the machine before you using a Dwell Move. Because the Dwell Move is based on Relative Moves, it can be run from any location.
- 2) The Dwell Move is stored in the internal memory of the SMD17K 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 36, it is also possible to save a Dwell Move to flash memory. This move is restored on power up and can be run as soon as you configure your SMD17K and enter Command Mode.

Controlled Stop Conditions

- The move completes without error.
- You issue a Hold Move command through the Network Output Data. When this occurs, the SMD17K decelerates the move at the deceleration rate of the present segment to the Starting Speed and ends the move. Note that your final position will most likely not be the one you commanded. A Dwell Move that is brought to a controlled stop with the Hold Move command cannot be restarted.

Immediate Stop Conditions

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

Assembled Move Programming

All of the segments in a Blend or Dwell Move must be written to the SMD17K before the move can be run. Segment programming is controlled with two bits in the Network Output Data and two bits in the Network Input Data. Blend and Dwell Moves are programmed in exactly the same way. When you start the move, a bit in the command data determines which type of Assembled Move is run. In the case of a Blend Move, the sign of each segment's Target Position is ignored and all segments are run in the same direction. In the case of a Dwell Move, the sign of each segment's Target Position determines the direction of the segment. For Dwell Moves, the Dwell Time is sent to the unit as part of the command.

Control Bits – Output Data

- **Program_Assembled bit** – A 0→1 transition on this bit tells the SMD17K that you want to program a Blend or Dwell Move Profile. The unit will respond by setting the *In_Assembled_Mode* bit in the Network Input Data. At the beginning of the programming cycle, the unit will also set the *Waiting_For_Assembled_Segment* bit to signify that it is ready for the first segment.
- **Read_Assembled_Data bit** – A 0→1 transition on this bit tells the SMD17K that the data for the next segment is available in the remaining data words.

Control Bits – Input Data

- **In_Assembled_Mode bit** – The SMD17K sets this bit to tell you that it is ready to accept segment programming data in the remaining output data words. The actual transfer of segment data is controlled by the *Waiting_For_Assembled_Segment* and *Read_Assembled_Data* bits.
- **Waiting_For_Assembled_Segment bit** – A 0→1 transition on this bit from the SMD17K is the signal to the host that the unit is ready to accept the data for the next segment.

Programming Routine

- 1)The host sets the *Program_Assembled* bit in the Network Output Data.
- 2)The SMD17K responds by setting both the *In_Assembled_Mode* and *Waiting_For_Assembled_Segment* bits in the Network Input Data.
- 3)When the host detects that the *Waiting_For_Assembled_Segment* bit is set, it writes the data for the first segment in the Network Output Data and sets the *Read_Assembled_Data* bit.
- 4)The SMD17K checks the data, and when finished, resets the *Waiting_For_Assembled_Segment* bit. If an error is detected, it also sets the *Command_Error* bit.
- 5)When the host detects that the *Waiting_For_Assembled_Segment* bit is reset, it resets the *Read_Assembled_Data* bit.
- 6)The SMD17K detects that the *Read_Assembled_Data* bit is reset, and sets the *Waiting_For_Assembled_Segment* bit to signal that it is ready to accept data for the next segment.
- 7)Steps 3 to 6 are repeated for the remaining segments until the entire move profile has been entered. The maximum number of segments per profile is sixteen.
- 8)After the last segment has been transferred, the host exits Assembled Move Programming Mode by resetting the *Program_Assembled* bit.
- 9)The unit resets the *In_Assembled_Mode* and the *Waiting_For_Assembled_Segment* bits.

Saving an Assembled Move in Flash

The SMD17K 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 SMD17K responds by resetting the *In_Assembled_Mode* and *Transmit Blend Move Segments* bits as usual and then it will flash the Status LED. If the LED is flashing green, the write to flash memory was successful. If it flashes red, then there was an error in writing the data. In either case, power must be cycled to the SMD17K before you can continue. With a limit of 10,000 write cycles, the design decision that requires you to cycle power to the unit was made to prevent an application from damaging the module by continuously writing to it.

Indexed Moves

All of the moves that have been explained in the chapter up to this point can be started by a transition on one of the inputs instead of a command from the network. If the *Indexed Move* bit is set when the command is issued, the SMD17K 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 SMD17K until the triggered move is finished.
- As stated above, a move can be run multiple times as long at the move command data remains unchanged. If you wish to program a second move and run it as an Indexed Move type, then you must have a 0→1 transition on the move command bit before the new parameters are accepted. The easiest way to accomplish this is by writing a value of 16#0000 to the command word between issuing move commands.
- A Jog Move that is started as an Indexed Move will come to a controlled stop when the command bit in the Network Output Data is reset to zero.
- It is possible to perform an indexed Registration Move by configuring two inputs for their respective functions. The first input, configured as a *Start Indexed Move Input*, starts the move and the second, configured as a *Stop Jog or Registration Move Input* causes the registration function to occur.
- You cannot issue a Hold Command with the Indexed Bit set and have the Hold Command trigger on the inactive-to-active transition of a physical input. Hold Commands are always acted upon as soon as they are accepted from the Network Output Data.
- You cannot issue an Immediate Stop Command with the Indexed Bit set and have the Immediate Stop Command trigger on the inactive-to-active transition of a physical input. Immediate Stop Commands are always acted upon as soon as they are accepted from the Network Output Data. If you need this functionality, consider programming the physical input as an E-Stop Input.
- You cannot issue a Clear Error Command with the Indexed Bit set and have the Clear Error Command trigger on the inactive-to-active transition of a physical input. Clear Error Commands are always acted upon as soon as they are accepted from the Network Output Data.

Controlling Moves In Progress

Each SMD17K has the ability to place a running move on hold and later resume the move if an error did not occur while the move was in its Hold state. One potential application for this feature is bringing a move to a controlled stop when your controller senses an end-of-stock condition. The move can be put in its Hold state until the stock is replenished and then the move can be resumed.

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

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

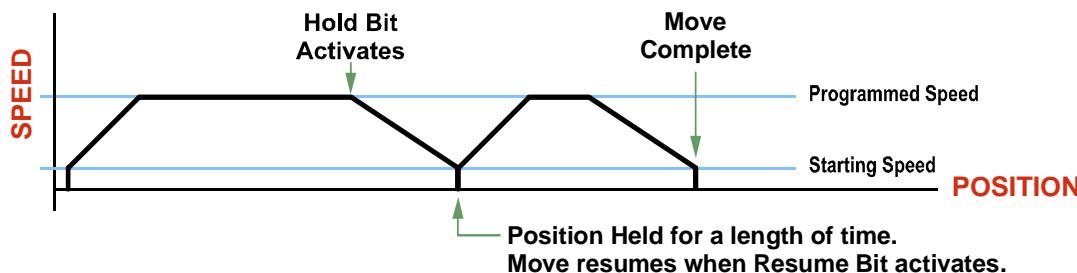


Figure R2.13 Hold/Resume a Move Profile

Jog Moves

Jog Moves can be placed in a Hold state and resumed if error conditions, such as programming errors, have not occurred. New Acceleration, Deceleration, and Programmed Speed parameters can be written to the SMD17K while a Jog Move is in its hold state. If these parameters are accepted without error, the move can be resumed and it will use the new parameter values.

Registration Moves

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

Absolute and Relative Moves

Absolute and Relative Moves can be placed in a Hold state and resumed if error conditions, such as programming errors, have not occurred. New Acceleration, Deceleration, and Programmed Speed parameters can be written to the SMD17K while these moves are in their hold states. If the parameters are accepted without error, the move can be resumed and it will use the new parameter values. Note that a change to the Target Position is ignored.

Assembled Moves

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

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 SMD17K 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/second²) for acceleration and deceleration. However, when programming the SMD17K, all acceleration and deceleration values must be programmed in the unit of measure of steps/second/millisecond.

- To convert from steps/second² 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 SMD17K.
- To convert from steps/second/millisecond to steps/second², multiply the value by 1000. This must be done when converting from the value programmed into the SMD17K 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/sec². For example, if the choose acceleration is 20,000 steps/sec², the smoothest transition occurs when the starting speed is 141. ($141^2 \approx 20,000$)

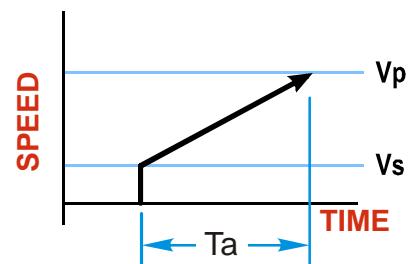


Figure R3.1 Constant Acceleration Curves

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/second²
- d = Deceleration value. Must be in the units of steps/second²
- 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	SMD17K 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:

$$\text{Time to accelerate: } T_A = (V_P - V_S)/a = (100,000 - 141)/20,000 = 4.993 \text{ seconds}$$

$$\text{Time to decelerate: } T_D = (V_P - V_S)/d = (100,000 - 141)/25,000 = 3.994 \text{ seconds}$$

$$\text{Distance to Accelerate: } D_A = T_A * (V_P + V_S)/2 = 4.993 * (100,000 + 141)/2 = 250,002 \text{ steps}$$

$$\text{Distance to Decelerate: } D_D = T_D * (V_P + V_S)/2 = 3.994 * (100,000 + 141)/2 = 199,982 \text{ steps}$$

$$\text{Total Distance needed to accelerate and decelerate: } 250,002 + 199,982 = 449,984 \text{ steps}$$

If a move with the above acceleration, deceleration, starting speed, and programmed speed has a length greater than 449,984 steps, the SMD17K will generate a Trapezoidal profile. If the move is equal to 449,984 steps, the unit will generate a Triangular profile and 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 .

$$D_A = T_A * (V_M + V_S)/2 \text{ and } T_A = (V_M - V_S)/a. \text{ By substitution:}$$

$$D_A = (V_M - V_S)/a * (V_M + V_S)/2 = (V_M^2 - V_S^2)/2a. \text{ By the same method,}$$

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

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.

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

Once you have calculated the maximum speed, you can substitute this value into the time and distance formulas in table 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.

$$D_P = (\text{Total Number of Steps}) - (D_A + D_D)$$

$$T_P = T_A + T_D + D_P/V_P$$

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 SMD17K 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 SMD17K below sixteen bits, the Acceleration Jerk parameter programmed into the driver does not have units of steps/sec³. The Acceleration Jerk parameter equals ($\{100 * \text{jerk in steps/sec}^3\} / \text{acceleration in steps/sec}^2$). This translates to the jerk property in steps/sec³ equalling ($\{\text{Acceleration Jerk parameter}/100\} * \text{acceleration in steps/sec}^2$).

With the range of values for the Acceleration Jerk parameter being 1 to 5,000, the jerk value ranges from $0.01a$ to $50a$ where “ a ” is the acceleration value in steps/sec². For example, if the acceleration is programmed to 20,000 steps/sec², then the value of the jerk property used by the unit can be programmed to be between 200 steps/sec³ ($0.01*20,000$) and 1,000,000 steps/sec³ ($50*20,000$). 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.

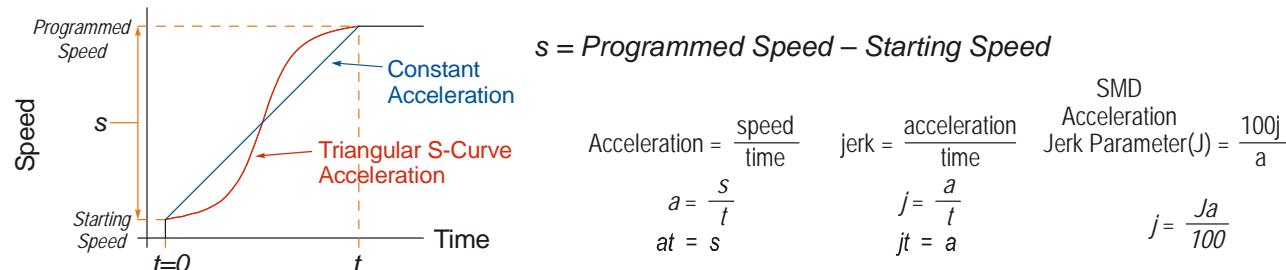


Figure R3.2 Move Profile Example

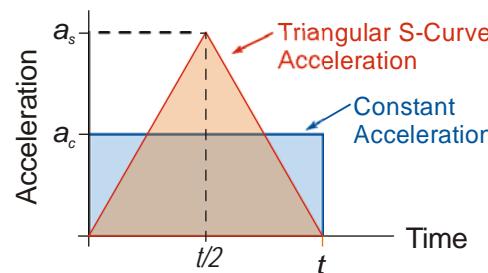


Figure R3.3 Triangular Acceleration

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.

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

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

$$\frac{Ja_s}{100} = \frac{2a_s}{t} \quad (j = \frac{Ja}{100})$$

$$Ja_s t = 200a_s$$

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

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/sec² that is applied for 2.0 seconds. If the acceleration value must remain at 20,000 steps/sec², 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.

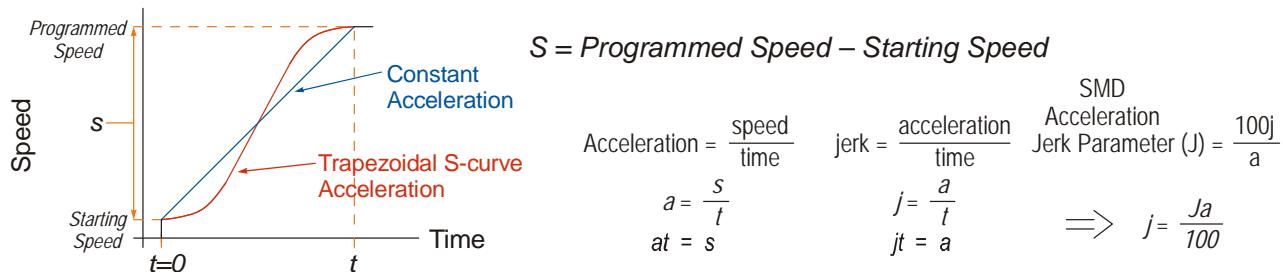


Figure R3.4 Move Profile Example

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

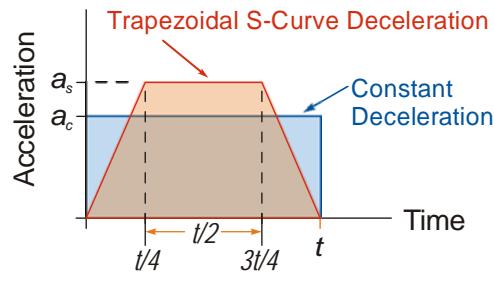


Figure R3.5 Trapezoidal Acceleration

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.

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

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

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

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

This means that a trapezoidal S-curve acceleration profile that 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.

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

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

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

$$Ja_s t = 400a_s$$

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

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.

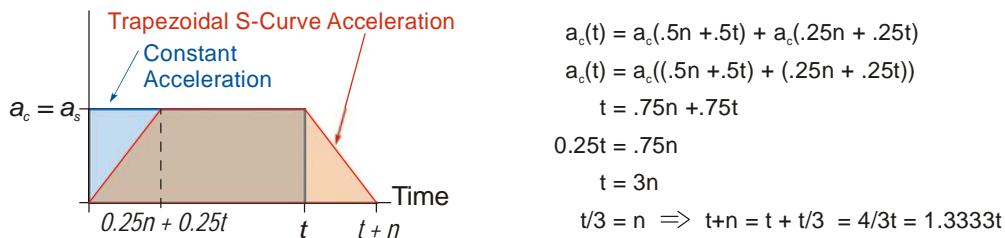


Figure R3.6 Trapezoidal S-curve Time Increase Example

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/sec² that is applied for 2.0 seconds. If the acceleration value must remain at 15,000 steps/sec², 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/sec². 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

$$S_m = \frac{30,000 \text{ steps/sec}}{2} = 15,000 \text{ steps/sec} \quad S_m = \text{midpoint of change in speed}$$

$$J = \frac{100j}{a} \Rightarrow j = \frac{Ja}{100} \quad J = \text{Acceleration Jerk parameter}$$

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

$$j = 11,600 \text{ steps/sec}^3 \quad a_f = \text{calculated final acceleration}$$

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

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

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

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

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/sec², the resulting acceleration is a Triangular S-curve. Total time to accelerate is twice the value calculated above, or 3.216 seconds.

S-Curve Acceleration Equations (continued)**Determining Waveforms by Values (continued)***Example 2, Jerk = 400*

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

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

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

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

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/sec², 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).

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

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

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

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

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

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

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

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

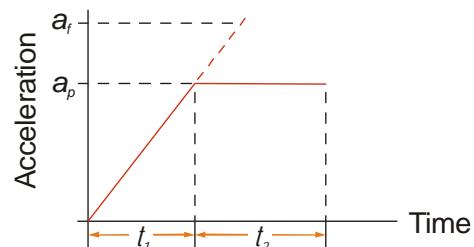


Figure R3.7 Calculating Trapezoidal S-Curve

The time for this acceleration phase is $2(t_1 + t_2)$, which equals $2(0.2500 \text{ sec} + 0.1336 \text{ sec})$ 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 SMD17K

This chapter explains the various ways of homing an SMD17K 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 SMD17K 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 SMD17K for speed control, don't require position data at all.

With the exception of Absolute Moves, the SMD17K can still perform all of its move commands if the Home Position is not defined.

Position Preset

One of the ways to define the Home Position is to issue the Preset Position command over the network. On units with integral encoders, both the motor position and the encoder position can be preset separately, and the motor position can also be preset to the encoder position. The motor and encoder position values can be preset anywhere in the range of -8,388,607 to +8,388,607.

NOTE When presetting the motor position to the encoder position, the programmed Steps per Turn and Counts per Turn parameter values are used to scale the encoder position before the motor position is set to it. For example, assume that the Encoder Counts per Turn is programmed to 2,048, and the Motor Steps per Turn is programmed to 2,000. If the encoder position is 2,048 when the preset command is issued, the motor position will be set to 2,000.

CW/CCW Find Home Commands

The other choice is to use the driver's Find Home commands to order the SMD17K to find the Home Position based on sensors brought into the unit. The CW Find Home command begins searching by rotating the motor shaft in the clockwise direction and ends when the home sensor triggers while the SMD17K is rotating in the clockwise direction *at the starting speed*. The CCW Find Home command operates in the same way but starts and ends with motion in the counter-clockwise direction.

Homing Inputs

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

Physical Inputs

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

Network Data Input

- **Backplane_Proximity_Bit:** An SMD17K 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.



This bit is disabled by default, and must be activated when you configure the SMD17K before it can be used. If you decide to activate this bit when you configure the unit and then never set it to a “1”, the SMD17K will never act on the physical Home Input.

Homing Configurations

A SMD17K must have one of its DC inputs configured as the home input before one of the *CW/CCW Find Home* commands can be issued.



- 1) You do not have to configure and use CW or CCW Limits. If you choose to configure the unit this way, then the SMD17K has no way to automatically prevent over travel during a homing operation. In linear applications, you must prevent over travel by some external means, or ensure that the homing command is issued in the direction that will result in reaching the homing input directly.
- 2) You can use a bit in the Network Output Data (the Backplane_Proximity_Bit) as a home proximity input. Using this bit is completely optional and prevents the Home Input from being acted upon until the Backplane_Proximity_Bit makes a 0→1 transition.

Homing Profiles



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.

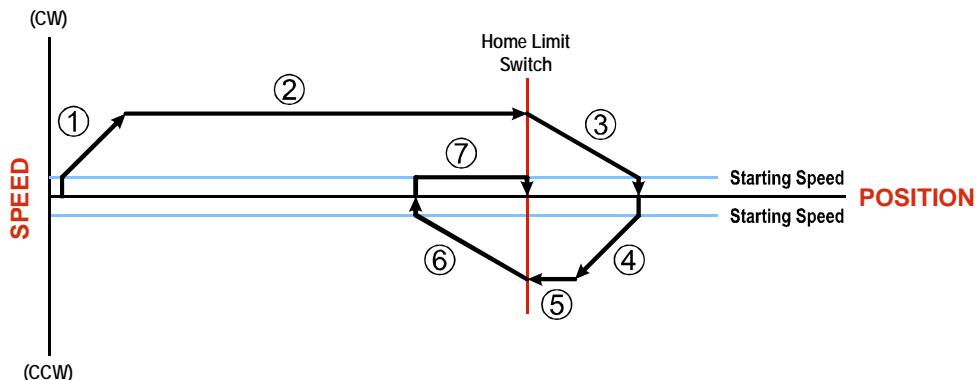


Figure R4.1 Home Input Profile

- 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.



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.

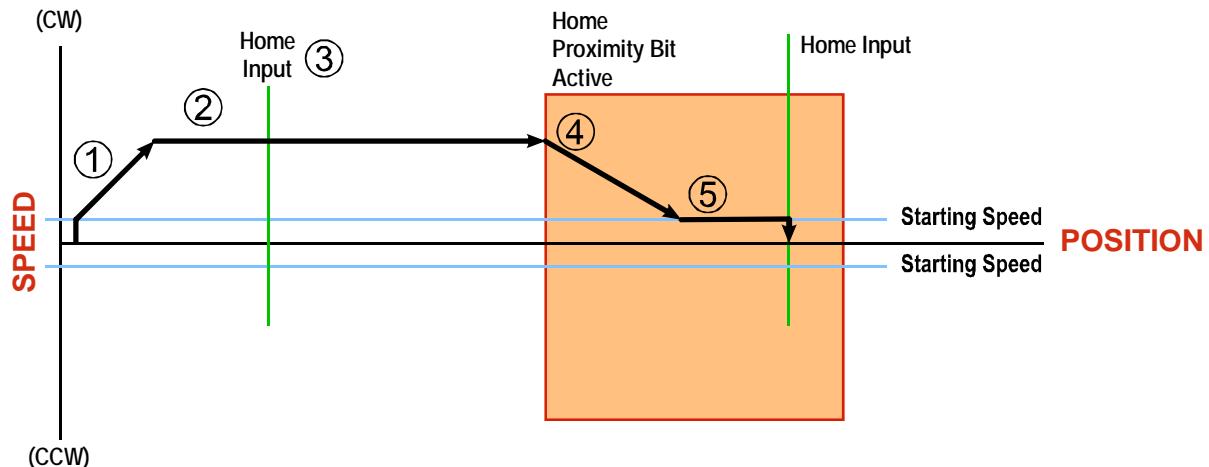


Figure R4.2 Homing with Proximity

- 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.



Figure R4.2 shows the Backplane_Proximity_Bit staying active until the SMD17K 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 SMD17K will stop and issue a *Home Invalid* error to your host if you activate the overtravel limit associated with travel in the opposite direction. i.e. Activating the CCW limit during a CW Find Home command. This can occur if the overtravel limits are not wired to the unit correctly, or not configured correctly when the unit was configured.

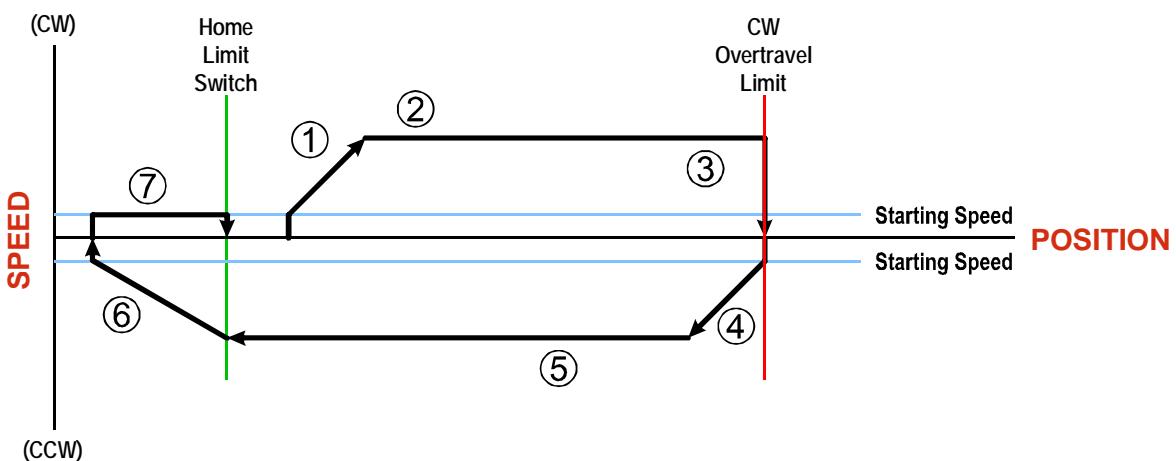


Figure R4.3 Profile with Overtravel Limit

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

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

Controlling Find Home Commands In Progress**Controlled Stop Conditions**

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

Immediate Stop Conditions

- You issue an Immediate Stop command through the Network Output Data.
- An inactive-to-active transition on an input configured as an E-Stop Input.
- The overtravel limit associated with travel in the opposite direction is activated. i.e. Activating the CCW limit during a CW Find Home command. This can occur if the overtravel limits are not wired to the SMD17K correctly, or not configured correctly when the unit was configured.

REFERENCE 5

CONFIGURATION DATA FORMAT

This chapter covers the format of the configuration data for an SMD17K. Configuration data is stored in registers available through the CANopen over Ethernet (CoE) interface.

CoE Registers

The SMD17K units use Service Data Objects (SDO) of the CANopen protocol to configure the unit. Typically, this configuration is specified in the system software and is written down to the SMD17K when communications is established. The TwinCAT system also allows you to programmatically read or change these values using the FB_EcCoeSdoRead and FB_EcCoeSdoWrite instructions.

Data Format

The correct format of the configuration data is shown below.

CANopen Index:SubIndex	Data Size	Configuration Data	Range
8010:01	UINT	CFG_Word_0	See below
8010:02	UINT	CFG_Word_1	See below
8010:03	UDINT	Starting_Speed	1 to 1,999,999 steps/sec.
8010:04	UINT	Motor_Resolution	200 to 32,767
8010:05	UINT	Encoder_Resolution	Set to 1,024, 2,048, or 4,096 for incremental encoder. Set to 2,048 for absolute encoder.
8010:06	UINT	Idle_current_reduction	0 to 100%
8010:07	UINT	Motor_Current (X10)	1 to 20, Represents 0.1 to 2.0 Arms

Table R5.1 CoE Configuration Data Layout

CFG_Word_0 Format

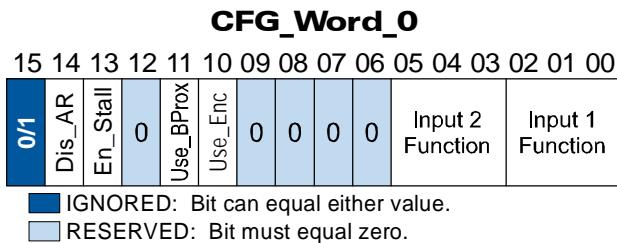


Figure R5.1 Configuration Word 0 Format

Bit 15: Reserved – State ignored.

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

Bit 13: Enable_Stall_Detection – “0” disables motor stall detection. “1” enables motor stall detection. Only valid on SMD17K units with built in encoders. The *Use_Encoder* bit, which is bit 10 of this word, must also be set to “1”. You must also program the *Encoder_Resolution* parameter in CoE index 8010:05.

Data Format (continued)**CFG_Word_0 Format (continued)**

Bit 12: Reserved – Must equal zero.

Bit 11: Use_Backplane_Proximity – “0” when the Backplane_Proximity_Bit is not used when homing the SMD17K. “1” when the Backplane_Proximity_Bit is used when homing the unit. Note that this bit is not the Backplane_Proximity_Bit, but enables or disables its operation. Do not use the Backplane_Proximity_Bit if you only want to home to a Home Limit Switch. (Leave this bit equal to “0.”) If you enable this bit and then never turn on the Backplane_Proximity_Bit, the SMD17K unit will ignore all transitions of the home limit switch and you will not be able to home the device.

Bit 10: Use_Encoder – Reset to “0” when the built-in encoder is not used or not available. Set to “1” to enable the built-in absolute or quadrature encoder. When enabled, you must also program the Encoder_Resolution parameter in CoE index 8010:05.

Bits 9-6: Reserved – Must equal “0”.

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

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

Bits			Function	Available On
5	4	3		
2	1	0		
0	0	0	General Purpose Input	The input is not used in any of the functions of the SMD17K, but its status is reported in the Network Data. This allows the input to be used as a discrete DC input to the host controller.
0	0	1	CW Limit	Input defines the mechanical end point for CW motion.
0	1	0	CCW Limit	Input defines the mechanical end point for CCW motion.
0	1	1	Start Indexed Move	Starts the move that is currently located in the output registers.
0	1	1	Start Indexed Move / Capture Encoder Value	When the encoder is enabled on an SMD17K, the encoder position value is captured whenever this input transitions. An inactive-to-active state transition will also trigger an Indexed Move if one is pending in the SMD17K.
1	0	0	Stop Jog or Registration Move	Brings a Jog or Registration Move to a controlled stop.
1	0	0	Stop Jog or Registration Move & Capture Encoder Value	When the encoder is enabled on an SMD17K, the encoder position value is captured when the input triggers a controlled stop to a Jog or Registration move.
1	0	1	Emergency Stop	All motion is immediately stopped when this input makes an inactive-to-active transition.
1	1	0	Home	Used to define the home position of the machine.
1	1	1	Invalid Combination	This bit combination is reserved.

Table R5.2 Configuration Data: Input Function Selections

Data Format (continued)**CFG_Word_0 Format (continued)**

To Make These Settings...		...Set These Bits To																
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Input 1*	General Purpose															0	0	0
	CW Limit															0	0	1
	CCW Limit															0	1	0
	Start Indexed Move [†]															0	1	1
	Stop Jog/Reg. Move [†]															1	0	0
	E-Stop															1	0	1
Input 2*	Home															1	1	0
	General Purpose															0	0	0
	CW Limit															0	0	1
	CCW Limit															0	1	0
	Start Indexed Move [†]															0	1	1
	Stop Jog/Reg. Move [†]															1	0	0
Use_Encoder	Disabled											0						
	Enabled											1						
Use_Backplane_Proximity	Disabled									0								
	Enabled								1									
Stall_Detection [‡]	Disabled				0													
	Enabled				1													
Antiresonance	To Enable		0															
	To Disable		1															
Resulting Bit Pattern:		0		0				0	0	0	0							
Hexadecimal Digits for CoE Configuration Setting:																		

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

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

‡ Enabling Stall_Detection will cause an Configuration Error if the encoder is not enabled. (Bit 10 must equal "1".)

Table R5.3 Configuration Word 0 Bits

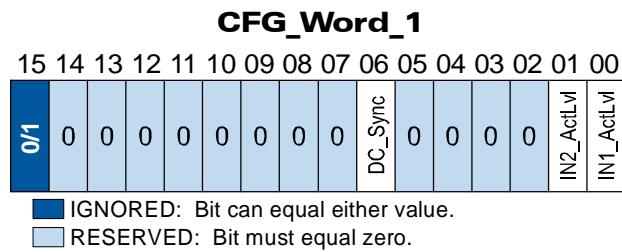
Data Format (continued)**CFG_Word_1 Format**

Figure R5.2 Configuration Word 1 Format

Bit 15: Reserved – State ignored.

Bits 14 - 7: Reserved – Must equal zero.

Bit 6: DC_Sync – When set to “0”, the SMD17K begins a move as soon as data is read from the EtherCAT Slave Controller (ESC). When set to “1”, the SMD17K will synchronize the start of a move on the SYNC0 output of the system’s Distributed Clock. This allows you to synchronize the start of moves over multiple devices.

Bits 5 - 2: Reserved – Must equal zero.

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

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

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

Range of Values for Configuration Word 1

Valid values for Configuration Word 1 are: 0, 1, 2, 3, 64, 65, 66, and 67. (0x0000, 0x0001, 0x0002, 0x0003, 0x0040, 0x0041, 0x0042, 0x0043)

Notes on Other Configuration Words

- Changes to the Idle Current only take effect at the *end of the first move after re-configuration*.

Invalid Configurations

The following configurations are invalid:

- 1) Setting any of the reserved bits in the configuration words.
- 2) Setting any parameter to a value outside of its valid range.
- 3) Configuring the two inputs to have the same function, such as two CW Limit Switches. (An error does not occur if both are configured as General Purpose Inputs.)
- 4) Setting the *Stall Detection Enable Bit* without configuring the SMD17K to use its built in encoder.
- 5) Setting the Input Configuration bits for any input to “111”. See table R5.3 on page 57 for more information.

REFERENCE 6

COMMAND DATA FORMAT

This chapter covers the format of the Network Output Data used to command the SMD17K and the format of the Network Input Data that contains the responses from the device. The parameter names of the input and output data are defined by the device ESI file. The size of each data element is also defined by the ESI file.

Command Bits Must Transition

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

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

Output Data Format

The following table shows the format of the output network data words when writing command data to the SMD17K.

ESI File Name	Data Size	Function
CMD_word0	UINT	Command Word 0
CMD_word1	UINT	Command Word 1
Position	DINT	Command Parameters
Velocity	UDINT	
Acceleration	UINT	
Deceleration	UINT	
Motor_Current	UINT	
Jerk	UINT	

Figure R6.1 Command Data Format

CMD_word0

CMD_word0															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	Prst_Enc	Run_AMov	Rd_AData	Prg_Asm	RSet_Err	Prst_Pos	Jog_CCW	Jog_CW	Home_CCW	Home_CW	I-Stop	Resm_Mv	Hold_Mv	Rel_Mv	Abs_Mv

Figure R6.2 Command Word 0 Format

Bit 15: **Mode_Select** – “0” for Command Mode programming.



It is possible to write configuration data to the unit using the output words, but the use of this method is discouraged. Configuration data should be written to the SMD17K using the CoE interface as described in the previous chapter.

Bit 14: **Preset_Encoder** – When this bit makes a 0→1 transition, the unit will preset the Encoder Position to the value stored in the *Position* double integer register defined by the ESI file.

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

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

Bits 12 & 11: Program_Assembled & Read_Assembled_Data – These bits are used to program the segments of an Assembled Move before the move can be run. Their use is explained in the [Assembled Move Programming](#) section of this manual starting on page 36.

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

Bit 9: **Preset_Position** – When this bit makes a 0→1 transition, the unit will preset the Motor Position. The value depends on the state of the *Preset_To_Encoder* bit (Command Word 1, bit 13). If the *Preset_To_Encoder* bit equals “0”, the Motor Position is preset to the value stored in the *Position* double integer register defined by the ESI file. If the *Preset_To_Encoder* bit equals “1”, the Motor Position is set to:

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

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

Bit 8: **Jog_CCW** – When this bit makes a 0→1 transition, the unit will run a Jog Move in the counter-clockwise direction. The full explanation of a [CW/CCW Jog Move](#) can be found starting on page 30.

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

CMD_word0 (continued)

- Bit 7:** **Jog_CW** – When this bit makes a 0→1 transition, the unit will run a Jog Move in the clockwise direction. The full explanation of a **CW/CCW Jog Move** can be found starting on page 30.
- **Registration_Move – Command Word 1, Bit 7:** When this bit equals “0”, and a Jog Move command is issued, it will run as a standard Jog Move. When this bit equals “1” and a Jog Move command is issued, the move will run as a Registration Move.
- Bit 6:** **Find_Home_CCW** – When this bit makes a 0→1 transition, the unit will attempt to find the Home Limit Switch by beginning the move in the counter-clockwise direction. A full explanation of homing can be found in the **Homing an SMD17K** reference chapter starting on page 49.
- Bit 5:** **Find_Home_CW** – When this bit makes a 0→1 transition, the unit will attempt to find the Home Limit Switch by beginning the move in the clockwise direction. A full explanation of homing can be found in the **Homing an SMD17K** reference chapter starting on page 49.
- Bit 4:** **Immediate_Stop** – When this bit makes a 0→1 transition, the unit will stop all motion without deceleration. The Motor Position value will become invalid if this bit is set during a move. Setting this bit when a move is not in progress will not cause the Motor Position to become invalid.
- Bit 3:** **Resume_Move** – When this bit makes a 0→1 transition, the unit will resume a move that you previously placed in a hold state. Use of the *Resume_Move* and *Hold_Move* bits can be found in the **Controlling Moves In Progress** section of this manual starting on page 38. Note that a move in its hold state need not be resumed. The move is automatically cancelled if another move is started in its place.
- Bit 2:** **Hold_Move** – When this bit makes a 0→1 transition, the unit will place a move in its hold state. The move will decelerate to its programmed Starting Speed and stop. The move can be completed by using the *Resume_Move* bit. Use of the *Hold_Move* and *Resume_Move* bits can be found in the **Controlling Moves In Progress** section of this manual starting on page 38.
- Bit 1:** **Relative_Move** – When this bit makes a 0→1 transition, the unit will perform a Relative Move using the data in the Command Parameters. The full explanation of a **Relative Move** can be found starting on page 28.
- Bit 0:** **Absolute_Move** – When this bit makes a 0→1 transition, the unit will perform an Absolute Move using the data in the Command Parameters. The full explanation of an **Absolute Move** can be found starting on page 29.

CMD_word1

CMD_word1															
En_Driver	0	Preset_to_Enc	0	BP_Prox	0	ASW_Type	Index_Cmd	Reg_Move	0	Save_to_Flash	Rev_BlendDir	0	0	Motor_Current	0

Figure R6.3 Command Word 1 Format

Bit 15: **Enable_Driver** – “0” to disable the motor current, “1” to enable motor current. The motor cannot be enabled if the SMD17K has a configuration error.

Bit 14: **Reserved** – Must equal zero.

Bit 13: **Preset_to_Encoder** – Only used when the Preset Motor Position bit (Command Word 0, Bit 9) is set to “1”. If this bit equals “0” when the Preset Motor Position bit equals “1”, the Motor Position is set to the value stored in the *Position* double integer register defined by the ESI file. If this bit equals “1” when the Preset Motor Position bit equals “1”, the Motor Position is set to:

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

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

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

Bit 11: **Backplane_Proximity_Bit** – When the SMD17K is configured to use the *Backplace_Proximity_Bit*, the unit will ignore the state of the Home Input as long as this bit equals “0”. This bit must equal “1” before a transition on the Home Input can be used to home the machine. Further information on using the *Backplace_Proximity_Bit* can be found in the [Profile with Backplane_Proximity_Bit](#) section found on page 52.

Bit 10: **Reserved** – Must equal “0”.

Bit 9: **Assembled_Move_Type** – When this bit equals “0”, a Blend Move is started when the Run Assembled Move bit, (Command Word 1, Bit 13) makes a 0→1 transition. When this bit equals “1”, a Dwell Move is started on the transition. The direction of a Blend Move is controlled by the *Reverse_Blend_Direction* bit, (Command Word 1, Bit 4). In a Dwell Move, the Dwell Time between segments is programmed in Word 9 of the command data.

Bit 8: **Indexed_Command** – If this bit is set when a move command is issued, the SMD17K will not run the move immediately, but will instead wait for an inactive-to-active transition on an input configured as a *Start Indexer Move* input. The move command data, including this bit, must remain in the Network Output Registers while performing an Indexed Move.

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

Bit 6: **Reserved** – Must equal “0”.

CMD_word1 (continued)

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

- When using this bit to save the programmed Assembled Move to flash memory, this bit must be set when the Program_Assembled bit (Command Word 0, bit 12) makes a 1 → 0 transition at the end of the programming cycle. The unit responds by flashing the ERR LED when the writing is complete. If the LED is flashing green at 4 Hz, the write to flash memory was successful. If it flashes green at 1 Hz, then there was an error in writing the data. In either case, power must be cycled to the unit before you can continue. This design decision is to protect the flash memory from constant write commands. The flash memory has a minimum of 10,000 write cycles.
- When using this bit to save the calculated absolute encoder offset value to flash memory, this bit must be set when the Preset Encoder command is issued. (Bit 14 of **CMD_word0** is set to “1”, see page 60.) If the offset is stored without error, the unit will respond by setting the Acknowledge bit. (Bit 13 of **STATUS_word1 Format**, see page 74.)

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

Bits 3-2: Reserved – Must equal “0”.

Bit 1: Motor Current – If reset to “0” when a move command is issued, the motor current will be the value specified in the CoE 8010:07 register. (See *Data Format* on page 55 for more information.) Set to “1” to program the motor current to the value stored in the *Motor_Current* single integer register defined by the ESI file. Motor current can set as a separate command or as part of a move command.

Bit 0: Reserved – Must equal “0”.

Command Blocks

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

Absolute Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	CMD_word0		16#0001
CMD_Word1	UINT	CMD_word1		See pg. 62
Position	DINT	Absolute Target Position	Steps	-8,388,608 and +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	0 to 20. Ignored if bit 1 of Command Word 1 is not set.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.1 Absolute Move Command Block

Command Blocks (continued)**Relative Move**

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0002
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Relative Target Position	Steps	-8,388,608 and +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	0 to 20. Ignored if bit 1 of Command Word 1 is not set.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.2 Relative Move Command Block

Hold Move

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0004
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R6.3 Hold Move Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Command Blocks (continued)**Resume Move**

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0008
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	0 to 20. Ignored if bit 1 of Command Word 1 is not set.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.4 Resume Move Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command. This is typically the case when resuming a move, the words are listed as “Unused” to highlight that the target position of a held move cannot be changed when the move is resumed.

Immediate Stop

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0010
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R6.5 Immediate Stop Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Command Blocks (continued)**Find Home CW**

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0020
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	0 to 20. Ignored if bit 1 of Command Word 1 is not set.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.6 Find Home CW Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Find Home CCW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0040
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	0 to 20. Ignored if bit 1 of Command Word 1 is not set.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.7 Find Home CCW Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Command Blocks (continued)**Jog CW**

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0080
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62 Bit 7 must equal “0”
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	0 to 20. Ignored if bit 1 of Command Word 1 is not set.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.8 Jog Move CW Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Registration Move CW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0080
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62 Bit 7 must equal “1”
Position	DINT	Stopping Distance	Steps	0 and +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current [†]	UINT	Minimum Registration Move Distance	steps	UDINT value between 0 and +8,388,607
Jerk [†]	UINT			

Table R6.9 Registration Move CW Command Block

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

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

Command Blocks (continued)**Jog CCW**

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0100
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62 Bit 7 must equal "0"
Position	DINT	Unused		See Note Below
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	Motor Current	0.1 amps	0 to 20. Ignored if bit 1 of Command Word 1 is not set.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.10 Jog CCW Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Registration Move CCW

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0100
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62 Bit 7 must equal "1"
Position	DINT	Stopping Distance	Steps	0 and +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current†	UINT	Minimum Registration Move Distance	steps	UDINT value between 0 and +8,388,607
Jerk†	UINT			

Table R6.11 Registration Move CCW Command Block

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

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

Command Blocks (continued)**Preset Position**

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0200
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62 Set bit 13 to preset the motor position to the encoder position
Position	DINT	Position Preset Value	Steps	-8,388,608 and +8,388,607
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R6.12 Preset Position Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

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

Reset Errors

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0400
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R6.13 Reset Errors Command Block

Unused words are ignored by the SMD17K, and can be any value, including parameter values from the previous command.

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

Command Blocks (continued)**Run Assembled Move**

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#2000
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62 Blend Move: Bit 9 = "0" Dwell Move: Bit 9 = "1" Move direction is controlled by Bit 4.
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused with Blend Move Dwell Time with Dwell Move	milliseconds	0 to 65,535

Table R6.14 Run Assembled Move Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Preset Encoder Position

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#4000
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Encoder Preset Value	Counts	-8,388,608 and +8,388,607
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R6.15 Preset Encoder Position Command Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Programming Blocks

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

First Block

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#0800
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Unused		See Note Below
Velocity	UDINT	Unused		See Note Below
Acceleration	UINT	Unused		See Note Below
Deceleration	UINT	Unused		See Note Below
Motor_Current	UINT	Unused		See Note Below
Jerk	UINT	Unused		See Note Below

Table R6.16 Assembled Move First Programming Block

Unused words are ignored by the SMD17K and can be any value, including parameter values from the previous command.

Once the first block is transmitted, the SMD17K responds by setting bits 8 and 9 in Status Word 0. (See *STATUS_word0 Format* starting on page 72.) Once these are set, you can then start transmitting Segment Blocks.

Segment Block

ESI File Name	Data Size	Function	Units	Range
CMD_Word0	UINT	<i>CMD_word0</i>		16#1800
CMD_Word1	UINT	<i>CMD_word1</i>		See pg. 62
Position	DINT	Relative Target Position	Steps	-8,388,608 and +8,388,607
Velocity	UDINT	Programmed Speed	Steps/Second	Value between the configured Starting Speed and 2,999,999
Acceleration	UINT	Acceleration	Steps/sec/ms	1 to 5000
Deceleration	UINT	Deceleration	Steps/sec/ms	1 to 5000
Motor_Current	UINT	<i>Reserved</i>		Must equal zero for compatibility with future releases.
Jerk	UINT	Acceleration Jerk		0 to 5000

Table R6.17 Assembled Move Segment Programming Block

Note that each Segment Block starts with bits 11 and 12 in Command Word 0 set to “1” (16#1800). When the unit sees bit 12 of Command Word 0 set, it will accept the block and reset bit 9 in Status Word 0. When your program sees this bit reset, it must respond by resetting bit 12 of Command Word 0. The SMD17K will respond to this by setting bit 9 in Status Word 0 and the next Segment Block can be written to the unit. You can write a maximum of sixteen Segment Blocks for each Assembled Move.

Input Data Format

The correct format for the Network Input Data when the SMD17K is in Command Mode is shown below.

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

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

Table R6.18 Network Input Data Format: Command Mode

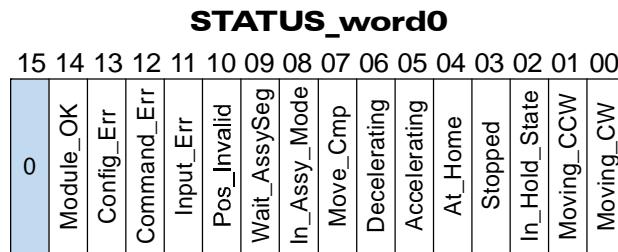
STATUS_word0 Format

Figure R6.4 Command Mode: Status Word 0 Format

Bit 15: **Mode_Flag** – Set to “0” when in Command Mode.

Bit 14: **Module_OK** – “1” when the SMD17K is operating without a fault, “0” when an internal fault condition exists.

Bit 13: **Configuration_Error** – “0” when the unit has a valid configuration in memory. “1” if there is one or more errors in the CoE SDO configuration data written to the unit. See [Configuration Data Format](#), which starts on page 55, for information on the format of the Configuration Data.

Bit 12: **Command_Error** – “1” when an invalid command has been written to the SMD17K. This bit can only be reset by the *Reset_Errors* bit, Command Word 0, Bit 10.

Bit 11: **Input_Error** – “1” when:

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

This bit is reset by a *Reset Errors* command. The format of the command is given on page 69.

Input Data Format (continued)**STATUS_word0 Format (continued)**

Bit 10: Position_Invalid – “1” when:

- 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.
- A configuration change is written to the unit through the CoE SDO interface.

Absolute moves cannot be performed while the position is invalid.

Bit 9: Waiting_For_Assembled_Segment – The SMD17K sets this bit to tell the host that it is ready to accept the data for the next segment of your assembled move profile. Its use is explained in the **Assembled Move Programming** section of this manual starting on page 36.

Bit 8: In_Assembled_Mode – The SMD17K sets this bit to signal the host that it is ready to accept assembled move profile programming data. Its use is explained in the **Assembled Move Programming** section of this manual starting on page 36.

Bit 7: Move_Complete – Set to “1” when the present Absolute, Relative, Jog, Registration, or Assembled Move command completes without error. This bit is reset to “0” when the next move command is written to the SMD17K, when the position is preset, or a Reset Errors command is issued to the unit. This bit is also set along with the *Command_Error* bit (Bit 12 of this word), when any Jog Move or Registration Move parameters are outside of their valid ranges. This bit is not set on a command error for any other type of command. Finally, this bit is not set at the end of a homing operation.

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

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

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

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

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

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

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

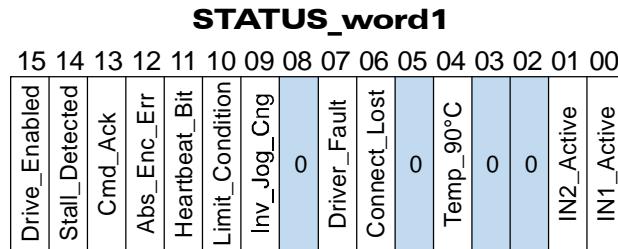
Input Data Format (continued)**STATUS_word1 Format**

Figure R6.5 Command Mode: Status Word 1 Format

Bit 15: **Drive_Is_Enabled** – Set to “1” when the motor driver section of the SMD17K 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. This bit will remain set if the motor current has been removed because motion is not occurring and the Idle Current Reduction is programmed to its 0% setting.

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

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

- Preset Position
- Preset Encoder Position
- Reset Errors

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

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

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

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

Bit 11: **Heartbeat_Bit** – This bit will change state approximately every 500 milliseconds. Monitor this bit to verify that the unit and network connection are operating correctly. Note that this bit is only available while the unit is in Command Mode.

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

Bit 9: **Invalid_Jog_Change** – Set during a Jog Move if parameters are changed to invalid values. Parameters that can be changed during a Jog Move are Programmed Speed, Acceleration, and Deceleration. Issuing a Reset Errors command will not reset this bit. This bit will reset when valid jog data is sent from the host controller.

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

Input Data Format (continued)**STATUS_word1 Format (continued)**

- Bit 7:** **Driver_Fault** – If the driver section of the SMD17K is enabled, this bit will be a “1” during a Over-temperature Fault. Even though the driver is enabled, it will not supply current to the motor until the motor’s temperature decreases to a safe value. At this point the fault will clear itself.
- Bit 6:** **Reserved** – Will always equal zero.
- Bit 5:** **PreOp_SafeOp** – Set to “1” when the network is in Pre_Op or Safe_Op modes. Moves cannot be commanded while in these modes. Reset to “0” when the network is in Operational mode. Moves can only be commanded when in Op mode.
- Bit 4:** **Temperature_Above_90°C** – This bit is set to “1” when the processor internal temperature exceeds 90°C. At this point, the heatsink temperature is typically near 83°C. If this bit trips often and you want to lower the operating temperature of the unit, consider changing how the device is mounted, or installing a fan to force additional airflow over the device.
- Bit 3:** **Reserved** – Will always equal zero.
- Bit 2:** **Reserved** – Will always equal zero.
- Bit 1:** **IN2_Active** – “1” when Input 2 is in its active state. The active state of the input is programmed as explained in the ***CFG_Word_1 Format*** section starting on page 58.
- Bit 0:** **IN1_Active** – “1” when Input 1 is in its active state. The active state of the input is programmed as explained in the ***CFG_Word_1 Format*** section starting on page 58.

Notes on Clearing a Driver Fault

A Driver Fault occurs when there is an over temperature condition. When a Driver Fault occurs, the SMD17K will set bit 7 of the Status Word 1 word in the Network Input Data. Even though the driver is enabled, it will not supply current to the motor. This fault is not self clearing. You must cycle power to the unit to clear this fault. If the temperature of the motor is too high after the power cycle, the over temperature fault will trip again.

Notes

TASK 1

INSTALLING THE SMD17K

1.1 Location

1.1.1 IP50 Rated Units (SMD17K-M12)

SMD17K 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 (SMD17K-M12S)

SMD17K 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 (SMD17K-M12P)

SMD17K 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



Electrostatic discharge can damage the SMD17K 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 M12 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 SMD17K units.

1.3 Operating Temperature Guidelines

Due to the onboard electronics, the maximum operating temperature of the SMD17K is limited to 203°F/95°C. Depending on the operating current setting, move profiles, idle time, and the idle current reduction setting, it is possible to exceed this temperature in a thermally isolated environment. As explained in the mounting section, mounting the SMD17K to a large metal heatsink is the best way to limit the operating temperature of the device. Operating temperature should be monitored during system startup to verify that the maximum motor temperature remains below this 203°F/95°C specification. The motor is rated to 266°F/130°C, but the operating temperature of the device as a whole should be limited to 203°F/95°C. SMD17K devices have an onboard thermistor that measures the temperature of the electronics and will remove motor current if the operating temperature exceeds this 203°F/95°C limit. This overtemperature fault is also reported in the Network Input Data.

1.4 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.



- 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.

1.4.1 SMD17K-M12 Mounting

The SMD17K-M12 units are not water tight. Their IP50 rating makes them acceptable for use in dusty environments with occasional condensation. These units should be mounted in such a way that condensation will naturally drain off of the unit instead of pooling at the motor shaft, where the motor wires exit the motor, or on the motor laminations.

1.4 Mounting (continued)

1.4.2 SMD17K-M12S Mounting

The SMD17K-M12S units are not water tight. Their IP64 rating makes them acceptable for use in dusty environments, environments with condensation, and environments where the unit may be exposed to splashing water. SMD17K-M12S units should be mounted in such a way that condensation and liquids will naturally drain off of the unit instead of pooling on the motor laminations.

1.4.3 SMD17K-M12P Mounting

The SMD17K-M12P 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.

1.4.4 SMD17K-80 Outline Drawing

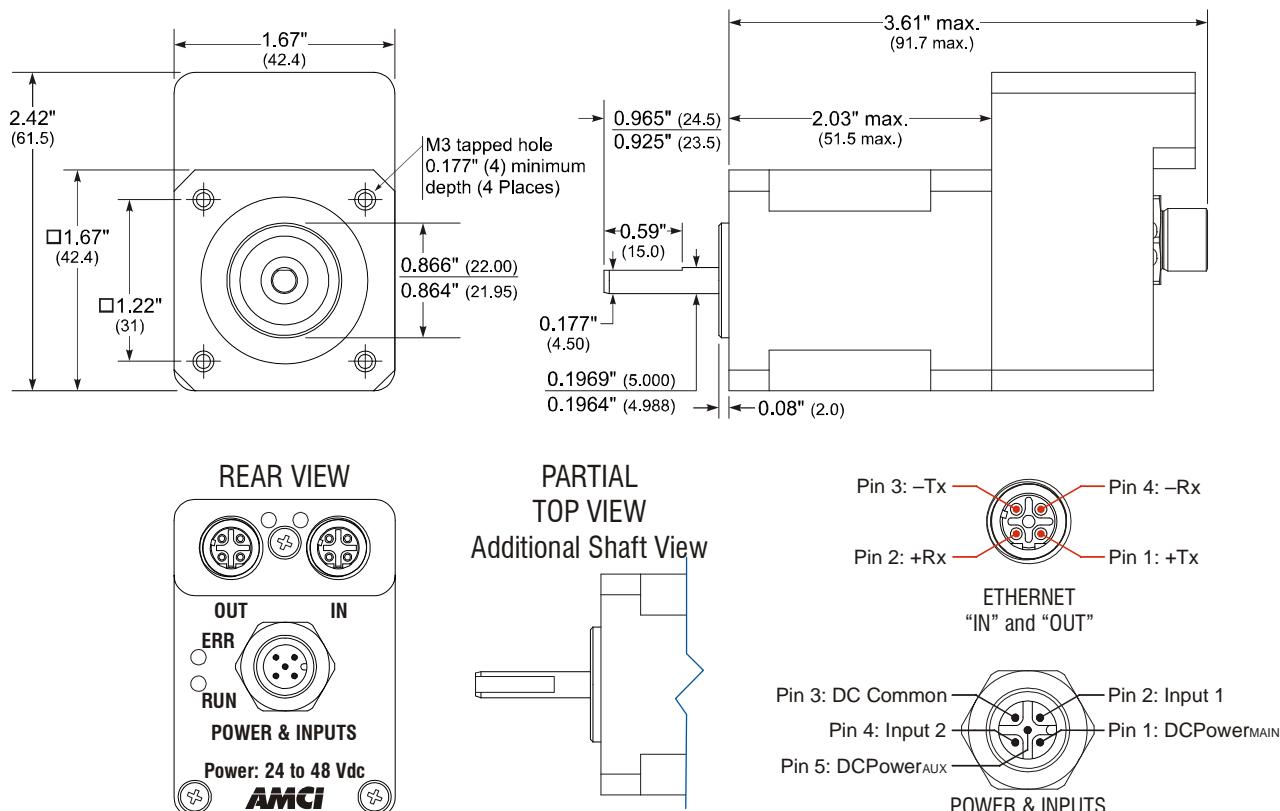


Figure T1.1 SMD17K-80 Outline Drawing

1.4.5 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)



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

1.5 Power and Input Connector

The Power and Input Connector is located on the back of the SMD17K unit below the Ethernet connectors. The connector is a standard five pin A-coded M12 connector that is rated to IP67 when the mate is properly attached. Figure T1.2 shows the pinout for the Input Connector when viewed from the back of the unit.

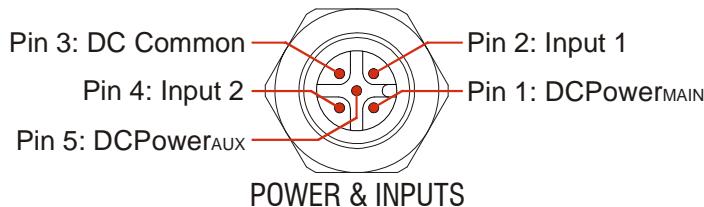


Figure T1.2 Power and Input Connector

Digital inputs on the SMD17K units are single ended and referenced to the DC Common pin. There are two power pins.

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

Using the DC Power_{AUX} pin is optional. If your application requires you to cut power to your motor under some conditions, using the DC Power_{AUX} pin allows you to cut power to your motor without losing your network connection.

1.5.1 Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17K units provided that the manufacturer follows the A-coded M12 standards. The following connector and cordset is available from AMCI.

Connector

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

Table T1.1 Compatible Connectors

Power Cordset

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

Table T1.2 Power Cordset

1.6 Power Wiring

The SMD17K accepts 24 to 48Vdc as its input power. This should be derived from a power limited power supply.

CAUTION

Do not apply 120 Vac to any pins of an SMD17K. If this occurs, the unit will be damaged and you will void the unit's warranty.

WARNING

The SMD17K 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.

The power and input connector is a standard five pin A-coded M12 connector. The MS-31 connector that is available from AMCI is IP67 rated when properly installed. The CNPL-2M and CNPL-5M cables are also IP67 rated when properly installed.

1.6.1 MS-31 Connector

The MS-31 Connector accepts 18 AWG wire on all of its contacts and a cable diameter of 6.00 to 8.00 mm (0.236" to 0.315"). AMCI strongly suggests using 18 AWG wire for the power connections.

1.6.2 CNPL-2M and CMPL-5M Cables

The CNPL cables are 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. If these cables do not satisfy these requirements in your application, an MS-31 connector and appropriately sized cable can be used.

1.6.3 Extending Power Leads

AMCI strongly suggests using 18 AWG or larger wire when the power leads to the SMD17K unit must be extended. This is to minimize power loss in the cable.

1.6.4 Main Power Wiring Only

Figure T1.3 below shows how to wire power to the SMD17K units when only powering with the DCPOWERMAIN connection. Colors in parentheses are the appropriate wire colors for the CNPL-2M and CNPL-5M cables.

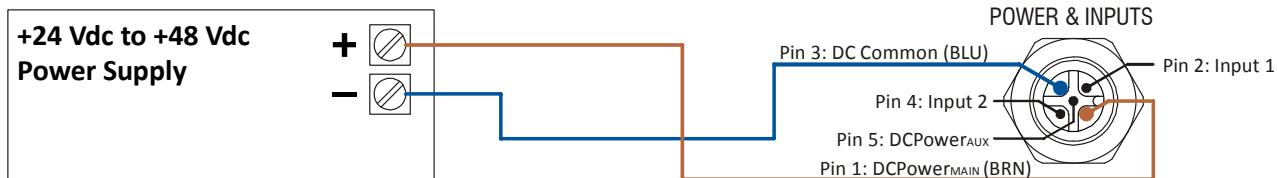


Figure T1.3 SMD17K Main Power Wiring

1.6 Power Wiring (continued)

1.6.5 Auxiliary Power, Single Supply

Figure T1.4 below shows how to wire main and auxiliary power to the SMD17K units when using a single power supply. Colors in parentheses are the appropriate wire colors for the CNPL-2M and CNPL-5M cables.

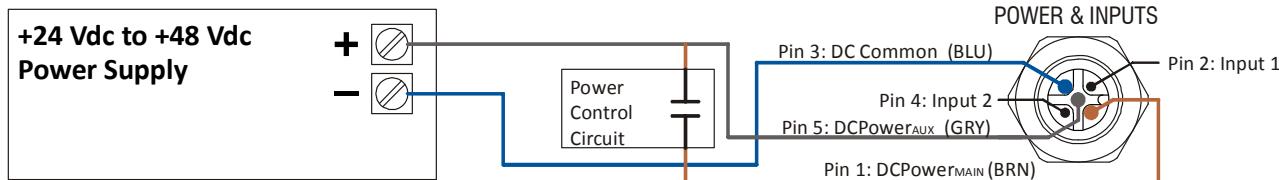


Figure T1.4 SMD17K Auxiliary Power, Single Supply Wiring

1.6.6 Auxiliary Power, Dual Supply

Figure T1.5 below shows how to wire main and auxiliary power to the SMD17K units when using two power supplies. Note that the two supplies can supply different voltages. Colors in parentheses are the appropriate wire colors for the CNPL-2M and CNPL-5M cables.

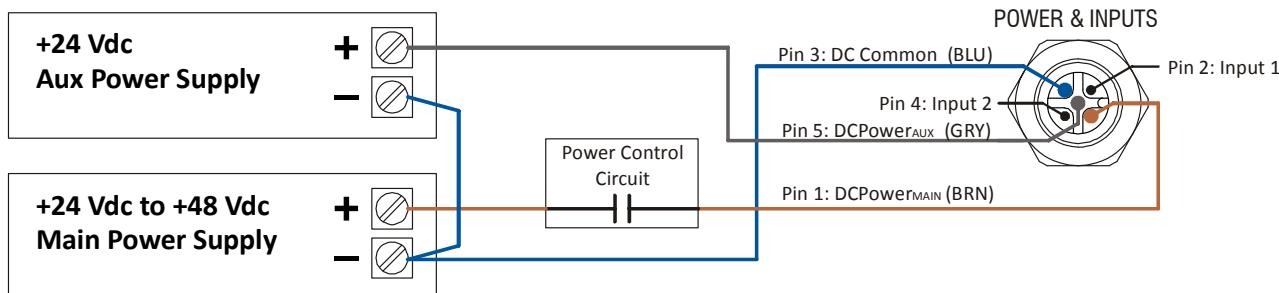


Figure T1.5 SMD17K Auxiliary Power, Dual Supply Wiring

1.6.7 Power Supply Troubleshooting

When troubleshooting systems that use the DCPOWERAUX supply, the SMD17K will respond to motion commands by updating the motor position and status bits even if the DCPOWERMAIN supply input is disconnected. The only indication that the DCPOWERMAIN supply is disconnected is that the motor has no holding torque and motion does not physically occur.

If Stall Detection is enabled, the *Stall_Detected* bit will trip if the move is greater than 45°. The *Stall_Detected* bit is bit 14 of Status Word 1.

Verify that the DCPOWERMAIN supply is connected to the SMD17K before proceeding.

1.7 Input Wiring

Inputs 1 and 2 are single ended inputs that share the DC Common return pin. They accept 3.5 to 27 Vdc without the need for an external current limiting resistor. Figure T1.6 below shows how to wire discrete DC sourcing and sinking sensors to inputs 1 and 2 of the SMD17K. Colors in parentheses are the appropriate wire color of the CNPL-5M cable.

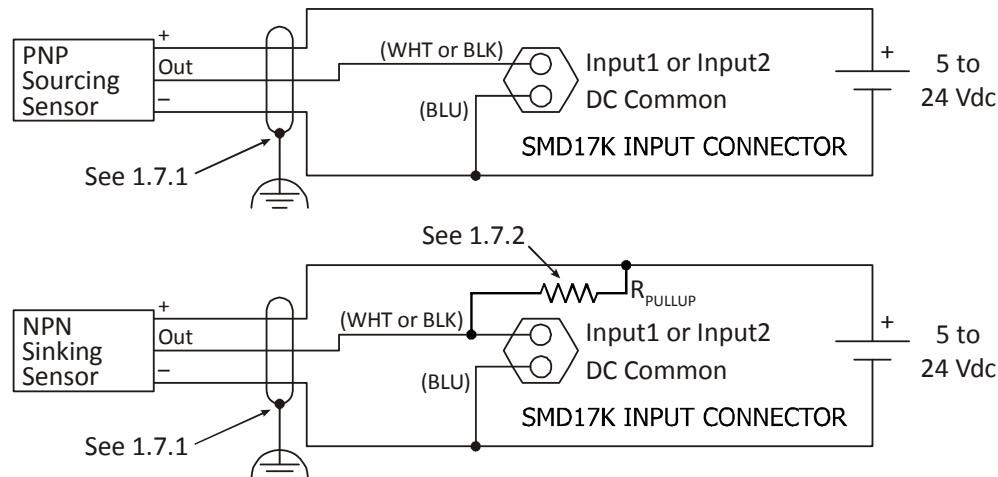


Figure T1.6 Input Wiring

1.7.1 Cable Shields

Because they are low power signals, cabling from the sensor to the SMD17K 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 SMD17K.

1.7.2 Sinking Sensors Require a Pull Up Resistor

Sinking output sensors require an external pull up resistor because inputs 1 and 2 of the SMD17K 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 Pull Up Resistor Values for DC Inputs

The logical states of the sensor and SMD17K input will be reversed. The SMD17K input is off when the sensor is active. You can set the logic state of the input when you configure the unit.

1.8 Network Connectors

Figure T1.7 shows the Ethernet connector pinout when viewed from the back of the SMD17K. The Ethernet ports on the units are both 100Base-TX and are auto MDI-X capable. This means that a standard cable can be used when connecting the SMD17K to any device.

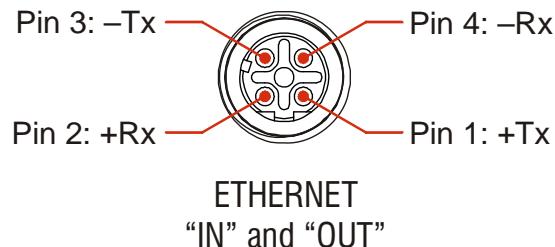


Figure T1.7 M12 Ethernet Connector Pinout

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

The two connectors are labeled “IN” and “OUT”. The IN port must be connected to the upstream devices in the EtherCAT network. The OUT port is for downstream devices.

1.8.1 Compatible Connectors and Cordsets

Many different connectors and cordsets are available on the market, all of which will work with the SMD17K provided that the manufacturer follows the connector and Ethernet standards. AMCI offers the following mating connector and cordsets that mate with the Ethernet port connectors.

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

Table T1.4 Compatible Ethernet Connectors and Cordsets

1.8.2 TIA/EIA-568 Color Codes

There are two color codes in common use when wiring Ethernet connections with twisted pairs. Either one of these standards is acceptable. The CNER-5M cable available from AMCI follows the 568B standard. Note that accidentally reversing the Tx/Rx pairs will not affect the operation of the SMD17K. Each unit has an “auto-sense” port that will automatically adjust for swapped pairs.

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

Table T1.5 TIA/EIA Color Codes

TASK 2

ETHERCAT SYSTEM CONFIGURATION

This chapter outlines how to add an SMD17K unit to an EtherCAT system. The TwinCAT version 3 software is used as an example.

2.1 Install the ESI file

2.1.1 Obtain the ESI file

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

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

Simply download the ZIP file and extract it.

2.1.2 Install the ESI file

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

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

2.1.3 Restart the Programming System If Needed

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

2.2 Add the SMD17K to the Project



This section assumes that the TwinCAT software is in Config Mode.

2.2.1 Scan for the SMD17K Device

- 1) Attach the SMD17K device to the network and power up the device.
- 2) At this point, the device is in its INIT state. The ERR LED will be on green and the RUN LED will be off.
- 3) Right click on the EtherCAT adapter that the SMD17K is attached to. In the drop down menu that opens, select the “Scan” option. (If the “Scan” option is not available, the TwinCAT software is not in Config Mode.)

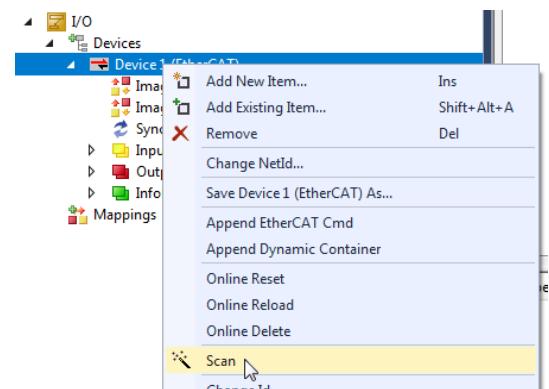


Figure T2.1 Scan for the SMD17K Device

2.2.2 Rename the Device

The SMD17K will appear in the device tree and the name will typically begin with “Box”.

- 1) Click on the unit in the device tree.
- 2) If needed, click on the “General” tab in the window that opens.
- 3) The device name for the SMD17K can be changed in the *Name:* field.

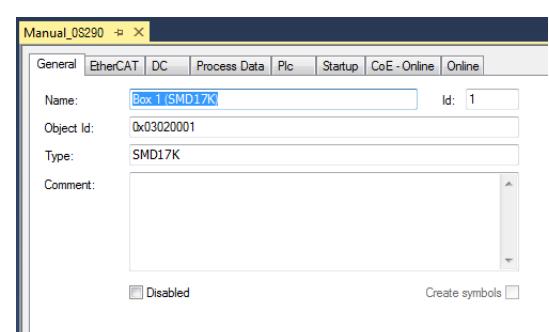


Figure T2.2 Rename the SMD17K Device

2.2 Add the SMD17K to the Project (continued)

2.2.3 Configure the SMD17K

- 1) Click on the “CoE - Online” tab.
- 2) Click on the [+] button next to the 6000:0 Index field (Inputs) to expand it. Note the value of the STATUS_word1 register (6000:01). This value is typically 0x4408.
- 3) Click on the [+] button next to the 8010:0 Index field to expand it.
- 4) The configuration data is available under the seven sub-index fields of the 8010:0 Index. *Configuration data can be temporarily changed here.* Double click on any field to open the *Set Value Dialog* screen to set the parameter to the desired value. Table R5.3, **Configuration Word 0 Bits** found on page 57, allows you to calculate the proper hexadecimal value of CFG_word0 for your application.
- 5) Once you set a parameter, check the value of the STATUS_word1 register (6000:01). If the value has bit 13 set to “1”, there is an error in your configuration setting. (An example is a STATUS_word1 register change from 0x4408 to 0x6408.)
- 6) To make these changes permanent, you must define values under the “Startup” tab. Click on the “Startup” tab.
- 7) Right click on the blank surface and select “Add New Item...” in the resulting pop up menu.
- 8) Click on the [+] button next to the 8010:0 Index field to expand it.
- 9) Double click on the field that must be changed. This opens a dialog that will allow you to set the field with a decimal or hexadecimal value.
- 10) Click the [OK] button in the dialog box to set the new value.
- 11) Click the [OK] button in the “Edit CANopen Startup Entry” screen to accept the new startup parameter setting.

Index	Name	Flags	Value
1C13:0	SyncManager 3 assignment	> 1 <	
1C32:0	SM output parameter	> 32 <	
1C33:0	SM input parameter	> 32 <	
6000:0	Inputs	> 7 <	
6000:01	STATUS_word0	RO P	0x4408 (17416)
6000:02	STATUS_word1	RO P	0x0004 (4)
6000:03	Motor_Position	RO P	0x00000000 (0)
6000:04	Encoder_Position	RO P	0xFFFFD3155 (-183979)
6000:05	Trapped_Encoder_Position	RO P	0x00000000 (0)
6000:06	Motor_Current	RO P	0x0014 (20)
6000:07	Jerk	RO P	0x0000 (0)
7010:0	Outputs	> 8 <	
8010:0	Configuration	> 7 <	
8010:01	CFG_word0	RW P	0x8400 (33792)
8010:02	CFG_word1	RW P	0x0003 (3)
8010:03	Starting_Speed	RW P	0x00000064 (100)
8010:04	Motor_Resolution	RW P	0x0800 (2048)
8010:05	Encoder_Resolution	RW P	0x0800 (2048)
8010:06	Idle_current_reduction	RW P	0x0014 (20)
8010:07	Motor_Current	RW P	0x0014 (20)

Figure T2.3 SMD17K Configuration Registers

Index	Name	Flags	Value
10F1:0	Error Settings	> 2 <	
10F8	Timestamp Object	RW P	0x0
1C32:0	SM output parameter	> 32 <	
1C33:0	SM input parameter	> 32 <	
7010:0	Outputs	> 8 <	
8010:0	Configuration	> 7 <	
8010:01	CFG_word0	RW P	0x8400 (33792)
8010:02	CFG_word1	RW P	0x0007 (7)
8010:03	Starting_Speed	RW P	0x00000064 (100)
8010:04	Motor_Resolution	RW P	0x0800 (2048)
8010:05	Encoder_Resolution	RW P	0x0800 (2048)
8010:06	Idle_current_reduction	RW P	0x0014 (20)
8010:07	Motor_Current	RW P	0x0011 (17)

Figure T2.4 SMD17K Configuration Registers

2.3 Create a DUT

AMCI sample programs use a DUT (Defined User Type) to group the I/O variables associated with each SMD17K. Once the DUT is defined, a separate variable is created for each device on the machine.

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

```
TYPE DUT_AMCI_SMD17K :  
STRUCT  
    //Outputs:  
    nCMD0 AT %Q*      : UINT;  
    nCMD1 AT %Q*      : UINT := 16#8000;  
    nTargetPos AT %Q* : DINT;  
    nTargetVel AT %Q* : UDINT;  
    nAccel AT %Q*     : UINT;  
    nDecel AT %Q*     : UINT;  
    nMtrCurr AT %Q*   : UINT;  
    nJerk AT %Q*      : UINT := 0;  
    //Inputs:  
    nStatus0 AT %I*   : UINT;  
    nStatus1 AT %I*   : UINT;  
    nMotorPos AT %I*  : DINT;  
    nEncPos AT %I*    : DINT;  
    nTrappedEnc AT %I*: DINT;  
    nLastCurrent AT %I*: UINT;  
    nLastJerk AT %I*   : UINT;  
END_STRUCT  
END_TYPE
```

2.4 Create Variables Based on the DUT

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

```
st_Axis1_SMD17K : DUT_AMCI_SMD17K;  
st_Axis2_SMD17K : DUT_AMCI_SMD17K;
```

2.5 Link Variable Names to I/O Words

- 1) From the menu, select *Build* → *Build Solution*. (Ctrl+Shift+B). The build will fail with a message that at least one variable must be linked to a task variable. Click [OK] to close the message.
- 2) In the I/O tree, expand the SMD17K until the input and output words are visible.
- 3) Right click on the STATUS_word0 input word. In the pop up menu that opens, select “Change Link...”

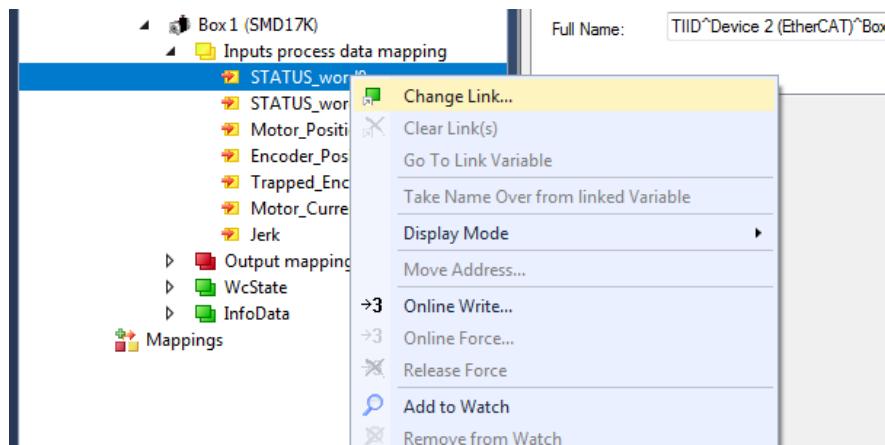


Figure T2.5 Change Link Pop up Menu

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

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

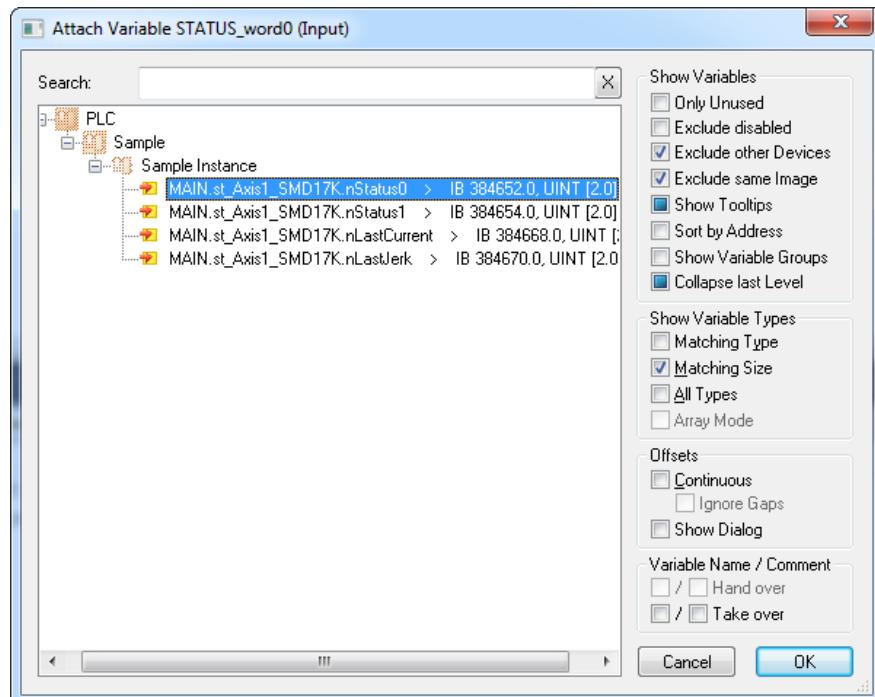


Figure T2.6 Change Link Pop up Menu

- 5) Repeat steps three and four for the remaining input and output words of the SMD17K.

TASK 3 (OPTIONAL)

DISTRIBUTED CLOCK - SYNC0 SETUP

This chapter outlines the steps need to configure the SMD17K to synchronize to the Sync0 signal from the Distributed Clock (DC) feature instead of the SyncManager 2 event. This configuration is optional.

By default, the SMD17K acts on the data from the EtherCAT master as soon as it is transmitted to the device. This mode of operation is called “SM-Synchron”. The data transfer is synchronized with the SyncManager 2 event. This synchronization is more than adequate for most machine types.

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

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

NOTE When using the Distributed Clock (DC) functionality of the EtherCAT system, Input 3 is not available for use. Internally, the Input 3 circuitry is used by the SYNC0 signal to trigger a synchronous command update.

3.1 Verify Main PLC Task Timing

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

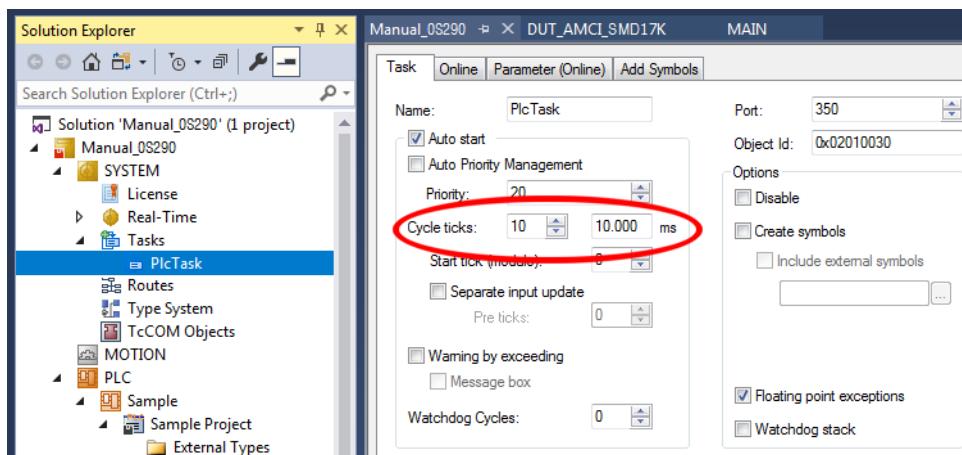


Figure T3.1 Main PLC Task Timing

- 2) If this value is between 2 and 50 milliseconds and the correct update time for the SMD17K, skip to step 3.3. When setting the *SYNC 0 -> Sync Unit Cycle* multiplier in step 6 of 3.3, use a value of 1.
- 3) If the timing in step 1 above is not correct for the SMD17K, but the correct time is an integer multiple of this timing, then skip to step 3.3. When setting the *SYNC 0 -> Sync Unit Cycle* multiplier in step 6 of 3.3, use the correct integer multiplier.
- 4) If the timing in step 1 above is not correct for the SMD17K, and the correct time is *not* an integer multiple of this timing, then proceed with step 3.2.

3.2 Create New PLC Task

You only have to follow this step if the Main PLC task timing verified in step 3.1 above is not the correct update time for the SMD17K and the correct time is not an integer multiple of the Main PLC task timing.

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

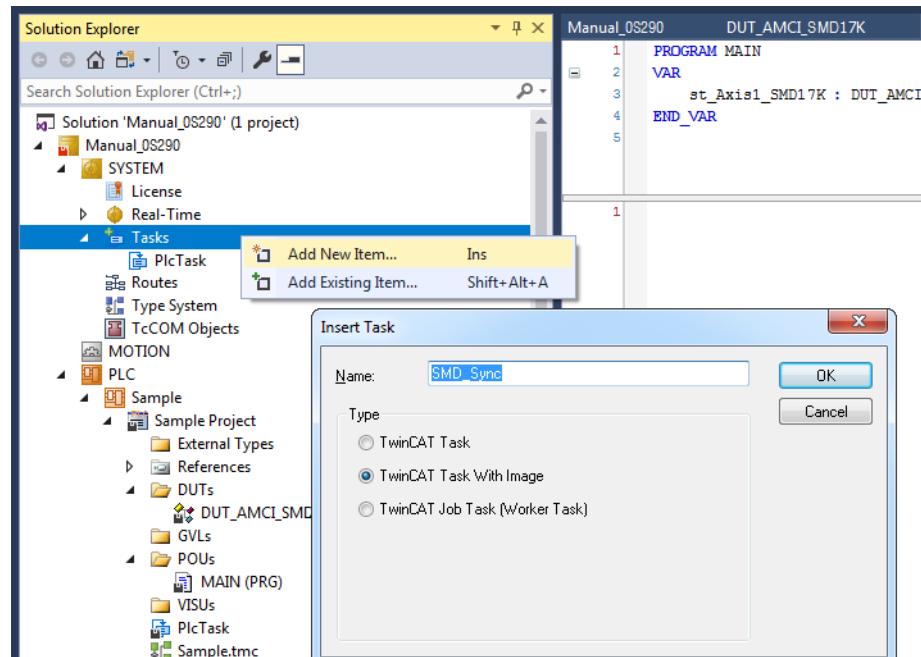


Figure T3.2 Create New Task

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

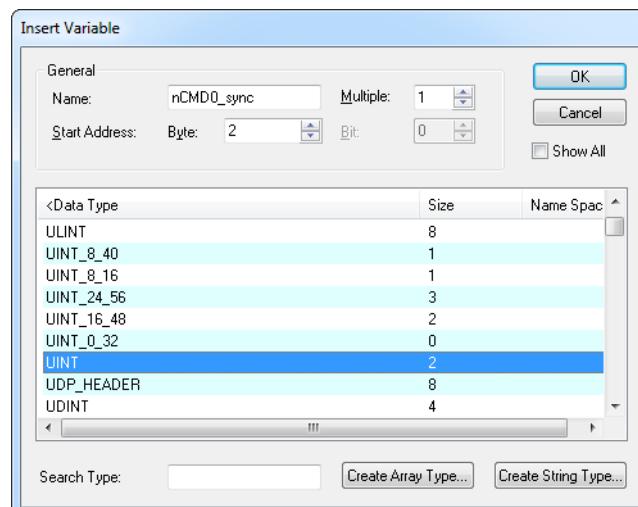


Figure R6.6 Setting Task Variable Name and Size

3.2 Create New PLC Task (continued)

- 7) In the *Solutions Explorer*, click on the name of the new variable to select it. The information on the variable will appear in a pane. Click on the [Linked to...] button in this pane to open the *Attach Variable* window.
- 8) Double click on the CMD_word0 variable of the correct SMD17K. This will link the CMD_word0 of the unit to the variable created for the task. The SMD17K will update at the time specified by the new task.

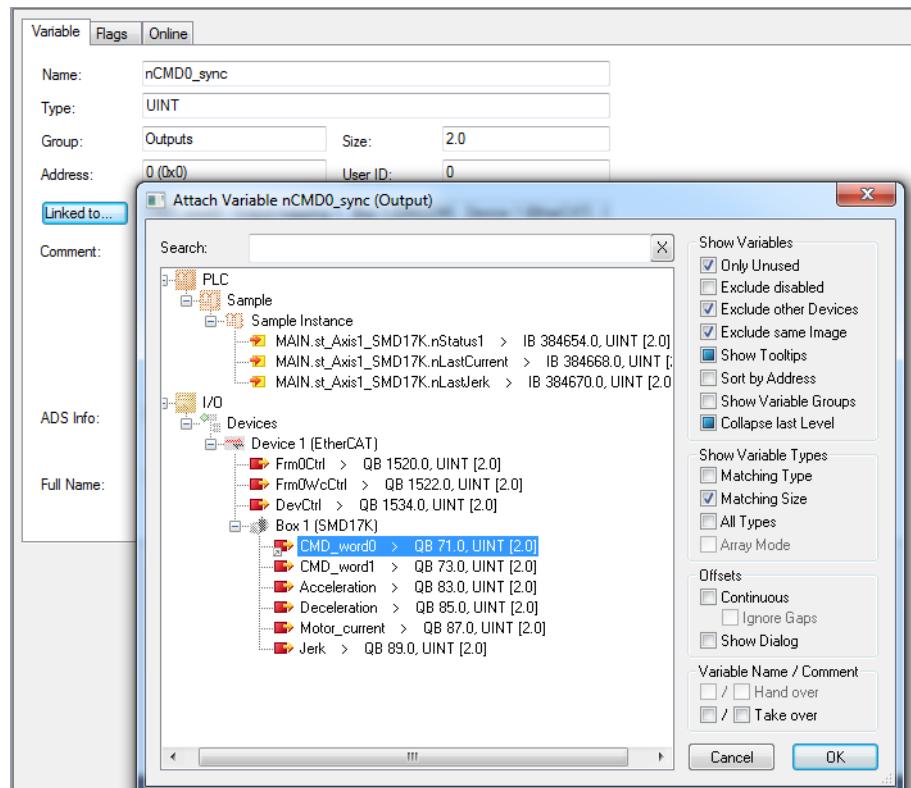


Figure T3.3 Linking Task Variable to SMD17K

3.3 Set Operational Mode

- 1) If necessary, expand the I/O Device tree until the SMD17K is visible.
- 2) Double click on the SMD17K to open the setting window for the device.
- 3) Click on the “DC” tab.
- 4) Under the Operation Mode setting, click on the drop down menu and select “DC_Synchron”.
- 5) Click on the [Advanced Settings...] button.
- 6) Verify or set the following:
 - *Cyclic Mode* -> *Enable* checkbox is checked.
 - *Sync Unit Cycle (μs)* is equal to the time of the correct task. If not, the variables are not linked correctly.
 - *SYNC 0* -> *Sync Unit Cycle* radio button is enabled. In the drop down menu, set the multiplier as needed so that the *Sync Unit Cycle (μs)* time multiplied by the multiplier value is between 2 and 50 milliseconds.
 - *Enable SYNC 0* checkbox is checked.
 - *Enable SYNC 1* checkbox is not checked.
- 7) Click [OK] to close the window.

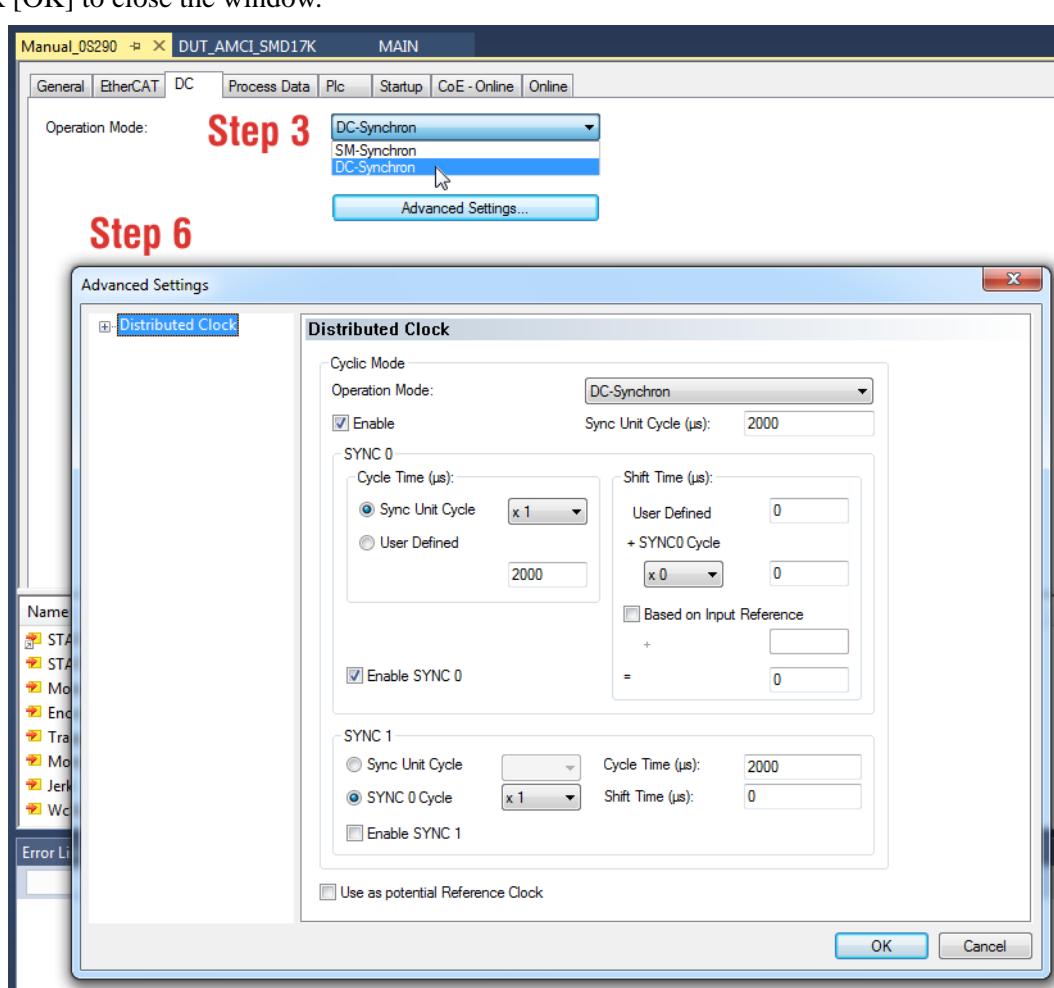


Figure T3.4 Distributed Clock Settings

3.4 Set CFG_word1 Value to Enable SYNC0

Continuing from step 3.3...

- 1) Click on the “CoE - Online” tab in the settings window.
- 2) Expand the 8010:0 tree to see the value of the 8010:02 register. (CFG_word1 value)
- 3) This value must have bit 6 set to enable the synchronous mode in the SMD17K. If this value is greater than or equal to 64, then the bit is set and you are finished with this step of the procedure. If the value is less than 64, then the bit is not set. This new value must be specified in the Startup tab in order to make the change permanent.
- 4) Click on the “Startup” tab.
- 5) Right click on the blank surface and select “Add New Item...” in the resulting pop up menu.
- 6) Click on the [+] button next to the 8010:0 Index field to expand it.
- 7) Double click on the 8010:02 field. This opens a dialog that will allow you to set the field with a decimal or hexadecimal value. If setting in decimal, the new value is the present value from step 3 + 64.
- 8) Click the [OK] button in the dialog box to set the new value.
- 9) Click the [OK] button in the Edit CANopen Startup Entry screen to accept the new startup parameter setting.

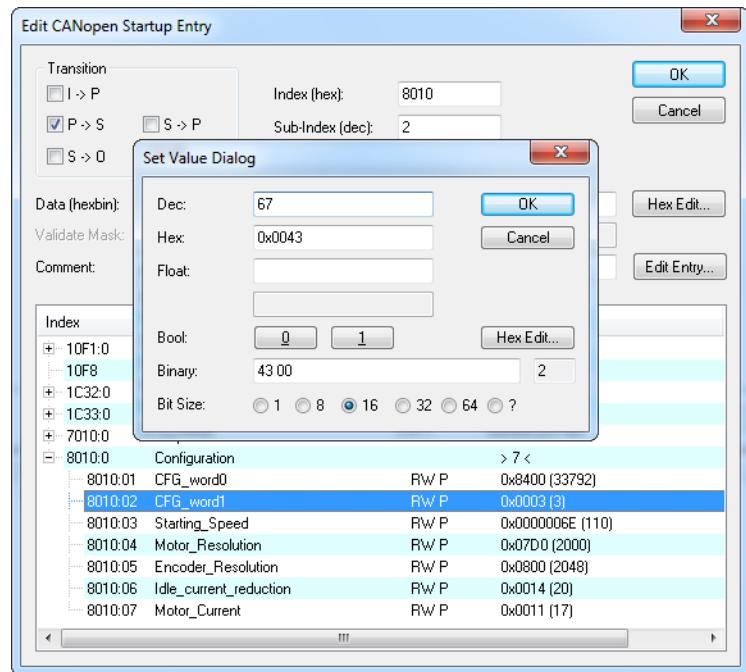


Figure T3.5 Set DC_Sync Bit in Configuration Word 1



ADVANCED MICRO CONTROLS INC.

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

www.amci.com

LEADERS IN ADVANCED CONTROL PRODUCTS

