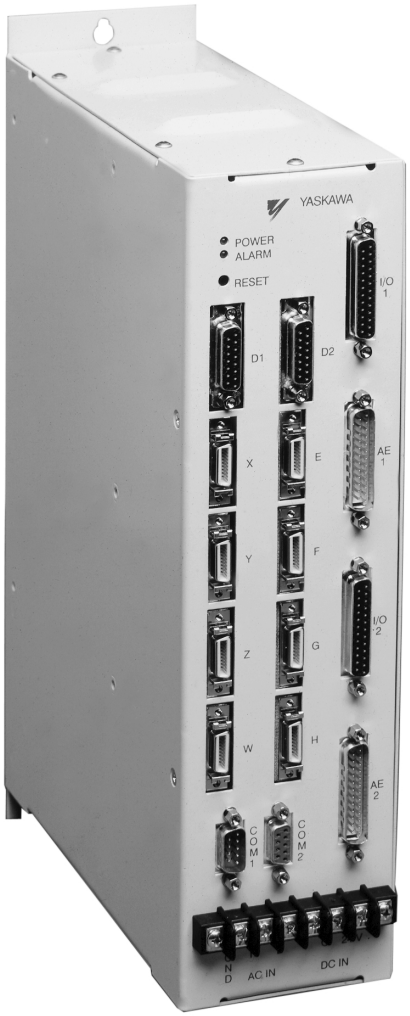
YASKAWA

A World of Automation Solutions™

SMC-2000 Multi-Axis Motion Controller

User's Guide Version 3.1



YASKAWA ELECTRIC AMERICA, INC.
Chicago-Corporate Headquarters 2121 Norman Drive South, Waukegan, IL 60085, U.S.A.
Phone: (847) 887-7000 Fax: (847) 887-7310 Internet: http://www.yaskawa.com
MOTOMAN INC.
805 Liberty Lane, West Carrollton, OH 45449, U.S.A.
Phone: (937) 847-6200 Fax: (937) 847-6277 Internet: http://www.motoman.com
YASKAWA ELECTRIC CORPORATION
New Pier Takeshiba South Tower, 1-16-1, Kaigan, Minatoku, Tokyo, 105-0022, Japan
Phone: 81-3-5402-4511 Fax: 81-3-5402-4580 Internet: http://www.yaskawa.co.jp
YASKAWA ELETRICO DO BRASIL COMERCIO LTDA.
Avenida Fagundes Filho, 620 Bairro Saude Sao Paolo-SP, Brasil CEP: 04304-000
Phone: 55-11-5071-2552 Fax: 55-11-5581-8795 Internet: http://www.yaskawa.com.br
YASKAWA ELECTRIC EUROPE GmbH
Am Kronberger Hang 2, 65824 Schwalbach, Germany
Phone: 49-6196-569-300 Fax: 49-6196-888-301 Internet: http://www.yaskawa.de
MOTOMAN ROBOTICS AB
Box 504 S38525, Torsas, Sweden
Phone: 46-486-48800 Fax: 46-486-41410
MOTOMAN ROBOTEC GmbH
Kammerfeldstrabe 1, 85391 Allershausen, Germany
Phone: 49-8166-900 Fax: 49-8166-9039
YASKAWA ELECTRIC UK LTD.
1 Hunt Hill Orchardton Woods Cumbernauld, G68 9LF, Scotland, United Kingdom
Phone: 44-12-3673-5000 Fax: 44-12-3645-8182
YASKAWA ELECTRIC KOREA CORPORATION
Paik Nam Bldg. 901 188-3, 1-Ga Euljiro, Joong-Gu, Seoul, Korea
Phone: 82-2-776-7844 Fax: 82-2-753-2639
YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.
Head Office: 151 Lorong Chuan, #04-01, New Tech Park Singapore 556741, SINGAPORE
Phone: 65-282-3003 Fax: 65-289-3003
TAIPEI OFFICE (AND YATEC ENGINEERING CORPORATION)
10F 146 Sung Chiang Road, Taipei, Taiwan
Phone: 886-2-2563-0010 Fax: 886-2-2567-4677
YASKAWA JASON (HK) COMPANY LIMITED
Rm. 2909-10, Hong Kong Plaza, 186-191 Connaught Road West, Hong Kong
Phone: 852-2803-2385 Fax: 852-2547-5773
BEIJING OFFICE
Room No. 301 Office Building of Beijing International Club,
21 Jianguomanwai Avenue, Beijing 100020, China
Phone: 86-10-6532-1850 Fax: 86-10-6532-1851
SHANGHAI OFFICE
27 Hui He Road Shanghai 200437 China
Phone: 86-21-6553-6600 Fax: 86-21-6531-4242
SHANGHAI YASKAWA-TONJI M & E CO., LTD.
27 Hui He Road Shanghai 200437 China
Phone: 86-21-6533-2828 Fax: 86-21-6553-6677
BEIJING YASKAWA BEIKE AUTOMATION ENGINEERING CO., LTD.
30 Xue Yuan Road, Haidian, Beijing 100083 China
Phone: 86-10-6232-9943 Fax: 86-10-6234-5002
SHOUGANG MOTOMAN ROBOT CO., LTD.
7, Yongchang-North Street, Beijing Economic & Technological Development Area,
Beijing 100076 China
Phone: 86-10-6788-0551 Fax: 86-10-6788-2878
YEA, TAICHUNG OFFICE IN TAIWAN
B1, 6F, No. 51, Section 2, Kung-Yi Road, Taichung City, Taiwan, R.O.C.
Phone: 886-4-2320-2227 Fax: 886-4-2320-2239
Phone: 55-11-5071-2552 Fax: 55-11-5581-8795 Internet: http://www.yaskawa.com.br

Yaskawa Electric America, Inc., September 2001 ATSA-SMC-2000, RevA Printed In U.S.A.

Overview

Introduction

The SMC-2000 Series are packaged motion controllers designed for stand-alone operation. Features include coordinated motion profiling, uncommitted inputs and outputs, non-volatile memory for stand-alone operation and RS232/RS422 communication. Extended performance capability includes: fast 8 MHz encoder input frequency, precise 16-bit motor command output DAC, +/-2 billion counts total travel per move, faster sample rate, and multitasking of up to four programs. The controllers provide increased performance and flexibility featuring plug and play operation.

Designed for maximum system flexibility, the SMC-2000 is available in one, two, four, and eight axes configurations.

Each axis accepts feedback from a quadrature linear or rotary encoder with input frequencies up to 8 million quadrature counts per second. For dual-loop applications that require one encoder on both the motor and the load, auxiliary encoder inputs are included for each axis.

The powerful controller provides several modes of motion including jogging, point-to-point positioning, linear and circular interpolation with infinite vector feed, electronic gearing and user-defined path following. Several motion parameters can be specified including acceleration and deceleration rates, and slew speed. To eliminate jerk, the SMC-2000 provides S-curve profiling.

For synchronizing motion with external events, the SMC-2000 includes 8 optically isolated inputs, eight programmable outputs and seven analog inputs (eight optional). I/O expansion boards provide additional inputs and outputs. Event triggers can automatically check for elapsed time, distance and motion complete.

Despite its full range of sophisticated features, the SMC-2000 is easy to program. Instructions are represented by two letter commands such as BG for Begin and SP for Speed. Conditional Instructions, Jump Statements, and arithmetic functions are included for writing self-contained applications programs.

To prevent system damage during machine operation, the SMC-2000 provides several error handling features. These include software and hardware limits, automatic shut-off on excessive error, abort input, and user-definable error and limit routines.

SMC-2000 Functional Elements

The SMC-2000 circuitry can be divided into the following functional groups as shown in Figure 1.1 and discussed below.

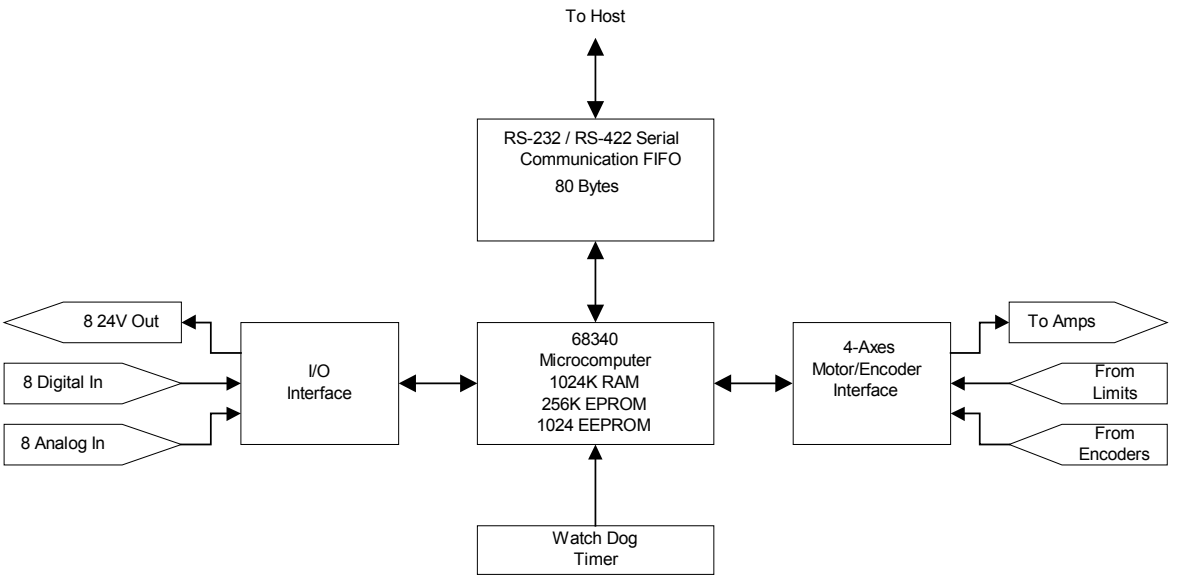


Figure 1.1 - SMC-2000 Functional Elements

Microcomputer Section

The main processing unit of the SMC-2000 is a specialized 32-bit Motorola 68340 Series Microcomputer with 256K RAM, 64 K EPROM and 128 K bytes EEPROM. The RAM provides memory for variables, array elements, and application programs. The EPROM stores the firmware of the SMC-2000. The EEPROM allows parameters and programs to be saved in non-volatile memory upon power down.

Motor Interface

For each axis, a sub-micron gate array performs quadrature decoding of the encoders at up to 8 MHz, generates the +/-10 Volt analog signal (16-Bit D-to-A) for input to a servo amplifier. Interface to hardware limits and home inputs is also included.

Communication

Communication to the SMC-2000 is via two separately addressable RS232 ports. The factory may also configure the ports for RS422. The serial ports may be daisy-chained to other SMC-2000 controllers.

General I/O

The SMC-2000 provides interface circuitry for eight opto-isolated inputs, eight general outputs, and seven (or eight) analog inputs (14-Bit ADC). The eight axis SMC-2000 provides 24 inputs and 16 outputs. Additional I/O is optional.

System Elements

As shown in Fig. 1.2, the SMC-2000 is part of a motion control system that includes amplifiers, motors, and encoders. These elements are described below

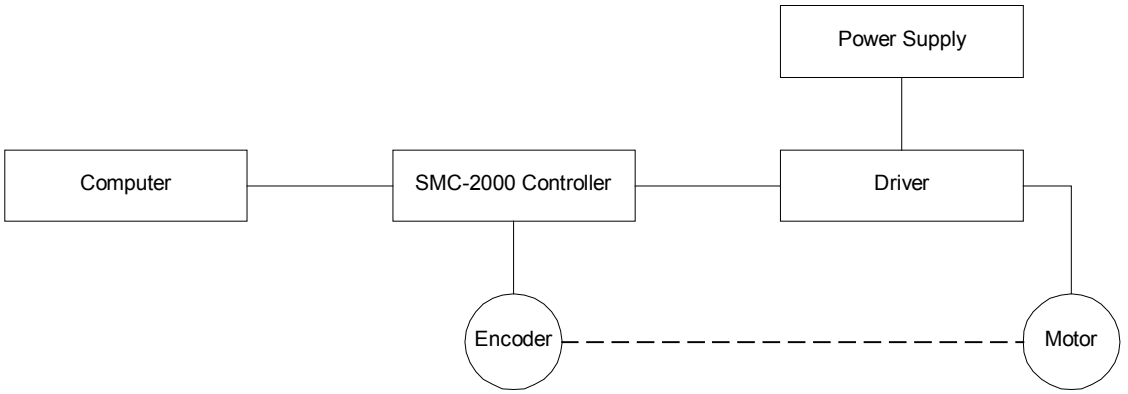


Figure 1.2 - Elements of Servo systems

Motor

A motor converts current into torque, which produces motion. Each axis of motion requires a motor sized properly to move the load at the required speed and acceleration. (Yaskawa's "YSize" software can help you with motor sizing).

The servo motor and can be brush-type or brushless, rotary or linear. Please refer to Yaskawa catalogs for more information.

Amplifier

For each axis, the power amplifier converts the +/-10 Volt signal from the controller into enough current to drive the motor. As such, the amplifier should be sized properly to meet the power requirements of the motor. For brushless motors, an amplifier that provides electronic commutation is required. The amplifier should be set up to operate in a torque control mode. Set the torque reference gain so that 10 Volts at the torque reference input will allow the amplifier/motor to operate at peak torque (typically 200-300% of rated torque). See Yaskawa technical manuals for specifications. Please call Yaskawa if you need help configuring your amplifier.

Encoder

An encoder translates motion into an electrical signal to be fed back into the controller. The SMC-2000 accepts feedback from either a rotary or linear encoder. The preferred encoder is the one with two channels in quadrature, CHA and CHB. This encoder may also have a third channel (or index) for synchronization. When necessary, the SMC-2000 can interface to encoders with pulse and direction signals.

There is no limit on encoder line density, however, the input frequency to the controller must not exceed 2,000,000 full encoder cycles/second (8,000,000 quadrature counts/sec). For example, if the encoder line density is 10000 cycles per inch, the maximum speed is 200 inches/second.

The encoder type may be either single-ended (CHA and CHB) or differential (CHA,CHA-,CHB,CHB-). The SMC-2000 decodes either type into quadrature states or four times the number of cycles.

The standard voltage level is TTL (zero to five volts), however, voltage levels up to 12 Volts are acceptable. (If using differential signals, 12 Volts can be input directly to the SMC-2000. Single-ended 12 Volt signals require a bias voltage input to the complementary inputs)

Getting Started

Elements You Need

Before you start, you must get all the necessary system elements. These include:

1. SMC-2000 Series Controller
2. Servo motors and amplifiers
3. 24 Volt Class 2 Power Supply for SMC-2000 and Amplifiers
4. PC (Personal Computer with RS232 port) with at least 4MB of RAM and Windows 3.1 or higher.
5. Communication Disk (YTerm-2000 software) from Yaskawa
6. All interface and communication cables

Warning: Follow the "Tuning Servo System" procedure before applying power to the SMC unit and the servo amplifier at the same time. Applying power to the SMC unit and the amplifiers at the same time may result in damage to the mechanical system if the initial gain parameters for the SMC unit are not properly set.

Installing the SMC-2000

Connecting AC and DC Power to the Controller

The SMC-2000 requires a single AC supply voltage, single phase, 50 Hz or 60 Hz, from 85 volts to 264 volts, and a +24 (±10%) Volt Class 2 DC supply for I/O. It is also recommended that AC and DC wiring is kept separate in order to avoid noise and interference.

Warning: Do NOT use the I/O 24 VDC power supply to power any holding brakes that may be connected to your servo motors. Use a dedicated supply for that purpose.

Warning: Dangerous voltages, current, temperatures and energy levels exist in this product and in its associated amplifier(s) and servo motor(s). Extreme caution should be exercised in the application of this equipment. Only qualified individuals should attempt to install, set up and operate this equipment.

The AC and DC power is applied to the power connector at the bottom of the front panel. The power connector is a 6-pin black screw-type terminal. Note that the AC power is applied to the LEFT side while the DC power is applied to the RIGHT. The five connections are:

Pin	Connect to:
GND	Earth Ground
N & L	AC In, 85V - 264V
0 & 24V	24 Volt DC and Common

Warning: Never open the controller box when AC power is applied to it.

Applying AC power will turn on the green light power indicator.

Connecting Servo Motors and the Amplifiers

Before connecting the amplifier to the controller, you need to verify that the ground level of the amplifier is either floating or at the same potential as earth.

WARNING: When the amplifier ground is not isolated from the power line or when it has a different potential than that of the computer ground serious damage will result to the computer controller and amplifier.

If you are not sure about the potential of the ground levels, connect the two ground signals by a 10 KΩ resistor and measure the voltage across the resistor. Only if the voltage is zero, proceed to connect the two ground signals directly.

Establishing Communication - RS232

Use the 9-Pin RS232 cable to connect the MAIN (Com 1) SMC-2000 serial port to your computer Com port. Your computer must be configured for a baud rate setting of 19.2 KB, full duplex, no parity, 8 bits data, one start bit, and one stop bit. The Yaskawa software "YTerm-2000" will accomplish this configuration.

At this point you should install YTerm-2000 software. This software requires the use of Windows 3.1 or above, and at least 4M of RAM. The YTerm-2000 communication disk from Yaskawa provides a terminal emulator / configuration program for your computer. Follow the steps below to install and run the terminal emulator.

Installation:

1. Insert Disk in drive A: (or B)
2. From Windows Program Manager or Start Menu, select <Run> command.
3. Run: A:\Setup (or B:\Setup)
4. After the Yaskawa group is created, make sure the SMC-2000 has AC power connected to it then double-click the YTerm-2000 icon to start the program.

Encoder Interface

Encoder interface is part of the Yaskawa supplied cable that connects the SMC with the Yaskawa amplifier.

See the pinout for connector AE1 or AE2 for Auxiliary Encoder interface connection, found in the appendix.

Tuning Servo System

Step 1. Setting servo(s) parameters

Yaskawa servo amplifier models SGD, SGDA, SGDB need to be set up to operate in a Torque Mode.

Parameter (SGD, SGDA)	Function	Setting
Cn-01, bit B , A	Torque Control Mode Selection	1,0
Cn-13	Torque Reference Gain	30
Cn-01, bit 2,3	Limit Switch Disable	1,1

Parameter (SGDB)	Function	Setting
Cn-2B	Torque Control Mode Selection	2
Cn-13	Torque Reference Gain	30
Cn-01, bit 2,3	Limit Switch Disable	1,1

NOTE: When using a motor with an absolute encoder please see the Absolute Encoder section in chapter 12 for additional parameter settings.

Step 2. Applying Power to SMC unit and servos

Apply power to SMC-2000. Input the command MO (CR), this will shut off control of the SMC to the servo(s).

Apply power to the servo amplifier.

Step 3. Setting Gain values in SMC unit

Set gains to default values:

Command	Function	Default value for SG** servo
KD	Derivative Constant	10
KP	Proportional Constant	1
KI	Integrator	0

Step 4. Enable Servo

In order to properly tune the servo system, enable one servo at a time with the **SH*** command (***=X, Y, Z, W, E, F, G, H**). After enabling a servo, maximize the gains.

Step 5. Maximize Gains

For more damping, you can increase KD (maximum is 4095). Increase gradually and stop after the motor vibrates. A vibration is noticed by the audible sound or by interrogation. If you send the command

TE X (CR) Tell error

a few times, and get varying responses, especially with reversing polarity, it indicates system vibration. When that is the case, simply reduce KD.

Next you need to increase the value of KP gradually (maximum allowed is 1023). You can monitor the improvement in the response with the Tell Error instruction.

KP 10 (CR)	Proportion gain
TE X (CR)	Tell error

As the proportional gain is increased, the error decreases.

Here again, the system may vibrate if the gain is too high. When that is the case, reduce KP. Typically, KP should not be greater than KD/4.

Finally, increase the value of KI, start with zero value and increase it gradually. The integrator eliminates the position error, resulting in improved accuracy. Therefore, the response to the instruction

TE X (CR)

becomes zero. As KI is increased, its effect is amplified and it may lead to vibrations. When that occurs, simply reduce KI.

After tuning one axis, disable the servo with the **MO*** command (*=X, Y, Z, W, E, F, G, H), and repeat the tuning process for the remaining axes.

After each servo has been properly tuned, the values now need to be burned into the EEROM. This is done by issuing the **BN** command. After the **BN** command has been issued the new values will remain effective.

Next, you are ready to try a few motion examples.

Motion Examples

Here are a few examples for using your controller.

Example 1- Profiled Move

Objective: Rotate the X-axis a distance of 10,000 counts at a slew speed of 20,000 counts/sec and an acceleration and deceleration rates of 100,000 counts/s^2.

Instruction	Interpretation
PR 10000	Distance
SP 20000	Speed
DC 100000	Deceleration
AC 100000	Acceleration
BG X	Start Motion

In response, the motor turns and stops.

Example 2 - Multiple Axes

Objective: To move four axes independently.

Instruction	Interpretation
PR 500,1000,600,-400	Distances of X,Y,Z,W
SP 10000,12000,20000,10000	Slew speeds of X,Y,Z,W
AC 100000,10000,100000,100000	Accelerations of X,Y,Z,W
DC 80000,40000,30000,50000	Decelerations of X,Y,Z,W
BG XZ	Start X and Z motion
BG YW	Start Y and W motion

Example 3 - Independent Moves

The motion parameters may be specified independently as illustrated below.

Instruction	Interpretation
PR ,300,-600	Distances of Y and Z
SP ,2000	Slew speed of Y
DC,80000	Deceleration of Y
AC,100000	Acceleration of Y
SP ,,40000	Slew speed of Z
AC ,,100000	Acceleration of Z
DC ,,150000	Deceleration of Z
BG Z	Start Z motion
BG Y	Start Y motion

Example 4 - Position Interrogation

The position of all axes may be interrogated with the instruction

TP	Tell position all axes

which returns all of the positions of the motors separated by commas.

Individual axis may be interrogated with the instructions:

TP X	Tell position - X axis only
TP Y	Tell position - Y axis only
TP Z	Tell position - Z axis only
TP W	Tell position - W axis only
TP E	Tell position - E axis only (SMC-2000-8 only)
TP F	Tell position - F axis only (SMC-2000-8 only)
TP G	Tell position - G axis only (SMC-2000-8 only)
TP H	Tell position - H axis only (SMC-2000-8 only)

The position error, which is the difference between the commanded position and the actual position, can be interrogated by the instructions

TE	Tell error - all axes
TE X	Tell error - X axis only
TE Y	Tell error - Y axis only
TE Z	Tell error - Z axis only
TE W	Tell error - W axis only

Example 5- Absolute Position

Objective: Command motion by specifying the absolute position.

Instruction	Interpretation
DP 0,2000	Define the current positions of X,Y as 0 and 2000
PA 7000,4000	Sets the desired absolute positions
BG X	Start X motion
BG Y	Start Y motion

After both motions are completed, command:

PA 0,0	Move to 0,0
BG XY	Start both motions

Example 6 - Velocity Control

Objective: Drive the X and Y motors at specified speeds.

Instruction	Interpretation
JG 10000,-20000	Set Jog Speeds and Directions
AC 100000, 40000	Set accelerations
DC 50000,50000	Set decelerations
BG XY	Start motion

after a few seconds, command:

JG -40000	New X speed and Direction
TV X	Returns X speed

and then

JG ,20000	New Y speed
TV Y	Returns Y speed

These cause velocity changes, including direction reversal. The motion can be stopped with the instruction

ST	Stop

Example 7 - Operation under Torque Limit

The magnitude of the motor command may be limited independently by the instruction TL. The following program illustrates that effect.

Instruction	Interpretation
TL 0.2	Set output limit of X axis to 0.2 volts
JG 10000	Set X speed
BG X	Start X motion

The X motor will probably not move as the output signal is not sufficient to overcome the friction. If the motion starts, it can be stopped easily by the touch of a finger.

Increase the torque level gradually by instructions such as

TL 1.0	Increase torque limit to 1 volt.
TL 9.998	Increase torque limit to maximum, 9.998 Volts.

The maximum level of 10 volts provides the full output torque.

Example 8 - Interrogation

The values of the parameters may be interrogated. For example, the instruction

KP ?	Return gain of X-axis.

returns the value of the proportional gain of the X axis. Similarly, the instruction

KP ,,?	Return gain of Z-axis.

returns the value of the Z axis gain.

KP ?,?,?,?	Return gains of all axes.

returns the gain values for the four axes.

The same procedure applies to other parameters such as KI, KD, FA, etc.

Example 9 - Operation in the Buffer Mode

The instructions may be buffered before execution as shown below.

PR 600000	Distance
SP 10000	Speed
WT 10000	Wait 10000 milliseconds before reading the next instruction
BG X	Start the motion

Example 10 - Motion Programs

Motion programs may be edited and stored in memory using Yaskawa's YTerm-2000 software. They may be executed at a later time.

#A	Define label
PR 700	Distance
SP 2000	Speed
BGX	Start X motion
EN	End program

Now the program may be executed with the command

XQ #A	Start the program running

Example 11 - Motion Programs with Loops

Motion programs may include conditional jumps as shown below.

Instruction	Interpretation
#A	Label
DP 0	Define current position as zero
V1=1000	Set initial value of V1
#LOOP	Label for loop
PA V1	Move X motor V1 counts
BG X	Start X motion
AM X	After X motion is complete
WT 500	Wait 500 ms
TP X	Tell position X
V1=V1+1000	Increase the value of V1
JP #LOOP,V1<10001	Repeat if V1<10001
EN	End

After the above program is entered and downloaded to the SMC-2000, use the following command to run the program:

XQ #A	Execute Program #A

Example 12 - Motion Programs with Trippoints

The motion programs may include trippoints as shown below.

Instruction	Interpretation
#B	Label
DP 0,0	Define initial positions
PR 30000,60000	Set targets
SP 5000,5000	Set speeds

BGX	Start X motion
AD 4000	Wait until X moved 4000
BGY	Start Y motion
AP 6000	Wait until position X=6000
SP 2000,50000	Change speeds
AP ,50000	Wait until position Y=50000
SP ,10000	Change speed of Y
EN	End program

To start the program, command:

XQ #B	Execute Program #B

Example 13 - Control Variables

Objective: To show how control variables may be utilized.

Instruction	Interpretation
#A;DP0	Label; Define current position as zero
PR 4000	Initial position
SP 2000	Set speed
BGX	Move X
AMX	Wait until move is complete
WT 500	Wait 500 ms
#B	
V1 = _TPX	Determine distance to zero
PR -V1/2	Command X move 1/2 the distance
BGX	Start X motion
AMX	After X moved
WT 500	Wait 500 ms
V1=	Report the value of V1
JP #C, V1=0	Exit if position=0
JP #B	Repeat otherwise
#C;EN	End

To start the program, command

XQ #A	Execute Program #A

This program moves X to an initial position of 1000 and returns it to zero on increments of half the distance. Note that _TPX is an internal variable that returns the value of the X position. Internal variables may be created by preceding a SMC-2000 instruction with an underscore, _.

Example 14 - Control Variables and Offset

Objective: Illustrate the use of variables in iterative loops and use of multiple instructions on one line.

Instruction	Interpretation
#A;KI0;DP0;V1=8	Set initial values
#B;OF V1;WT 200	Set offset value
V2=_TPX;JP #C,@ABS[V2]<2;V2=	Exit if error small, report position
V1=V1-1;JP #B	Decrease Offset
#C;EN	End

This program starts with a large offset and gradually decreases its value, resulting in decreasing error.

Example 15 - Linear Interpolation

Objective: Move X,Y,Z motors distance of 7000,3000,6000, respectively, along linear trajectory. Namely, motors start and stop together.

Instruction	Interpretation
LM XYZ	Specify linear interpolation axes
LI 7000,3000,6000	Relative distances for linear interpolation
LE	Linear End
VS 6000	Vector speed
VA 20000	Vector acceleration
VD 20000	Vector deceleration
BGS	Start motion

Example 16 - Circular Interpolation

Objective: Move the XY axes in circular mode to form the path shown on Fig. 2.3.

Instruction	Interpretation
VM XY	Select XY axes for circular interpolation
VP -4000,0	Linear segment
CR 2000,270,-180	Circular segment
VP 0,4000	Linear segment
CR 2000,90,-180	Circular segment
VS 1000	Vector speed
VA 50000	Vector acceleration
VD 50000	Vector deceleration
VE	End vector sequence
BGS	Start motion

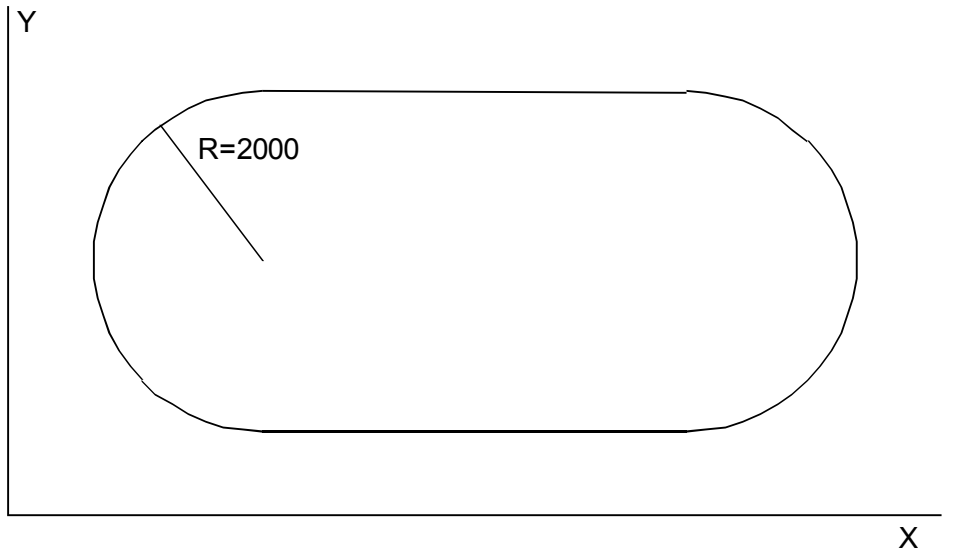


Figure 2-3 - Motion Path for Example 16

Hardware Interface

Cable Shielding, Segregation and Noise Immunity

Yaskawa recommends the following shielding and wiring precautions to maximize the performance of the SMC-2000:

a) Signal cables (encoder, communication, I/O) should be routed away from AC power/signal wiring such as motor power and amplifier power wiring

b) Separate metal conduit should be used for running signal and power wiring from the enclosure

c) Parallel runs of signal and power wiring should be avoided. If unavoidable, parallel runs should be in a separate wire-way spaced at least 2 inches apart.

d) Signal and power wires should cross at right angles.

e) Shielded cables should be properly terminated by grounding the shielding conductor at one end only.

f) The shield should continue throughout the cable from device to device. The shield should be continuous across plugs/receptacles and terminal blocks, or the shields may be grounded separately by grounding one end and tying the shield back at the other (See Fig. 3-1a).

DO NOT ground shields at both ends as this can create ground loops (See Fig. 3-2a).

DO NOT allow the shield to remain ungrounded, this causes the shield to actually pick up and transmit noise.

To improve noise immunity, all inductive loads (Brakes, Relay Coils, etc.) should have a flyback diode connected across them to absorb and back EMF produced by that load. The flyback diode should be placed as close to the load as possible (See Fig. 3.2a).

Proper Shield Terminations

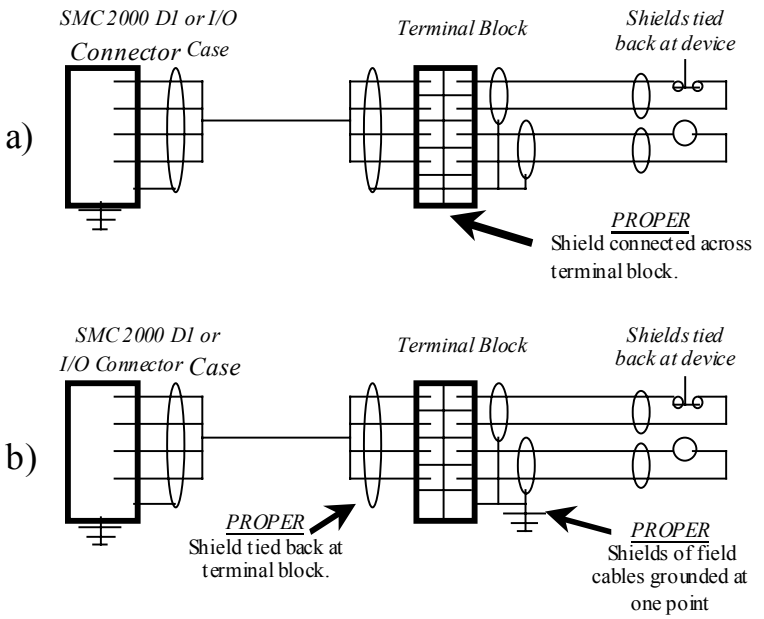


Figure 3-1 – Proper shield terminations

Improper Shield Terminations

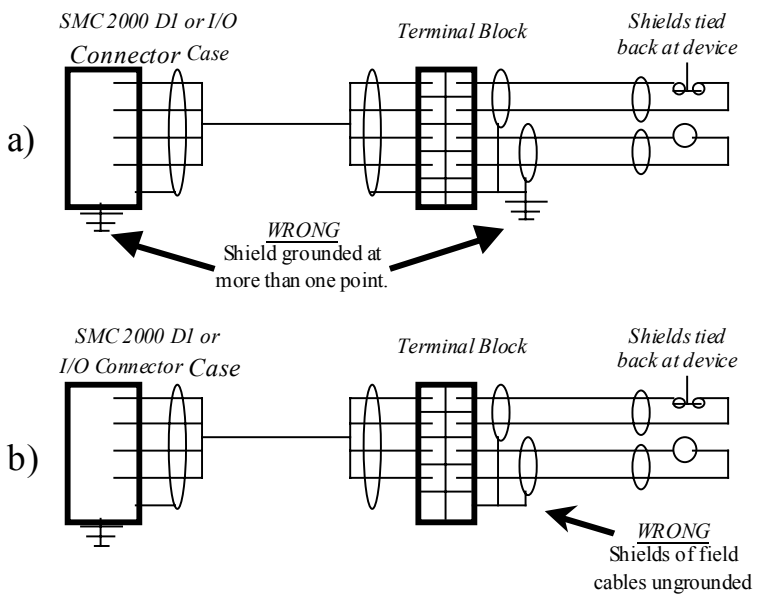


Figure 3-2 – Improper shield terminations

Encoder Interface

For each axis of motion, the SMC-2000 accepts inputs from incremental encoders with two channels in quadrature, or 90 electrical degrees out of phase. The SMC-2000 performs quadrature decoding of the two signals, resulting in bi-directional position information with a resolution of four times the number of full encoder cycles. For example, a 500 cycle encoder is decoded into 2000 quadrature counts per revolution. An optional third channel or index pulse may be used for homing or synchronization. Several types of incremental encoders may be used: linear or rotary, analog or digital, single-ended or differential. Any line resolution may be used, the only limitation being that the encoder input frequency must not exceed 2,000,000 full cycles/sec (or 8,000,000 quadrature counts/sec). The SMC-2000 also accepts inputs from an additional encoder for each axis. These are called auxiliary encoders and can be used for dual-loop applications.

The encoder inputs are not isolated.

Connections for the various types of encoders are described below.

Pin # of X, Y, ...	Signal
1	Channel B
2	Channel B complementary
3	Channel A
4	Channel A complementary
5	Index
6	Index complementary

Use the above table to connect the signals as needed. For example, when connecting an encoder with Channels A, B single ended, use pins 1 and 3, and ignore 2 and 4-6.

In a similar manner, the auxiliary encoders may be connected by using the pin-out for connector AE1 or AE2 found in the appendix.

The SMC-2000 can interface to incremental encoders of the pulse and direction type, instead of two channels in quadrature. In that case, replace Channel A by the pulse signal, and Channel B by the direction, and use the CE command to configure the SMC-2000 for pulse and direction encoder format. For pulse and direction format, the SMC-2000 provides a resolution of 1X counts per pulse.

Note that while TTL level signals are common, the SMC-2000 encoder inputs accept signals in the range of +/- 12V. If you are using a non-TTL single-ended encoder signal (no complement), to assure proper bias, connect a voltage equal to the average signal to the complementary input. For example, if Channel A varies between 2 and 12V, connect 7 volts to Channel A complement input.

Opto-isolated Inputs

The SMC-2000 provides opto-isolated digital inputs for limit, home, abort, and the uncommitted inputs. All inputs have the same common ground and are sinking inputs.

If nothing is connected to the inputs, no current flows, resulting in a logic one. A logic zero is generated when at least 1 mA of current flows through the input.

The 8-Axis SMC provides 16 isolated inputs and 8 additional TTL inputs.

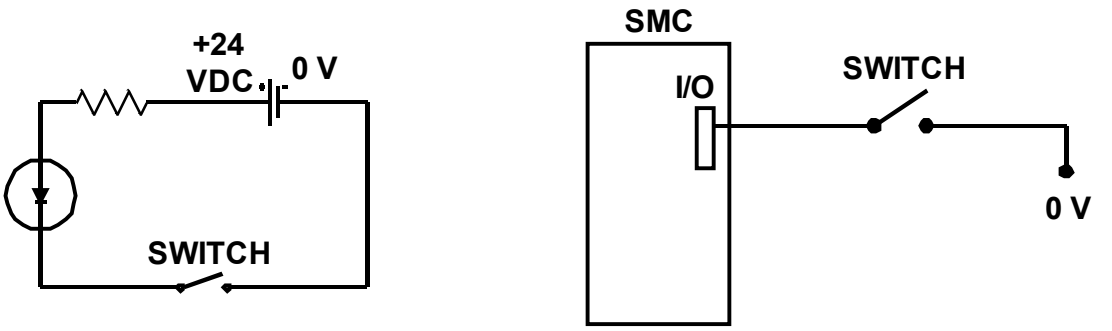


Figure 3-3 - Digital input diagram

Outputs

The SMC-2000 provides several output signals including eight general outputs, and four amplifier enable signals AEN. All the output signals are 24 volts and are sinking outputs. The maximum current draw is 600 mA per point, and a total of 800 mA per group of eight i.e. outputs 1-8, 9-16 ... The 8-Axis SMC provides an additional eight outputs.

WARNING: All inductive loads (Brakes, Relay Coils,...) should have a flyback diode connected across them to absorb any back EMF produced by that load. The flyback diode should be placed as close to the load as possible.

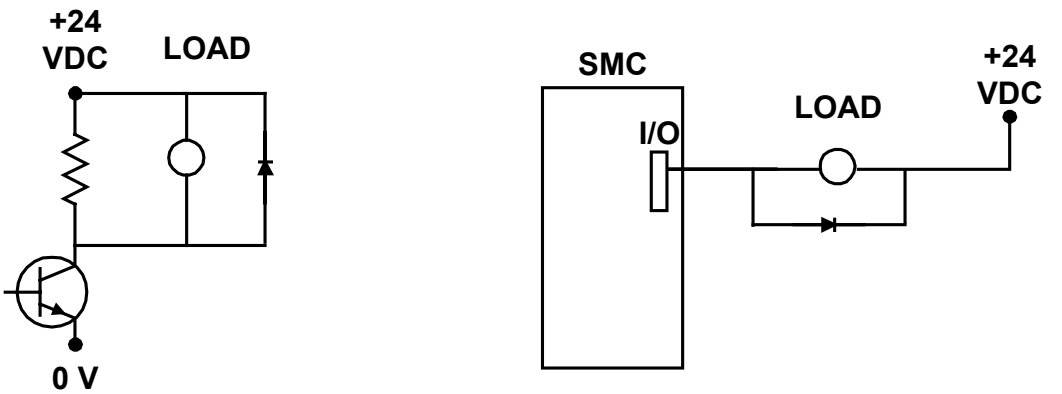


Figure 3-4 - Digital output diagram

Analog Inputs

The SMC-2000 has seven analog inputs configured for the range between -10V and 10V. The inputs are decoded by a 14-bit A/D decoder. The impedance of these inputs is 10 KΩ.

Amplifier Interface

The SMC-2000 generates +/-10 Volt range analog signal, ACMD, and ground for each axis. This signal is input to power amplifiers which have been sized to drive the motors and load. For best performance, the amplifiers should be configured for a current mode of operation with no additional compensation. The gain should be set such that a 10 Volt input results in the maximum required current.

The SMC-2000 also provides an AEN, amplifier enable signal, to control the status of the amplifier. This signal toggles when the watchdog timer activates, when a motor-off command is given, or when OE1 (Off-on-error is enabled) command is given and the position error exceeds the error limit. As shown in Figure 3-3, AEN can be used to disable the amplifier for these conditions.

The standard configuration of the AEN signal is 24 VDC active low.

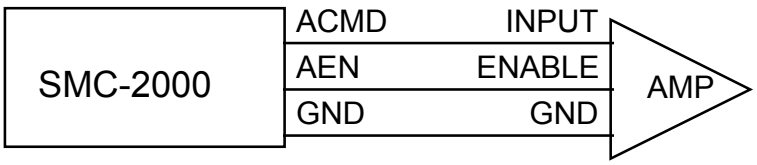


Figure 3-5 - Connecting AEN to an amplifier

Motors with Brakes

A separate 24 VDC power supply should be used to power the brakes because holding brakes typically generate large power spikes when they are de-energized.

Severe damage may result when connecting the same power supply to the SMC and the brake!

Yaskawa recommends that all inductive loads have a diode across them to absorb back EMF.

Step Motors

To connect step motors to the SMC 2000 you must follow this procedure:

1. For each axis that is a stepper, a jumper wire is necessary between ground pin 23 on the AE1 or AE2 connector and the axis stepper mode jumper pin on the AE connector. Newer controllers already have this jumper wire installed during assembly.
2. Connect step and direction signals from the axis connector pins with the labels, STEP (pin 12) and SEN/DIR (pin 13) to respective signals on your step motor amplifier. The signals are 5V TTL level. Consult the documentation for your step motor amplifier.
3. Configure the SMC 2000 for motor type using the MT command. You can configure the SMC 2000 for active high or active low pulses. Use the command MT 2 for active high step motor pulses and MT -2 for active low step motor pulses. See the Commands section of this manual for details.

The pulse output signal has a 50% duty cycle. Step motors operate open loop and do not require encoder feedback. When a stepper is used, the auxiliary encoder for the corresponding axis is unavailable for an external connection. If an encoder is used for position feedback, connect the encoder to the main encoder input corresponding to that axis. The commanded position of the stepper can be interrogated with RP or DE. The encoder position can be interrogated with TP.

The frequency of the step motor pulses can be smoothed with the filter parameter, KS. The KS parameter has a range between 1 and 16, where 16 implies the largest amount of smoothing.

The SMC 2000 profiler commands the step motor amplifier. All SMC 2000 motion commands apply such as PR, PA, VP, CR and JG. The acceleration, acceleration, slew speed, and S-curve filtering are also used. However, since step motors run open loop, the PID filter does not function and the position error is not generated.

When configured for stepper motor operation, the SMC 2000 can accept encoder signals into the main encoder inputs. This is useful for monitoring encoder position to insure that encoder position is consistent with commanded position.

Note: When configured for step motors, the encoder inputs can not be used for closed loop position control and the auxiliary encoder inputs are not available.

Communication - RS232

RS232 Ports

The SMC-2000 has two RS232 ports. The main port can be configured by the factory, and the auxiliary port can be configured with the software command CC. The auxiliary port can either be configured as a general port or for the daisy-chain communications. The auxiliary port configuration can be saved using the Burn (BN) instruction. The RS232 ports also have a clock synchronizing line that allows synchronization of motion on more than one controller.

The RS232 pin-out description for the main and auxiliary port is given below. Note, the auxiliary port is essentially the same as the main port except inputs and outputs are reversed. The SMC-2000 may also be configured by the factory for RS422. These pin-outs are also listed below.

RS232 - Main Port {COM 1}

1 CTS (-) output	6 CTS (-) output
2 Transmit Data (-) output	7 RTS (-) input
3 Receive Data (-) input	8 CTS (-) output
4 RTS (-) input	9 No connect - or - (5V or sample clock with jumpers)
5 Ground	

RS232 - Auxiliary Port {COM 2}

1 CTS (-) input	6 CTS (-) input
2 Transmit Data (-) input	7 RTS (-) output
3 Receive Data (-) output	8 CTS (-) input
4 RTS (-) output	9 5V - or - (no connect or sample clock with jumpers)
5 Ground	

*RS422 - Main Port {COM 1}

1 CTS (-) output	6 CTS (+) output
2 Transmit Data (-) output	7 Transmit Data (+) output
3 Receive Data (-) input	8 Receive Data (+) input
4 RTS (-) input	9 RTS (+) input
5 Ground	

*RS422 - Auxiliary Port {COM 2}

1 CTS (-) input	6 CTS (+) input
2 Transmit Data (-) input	7 Transmit Data (+) input
3 Receive Data (-) output	8 Receive Data (+) output
4 RTS (-) output	9 RTS (+) output
5 Ground	

*Configured for RS422 by factory

Configuration

Configure your PC for 8-bit data, one start-bit, one stop-bit, full duplex and no parity. The baud rate for the RS232 communication is 19.2 K baud. A lower baud rate may be configured at the factory.

The RS232 main port is configured for handshake mode. In this mode, the RTS and CTS lines are used. The CTS line will go high whenever the SMC-2000 is not ready to receive additional characters. The RTS line will inhibit the SMC-2000 from sending additional characters. Note the RTS line goes high for inhibit.

The auxiliary port of the SMC-2000 can be configured either as a general port or for the daisy chain. When configured as a general port, the port can be commanded to send ASCII messages to another SMC-2000 controller or to a display terminal or panel.

(Configure Communication) at port 2. The command is in the format of:

CC m,n,r,p

where m sets the baud rate, n sets for either handshake or non-handshake mode, r sets for general port or the auxiliary port, and p turns echo on or off.

m - Baud Rate - 300,1200,4800,9600,19200,38400

n - Handshake - 0=No; 1=Yes

r - Mode - 0=General Port; 1=Daisy-chain

p - Echo - 0=Off; 1=On; Valid only if r=0

Note, for the handshake of the auxiliary port, the roles for the RTS and CTS lines are reversed.

Example:

CC 1200,0,0,1	Configure communication at port 2, with 1200 baud, no handshake, general port and echo turned on.

Daisy-Chaining

Up to eight SMC-2000 controllers may be connected in a daisy chain. The daisy-chain connection is straightforward. One SMC-2000 is connected to the host terminal via the RS232 at port 1, or the main port. Port 2, or the auxiliary port, of that SMC-2000 is then brought into port 1 of the next SMC-2000, and so on. The default address of the SMC-2000 is zero, if another address is required each of the SMC-2000's must be configured by the factory. Please contact Yaskawa if your application requires daisy-chaining.

To communicate with any one of the SMC-2000s, give the command of %A, where A is the address of the SMC that you want to communicate with. All instructions following this command will be sent only to the SMC with that address. Only when a new %A command is given will the instruction be sent to another SMC. The only exception is "!" command. To talk to all the SMC-2000s in the daisy-chain at one time, insert the character "!" before the software command. All SMCs receive the command, but only address 0 will echo.

Note: The CC command must be specified to configure the port {P2} of each unit.

Example:

Problem: 6-axis motion system. Address 0 is a 4-axis SMC-2000-4. A 2-axis SMC-2000-2 is set for Address 1.	Required Motion:
Address 0	X Axis is 500 counts Y Axis is 1000 counts Z Axis is 2000 counts W Axis is 1500 counts
Address 1	X Axis is 700 counts Y Axis is 1500 counts

Software Command	Interpretation
%0	Talk only to controller 0 (SMC-2000-4)
PR 500,1000,2000,1500	Specify X,Y,Z,W distances
%1	Talk only to controller 1 (SMC-2000-2)
PR 700,1500	Specify X,Y distances
!BG	Begin motion on both controllers

Synchronizing Sample Clocks

It is possible to synchronize the sample clocks of all SMC-2000's in the daisy chain. This involves burning in the command, TM-1, in all SMC-2000's except for one SMC-2000, which will be the source. If it is necessary to synchronize the sample clocks please contact Yaskawa.

Operator Interface

To program an operator interface you need to select a port (either 1 or 2), If you select port 2 it must be configured by using the Configure Communication (CC) command as shown on page 26. You must also decide if the port will be a general port or an operator data entry port. NOTE: configuring a port as an operator data

entry port will disable ALL commands sent to that port, see Operator Data Entry Mode in chapter 7 for a complete description. All serial commands, such as message (MG) or input variable (IN) default to port 1. To assign serial commands to port 2, you must follow the command with a "{P2}" such as:

IN {P2} "Enter a value",VALUE

Which will send out the prompt "Enter a value" to port 2, then wait until a return or semi-colon is sent out port 2, and assign the value of the preceding characters to the variable VALUE.

Controller Response to Data

Most SMC-2000 instructions are represented by two characters followed by the appropriate parameters. Each instruction must be terminated by a carriage return or semicolon.

Instructions are sent in ASCII, and the SMC-2000 decodes each ASCII character (one byte) one at a time. It takes approximately .5 msec for the controller to decode each command.

After the instruction is decoded, the SMC-2000 returns a colon (:) if the instruction was valid or a question mark (?) if the instruction was not valid or was not recognized.

For instructions requiring data, such at Tell Position (TP), the SMC-2000 will return the data followed by a carriage return, line feed and : .

It is good practice to check for : after each command is sent to prevent errors. An echo function is provided to enable associating the SMC-2000 response with the data sent. The echo is enabled by sending the command EO 1 to the controller.

Programming Basics

Introduction

The SMC-2000 provides over 100 commands for specifying motion and machine parameters. Commands are included to initiate action, interrogate status and configure the digital filter.

The SMC-2000 instruction set is BASIC-like and easy to use. Instructions consist of two uppercase letters that correspond phonetically with the appropriate function. For example, the instruction BG begins motion, and ST stops the motion.

Commands can be sent "live" for immediate execution by the SMC-2000, or an entire group of commands can be downloaded into the SMC-2000 memory for execution at a later time. Combining commands into groups for later execution is referred to as Applications Programming and is discussed in the following chapter.

This section describes the SMC-2000 instruction set and syntax. A complete listing of all SMC-2000 instructions is included in the command reference section.

Command Syntax

SMC-2000 instructions are represented by two ASCII upper case characters followed by applicable arguments. A space may be inserted between the instruction and arguments. A semicolon or <enter> is used to terminate the instruction for processing by the SMC-2000 command interpreter.

IMPORTANT: All SMC-2000 commands are sent in upper case.

For example, the command

PR 4000 <enter> Position relative

PR is the two character instruction for position relative. 4000 is the argument which represents the required position value in counts. The <enter> terminates the instruction. The space between PR and 4000 is optional.

For specifying data for the X,Y,Z and W axes, commas are used to separate the axes and preserve axis order as X,Y,Z and W. If no data is specified for an axis, a comma is still needed as shown in the examples below. If no data is specified for an axis, the previous value is maintained. The space between the data and instruction is optional. For the SMC-2000-8, the eight axes are referred to A,B,C,D,E,F,G,H where X,Y,Z,W and A,B,C,D may be used interchangeably.

To view the current values for each command, specify the command followed by a ? for each axis requested. The SMC-2000 provides an alternative method for specifying data.

Here data is specified individually using a single axis specified such as X,Y,Z or W (or A,B,C,D,E,F,G or H for the SMC-2000-8). An equals sign is used to assign data to that axis. For example:

PRZ=1000 Sets the Z axis data as 1000

All axes data may be specified at once using the * symbol. This sets all axes to have the same data. For example:

PR*=1000 Sets all axes to 1000

Example XYZW Syntax for Specifying Data

PR*=1000	Specify data on all axes as 1000
PRY=1000	Specify Y as 1000
PR 1000	Specify X only as 1000
PR ,2000	Specify Y only as 2000
PR ,,3000	Specify Z only as 3000
PR ,,,4000	Specify W only as 4000
PR 2000,4000,6000,8000	Specify X,Y,Z, and W
PR ,8000,,9000	Specify Y and W only
PR*=?	Request X,Y,Z,W values
PR ,?	Request Y value only

For SMC-2000-8 only:

PR,,,,,,8000	Specify G axis data as 8000
PRG=8000	Alternative method for specifying G data

Instead of data, some commands request action to occur on an axis or group of axes. For example, ST XY stops motion on both the X and Y axes. Commas are not required in this case since the particular axis is specified by the appropriate letter X Y Z or W. If no parameters follow the instruction, action will take place on all axes. The letter S is used to specify a coordinated motion sequence. For the SMC-2000-8, the eight axes are commanded with ABCDEFGH or XYZWEFGH where XYZW is used interchangeably with ABCD.

Example XYZW syntax for Requesting Action:

BG X	Begin X only
BG Y	Begin Y only
BG XYZW	Begin all axes
BG YW	Begin Y and W only
BG	Begin all axes
BG S	Begin coordinated sequence
BG SW	Begin coordinated sequence and W axis

For the SMC-2000-8 only:

BG ABCDEFGH	Begin all axes
BG D	Begin D only

Controller Response to Commands

For each valid command entered, the SMC-2000 returns a colon (:). If the SMC-2000 decodes a command as invalid, it returns a question mark (?).

Note:

The SMC-2000 returns a **:** for valid commands.

The SMC-2000 returns a ? for invalid commands.

For example, if the command BG is sent in lower case, the SMC-2000 will return a ?.

:bg <enter>	invalid command, lower case
?	SMC-2000 returns a ?

The command Tell Code, TC1, will return the reason for the ? received for the last invalid command.

:TC1 <enter>	Tell Code command
1 Unrecognized command	Returned response

There are several coded reasons for receiving a ?. Example codes include unrecognized command (such as typographical entry or lower case), a command given at improper time, or a command out of range, such as exceeding maximum speed. A complete listing of all codes is listed in the TC command in the Command Reference section.

For interrogation instructions such as Tell Position (TP) or Tell Status (TS), the SMC-2000 returns the requested data on the next line followed by a carriage return and line feed. The data returned is in decimal format.

Tell Position X	:TP X <enter>
data returned	0000000000
Tell Position X and Y	:TP XY <enter>
data returned	0000000000,0000000000

The format of the returned data can be set using the Position Format (PF) and Variable Format (VF) command.

:PF 4 <enter>	Position Format is 4 integers
:TP X <enter>	Tell Position

0000	returned data

Command Summary

Each SMC-2000 command is described fully in the command reference section at the end of this manual. A summary of the commands follows.

The commands are grouped in this summary by the following functional categories:

- Motion
- Program Flow
- General Configuration
- Control Settings
- Status and Error/Limits.

Motion commands are those to specify modes of motion such as Jog Mode or Linear Interpolation, and to specify motion parameters such as speed, acceleration and deceleration, and distance.

Program flow commands are used in Application Programming to control the program sequencer. They include the jump on condition command and event triggers such as after position and after elapsed time.

General configuration commands are used to set controller configurations such as setting and clearing outputs, formatting variables, and motor/encoder type.

The control setting commands include filter settings such as KP, KD, and KI and sample time.

Error/Limit commands are used to configure software limits and position error limits.

Motion

AB	Abort Motion
AC	Acceleration
BG	Begin Motion
CD	Contour Data
CM	Contour Mode
CR	Circle
CS	Clear Motion Sequence
DC	Deceleration
DT	Contour Time Interval
EA	Select Master CAM axis
EB	Enable CAM mode
EG	Start CAM motion for slaves
EM	Define CAM cycles for each axis
EP	Define CAM table intervals & start point
EQ	Stop CAM motion for slaves
ES	Ellipse Scaling
ET	CAM table entries for slave axes
FE	Find Edge
FI	Find Index
GA	Master Axis for Gearing
GR	Gear Ratio
HM	Home
IP	Increment Position
JG	Jog Mode
LE	Linear Interpolation End
LI	Linear Interpolation Distance
LM	Linear Interpolation mode
LT	Latch Target
PA	Position Absolute
PR	Position Relative
SP	Speed
ST	Stop
TN	Tangent
VA	Vector acceleration
VD	Vector Deceleration
VE	Vector Sequence End
VM	Coordinated Motion Mode
VP	Vector Position
VR	Vector speed ratio

AB	Abort Motion
VS	Vector Speed

Program Flow

AD	After Distance
AI	After Input
AM	After Motion Complete
AP	After Absolute Position
AR	After Relative Distance
AS	At Speed
AT	After Time
AV	After Vector Distance
EN	End Program
HX	Halt Task
IN	Input Variable
II	Input Interrupt
JP	Jump To Program Location
JS	Jump To Subroutine
MC	After motor is in position
MF	After motion -- forward direction
MG	Message
MR	After motion -- reverse direction
NO	No operation
RE	Return from Error Subroutine
RI	Return from Interrupt
TW	Timeout for in position
WC	Wait for Contour Data
WT	Wait
XQ	Execute Program
ZS	Zero Subroutine Stack

General Configuration

AE	Absolute Encoder
AF	Analog Feedback
AL	Arm Latch
BN	Burn
BP	Burn Program

BV	Burn Variables
CB	Clear Bit
CC	Configure Communications Port 2
CE	Configure Encoder Type
CN	Configure Switches and Stepper
DA	De-Allocate Arrays
DE	Define Dual Encoder Position
DL	Download
DM	Dimension Arrays
DP	Define Position
EO	Echo Off
LS	List
MO	Motor Off
MT	Motor Type Define
OB	Output Bit
OP	Output Port
PF	Position Format
QU	Upload Array
QD	Download Array
RA	Record Array
RC	Record
RD	Record Data
RI	Interrupt Mask
RS	Reset
SB	Set Bit
UL	Upload
VF	Variable Format

Control Filter Settings

DV	Damping for dual loop
FA	Acceleration Feed Forward
FV	Velocity Feed Forward
GN	Gain
IL	Integrator Limit
IT	Smoothing Time Constant - Independent
KD	Derivative Constant
KI	Integrator Constant
KP	Proportional Constant
OF	Offset
SH	Servo Here
TL	Torque Limit
TM	Sample Time
VT	Smoothing Time Constant - Vector
ZR	Zero

Status

QY	Query Yaskawa Encoder
RP	Report Command Position
RL	Report Latch
SC	Stop Code
TB	Tell Status
TC	Tell Error Code
TD	Tell Dual Encoder
TE	Tell Error
TI	Tell Input
TP	Tell Position
TR	Trace
TS	Tell Switches
TT	Tell Torque
TV	Tell Velocity
TY	Tell Yaskawa Encoder

Error And Limits

BL	Reverse Software Limit
ER	Error Limit

FL	Forward Software Limit
OE	Off on Error

Arithmetic Functions

@SIN	Sine
@COS	Cosine
@ABS	Absolute value
@ASIN	Arc Sine
@ACOS	Arc Cosine
@FRAC	Fraction portion
@INT	Integer portion
@RND	Round
@SQR	Square root
@COM	Return 2's Complement
@IN	Return digital input
@AN	Return analog input
+	Add
-	Subtract
*	Multiply
/	Divide
&	And
|	Or
( )	Parentheses

Instruction Set Examples

Below are some examples of simple instructions. It is assumed your system is hooked-up and the motors are under stable servo control. Note, the colon (:) is returned by the controller and appears on the screen. You do not need to type the :.

:DP*=0 <enter>	Define all axis positions as 0
:PF 6,6,6,6 <enter>	Define position format as 6 digits
:PR 100,200,300,400 <enter>	Specify X,Y,Z,W position command
:BG <enter>	Begin Motion
:TP <enter>	Tell Position
00100,00200,00300,00400	Returned Position data
:PR?,?,?,? <enter>	Request Position Command
00100,00200,00300,00400	Returned data
:BGX <enter>	Begin X axis only
:TPX <enter>	Tell X position only
00200	Returned position data
:tpx <enter>	Enter invalid command
? TC1 <enter>	Controller response - Request error code
1 Unrecognized command	Controller response
:VM XY <enter>	Specify Vector Mode for XY
:VS 10000 <enter>	Specify Vector Speed
:VP 2000,3000 <enter>	Specify Vector Segment
:VP 4000,5000 <enter>	Specify Vector
:LE <enter>	Segment End Vector
:BGS <enter>	Begin Coordinated Sequence
:TPXY <enter>	Tell X and Y position
004200,005200	Returned data

Command Interrogation List

Command	Firmware	Definition	units	min	max	default
_AB	2.0g	Status of abort input	status	0=Aborted	1=OK	n/a
_ACx	all	Axis acceleration rate	counts/sec^2	1024	67107840	256000
_AEx	D150n19h & up	The last absolute encoder axis that was read	axis 0,1,2 = X,Y,Z	0	7	n/a
_AFx	all	Analog or digital feedback?	status	0=DIGITAL	1=ANALOG	0
_ALx	all	High speed position capture status	status	0=TRIPPED	1=NOT YET	0
_AV	all	Distance from the start of vector sequence	counts	0	2147483647	0
_BGx	all	Is axis in motion?	status	0=NO	1=YES	n/a
_BLx	all	Reverse software limit	counts	-2147483648	2147483647	-2147483648
_BN	all	Serial number of the SMC2000	n/a			n/a
_BV	all	Size of the EEPROM	bytes	1 megabyte	4 megabyte	n/a
_CEx	all	Type of encoder selected	configuration	0	15	0
_CM	all	Is the contour mode buffer full?	status	0=NO	1=YES	0
_CS	all	Current segment number for Vector Mode	n/a	0	511	n/a
_CW	all	Port #1 data adjustment (MG from program, characters have bit 8 set)	status	1=SET	2=OFF	2
_DA	all	Number of available arrays	n/a	0	30	30
_DCx	all	Axis deceleration rate	counts/sec^2	1024	67107840	256000
_DEx	all	Encoder position of the auxiliary encoder	counts	-2147483648	2147483647	n/a
_DL	all	Number of available labels	n/a	0	254	254
_DM	all	Number of available array locations	n/a	0	8000	8000
_DPx	all	Current encoder position of axis	counts	-2147483648	2147483647	n/a
_DT	all	Time interval for contour mode	2^N mSec	0	8	0
_DVx	all	Is the axis using dual loop PID?	status	0=NO	1=YES	0
_EB	all	Is CAM mode enabled?	status	0=NO	1=YES	0
_ED	all	The last line that caused a CMDERR	line number	0	999	n/a
_EGx	all	Is CAMMING axis engaged?	status	0=NO	1=YES	0
_EMx	all	Cam cycle for camming (master or slave)	counts	0	2147483647	0
_EO	all	Is echo mode on?	status	0=NO	1=YES	1
_EP	all	CAMMING interval (resolution)	counts	1	32767	256
_EQx	all	Status of ECAM slave	status	0	3	0
_ERx	all	Axis following error limit (2.0g firmware for Erx=0 to disable)	counts	0	32767	16384

_ES	all	Ellipse scale ratio	n/a	0.0001	1	1
_FAx	all	Axis acceleration feed forward	constant	0	8191	0
_FLx	all	Forward software limit	counts	-2147483648	2147483647	2147483647
_FVx	all	Axis velocity feed forward	constant	0	8191	0
_GRx	all	Gear ratio of the axis	constant	-127.9999	127.9999	0
_HMx	all	State of the home switch	status	0=ACTIVE	1=INACTIVE	n/a
_HXx	all	Thread info	0=NOT RUNNING	1=RUNNING	2=AT TRIPPOINT	n/a
ID	D150n19h & up	The part number of an SMC-2000				n/a
_ILx	all	Integrator limit of the axis	voltage	-9.9988	9.9988	9.9988
_IPX	all	Current encoder position of axis	counts	-2147483648	2147483647	n/a
_ITx	all	S curve smoothing function value	constant	0.004	1	1
_JGx	all	Jog speed for that axis	counts/sec	0	8000000	25000
_KDx	all	Derivative Constant for PID loop	constant	0	4095.875	64 / 10
_KIx	all	Integrator for PID loop	constant	0	2047.875	0 / 0
_KPx	all	Proportional Constant for PID loop	constant	0	1023.875	6 / 1
_LE	all	Length of the vector	counts	0	2147483647	0
_LFx	all	Forward Limit Switch	status	0=ACTIVE	1=INACTIVE	n/a
_LM	all	Number of free locations in linear mode buffer	n/a	0	511	n/a
_LRx	all	Reverse Limit Switch	status	0=ACTIVE	1=INACTIVE	n/a
_LS	all	Next line that will be executed after current subroutine ends	line number	0	999	n/a
_LTx	D150n19j & up	Distance until stop after a registration mark	counts	1	2147483647	??
_LZ	v2.0 & up	Serial port leading zero removal	status	0=OFF	1=ON	0
_MOx	all	Current state of motor, enabled or not	status	0=ENABLED	1=DISABLED	0 / 1
_MTx	all	Type of motor	configuration	-2.5	2.5	1
_OEx	all	Indicates if servo enable signal will shut off if "_ERX" is exceeded	status	0=NO	1=YES	0
_OFx	all	Axis command offset	voltage	-9.9988	9.9988	0
_OPx	all	Entire byte or word of output port (x = output bank 0-3)	byte or word	0	65535	0
PxCD	all	Status code of serial port (x = 1 or 2)	status	-1	3	n/a
PxCH	all	The last character received from serial port (x = 1 or 2)	character	0	255	n/a
PxNM	all	The last number received from serial port (x=1 or 2)	number	-2147483648	2147483647	n/a
PxST	all	The last string received from serial port (x = 1 or	string		6 chars max	n/a

		2)				
_PAx	all	Last commanded absolute position if moving, otherwise current position	counts	-2147483648	2147483647	0
_PF	all	Encoder position format	digits before & after	-8.4	10.4	10.4
_PRx	all	Current incremental distance to move (Even if move set by PA)	counts	-2147483648	2147483647	0
QY	D150n19h & up	The last ASCII string received from an absolute encoder	string		6 chars max	n/a
_RC	all	Status of record mode	status	0= NOT RECORDING	1=RECORDING	0
_RD	all	Array index that record mode will use next	index	0	7999	0
_RLx	all	Encoder value of last latched position	counts	-2147483648	2147483647	0
_RPx	all	Current commanded position of the motor	counts	-2147483648	2147483647	0
_SCx	all	The Stop Code of the axis	code	0	150	1
_SPx	all	Speed parameter of the axis	counts/sec	0	8000000	25000
_TB	all	Status information from controller	byte	0	255	1
_TC1	all	Error code and message from controller	number	0	150	0
_TDx	all	Current auxiliary encoder position	counts	-2147483648	2147483647	n/a
_TEx	all	Difference between commanded & actual axis position	counts	-2147483648	2147483647	n/a
_TIx	all	8 inputs as a decimal or hex value (x = input bank 0-7)	byte	0	255	n/a
TIME	all	Counter since SMC2000 powered on	milliseconds	0	2147483647	0
_TLx	all	Torque limit of axis	voltage	0	9.9988	9.9988
_TM	all	Servo update cycle for all axes	µSec	375	20000	1000
_TN	all	Position of first tangent point	counts	-2147483648	2147483647	0
_TPx	all	Current encoder position of axis	counts	-2147483648	2147483647	n/a
_TSx	all	Status of switches for axis	byte	0	255	n/a
_TTx	all	Current output voltage to amplifier	voltage	-9.9988	9.9988	0
_TVx	all	Velocity of axis (averaged over 256 servo cycles)	counts/sec	0	8000000	n/a
_TYx	D150n19h & up	The position of an absolute encoder at time of reading	counts	-2147483648	2147483647	-2147483648
_TWx	all	Time limit that program will wait for axis to get to target position (MCx)	milliseconds	-1	32766	32766
_UL	all	Number of variables available	n/a	0	254	254
_VA	all	acceleration value for vector mode	counts/sec^2	1024	68431360	256000
_VD	all	Deceleration value for vector mode	counts/sec^2	1024	68431360	256000
_VE	all	Length of vector (all moves in coordinated move sequence)	counts	0	2147483647	0

_VF	all	Setting of variable formatting	n/a	0	10.4	10.4
_VM	all	Number of free locations in vector mode buffer	n/a	0	511	511
_VPx	all	Absolute coordinate of the axis in the last segment	counts	-2147483648	2147483647	0
_VR	all	Vector speed ratio	n/a	0	10	1
_VS	all	Vector Speed	counts/sec	2	8000000	25000
_VT	all	S curve smoothing value for vector mode	constant	0.004	1	1
_XQx	all	Current line number being executed (x = thread #)	line number	-1	999	n/a
_ZS	all	Current subroutine depth	number	0	16	n/a

Programming Motion

Overview

The SMC-2000 provides several modes of motion, including independent positioning and jogging of any axis, coordinated motion, and electronic gearing. Each one of these modes is discussed in the following sections. Please note the SMC-2000-2 uses X and Y, the SMC-2000-4 uses X, Y, Z and W.

The SMC-2000-8 uses the axes A, B, C, D, E, F, G, and H. For SMC-2000-8, the axes A, B, C, D can be referred to interchangeably as X,Y,Z,W.

The example applications described below will help guide you to the appropriate mode of motion.

Example Application	Mode of Motion	Commands
Absolute or relative positioning where each axis is independent and follows prescribed velocity profile.	Independent Axis Positioning	PA,PR SP,AC,DC
Velocity control where no final endpoint is prescribed. Motion stops on Stop command.	Independent Jogging	JG AC,DC ST
Motion Path described as incremental position points versus time.	Contour Mode	CM CD DT WC
2,3 or 4 axis coordinated motion where path is described by linear segments.	Linear Interpolation	LM LI,LE VS VA,VD
2-D motion path consisting of arc segments and linear segments, such as engraving or quilting.	Coordinated Motion	VM VP CR VS VA,VD VE

Third axis must remain tangent to 2-D motion path, such as knife cutting.	Coordinated motion with tangent axis specified	VM VP CR VS,VA,VD TN VE
Electronic gearing where slave axes are scaled to master axis which can move in both directions.	Electronic Gearing	GA GR
Master/slave where slave axes must follow a master such as conveyer speed.	Electronic Gearing	GA GR
Moving along arbitrary profiles or mathematically prescribed profiles such as sine or cosine trajectories.	Contour Mode	CM CD DT WC
Teaching or Record and Play Back	Contour Mode with Automatic Array Capture	CM CD DT WC RA RD RC
Backlash Correction	Dual Loop	DE
Motion Smoothing	Applies to all independent modes of motion i.e. PR, PA, JG. Smoothes motion to eliminate vibrations due to jerk (discontinuities in acceleration)	IT

Independent Axis Positioning

In this mode, motion between the specified axes is independent, and each axis follows its own profile. The user specifies the desired absolute (PA) or relative position (PR), slew speed (SP), acceleration ramp (AC), and deceleration ramp (DC), for each axis. On begin (BG), the SMC-2000 profiler generates the corresponding trapezoidal or triangular velocity profile and position trajectory. A new command position along the trajectory is generated every sample period. Motion is complete when the last position command or target position is generated by the SMC-2000 profiler. The actual motor motion may not be complete at this point, however, the next motion command may be specified.

The Begin (BG) command can be issued for all axes either simultaneously or independently. XYZ or W axis specifiers are required to select the axes for motion. No axes specifier implies motion on all the axes. For the SMC-2000-8, ABCDEFGH axes specifiers are used where XYZ and W may be interchanged with ABCD.

The speed (SP) and the acceleration (AC) can be changed at any time during motion, however, the deceleration (DC) and position (PR or PA) cannot be changed until motion is complete. Remember, motion is complete when the profiler is finished, not when the actual motor is in position. The Stop command (ST) can be issued at any time to decelerate the motor to a stop before it reaches its final position.

A new position target (IP) may be specified during motion as long as the additional move is in the same direction. Here, the user specifies the desired position increment, n. The new target is equal to the old target plus the increment, n. Upon receiving the IP command, a revised profile will be generated for motion towards the new end position. The IP command does not require a begin. Note: If the motor is not moving, the IP command is equivalent to the PR and BG command combination.

Independent Axis Command Summary

PR x,y,z,w	Specifies relative distance
PA x,y,z,w	Specifies absolute position
SP x,y,z,w	Specifies slew speed
AC x,y,z,w	Specifies acceleration rate
DC x,y,z,w	Specifies deceleration rate
BG XYZW	Starts motion
ST XYZW	Stops motion before end of move
IP x,y,z,w	Changes position target
AM XYZW	Trippoint for profiler complete
MC XYZW	Trippoint for "in position"

For the SMC-2000-8:

Use a,b,c,d,e,f,g,h to specify axis data above.

Example - Absolute Position

PA 10000,20000	Specify absolute X,Y position
AC 1000000,1000000	Acceleration for X,Y
DC 1000000,1000000	Deceleration for X,Y
SP 50000,30000	Speeds for X,Y
BG XY	Begin motion

Example - Multiple Move Sequence

Required Motion Profiles

X-Axis	500 counts	Position
	10000 count/sec	Speed
	500000 counts/sec^2	Acceleration
Y-Axis	1000 counts	Position
	15000 count/sec	Speed
	500000 counts/sec^2	Acceleration
Z-Axis	100 counts	Position
	5000 counts/sec	Speed
	500000 counts/sec^2	Acceleration

Start X and Y motion at the same time. After 20 msec, start Z motion. If input 1 is high, stop Y motion.

#A	Begin Program
PR 500,1000,100	Specify position
SP 10000,15000,5000	Specify speed
AC 500000,500000,500000	Specify acceleration
DC 500000,500000,500000	Specify deceleration
BG XY	Begin X and Y
WT 20;BG Z	Wait 20 msec and begin
JP #B,@IN[1]=0	Jump if input 1 is low
STY	Stop Y
#B;EN	End Program

Fig. 6.1 shows the velocity profiles for the X,Y and Z axis.

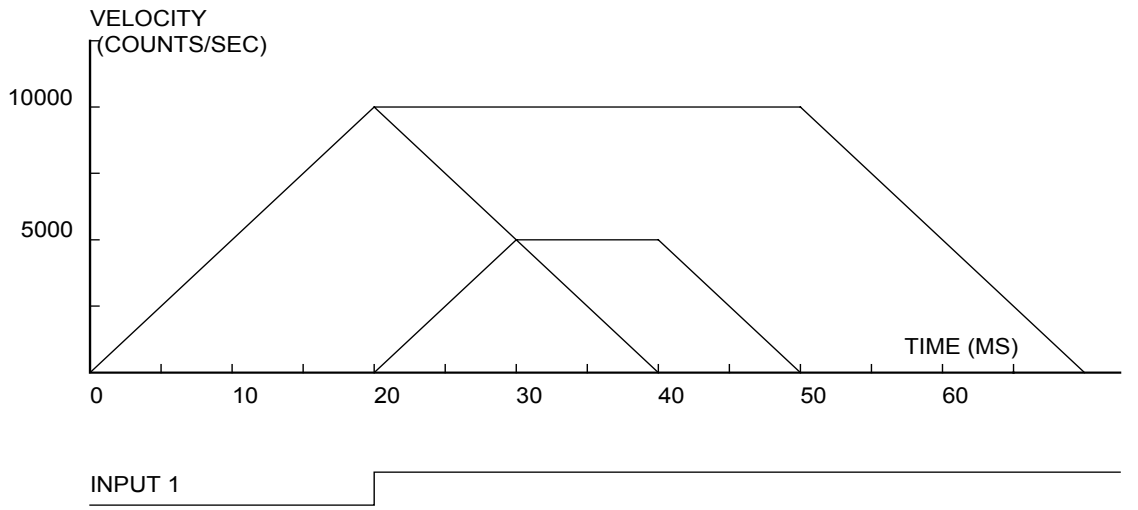


Figure 6.1 - Velocity Profiles of XYZ

Independent Jogging

In this mode, the user specifies the jog speed (JG), acceleration (AC), and the deceleration (DC) rate for each axis. On begin (BG), the motor accelerates up to speed and continues to jog at that speed until a new speed or stop (ST) command is issued. The direction of motion is specified by the sign of the JG parameters.

The jog mode of motion is very flexible because the speed, direction, and acceleration can be changed during motion. The IP command can also be used to instantly change the motor position. Upon receiving this command, the motor will instantly try to servo to a position, which is equal to the specified increment plus the current position. This command is useful when trying to synchronize the position of two motors while they are moving.

It should be noted that the controller operates as a closed-loop position controller even while in the jog mode. The SMC-2000 converts the velocity profile into a position trajectory where a new position target is generated every sample period. This method of control results in precise speed regulation with phase lock accuracy.

Jogging Command Summary

JG +/-x,y,z,w	Specifies jog speed and direction
AC x,y,z,w	Specifies acceleration rate
DC x,y,z,w	Specifies deceleration rate
BG XYZW	Begins motion
ST XYZW	Stops motion
IP x,y,z,w	Increments position instantly

For the SMC-2000-8:

Use a,b,c,d,e,f,g,h to specify axis data above.

Example - Jog in X only

Jog X motor at 50000 count/s. After X motor is at its jog speed, begin jogging Z in reverse direction at 25000 count/s.

#A	
AC 20000,,20000	Specify X,Z acceleration
DC 20000,,20000	Specify X,Z deceleration
JG 50000,,-25000	Specify X,Z speed and direction
BG X	Begin X motion
AS X	After X at speed
BG Z	Begin Z motion
EN	

Example - Joystick Jogging

The jog speed can also be changed using an analog input such as a joystick. Assume that for a 10 Volt input the speed must be 50000 counts/sec. Therefore, the calibration factor is 50000/8191 since the SMC-2000 uses a 14-bit ADC resulting in 8191 counts for 10 Volts.

#JOY	Label
JG0	Set in Jog Mode
BGX	Begin motion
#B	Label for loop
V1 =@AN[1]	Read analog input
VEL=V1*50000/8191	Compute speed
JG VEL	Change JG speed
JP #B	Loop

Linear Interpolation Mode

The SMC-2000 provides a linear interpolation mode for 2,3 or 4 axes (up to 8 axes for the SMC-2000-8). Here, motion between the axes is coordinated to maintain the prescribed vector speed, acceleration, and deceleration along the specified path. The motion path is described in terms of incremental distances for each axis. Several incremental segments may be given in a continuous move sequence, making the linear interpolation mode ideal for following a piece-wise linear path. There is no limit to the total move length.

The LM XYZW command selects the Linear Interpolation mode and axes for interpolation. For example, LM YZ selects only the Y and Z-axes for linear interpolation. For the SMC-2000-8, use ABCDEFGH axis specifies where XYZW may be used interchangeably with ABCD.

The LM command only needs to be specified once unless the axes for linear interpolation change, or another mode such as VM is given.

The LI x,y,z,w or LI a,b,c,d,e,f,g,h command specifies the incremental move distance for each axis. This means motion is prescribed with respect to the current axis position. Up to 511 incremental move segments may be given prior to the Begin Sequence (BGS) command. Once motion has begun, additional LI segments may be specified.

The clear sequence (CS) command can be used to remove LI segments stored in the buffer prior to the start of the motion. To stop the motion, use the instructions STS or AB. The ST command causes a decelerated stop, while the AB command gives an instantaneous stop and aborts the program, while AB1 aborts the motion only.

The Linear End (LE) command must be used to specify the end of a linear move sequence. This command tells the controller to decelerate to a stop following the last LI command. If an LE command is not given, an Abort AB1 must be used to abort the motion sequence.

It is the responsibility of the user to keep enough LI segments in the SMC-2000 sequence buffer to ensure continuous motion. If the controller receives no additional LI segments and no LE command, the controller will stop motion instantly at the last vector. There will be no controlled deceleration. LM? or _LM returns the available spaces for LI segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 LI segments can be sent. A zero means the buffer is full and no additional segments can be sent. As long as the buffer is not full, LI segments can be sent at the COM port baud rate.

The instruction _CS returns the segment counter. As the segments are processed, _CS increases, starting at zero. This function allows the host computer to determine which segment is being processed.

Additional commands for linear interpolation are VS n, VA n, and VD n for specifying the vector speed, acceleration and deceleration. The AV n command is the After Vector trippoint, which waits for the vector distance of n to occur.

For example, note the following program:

DP 0,0	Define position
LMXY	Specify axes for linear interpolation
LI 5000,0	Specify XY distances
LI 0,5000	Specify XY distances
LE	Specify end move
VS 4000	Specify vector speed
BGS	Begin sequence
AV 4000	After vector distance 4000
VS 1000	Specify vector speed
AV 5000	After vector distance 5000
VS 4000	Specify vector speed

EN	End program

Here the XY system is required to perform 90° turn. In order to slow the speed around the corner, we use the AV 4000 trippoint, which slows the speed to 1000 count/s. Once the motors reach the corner, we can increase the speed, back to 4000 cts/s, with the trippoint AV 5000.

The instruction AV can be used as an operand. _AV returns the distance along the motion sequence.

The instruction _VP returns the absolute coordinate of the last data point along the trajectory. This enables the host to command motion backward in case of tool break.

For example, note the program shown above. Consider the first motion segment, where the X-axis moves toward the point X=5000. Now suppose that when X=3000, the controller is interrogated.

The response to _AV will be 3000. The response to _CS is 0 and the responses to _VPX and _VPY are zeros for both.

Now suppose that the interrogation is repeated at the second segment when Y=2000. The response to _AV at this point is 7000, _CS equals 1, _VPX=5000 and _VPY=0.

It should be noted that the SMC-2000 computes the vector speed based on the axes specified in the LM mode. For example, LM XYZ designates linear interpolation for the X, Y, and Z-axes. The speed of these axes will be computed from $VS^2=XS^2+YS^2+ZS^2$, where XS, YS and ZS are the speed of the X, Y and Z-axes. If the LI command specifies only X and Y, the speed of Z will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed.

Command Summary - Linear Interpolation

LM XYZW	Specify axes for linear interpolation
LM ABCDEFGH	Specify axes for linear interpolation (SMC-2000-8)
LI x,y,z,w LI a,b,c,d,e,f,g,h	Specify incremental distances relative to current position
_LM or LM?	Returns number of available spaces for linear segments in SMC-2000 sequence buffer. Zero means buffer full. 511 means buffer empty.
VS n	Specify vector speed
VA n	Specify vector acceleration
VD n	Specify vector deceleration
BGS	Begin Linear Sequence
CS	Clear sequence
_CS	Segment counter
_VPm	Return coordinate of last point, where m=X,Y,Z or W or A,B,C,D,E,F,G or H
LE	Linear End- Required at end of LI command sequence
_LE or LE?	Returns length of vector (resets after 2147483647)
AMS	Trippoint for After Sequence complete
AV n	Trippoint for After Relative Vector distance ,n
_AV	Return distance traveled

Example - Linear Move

Make a coordinated linear move in the ZW plane. Move to coordinates 40000,30000 counts at a vector speed of 100,000 counts/sec and vector acceleration of 1000000 counts/sec^2.

LM ZW	Specify axes for linear interpolation
LI,,40000,30000	Specify ZW distances
LE	Specify end move
VS 100000	Specify vector speed
VA 1000000	Specify vector acceleration
VD 1000000	Specify vector deceleration
BGS	Begin sequence

Note that the above program specifies the vector speed, VS, and not the actual axis speeds VZ and VW. The axis speeds are determined by the SMC-2000 from:

$$VS = \sqrt{VZ^2 + VW^2}$$

The resulting profile is shown in Figure 6.2.

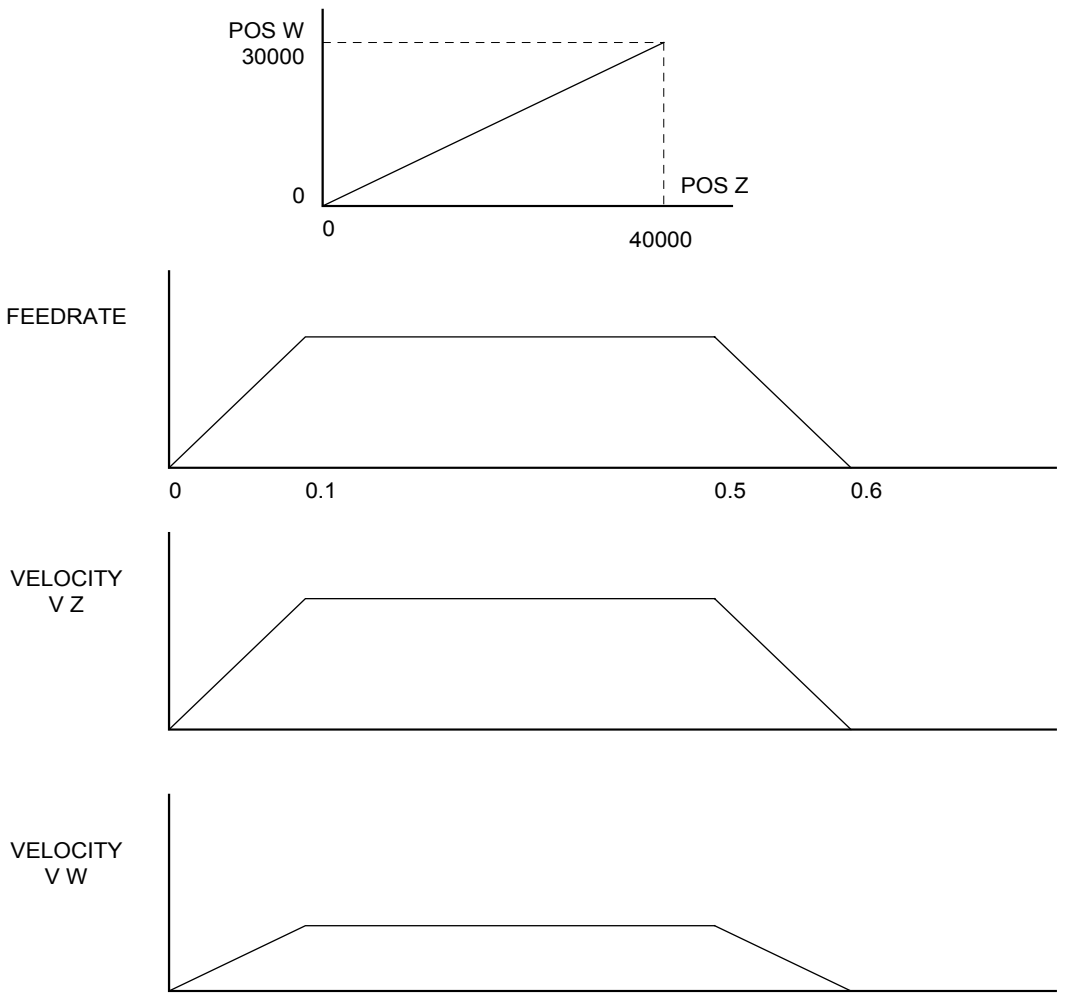


Figure 6.2 - Linear Interpolation

Example - Multiple Moves

Make a coordinated linear move in the XY plane. The Arrays VX and VY are used to store 750 incremental distances that have been filled by the program #LOAD.

#LOAD	Load Program
DM VX [750],VY [750]	Define Array
COUNT=0	Initialize Counter
N=0	Initialize position increment
#LOOP	LOOP
VX [COUNT]=N	Fill Array VX
VY [COUNT]=N	Fill Array VY
N=N+10	Increment position
COUNT=COUNT+1	Increment counter
JP #LOOP,COUNT<750	Loop if array not full
#A	Label
LM XY	Specify linear mode for XY
COUNT=0	Initialize array counter
#LOOP2;JP#LOOP2,_LM=0	If sequence buffer full, wait
JS#C,COUNT=500	Begin motion on 500th segment
LI VX[COUNT],VY[COUNT]	Specify linear segment
COUNT=COUNT+1	Increment array counter
JP #LOOP2,COUNT<750	Repeat until array done
LE	End Linear Move
AMS	After Move sequence done
MG "DONE"	Send Message
EN	End program
#C;BGS;EN	Begin Motion Subroutine

Coordinated Motion Sequences

The SMC-2000 allows a long 2-D path consisting of linear and arc segments to be prescribed. Motion along the path is continuous at the prescribed vector speed even at transitions between linear and circular segments. The SMC-2000 performs all the complex computations of linear and circular interpolation, freeing the host PC from this time intensive task.

The coordinated motion mode is similar to the linear interpolation mode. Here, any pair of two axes may be selected for coordinated motion consisting of linear and circular segments. In addition, a third axis can be controlled such that it remains tangent to the motion of the selected pair of axes.

The VM m,n,p command specifies the axes. m,n are the coordinated pair and p is the tangent. For example, VM X, W, Z selects the XW axes for coordinated motion and the Z-axis as the tangent. Commas are not required.

The motion segments are described by two commands, VP for linear and CR for circular segments. The VP x,y command specifies the end point coordinate of the linear segment, in reference to the starting point. CR r, θ, δ define a circular arc with a radius r, starting angle of θ, and a traversed angle δ. The notation for θ is that zero corresponds to the positive horizontal direction and for both θ and δ, the counter-clockwise (CCW) rotation is positive.

Up to 511 segments of CR or VP may be given prior to the Begin Sequence (BGS) command. Once motion starts, additional segments may be added.

The Clear Sequence (CS) command can be used to remove VP and CR stored in the buffer prior to the start of the motion. To stop the motion, use the instructions STS or AB1. ST stops motion at the specified deceleration. AB1 aborts the motion instantaneously.

The Vector End (VE) command must be used to specify the end of the coordinated motion. This command requires the controller to decelerate to a stop following the last motion requirement. If a VE command is not given, an Abort (AB1) must be used to abort the coordinated motion sequence.

It is the responsibility of the user to keep enough motion segments in the SMC-2000 sequence buffer to ensure continuous motion. If the controller receives no additional motion segments and no VE command, the controller will stop motion instantly at the last vector. There will be no controlled deceleration. LM? or _LM returns the available spaces for motion segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 segments can be sent. A zero means the buffer is full and no additional segments can be sent. As long as the buffer is not full, additional segments can be sent at the COM port baud rate.

The instruction _CS returns the segment counter. It allows the host to determine the motion segment being executed.

Additional commands for coordinated motion are VS n, VA n and VD n for specifying the vector speed, acceleration, and deceleration. The AV n command is the After Vector trippoint, which waits for the vector relative distance of n to occur.

The AV trippoint is useful in changing the parameters, such as the vector speed along the sequence.

When AV is used as an operand, _AV returns the distance traveled along the sequence.

The instruction _VPX and _VPY can be used to return the coordinates of the last point specified along the path.

Example:

Traverse the path shown in Fig. 6.3. Feed rate is 20000 counts/sec. Plane of motion is XY.

VM XY	Specify motion plane
VS 20000	Specify vector speed
VA 1000000	Specify vector acceleration
VD 1000000	Specify vector deceleration
VP -4000,0	Segment AB
CR 1500,270,-180	Segment BC
VP 0,3000	Segment CD
CR 1500,90,-180	Segment DA
VE	End of sequence
BGS	Begin Sequence

The resulting motion starts at the point A and moves toward points B, C, D, A. Suppose that we interrogate the controller when the motion is halfway between the points A and B.

_AV returns 2000

_CS returns 0

_VPX and _VPY return the absolute coordinate of the point A

Next, suppose that the interrogation is repeated at a point, halfway between the points C and D.

_AV returns 4000+1500π+2000=10,712

_CS returns 2

_VPX,_VPY return the coordinates of the point C

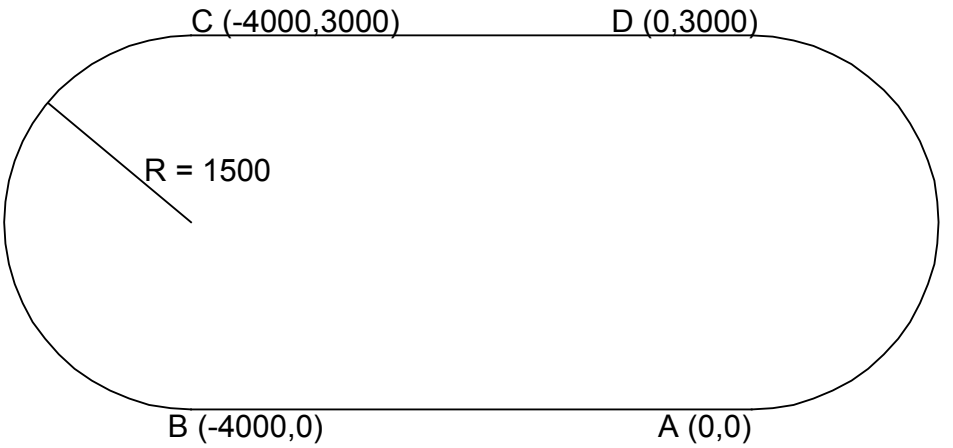


Figure 6.3 - The Required Path

Tangent Motion

Several applications, such as cutting, require a third axis (i.e. a knife blade), to remain tangent to the coordinated motion path. To handle these applications, the SMC-2000 allows one axis to be specified as the tangent axis. The VM command provides parameter specifications for describing the coordinated axes and the tangent axis.

VM m,n,p	m,n specifies coordinated axes p specifies tangent axis such as X,Y,Z,W or A,B,C,D,E,F,G,H. p=N turns off tangent axis

Before the tangent mode can operate, it is necessary to assign an axis via the VM command and define its offset and scale factor via the TN m,n command. m defines the scale factor in counts/degree and n defines the tangent position that equals zero degrees in the coordinated motion plane. The _TN can be used to return the initial position of the tangent axis.

Example:

Assume an XY table with the Z-axis controlling a knife. The Z-axis has a 2000 quad counts/rev encoder and has been initialized after power-up to point the knife in the +Y direction. A 180° circular cut is desired, with a radius of 3000, center at the origin and a starting point at (3000,0). The motion is CCW, ending at (-3000,0).

Note that the 0° position in the XY plane is in the +X direction. This corresponds to the position -500 in the Z-axis, and defines the offset. The motion has two parts. First, X,Y and Z are driven to the starting point, and later, the cut is performed. Assume that the knife is engaged with output bit 1.

#EXAMPLE	Example program
VM XYZ	XY coordinate with Z as tangent
TN 2000/360,-500	2000/360 counts/degree, position -500 is 0 degrees in XY plane
CR 3000,0,180	3000 count radius, start at 0 and go to 180 CCW
VE	End vector
CB1	Disengage knife
PA 3000,0,_TN	Move X and Y to starting position, move Z to initial tangent position
BG XYZ	Start the move to get into position
AM XYZ	When the move is complete
SB1	Engage knife
WT50	Wait 50 msec for the knife to engage
BGS	Do the circular cut
AMS	After the coordinated move is complete
CB1	Disengage knife
MG "ALL DONE"	
EN	End program

Coordinated Motion Sequence Instructions - Summary

VM m,n,p	Specifies plane for the motion sequence such as X,Y or Z,W. p specifies tangent axis.
VP m,n	Return coordinate of last point, where m=X,Y,Z or W.
_VPm	Specifies the end point of a segment in reference to the starting point of the sequence.
CR *r*,Θ, ±ΔΘ	Specifies arc segment where r is the radius, Θ is the starting angle and ΔΘ is the travel angle. Positive direction is CCW.
VS n	Specifies vector speed or feed rate of sequence.
VA n	Specifies vector acceleration along the sequence.
VD n	Specifies vector deceleration along the sequence.
BGS	Begin motion sequence.
AV n	Trippoint for After Relative Vector distance, n.
_AV	Return distance traveled.
AMS	Holds execution of the next command until the Motion Sequence is completed.

_LM or LM?	Return number of available spaces for linear and circular segments in SMC-2000 sequence buffer. Zero means buffer is full. 511 means buffer is empty.
TN m,n	Tangent scale and offset.
ES m,n	Ellipse scale factor.
CS	Clear sequence.
_CS	Segment counter.

Electronic Gearing

This mode allows 1,2 or 3 axes (or 4,5,6,7 axes for the SMC-2000-8) to be electronically geared to one driven master axis, or all axes to be geared to an auxiliary encoder. The master may rotate in both directions and the geared axes will follow at the specified gear ratio. The gear ratio may be different for each axis and changed during motion.

The command GAX or GAY or GAZ or GAW (or GAA or GAB or GAC or GAD or GAE or GAF or GAG or GAH for SMC-2000-8) specifies the master axis. There may only be one master. GR x,y,z,w specifies the gear ratios for the slaves where the ratio may be a number between +/-127.9999 with a fractional resolution of .0001. GR 0,0,0,0 turns off electronic gearing for any set of axes. A limit switch will also disable electronic gearing for that axis. GR causes the specified axes to be geared to the actual position of the master. The master axis is commanded with motion commands such as PR, PA or JG.

An alternative gearing method is to synchronize the slave motor to the commanded vector motion of several axes performed by GAS. For example, if the X and Y motor form a circular motion, the Z axis may move in proportion to the vector move. Similarly, if X,Y and Z perform a linear interpolation move, W can be geared to the vector move.

Electronic gearing allows the geared motor to perform a second independent or coordinated move in addition to the gearing. For example, when a geared motor follows a master at a ratio of 1:1, it may be advanced an additional distance with PR, or JG, commands, or VP, or LI.

Command Summary - Electronic Gearing

GA n	Specifies master axis for gearing where n = X,Y,Z or W or A,B,C,D,E,F,G,H for main encoder as master
	n = XC,YC,ZC or WC or AC, BC, CC, DC, EC, FC,GC,HC for commanded position as master
	n=S vector move for master
GR x,y,z,w	Sets gearing mode and gear ratio for slave axes. 0 disables electronic gearing for specified axis.
GR a,b,c,d,e,f,g,h	Sets gearing mode and gear ratio for slave axes. 0 disables electronic gearing for specified axis.
MR x,y,z,w	Trippoint for motion past assigned point in reverse direction. Only one field may be used.
MF x,y,z,w	Trippoint for motion past assigned point in forward direction. Only one field may be used.

Example - Simple Master Slave

Master axis moves 10000 counts at slew speed of 100000 counts/sec. Y is defined as the master. X,Z,W are geared to master at ratios of 5,-.5 and 10 respectively.

GAY	Specify master axes as Y
GR 5,,-.5,10	Set gear ratios
PR ,10000	Specify Y position
SP ,100000	Specify Y speed
BGY	Begin motion

Example - Electronic Gearing

Run two geared motors at speeds of 1.132 and -0.045 times the speed of an external master. The master motor is driven externally at speeds between 0 and 1800 RPM (2000 counts/rev encoder).

Solution: Use a SMC-2000-4 controller, where the Z-axis is the master and X and Y are the geared axes.

M0 Z	Turn Z off, for external master
GA Z	Specify master axis
GR 1.132,-.045	Specify gear ratios

Now suppose the gear ratio of the X-axis is to change on the fly to 2. This can be achieved by commanding:

GR 2	

In several applications where both the master and the follower are controlled by the SMC-2000 controller, it may be desired to synchronize the follower with the commanded position of the master, rather than the actual position. This eliminates the coupling between the axes that may lead to oscillations.

For example, assume that a gantry is driven by two axes, X,Y, on both sides. The X-axis is the master and the Y-axis is the follower. To synchronize Y with the commanded position of X, use the instructions:

GACX	Specify master as commanded position of X
GR,1	Set gear ratio for Y as 1:1
PR 3000	Command X motion
BG X	Start motion

You may also perform profiled position corrections in the electronic gearing mode. Suppose, for example, that you need to advance the slave 10 counts. Simply command

IP,10	

which is equivalent to PR,10; BGY.

Often the correction is quite large. Such requirements are common on synchronizing cutting knives or conveyor belts.

Example - Synchronize two conveyor belts with trapezoidal velocity correction.

GAX	Define master axis as X
GR,2	Set gear ratio 2:1 for Y
PR,300	Specify correction distance
SP,5000	Specify correction speed
AC,100000	Specify correction acceleration
DC,100000	Specify correction deceleration
BGY	Start correction

Contour Mode

The SMC-2000 also provides a contouring mode. This mode allows any arbitrary position curve for 1,2,3 or 4 (5,6,7 or 8 axes for SMC-2000-8) axes to be prescribed which is ideal for following computer generated paths such as parabolic, spherical or user-defined profiles. Here, the path is not limited to straight line and arc segments. Also, the path length may be infinite.

The Contour Mode (CM) command specifies which axes are to be contoured. Any combination of 1,2,3 or 4 axes (5,6,7 or 8 axes for SMC-2000-8) may be used. For example, CMXZ specifies contouring on the X and Z-axes. Axes non-contouring may be operated in other modes.

The contour is described by position increments, CD x,y,z,w over a time interval, DT n. For the SMC-2000-8, the contour is described by CD a,b,c,d,e,f,g,h.

The time interval must be 2^n ms, where n is a number between 1 and 8. The controller performs linear interpolation between the specified increments, where one point is generated for each millisecond.

Consider, for example, the trajectory shown in Fig. 6.4. The position X may be described by the points.

Point 1	X=0 at T=0ms
Point 2	X=48 at T=4ms
Point 3	X=138 at T=12ms
Point 4	X=302 at T=28ms

The same trajectory may be represented by the increments

Increment 1	DX=48	Time=4	DT=4
Increment 2	DX=90	Time=8	DT=8
Increment 3	DX=164	Time=16	DT=16

When the controller receives the command to generate a trajectory along these points, it interpolates linearly between the points. The resulting interpolated points include the position 12 at 1 msec, position 24 at 2 msec, etc.

The programmed commands to specify the above example are:

#A	
CMX	Specifies X axis for contour mode
DT 2	Specifies first time interval, 2^2

CD 48;WC	Specifies first position increment
DT 3	Specifies second time interval, 2^3
CD 90;WC	Specifies second position increment
DT 4	Specifies the third time interval, 2^4
CD 164;WC	Specifies the third position increment
DT0;CD0	Exits contour mode
EN	

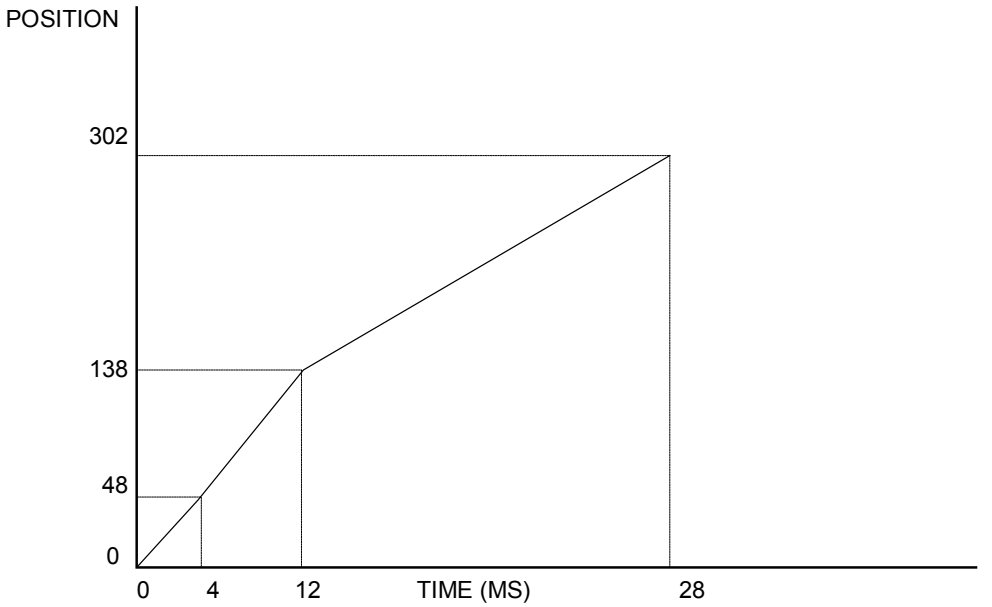


Figure 6.4 - The Required Trajectory

The command, WC, is used as a trippoint "When Complete". This allows the SMC-2000 to use the next increment only when it is finished with the previous one. Zero parameters for DT or CD exit the contour mode.

If no new data record is found and the controller is still in the contour mode, the controller waits for new data. No new motion commands are generated while waiting. If bad data is received, the controller responds with a ?.

The command _CS, the segment counter, returns the number of the segment being processed. This information allows the host computer to determine when to send additional data.

Summary of Commands for Contour Mode:

CM XYZW	Specifies which axes for contouring mode. Any non-contouring axes may be operated in other modes.
CM ABCDEFGH	Contour axes for SMC-2000-8
CD x,y,z,w	Specifies position increment over time interval. Range is +/-32,767. Zero ends contour mode.
CD a,b,c,d,e,f,g,h	Position increment data for SMC-2000-8

DT n	Specifies time interval 2^n msec for position increment, where n is an integer between 1 and 8. Zero ends contour mode. If n does not change, it does not need to be specified with each CD.
WC	Waits for previous time interval to be complete before next data record is processed.
_CS	Return segment number

General Velocity Profiles

The Contour Mode is ideal for generating any arbitrary velocity profiles. The velocity profile can be specified as a mathematical function or as a collection of points.

The design includes two parts: Generating an array with data points and running the program.

Generating an Array

Consider for example the velocity and position profiles shown in Fig. 6.5. The objective is to rotate a motor a distance of 6000 counts in 120 ms. The velocity profile is sinusoidal to reduce the jerk and the system vibration. When the position displacement is A counts in B milliseconds, the general expression for the velocity and position profile, where T is the time in milliseconds, is:

$$\omega = \frac{A}{B}\left(1 - \cos(2\pi/B)\right)$$

$$X = \frac{AT}{B} - \frac{A}{2\pi}\sin(2\pi/B)$$

In the given example, A=6000 and B=120, the position and velocity profiles are:

$$X = 50T - (6000/2\pi) \sin (2\pi\ T/120)$$

Note that the velocity, ω, in count/ms, is

$$\omega = 50\ [1 - \cos 2\pi\ T/120]$$

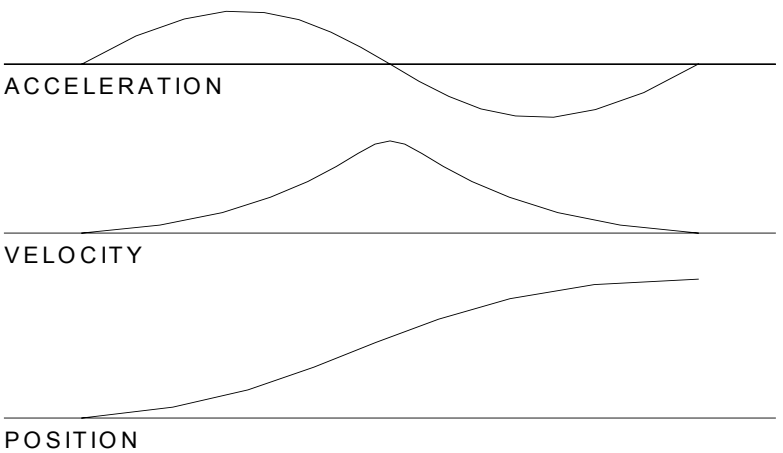


Figure 6.5 - Velocity Profile with Sinusoidal Acceleration

The SMC-2000 can compute trigonometric functions. However, the argument must be expressed in degrees. Accordingly, the equation of X is written as:

X = 50T - 955 sin 3T

To generate an array, we compute the position value at intervals of 8 ms. This is stored at the array POS. Later, the difference between the positions is computed and is stored in the array DIF.

The program for storing the values is given below.

Instruction	Interpretation
#POINTS	Program defines X points
DM POS[16]	Allocate memory
DM DIF[15]	
C=0	Set initial conditions, C is index
T=0	T is time in ms
#A	
V1=50*T	
V2=3*T	Argument in degrees
V3=-955*@SIN[V2]+V1	Compute position
V4=@INT[V3]	Integer value of V3
POS[C]=V4	Store in array POS
T=T+8	
C=C+1	
JP #A,C<16	
#B	Program to find position differences
C=0	
#C	
D=C+1	
DIF[C]=POS[D]-POS[C]	Compute the difference and store
C=C+1	
JP #C,C<15	
EN	End first program
#RUN	Program to run motor
CMX	Contour Mode
DT3	4 millisecond intervals
C=0	
#E	
CD DIF[C]	Contour Distance is in DIF
WC	Wait for completion
C=C+1	
JP #E,C<15	
DT0	
CD0	Stop Contour
EN	End the program

Teach (Record and Play-Back)

Several applications require teaching the machine a motion trajectory. Teaching can be accomplished by using the SMC-2000 automatic array capture feature to capture position data. The captured data may then be played back in the contour mode. The following array commands are used:

DM C[n]	Dimension array
RA C[]	Specify array for automatic record (up to 8)
RD _TPX	Specify data for capturing (such as _TPX or _TPZ)
RC n,m	Specify capture time interval where n is 2^n msec, m is number of records to be captured
RC? or _RC	Returns a 1 if recording

Example:

#RECORD	Begin Program
DM XPOS[501]	Dimension array with 501 elements
RA XPOS[]	Specify automatic record
RD _TPX	Specify X position to be captured
MOX	Turn X motor off
RC2	Begin recording; 4 msec interval
#A;JP#A,_RC=1	Continue until done recording
#COMPUTE	Compute DX
DM DX[500]	Dimension Array for DX
C=0	Initialize counter
#L	Label
D=C+1	
DELTA=XPOS[D]-XPOS[C]	Compute the difference
DX[C]=DELTA	Store difference in array
C=C+1	Increment index
JP #L,C<500	Repeat until done
#PLAYBCK	Begin Playback
CMX	Specify contour mode
DT2	Specify time increment
I=0	Initialize array counter
#B	Loop counter
CD DX[I];WC	Specify contour data
I=I+1	Increment array counter
JP #B,I<500	Loop until done
DT 0;CD0	End contour mode
EN	End program

For additional information about automatic array capture, see Chapter 7, Arrays.

Dual Loop (Auxiliary Encoder)

The SMC-2000 provides an interface for a second encoder per axis. The second encoder may be mounted on the motor, the load or in any position.

The second encoder may be of the standard quadrature type, or it may be of the pulse and direction type. The controller also offers the provision for inverting the direction of the encoder rotation.

The configuration of the auxiliary encoder is done by the CE (Configure Encoder) command. This command configures both the main and the second encoder.

The command form is CE x,y,z,w or a,b,c,d,e,f,g,h for SMC-2000-8 where the parameters x,y,z,w each equals the sum of two integers m and n. m configures the main encoder and n configures the second encoder.

m=	Main Encoder	n=	Second Encoder
0	Normal quadrature	0	Normal quadrature
1	Pulse & direction	4	Pulse & direction
2	Reverse quadrature	8	Reversed quadrature
3	Reverse pulse & direction	12	Reversed pulse & direction

For example, to configure the main encoder for reversed quadrature, m=2, and a second encoder of pulse and direction, n=4, the total is 6, and the command for the X axis is

```
CE 6
```

The DE x,y,z,w command can be used to define the position of the auxiliary encoders. For example,

```
DE 0,500,-30,300
```

sets their initial values.

The positions of the auxiliary encoders may be interrogated with DE?. For example

```
DE ?,,?
```

returns the value of the X and Z auxiliary encoders.

The auxiliary encoder position may be assigned to variables with the instructions

```
V1=_DEX
```

Backlash Compensation

The dual loop methods can be used for backlash compensation. This can be done by two approaches:

Continuous dual loop

Sampled dual loop

To illustrate the problem, consider that the coupling between the motor and the load has a backlash. The approach is to mount position encoders on both the motor and the load.

The continuous dual loop combines the two feedback signals to achieve stability. This method requires careful system tuning, and depends on the magnitude of the backlash. However, once successful, this method compensates for the backlash continuously.

The second method, the sampled dual loop, reads the load encoder only at the end point and performs a correction. This method is independent of the size of the backlash. However, it is effective only in point-to-point motion systems which require position accuracy only at the endpoint.

Continuous Dual Loop - Example

Connect the load encoder to the main encoder port and connect the motor encoder to the dual encoder port. The dual loop method splits the filter function between the two encoders. It applies the KP (proportional) and KI (integral) terms to the position error, based on the load encoder, and applies the KD (derivative) term to the motor encoder. This method results in a stable system.

The dual loop method is activated with the instruction DV (Dual Velocity), where

```
DV      1,1,1,1
```

activates the dual loop for the four axes and

```
DV      0,0,0,0
```

disables the dual loop.

Note that the dual loop compensation depends on the backlash magnitude, and in extreme cases will not stabilize the loop. The proposed compensation procedure is to start with KP=0, KI=0 and to maximize the value of KD under the condition DV1. Once KD is found, increase KP gradually to a maximum value, and finally, increase KI, if necessary.

Sampled Dual Loop - Example

Run a linear slide by a rotary motor via a lead screw. As the lead screw has a backlash, it is necessary to use a linear encoder to monitor the position of the slide. In addition, for stability reasons, it is best to use a rotary encoder on the motor.

Connect the rotary encoder to the X-axis and connect the linear encoder to the auxiliary encoder of X. Let the required motion distance be one inch, and assume that this corresponds to 40,000 counts of the rotary encoder and 10,000 counts of the linear encoder.

The design approach is to drive the motor a distance, which corresponds to 40,000 rotary counts. Once the motion is complete, the controller monitors the position of the linear encoder and performs position corrections.

This is done by the following program.

Instruction	Interpretation
#DUALOOP	Label
CE 0	Configure encoder
DE0	Set initial value
PR 40000	Main move
BGX	Start motion
#CORRECT	Correction loop
AMX	Wait for motion completion
V1=10000-_DEX	Find linear encoder error
V2=-_TEX/4+V1	Compensate for motor error
JP#END,@ABS[V2]<2	Exit if error is small
PR V2*4	Correction move
BGX	Start correction
JP#CORRECT	Repeat

EN	#END

Motion Smoothing (S curve profiling)

The SMC-2000 controller allows the smoothing of the velocity profile to reduce the mechanical vibration of the system. The resulting velocity profile is known as S curve.

Trapezoidal velocity profiles have acceleration rates that change abruptly from zero to maximum value. The discontinuous acceleration results in infinite jerk that causes vibration. The smoothing of the acceleration profile leads to a continuous acceleration profile and a finite jerk, which reduces the mechanical shock and vibration.

The smoothing is accomplished by filtering the acceleration profile. The degree of the smoothing is specified by the commands:

IT x,y,z,w	Independent time constant
VT n	Vector time constant

IT is used for smoothing independent moves of the type JG, PR, PA, whereas VT is used to smooth vector moves of the type VM and LM.

The smoothing parameters, x,y,z,w and n are numbers between 0 and 1 and determine the degree of filtering, where the maximum value of 1 implies no filtering, resulting in trapezoidal velocity profiles. Smaller values of the smoothing parameters imply heavier filtering and smoother moves.

The following diagrams illustrate the effect of the smoothing. Fig. 6.6 shows the trapezoidal velocity profile and the modified acceleration and velocity.

Also note that the smoothing process results in longer motion time.

Example - Smoothing

PR 20000	Position
AC 100000	Acceleration
DC 100000	Deceleration
SP 5000	Speed
IT .5	Filter for S-curve
BG X	Begin

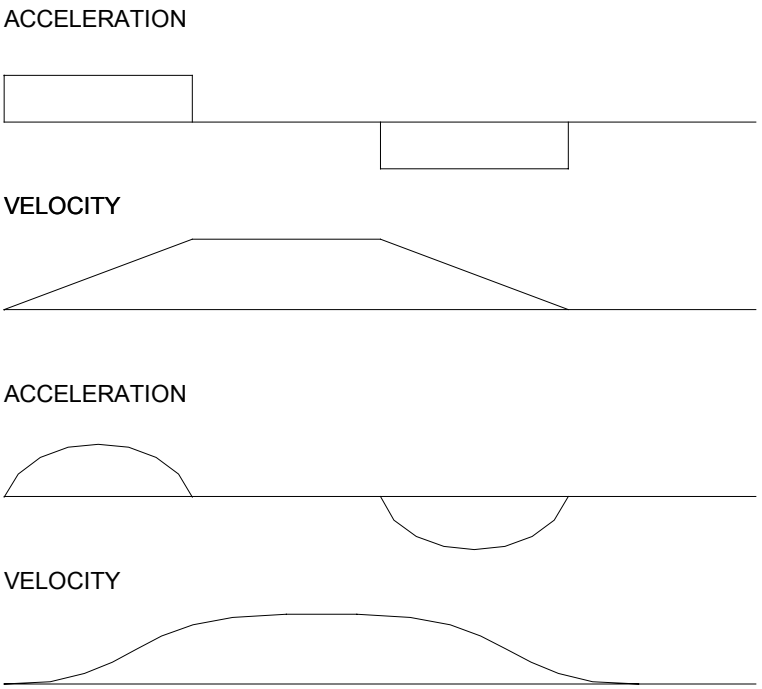


Figure 6.6 - Trapezoidal velocity and smooth velocity profiles

Homing

The Find Edge (FE) and Home (HM) instructions may be used to home the motor to a mechanical reference. This reference is connected to the Home input line. The HM command initializes the motor to the encoder index pulse in addition to the Home input. The configure command (CN) is used to define the polarity of the home input.

The Find Edge (FE) instruction is useful for initializing the motor to a home switch. The home switch is connected to the Homing Input. When the Find Edge command and Begin is used, the motor will accelerate up to the slew speed and slew until a transition is detected on the Homing line. The motor will then decelerate to a stop. A high deceleration value must be input before the find edge command is issued for the motor to decelerate rapidly after sensing the home switch. The velocity profile generated is shown in Fig. 6.7.

The Home (HM) command can be used to position the motor on the index pulse after the home switch is detected. This allows for finer positioning on initialization. The HM command and BG command cause the following sequence of events to occur.

1. Upon begin, motor accelerates to the slew speed. The direction of its motion is determined by the state of the homing input. A zero (GND) will cause the motor to start in the forward direction; +24V will cause it to start in the reverse direction. The CN command is used to define the polarity of the home input.
2. Upon detecting the home switch changing state, the motor begins decelerating to a stop.
3. The motor then traverses very slowly back until the home switch toggles again.
4. The motor then traverses forward until the encoder index pulse is detected.
5. The SMC-2000 defines the home position (0) as the position at which the index was detected.

Example:

#HOME	Label
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
SP 5000	Speed for Home Search
HM X	Home X
BG X	Begin Motion
AM X	After Complete
MG "AT HOME"	Send Message
EN	End
#EDGE	Label
AC ,2000000	Acceleration rate
DC ,2000000	Deceleration rate
SP ,8000	Speed
FE Y	Find edge command
BG Y	Begin motion
AM Y	After complete
MG "FOUND HOME"	Print message
DP,0	Define position as 0
EN	End

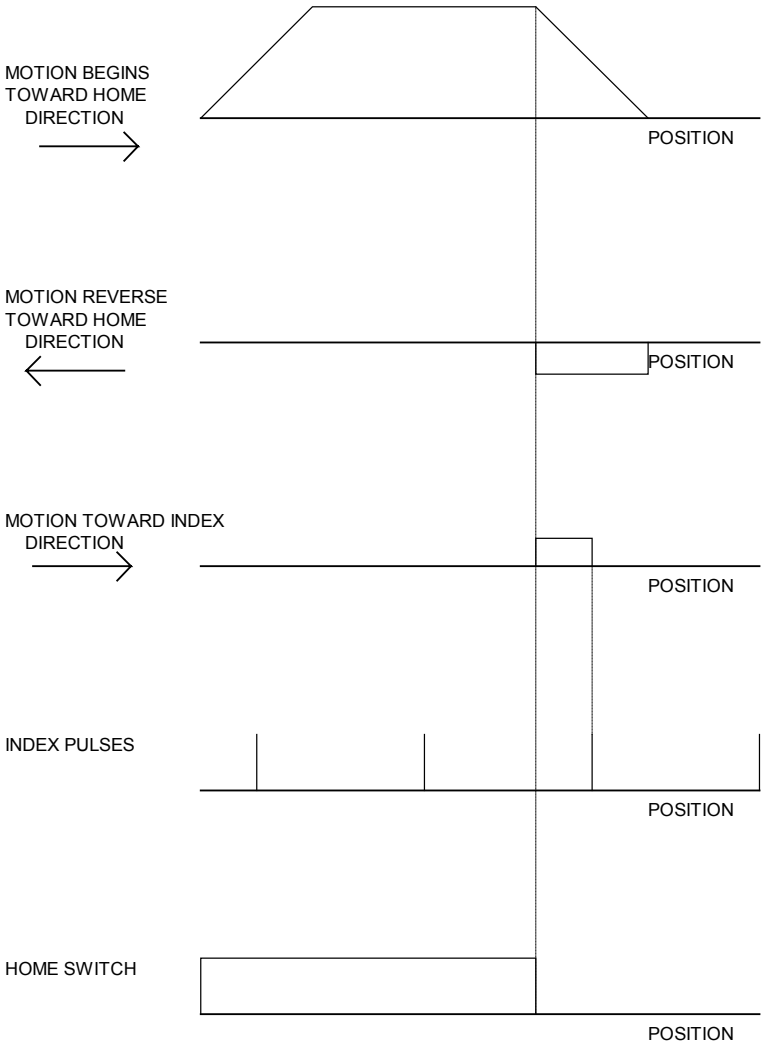


Figure 6.7 - Motion intervals in the Home sequence

High Speed Position Capture

Often it is desirable to capture the position precisely for registration applications. The SMC-2000 provides a position latch feature. This feature allows the position of X,Y,Z or W to be captured within 25 microseconds of an external signal. The external signal is input at inputs 1 through 4, and 9 through 12 on the SMC-2000-8.

IN1 X-axis latch	IN 9 E-axis latch
IN2 Y-axis latch	IN10 F-axis latch
IN3 Z-axis latch	IN11 G-axis latch
IN4 W-axis latch	IN12 H-axis latch

The SMC-2000 software commands, AL, LT, and RL, are used to arm the latch and report the latched position. The steps to use the latch are as follows:

1. Give the AL XYZW command, or ABCDEFGH for SMC-2000-8, to arm the latch for the specified axis or axes.
2. Test to see if the latch has occurred (Input goes low) by using the _AL X or Y or Z or W command. Example, V1=_ALX returns the state of the X latch into V1. V1 is 1 if the latch has not occurred.
3. After the latch has occurred, read the captured position with the RL XYZW command or _RL XYZW.

Note: The latch must be re-armed after each latching event.

Example:

#LATCH	Latch program
JG,5000	Jog Y
BG Y	Begin Y
AL Y	Arm Latch
#WAIT	Loop for Latch=1
JP #WAIT,_ALY=1	Wait for latch
RESULT=_RLY	Report position
RESULT=	Print result
EN	End

Electronic Cam

The electronic cam is a motion control mode which enables the periodic synchronization of several axes of motion when one of the axes is independent and is not necessarily driven by the motion controller.

The electronic cam is a more general type of electronic gearing which allows a table based relationship between the axes. It allows synchronizing all the controller axes. Therefore, the SMC-2000-8 may have one master and up to seven slaves. To simplify the presentation, we will limit the description to a 4-axis controller.

EAp where p=X,Y,Z,W

p is the selected master axis

To illustrate the procedure of setting cam mode the master position M_0, consider the cam relationship for the slave axis Y, when the master is X. Such a graphic relationship is shown in Figure 6.8.

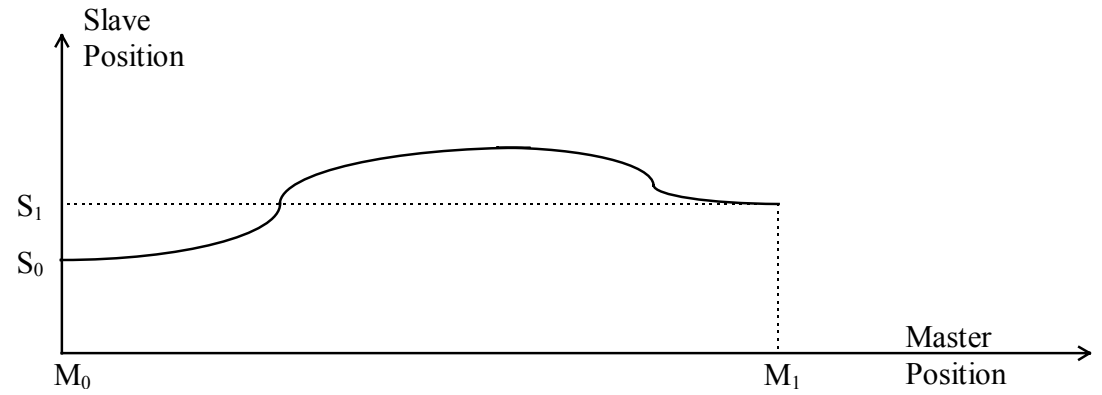


Figure 6.8 - Electronic Cam Cycle

The cam cycle starts at the master position M_0 and ends at the master position M_1. This implies that the cycle for the master, CM is

$$CM=M_1 - M_0$$

The slave axis must equal S_0 . When the master position is M_0 and S_1 when the when the master position equals M_1. Over one cycle, the change in slave position, CS, equals

$$CS=S_1 - S_0$$

In the cam mode, the positions of the master and the slave are redefined to fit the values shown in Figure 6.8. This implies that if the master axis position increases beyond M_1 , the master position is decreased by CM and the slave position is decreased by CS. On the other hand, if the master axis moves in the negative direction through the point M_0 , the master position is increased by CM and the slave position is increased by CS. To specify the values of CM and CS, we use the instruction

EM x,y,z,w

where the values x,y,z,w are the CM or CS values for the corresponding axes.

The range of CM is an integer between 1 and 8,388,607 and the range of CS is an integer between 0 and 2,147,483,647. If CS is negative, its absolute value is specified.

For example, suppose that the cam relationship is as expressed in Fig 6.9.

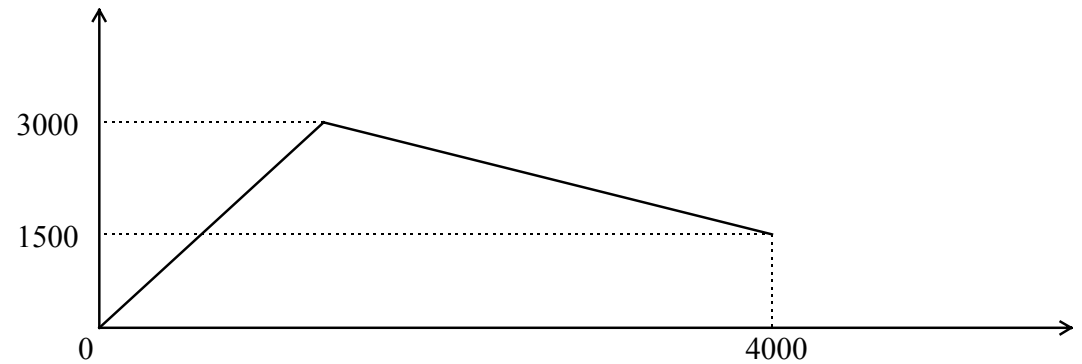


Figure 6.9 - Electronic Cam Example

Since the cycles are 4000 and 1500 for X and Y respectively, the command is

EP m,n

For example, EP 100,500 indicates that the table starts at the master position of 500 and that the following master points are 600, 700, etc.

Finally, the table parameters are defined with the instruction

ET[n]=x,y,z where n starts at 0 and may go up to 256

For example,

ET[7]=100,300,-200

defines a row in the table. It indicates that the position X=100, Y=300, and Z=-200 must be synchronized.

The table generated with ET[n] can be stored in the SMC-2000 with the BN command.

Note that only the slave points must be given. The master points are defined by the EP instruction.

To illustrate the process, consider the simple example where X, the master, has a cycle of 4000 counts and Y, the slave, must be synchronized with a gear ratio of 1 during the first half of the cycle and a zero ratio during the second half.

To construct the table we start by selecting the X axis as the master.

EAX

Define the cycles as

EM 4000,2000

Since the table is quite simple, it can be defined by a few points with an interval of 1000.

EP 1000,0

The first point, when X=0 is

ET[0]=,0

It is followed by:

ET[1]=,1000

ET[2]=,2000

ET[3]=,2000

ET[4]=,2000

Once the cam mode is defined, it can be enabled or disabled with the instruction

EB n where n=1 enables the cam mode and n=0 disables it.

When the cam mode is enabled, the position of the mater is monitored and is re-defined as a value within the cycle.

To engage the slave axes at a programmed point, we use the command

EG x,y,z,w where x,y,z,w are the master positions at which the corresponding slave axes must be engaged.

If the value of any parameter is outside the range specified by the master cycle, the cam engages that axis immediately. When a slave motor is engaged, its position is redefined to fit with the cycle.

To stop a slave axis, use the instruction

EQ x,y,z,w — where x,y,z,w are the master positions at which the corresponding slave axes must be disengaged.

This disengages the slave axis at a specified master position. If the parameter is outside the master cycle, the stopping is instantaneous.

Programmed start and stop can be used only when the master moves forward.

To illustrate the complete process, consider the cam relationship described by the equation:

$$Y=0.5 * 100\sin(0.18 * X)$$

where X is the master, with a cycle of 2000 counts.

The cam table can be constructed manually, point by point, or automatically by a program. The following program includes the set-up.

The instruction EAX defines X as the master axis. The cycle of the master is CM=2000. Over that cycle, X varies by CS=1000. This leads to the instruction EM 2000,1000.

Suppose we want to define a table with 100 segments. This implies increments of 20 counts each. If the master points are to start at zero, the required instruction is EP 20,0.

The following routine computes the table points. As the phase equals 0.18X and X varies in increments of 20, the phase varies by increments of 3.6^{o}. The program then computes the values of X according to the equation and assigns the values to the table with the instruction ET[N]=,Y.

#SETUP	Label
EAX	Select X as master
EM 2000,1000	Cam cycles
EP 20,0	Master position increments
N=0	index
#LOOP	Loop to construct table from equation
P=N*3.6	Note 3.6 = 0.18*20
S=@SIN[P]*10	Define sine position
Y=N*10+S	Define slave position
ET[N]=,Y	Define table
N=N+1	
JP #LOOP,N<=100	Repeat the process
EN	

Now suppose that the slave axis is engaged with a start signal, input 1, but that both the engagement and disengagement points must be done at the center of the cycle: X=1000 and Y=500. This implies that Y must be driven to that point to avoid a jump.

This is done with the program:

#RUN	Label

EB1	Enable Cam
PA,500	Starting position
SP,5000	Y Speed
BGY	Move Y Motor
AM	After Y moved
AI1	Wait for start signal
EG,1000	Disengage slave
AI-1	Wait for stop signal
EQ,1000	Disengage slave
EN	End

The following example illustrates a cam program with a master axis, Z, and two slaves, X and Y.

#A;V1=0	Label; Initialize variable
PA 0,0; BGXY; AMZXY	Go to position 0,0 on X and Y axes
EAZ	Z axis as the Master for ECAM
EM0,0,4000	Change for Z is 4000, zero for X,Y
EP400,0	ECAM interval is 400 counts with zero start
ET[0]=0,0	When master is at zero position; first point
ET[1]=40,20	2nd point in the ECAM table
ET[2]=120,60	3rd point in the ECAM table
ET[3]=240,120	4th point in the ECAM table
ET[4]=280,140	5th point in the ECAM table
ET[5]=280,140	6th point in the ECAM table
ET[6]=280,140	7th point in the ECAM table
ET[7]=240,120	8th point in the ECAM table
ET[8]=120,60	9th point in the ECAM table
ET[9]=40,20	10th point in the ECAM table
ET[10]=0,0	Starting for the next cycle
EB1	Enable ECAM mode
JGZ=4000	Set Z to jog at 4000
EG 0,0	Engage both X and Y when Master=0
BGZ	Begin jog on Z axis
#LOOP; JP#LOOP,V1=0	Loop until the next variable is set
EQ2000,2000	Disengage X and Y when Master = 2000
MF 2000	Wait until the master goes to 2000
ST Z	Stop the Z axis motion
EB 0	Exit the ECAM mode
EN	End of the program

The above example shows how the ECAM program is instructed and how the commands can be given to the controller.

Application Programming

Introduction

The SMC-2000 programming language is a powerful language that allows users to customize a program to handle their particular application. Complex programs can be downloaded into the SMC-2000 memory for later execution. Utilizing the SMC-2000 to execute sophisticated programs frees the host computer for other tasks. However, the host computer can still send commands to the controller at any time, even while a program is being executed.

In addition to standard motion commands, the SMC-2000 provides several commands that allow the SMC-2000 to make its own decisions. These commands include conditional jumps, event triggers, and subroutines. For example, the command JP#LOOP, N<10 causes a jump to the label #LOOP if the variable N is less than 10.

For greater programming flexibility, the SMC-2000 provides 254 user-defined variables, arrays and arithmetic functions. For example, the length in a cut-to-length operation can be specified as a variable in a program and then be assigned by an operator.

The following sections in this chapter discuss all aspects of creating applications programs.

Program Format

A SMC-2000 program consists of several SMC-2000 instructions combined to solve a machine control application. Action instructions, such as starting and stopping motion, are combined with Program Flow instructions to form the complete program. Program Flow instructions evaluate real-time conditions, such as elapsed time or motion complete, and alter program flow accordingly.

A delimiter must separate each SMC-2000 instruction in a program. Valid delimiters are the semicolon (;) or carriage return. The semicolon is used to separate multiple instructions on a single program line. A carriage return enters the final command on a program line.

All SMC-2000 programs must begin with a label and end with an End (EN) statement. Labels start with the pound (#) sign followed by a maximum of seven characters. The first character must be a letter; after that, numbers are permitted. Spaces are not permitted. NOTE: All letter must be UPPER CASE.

The maximum number of labels that may be defined is 254.

Valid labels

#BEGIN

#SQUARE

#X1

#BEGIN1

Invalid labels

#1Square

#123

Special Labels

There are also some special labels, which are used to define input interrupt subroutines, limit switch subroutines, error handling subroutines, and command error subroutines.

#LIMSWI	Label for Limit Switch subroutine
#POSERR	Label for excess Position Error subroutine
#ININT	Label for Input Interrupt subroutine
#CMDERR	Label for incorrect command subroutine
#COMINT	Label for communication interrupt
#MCTIME	Label for timeout if encoder is not in-position within time specified by TW.
#AUTO	Label for automatic program start

Example Program:

#AUTO	Beginning of the Program
SH	Turn motors on
PR 10000,20000;BG XY	Specify relative distances on X and Y axes; Begin Motion
AM	Wait for motion complete
WT 2000	Wait 2 sec
JP # AUTO	Jump to label AUTO
EN	End of Program

The above program will execute automatically at power up and move X and Y 10000 and 20000 units. After the motion is complete, the motors rest for 2 seconds. The cycle repeats indefinitely until the stop command is issued.

Executing Programs - Multitasking

Up to four programs can run independently. The programs, called threads, are numbered 0 through 3, where 0 is the main thread.

The main thread differs from the others in the following points:

1. Only the main thread may use the input command, IN.

2. In a case of interrupts, due to inputs, limit switches, position errors or command errors, it is the program in thread 0 which jumps to those subroutines.

The execution of the various programs is done with the instruction:

XQ #A, n

Where n indicates the thread number. To halt the execution of any thread, use the instruction

HX n

where n is the thread number.

Note that both the XQ and HX functions can be performed by an executing program.

Multitasking is useful for executing independent operations such as PLC functions that occur independently of motion. The example below produces a waveform on Output 1 independent of a move.

#TASK1	Task1 label
AT0	Initialize reference time
CB1	Clear Output 1
#LOOP1	Loop1 label
AT 10	Wait 10 msec from reference time
SB1	Set Output 1
AT -40	Wait 40 msec from reference time, then initialize reference
CB1	Clear Output 1
JP #LOOP1	Repeat Loop1
#TASK2	Task2 label
XQ #TASK1,1	Execute Task1
#LOOP2	Loop2 label
PR 1000	Define relative distance
BGX	Begin motion
AMX	After motion done
WT 10	Wait 10 msec
JP #LOOP2,@IN[2]=1	Repeat motion unless Input 2 is low
HX	Halt all tasks

The program above is executed with the instruction XQ #TASK2,0 which designates TASK2 as the main thread. #TASK1 is executed within TASK2.

Debugging Programs

The SMC-2000 provides trace and error code commands which are used in debugging programs. The trace command may be activated using the command, TR1. This command causes each line in a program to be sent out to the communications port immediately prior to execution. The TR1 command is useful for debugging programs. TR0 disables the trace function. The TR command may also be included as part of a program.

If there is a program error, the SMC-2000 will halt program execution at the line number at which an error occurs and display the line. The user can obtain information about the type of error condition that occurred by using the command, TC1. This command reports back a number and error condition as follows:

Error Codes:

1	Unrecognized command
2	Command only valid from program
3	Command not valid in program
4	Operand error
5	Input buffer full
6	Number out of range
7	Command not valid while running
8	Command not valid while not running
9	Variable error
10	Empty program line or undefined label
11	Invalid label or line number
12	Subroutine more than 16 deep
13	JG only valid when running in jog mode
14	EEPROM check sum error
15	EEPROM write error
16	IP incorrect sign during position move or IP given during forced deceleration
17	ED, BN and DL not valid while program running
18	Command not valid when contouring
19	Application program/strand already executed
20	Begin not valid with motor off
21	Begin not valid while running
22	Begin not possible due to Limit Switch
24	Begin not valid because no sequence defined
25	Variable not given in IN command
28	S operand not valid
29	Not valid during coordinated move
30	Sequence segment too short
31	Total move distance in a sequence > 2 billion
32	More than 511 segments in a sequence
41	Contouring record range error
42	Contour data being sent too slowly
46	Gear axis both master and follower
50	Not enough fields
51	Question mark not valid
52	Missing " or string too long
53	Error in {}
54	Question mark part of string

55 Missing [or []
56 Array index invalid or out of range
57 Bad function or array
58 Unrecognized command in a command response (i.e._TPQ)
59 Mismatched parentheses
60 Download error - line too long or too many lines
61 Duplicate or bad label
62 Too many labels
65 IN command must have a comma
66 Array space full
67 Too many arrays or variables
71 IN only valid in task #0
80 Record mode already running
81 No array or source specified
82 Undefined array
83 Not a valid number
84 Too many elements
90 Only X,Y,Z,W or A,B,C,D,E,F,G,H valid operand
96 SM jumper needs to be installed for stepper motor operation
100 Not valid when running ECAM
101 Improper index to ET (must be 0-256)
102 No master axis for ECAM
103 Master axis modulus greater than 256*EP value
104 Not valid when axis performing ECAM
105 EB1 command must be given first
114 Absolute Encoder option not installed
115 Motor must be in MO for this comment
116 Absolute Encoder responded with an alarm
117 Absolute Encoder did not respond
118 Controller has GL1600, not GL1800

Note: TC0 or TC will return the error code only without the text message.

Example:

#A	Program Label
PR1000	Position Relative 1000
BGX	Begin
PR5000	Position Relative 5000
EN	End

:XQ #A	Execute #A
?003 PR5000	Error on Line 3
:TC1	Tell Error Code
AMX;PR5000;BGX	Add After Motion Done
:XQ #A	Execute #A

Program Flow Commands

The SMC-2000 provides several instructions that control program flow. Normally, the SMC-2000 program sequencer executes instructions in a program sequentially. Program Flow commands, however, may be used to redirect program flow. A summary of these commands is given below and they are detailed in the following sections.

Program Flow Command Summary

JP	Conditional Jump
JS	Conditional Jump to Subroutine
AD	After Distance Trigger
AI	After Input Trigger
AM	After Motion Complete Trigger
AP	After Absolute Position Trigger
AR	Relative Distance Trigger
AS	After Speed Trigger
AT	Wait for time with respect to reference
AV	After Vector Distance Trigger
MC	Trigger "In position" trigger (TW x,y,z,w sets timeout for in-position)
MF	Trigger Forward motion
MR	Trigger Reverse motion
WC	Wait for Contour Data
WT	Wait for time to elapse

Event Triggers & Trippoints

To function independently from the host computer, the SMC-2000 can be programmed to make decisions based on the occurrence of an event. Such events include waiting for motion to be complete, waiting for a specified amount of time to elapse, or waiting for an input to change logic levels.

The SMC-2000 provides several event triggers that cause the program sequencer to halt until the specified event occurs. Normally, a program is automatically executed sequentially one line at a time. When an event trigger instruction is decoded, however, the actual program sequence is halted. The program sequence does not continue until the event trigger is "tripped". For example, the motion complete trigger can be used to separate two move sequences in a program. The commands for the second move sequence will not be executed until the motion is complete on the first motion sequence. In this way, the SMC-2000 can make decisions based on its own status or external events without intervention from a host computer.

SMC-2000 Event Triggers

Command	Function
AM X Y Z W or S (A B C D E F G H)	Halts program execution until motion is complete on the specified axes or motion sequence(s). AM with no parameter tests for motion complete on all axes. This command is useful for separating motion sequences in a program.
AD X or Y or Z or W (A or B or C or D or E or F or G or H)	Halts program execution until position command has reached the specified relative distance from the start of the move. Only one axis may be specified at a time.
AR X or Y or Z or W (A or B or C or D or E or F or G or H)	Halts program execution until after specified distance from the last AR or AD command has elapsed. Only one axis may be specified at a time.
AP X or Y or Z or W (A or B or C or D or E or F or G or H)	Halts program execution until after absolute position occurs. Only one axis may be specified at a time.
AI +/-n	Halts program execution until after specified input is at specified logic level. n specifies input line. Positive is high logic level, negative is low level.
AS X Y Z W S (A B C D E F G H)	Halts program execution until specified axis has reached its slew speed.
AT +/-n	Halts program execution until n msec from reference time. AT 0 sets reference. AT n waits n msec from reference. AT -n waits n msec from reference and sets new reference after elapsed time.
AV n	Halts program execution until specified distance along a coordinated path has occurred.
MC X or Y or Z or W (A or B or C or D or E or F or G or H)	Halt program execution until after the motion profile has been completed and the encoder has entered or passed the specified position. TW x,y,z,w sets timeout to declare an error if not in position. If timeout occurs, then the trippoint will clear and the stop code will be set to 99. An application program will jump to label #MCTIME.
MF X or Y or Z or W (A or B or C or D or E or F or G or H)	Halt program execution until after forward motion reached absolute position. Only one axis may be specified. If position is already past the point, then MF will trip immediately.
MR X or Y or Z or W (A or B or C or D or E or F or G or H)	Halt program execution until after reverse motion reached absolute position. Only one axis may be specified. If position is already past the point, then MR will trip immediately.
WT n	Halts program execution until specified time in msec has elapsed.

Event Trigger Examples:

Event Trigger - Multiple Move Sequence

The AM trippoint is used to separate the two PR moves. If AM is not used, the controller returns a ? for the second PR command because a new PR cannot be given until motion is complete.

#TWOMOVE	Label

PR 2000	Position Command
BGX	Begin Motion
AMX	Wait for Motion Complete
PR 4000	Next Position Move
BGX	Begin 2nd move
EN	End program

In the above example, the AM trippoint is used to separate the two PR moves. If AM is not used, the controller returns a ? for the second PR command because a new PR cannot be given until motion is complete.

Event Trigger - Set Output after Distance

Set output bit 1 after a distance of 1000 counts from the start of the move. The accuracy of the trippoint is the speed multiplied by the sample period.

#SETBIT	Label
SP 10000	Speed is 10000
PA 20000	Specify Absolute position
BGX	Begin motion
AD 1000	Wait until 1000 counts
SB1	Set output bit 1
EN	End program

The above example sets output bit 1 after a distance of 1000 counts from the start of the move. The accuracy of the trippoint is the speed multiplied by the sample period.

Event Trigger - Repetitive Position Trigger

To set the output bit every 10000 counts during a move, the AR trippoint is used as shown in the next example.

#TRIP	Label
JG 50000	Specify Jog Speed
BGX;N=0	Begin Motion
#REPEAT	# Repeat Loop
AR 10000	Wait 10000 counts
TPX	Tell Position
SB1	Set output 1
WT50	Wait 50 msec
CB1	Clear output 1
N=N+1	Increment counter
JP #REPEAT,N<5	Repeat 5 times
STX	Stop
EN	End

Event Trigger - Start Motion on Input

This example waits for input 1 to go low and then starts motion. Note: The AI command actually halts execution of the program until the input occurs. If you do not want to halt the program sequences, you can use the Input Interrupt function (II) or use a conditional jump on an input, such as JP #GO,@IN[1] = 0.

#INPUT	Program Label
AI-1	Wait for input 1 low
PR 10000	Position command
BGX	Begin motion
EN	End program

Event Trigger - Set output when At speed

#ATSPEED	Program Label
JG 50000	Specify jog speed
AC 10000	Acceleration rate
BGX	Begin motion
ASX	Wait for at slew speed 50000
SB1	Set output 1
EN	End program

Event Trigger - Change Speed along Vector Path

The following program changes the feed rate or vector speed at the specified distance along the vector. The vector distance is measured from the start of the move or from the last AV command.

#VECTOR	Label
VMXY;VS 5000	Coordinated path
VP 10000,20000	Vector position
VP 20000,30000	Vector position
VE	End vector
BGS	Begin sequence
AV 5000	After vector distance
VS 1000	Reduce speed
EN	End

Event Trigger - Multiple move with wait

#MOVES	Label
PR 12000	Distance
SP 20000	Speed
AC 100000	Acceleration
BGX	Start Motion
AD 10000	Wait a distance of 10,000 counts
SP 5000	New Speed
AMX	Wait until motion is completed
WT 200	Wait 200 ms
PR -10000	New Position
SP 30000	New Speed
AC 150000	New Acceleration

BGX	Start Motion
EN	End

Define Output Waveform Using AT

The following program causes Output 1 to be high for 10 msec and low for 40 msec. The cycle repeats every 50 msec.

#OUTPUT	Program label
AT0	Initialize time reference
SB1	Set Output 1
#LOOP	Loop
AT 10	After 10 msec from reference,
CB1	Clear Output 1
AT -40	Wait 40 msec from reference and reset reference
SB1	Set Output 1
JP #LOOP	Loop
EN	

Conditional Jumps

The SMC-2000 provides Conditional Jump (JP) and Conditional Jump to Subroutine (JS) instructions for branching to a new program location based on a specified condition. Unlike event triggers, the conditional jump instruction does not halt the program sequence. Instead, it tests to see if a condition is satisfied and then branches to a new location or subroutine. (A subroutine is a group of commands defined by a label and EN command. After all the commands in the subroutine are executed, a return is made to the main program). If the condition is not satisfied, the program sequence continues to the next program line.

The JP and JS instructions have the following format:

Format:	**Meaning**
JS destination, logical condition	Jump to subroutine if logical condition is satisfied
JP destination, logical condition	Jump to location if logical condition is satisfied

The destination is a program line number or label. The destination is where the program sequencer jumps to if the specified condition is satisfied. The comma designates "IF". The logical condition tests two operands with logical operators. The operands can be any valid SMC-2000 numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions.

Logical operators:

<	less than
>	greater than
=	equal to
<=	less than or equal to
>=	greater than or equal to
<>	not equal

Operands:

Type	**Examples**

Number	V1=6
Numeric Expression	V1=V7*6
	@ABS[V1]>10
Array Element	V1<Count[2]
Variable	V1<V2
Internal Variable	_TPX=0
	_TVX>500
I/O	V1>@AN[2]
	@IN[1]=0

The jump statement may also be used without a condition.

Example conditional jump statements are given below:

Conditional	**Meaning**
JP #LOOP,COUNT<10	Jump to #LOOP if the variable, COUNT, is less than 10
JS #MOVE2,@IN[1]=1	Jump to subroutine #MOVE2 if input 1 is logic level high. After the subroutine MOVE2 is executed, the program sequencer returns to the main program location where the subroutine was called.
JP #BLUE,@ABS[V2]>2	Jump to #BLUE if the absolute value of variable, V2, is greater than 2
JP #C,V1*V7<=V8*V2	Jump to #C if the value of V1 times V7 is less than or equal to the value of V8*V2
JP#A	Jump to #A

Conditional jumps are useful for testing events in real-time. They allow the SMC-2000 to make decisions without a host computer. For example, the SMC-2000 can decide between two motion profiles based on the state of an input line. Or, the SMC-2000 can keep track of how many times a motion profile is executed.

NOTE: Conditions may NOT be grouped using the AND (&) or OR (|) operators.

Example:

Move the X motor to absolute position 1000 counts and back to zero ten times. Wait 100 msec between moves.

#BEGIN	Begin Program
COUNT=10	Initialize loop counter
#LOOP	Begin loop
PA 1000	Position absolute 1000
BGX	Begin move
AMX	Wait for motion complete
WT 100	Wait 100 msec
PA 0	Position absolute 0
BGX	Begin move
AMX	Wait for motion complete
WT 100	Wait 100 msec
COUNT=COUNT-1	Decrement loop counter
JP #LOOP,COUNT>0	Test for 10 times through loop

EN	End Program

Subroutines

A subroutine is a group of instructions beginning with a label and ending with an END (EN). Subroutines are called from the main program with the jump subroutine instruction JS, followed by a label or line number, and conditional statement. Up to 16 subroutines can be nested. After the subroutine is executed, the program sequencer returns to the program location where the subroutine was called unless the subroutine stack is manipulated as described in the following section.

Example:

An example of a subroutine to draw a square 500 counts per side is given below. The square is drawn at vector position 1000,1000.

#M	Begin Main Program
CB1	Clear Output Bit 1 (pick up pen)
VP 1000,1000;LE;BGS;AMS	Define vector position; move pen
SB1	Set Output Bit 1 (put down pen)
JS #SQUARE;CB1	Jump to square subroutine
EN	End Main Program
#SQUARE	Square subroutine
V1=500;JS #L	Define length of side
V1=-V1;JS #L	Switch direction
EN	End subroutine
#L;PR V1,V1;BGX	Define X,Y; Begin X
AMX;BGY;AMY	After motion on X, Begin Y
EN	End subroutine

Stack Manipulation

It is possible to manipulate the subroutine stack by using the ZS command. Every time a JS instruction, interrupt or automatic routine (such as #POSERR or #LIMSWI) is executed, the subroutine stack is incremented by 1. Normally the stack is restored with an EN instruction. Occasionally it is desirable not to return back to the program line where the subroutine or interrupt was called. The ZS1 command clears 1 level of the stack. This allows the program sequencer to continue to the next line. The ZS0 command resets the stack to its initial value. For example, if a limit occurs and the #LIMSWI routine is executed, it is often desirable to restart the program sequence instead of returning to the location where the limit occurred. To do this, give a ZS command at the end of the #LIMSWI routine.

Automatic Subroutines for Monitoring Conditions

Often it is desirable to monitor certain conditions continuously without tying up the host or SMC-2000 program sequences. The SMC-2000 can monitor several important conditions in the background. These conditions include checking for the occurrence of a limit switch, a defined input, position error, or a command error. Automatic monitoring is enabled by inserting a special, predefined label in the applications program. The pre-defined labels are:

#LIMSWI	Limit switch on any axis goes low

#POSERR	Position error exceeds limit specified by ER
#ININT	Input specified by II goes low
#CMDERR	Bad command given
#COMINT	Communication interrupt occurred
#MCTIME	Timeout for In-position trippoint, MC
#AUTO	Auto start program on power-up

For example, the #POSERR subroutine will automatically be executed when any axis exceeds its position error limit. The commands in the #POSERR subroutine could decode which axis is in error and take the appropriate action. In another example, the #ININT label could be used to designate an input interrupt subroutine. When the specified input occurs, the program will be executed automatically.

Note: An application program must be running for automatic monitoring to function.

Example - Limit Switch:

This program prints a message upon the occurrence of a limit switch. Note, for the #LIMSWI routine to function, the SMC-2000 must be executing an applications program from memory. This can be a very simple program that does nothing but loop on a statement, such as #LOOP;JP #LOOP;EN. Motion commands, such as JG 5000 can still be sent from the PC even while the "dummy" applications program is being executed.

#LOOP	Dummy Program
JP #LOOP;EN	Jump to Loop
#LIMSWI	Limit Switch Label
MG "LIMIT OCCURRED"	Print Message
RE	Return to main program
:XQ #LOOP	Execute Dummy Program
:JG 5000	Jog
:BGX	Begin Motion

Now, when a forward limit switch occurs on the X axis, the #LIMSWI subroutine will be executed.

Note: The RE command is used to return from the #LIMSWI subroutine.

Note: The #LIMSWI will continue to be executed until the limit switch is cleared (goes high).

Example - Position Error

#LOOP	Dummy Program
JP #LOOP;EN	Loop
#POSERR	Position Error Routine
V1=_TEX	Read Position Error
MG "EXCESS POSITION ERROR"	Print Message
MG "ERROR=",V1=	Print Error
RE	Return from Error
:XQ #LOOP	Execute Dummy Program
:JG 100000	Jog at High Speed
:BGX	Begin Motion

Now, if the position error on the X axis exceeds that specified by the ER command, the #POSERR routine will execute.

Note: The RE command is used to return from the #POSERR subroutine

Note: The #POSERR routine will continue to be executed until the position error is cleared (is less than the ER limit).

Input Interrupt Example:

#A	Label
II1	Input Interrupt on 1
JG 30000,,,60000	Jog
BGXW	Begin Motion
#LOOP;JP#LOOP;EN	Loop
#ININT	Input Interrupt
ST;AM	Stop Motion
#TEST;JP #TEST, @IN[1]=0	Test for Input 1 still low
BGXW;RI	Begin motion and Return to Main Program
EN	

NOTE: Use the RI command to return from #ININT subroutine.

Bad Command Example

#BEGIN	Begin main program
IN "ENTER SPEED", SPEED	Prompt for speed
JG SPEED;BGX;	Begin motion
JP #BEGIN	Repeat
EN	End main program
#CMDERR	Command error utility
JP#DONE,_TC<>6	Check if out of range
MG "SPEED TOO HIGH"	Send message
MG "TRY AGAIN"	Send message
ZS1	Adjust stack
JP #BEGIN	Return to main program
#DONE	End program if other error
ZS0	Zero stack
EN	End program

The above program prompts the operator to enter a jog speed. If the operator enters a number out of range (greater than 8 million), the #CMDERR routine will be executed prompting the operator to enter a new number.

Mathematical and Functional Expressions

For manipulation of data, the SMC-2000 provides the use of the following mathematical operators:

Operator	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division
&	Logical And (Bit-wise)
|	Logical Or (On some computers, a solid vertical line appears as a broken line)
( )	Parenthesis

The numeric range for addition, subtraction and multiplication operations is +/-2,147,483,647.9999. The precision for division is 1/65,000.

Mathematical operations are executed from left to right. Parentheses can be used and nested four deep. Calculations within a parentheses have precedence.

Examples:

SPEED=7.5*V1/2	The variable, SPEED, is equal to 7.5 multiplied by V1 and divided by 2
COUNT=COUNT+2	The variable, COUNT, is equal to the current value plus 2.
RESULT=_TPX-(@COS[45]*40)	Puts the position of X - 28.28 in RESULT. 40 * cosine of 45° is 28.28
TEMP=@IN[1]&@IN[2]	TEMP is equal to 1 only if Input 1 and Input 2 are high

The SMC-2000 also provides the following functions:

Function	Command Meaning
@ABS	Absolute Value
@SIN	Sine
@COS	Cosine
@COM	2's Complement
@FRAC	Fraction
@INT	Integer
@RND	Rounds number .5 and up to next integer
@IN[n]	Read digital input n
@AN[n]	Read analog input n
@SQR[n]	Square Root Function; Accuracy is +/-.0004

Functions may be combined with mathematical expressions. The order of execution is from left to right. The units of the SIN and COS functions are in degrees with resolution of 1/128 degrees. The values can be up to +/- 4 billion degrees.

Example:

V1=@ABS[V7]	The variable, V1, is equal to the absolute value of variable V7.
V2=5*@SIN[POS]	The variable, V2, is equal to five times the sine of the variable, POS.
V3=@IN[1]	The variable, V3, is equal to the digital value of input 1.
V4=@AN[5]	The variable, V4, is equal to the digital value of analog input 5.

Variables

Many motion applications include parameters that are variable. For example, a cut-to-length application often requires that the cut length be variable. The motion process is the same, however the length is changing.

To accommodate these applications, the SMC-2000 provides for the use of both numeric and string variables. A program can be written in which certain parameters, such as position or speed, are defined as variables. The variables can later be assigned by the operator or determined by the program calculations.

Example:

PR POSX	Assigns variable POSX to PR command
JG RPMY*70	Assigns variable RPMY multiplied by 70 to JG command.

Programmable Variables

The SMC-2000 allows the user to create up to 254 variables. Each variable is defined by a name which can be up to eight characters. The name must start with an alphabetic character, however, numbers are permitted in the rest of the name. Spaces are not permitted. Examples of valid and invalid variable names are:

Valid Variable Names

POSX

POS1

SPEEDZ

Invalid Variable Names

1POS

123

SPEED Z

It is recommended that variable names not be the same as SMC-2000 instructions. For example, PR is not a good choice for a variable name.

The range for numeric variable values is 4 bytes of integer followed by two bytes of fraction (+/- 2,147,483,647.9999).

String variables can contain up to six characters which must be in quotation. Example: VAR="STRING".

Numeric values can be assigned to programmable variables using the equal sign. Assigned values can be numbers, internal variables and keywords, and functions. String values can be assigned to variables using quotations.

Any valid SMC-2000 function can be used to return a value such as V1=@ABS[V2] or V2=@IN[1]. Arithmetic operations are also permitted.

Example:

POSX=_TPX	Assigns returned value from TPX command to variable POSX.
SPEED=5.75	Assigns value 5.75 to variable SPEED
INPUT=@IN[2]	Assigns logical value of input 2 to variable INPUT
V2=V1+V3*V4	Assigns the value of V1 plus V3 times V4 to the variable V2.
VAR="CAT"	Assign the string, CAT, to VAR

Variable values may be assigned to controller parameters such as GN or PR. Here, an equal is not used. For example:

PR V1	Assign V1 to PR command
SP VS*2000	Assign VS*2000 to SP command

Example - Using Variables for Joystick

The example below reads the voltage of an X-Y joystick and assigns it to variables VX and VY to drive the motors at proportional velocities, where

10 Volts = 8191 counts --> 3000 rpm = 200000 c/sec

Speed/Analog input = 200000/8191 = 24.4

#JOYSTICK	Label
JG 0,0	Set in Jog mode
BGXY	Begin Motion
#LOOP	Loop
VX=@AN[1]*24.4	Read joystick X
VY=@AN[2]*24.4	Read joystick Y
JG VX,VY	Jog at variable VX,VY
JP#LOOP	Repeat
EN	End

Internal Variables & Keywords

Internal variables allow motion or status parameters from SMC-2000 commands to be incorporated into programmable variables and expressions. Internal variables are designated by adding an underscore (_) prior to the SMC-2000 command. SMC-2000 commands which can be used as internal variables are listed in the Command Reference as "Used as an Operand".

Most SMC-2000 commands can be used as internal variables. Status commands such as Tell Position return actual values, whereas action commands such as GN or SP return the values in the SMC-2000 registers. The X,Y,Z or W or A,B,C,D,E,F,G,H for the SMC-2000-8, axis designation is required following the command.

Examples:

POSX=_TPX	Assigns value from Tell Position X to the variable POSX.
GAIN=_GNZ*2	Assigns value from GNZ multiplied by two to variable, GAIN.
JP #LOOP,_TEX>5	Jump to #LOOP if the position error of X is greater than 5
JP #ERROR,_TC=1	Jump to #ERROR if the error code equals 1.

Internal variables can be used in an expression and assigned to a programmable variable, but they cannot be assigned a value. For example: _GNX=2 is invalid.

The SMC-2000 also provides a few keywords which give access to internal variables that are not accessible by standard SMC-2000 commands.

Keyword	**Function**
_BGX or _BGY or _BGW	Motion Done if 1. Moving if 0.
_LFX or _LFY or _LFZ or_LFW	Forward Limit (equals 0 or 1)
_LRX or _LRY or _LRZ or LRW	Reverse Limit (equals 0 or 1)
TIME	Free-Running Real Time Clock* (off by 2.4% - Reset on power-on). Note: TIME does not use _.
_HMX or _HMY or _HMZ or HMW	Home Switch (equals 0 or 1)

Examples:

V1=_LFX	Assign V1 the logical state of the Forward Limit Switch on the X-axis
V3=TIME	Assign V3 the current value of the time clock
V4=_HMW	Assign V4 the logical state of the Home input on the W-axis

Example Program:

#TIMER	Timer
INITIME=TIME	Initialize time variable
PR50000;BGX	Begin move
AMX	After move
ELAPSED=TIME-INTIME	Compute elapsed time
EN	End program
#LIMSWI	Limit Switch Routine
JP #FORWARD,_LFX=0	Jump if Forward Limit
AMX	Wait for Motion Done
PR 1000;BGX;AMX	Move Away from Reverse Limit
JP #END	Exit
#FORWARD	Forward Label
PR -1000;BGX;AMX	Move Away from Forward Limit
#END	Exit
RE	Return to Main Program

Arrays

For storing and collecting numerical data, the SMC-2000 provides array space for 8000 elements in up to 30 arrays. Arrays can be used to capture real-time data, such as position, torque and analog input values. In the contouring mode, arrays are convenient for learning a position trajectory and later playing it back.

Defining Arrays

An array is defined by a name and number of entries using the DM command. The name can contain up to eight characters, starting with an uppercase alphabetic character.

The number of entries in the defined array is enclosed in [ ].

Up to 30 different arrays may be defined. The arrays are one dimensional.

Example:

DM POSX[7]	Defines an array named POSX with seven entries
DM SPEED[100]	Defines an array named speed with 100 entries
DM POSX[0]	Frees array space

Each array element has a numeric range of 4 bytes of integer (2^{31})followed by two bytes of fraction (+/- 2,147,483,647.9999).

Array space may be de-allocated using the DA command followed by the array name. DA*[0] de-allocates all the arrays.

Assignment of Array Entries

Like variables, each array element can be assigned a value. Assigned values can be numbers or returned values from instructions, functions and keywords.

Values are assigned to array entries using the equal sign. Assignments are made one element at a time by specifying the element number with the associated array name.

NOTE: Remember to define arrays using the DM command before assigning entry values.

Examples:

DM SPEED[10]	Dimension Speed Array
SPEED[1]=7650.2	Assigns the first element of the array, SPEED the value 7650.2
SPEED[1]=	Returns array element value
POSX[10]=_TPX	Assigns the 10th element of the array POSX the returned value from the tell position command.
CON[2]=@COS[POS]*2	Assigns the second element of the array CON the cosine of the variable POS multiplied by 2.
TIMER[1]=TIME	Assigns the first element of the array timer the returned value of the TIME keyword.

An array element number can also be a variable. This allows array entries to be assigned sequentially using a counter.

For example:

#A	Begin Program
COUNT=0;DM POS[10]	Initialize counter and define array
#LOOP	Begin loop
WT 10	Wait 10 msec
POS[COUNT]=_TPX	Record position into array element
POS[COUNT]=	Report position
COUNT=COUNT+1	Increment counter
JP #LOOP,COUNT<10	Loop until 10 elements have been stored
EN	End Program

The above example records 10 position values at a rate of one value per 10 msec. The values are stored in an array named POS. The variable, COUNT, is used to increment the array element counter. The above example can also be executed with the automatic data capture feature described below.

Arrays may be uploaded and downloaded using the QU and QD commands.

QU array[],start,end,comma

QD array[],start,end

where array is an array name such as A[].

Start is the first element of array (default=0)

End is the last element of array (default=last element)

Comma -- if comma is a 1, then the array elements are separated by a comma. If not a 1, then the elements are separated by a carriage return.

The file is terminated using <control>Z, <control>Q, <control>D or \.

Automatic Data Capture into Arrays

The SMC-2000 provides a special feature for automatic capture of data such as position, position error, inputs or torque. This is useful for teaching motion trajectories or observing system performance. Up to eight types of data can be captured and stored in eight arrays. The capture rate or time interval may be specified.

Commands used:

RA n[],m[],o[],p[]	Selects up to four arrays (eight arrays for SMC-2000-8) for data capture. The arrays must be defined with the DM command.
RD_TI,_TPX,_SCZ,_TSY	Selects the type of data to be recorded. See the table below for the various types of data. The order of data type is important and corresponds with the order of n,m,o,p arrays in the RA command. In this example, the _TI input data is stored in the first array selected by the RA command.
RC n,m	The RC command begins data collection. Sets data capture time interval where n is an integer between 1 and 8 and designates 2n msec between data. m is optional and specifies the number of elements to be captured. If m is not defined, the number of elements defaults to the smallest array defined by DM. n=0 stops recording.
RC? or V=_RC	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress

Data Types for Recording

_DEX	2nd encoder position (dual encoder)
_TPX	Encoder position
_TEX	Position error
_RPX	Commanded position
_RLX	Latched position
_TI	Inputs
_OP	Output
_TSX	Switches (only bit 0-4 valid)
_SCX	Stop code
_TBX	Status bits
_TTX	Torque (reports digital value +/-32703)

Note: X may be replaced by Y,Z or W for capturing data on other axes, or A,B,C,D,E,F,G,H for SMC-2000-8.

Example - Recording into An Array

During a position move, store the X and Y positions and position error every 2 msec.

#RECORD	Begin program
DM XPOS[300],YPOS[300]	Define X,Y position arrays
DM XERR[300],YERR[300]	Define X,Y error arrays
RA XPOS[],XERR[],YPOS[],YERR[]	Select arrays for capture
RD _TPX,_TEX,_TPY,_TEY	Select data types

PR 10000,20000	Specify move distance
RC1	Start recording now, at rate of 2 msec
BG XY	Begin motion
#A;JP #A,RC=1	Loop until done
MG "DONE"	Print message
EN	End program
#PLAY	Play back
N=0	Initial Counter
JP# DONE,N>300	Exit if done
N=	Print Counter
X POS[N]=	Print X position
Y POS[N]=	Print Y position
XERR[N]=	Print X error
YERR[N]=	Print Y error
N=N+1	Increment Counter
#DONE	Done

Input and Output of Data

Sending Messages

Messages may be sent to the bus using the message command, MG. This command sends specified text and numerical or string data from variables or arrays to the screen.

Text strings are specified in quotes and variable or array data is designated by the name of the variable or array. For formatting string variables, the {Sn} specifier is required where n is the number of characters, 1 through 6. Example:

```
MG STR {S3}
```

The above statement returns 3 characters of the string variable named STR.

Numeric data may be formatted using the {Fn.m} expression following the completed MG statement. {$n.m} formats data in HEX instead of decimal. Example:

```
MG "The Final Value is", RESULT {F5.2}
```

The above statement sends the message:

```
The Final Value is xxxxx.xx
```

The actual numerical value for the variable, RESULT, is substituted with the format of 5 digits to the left of the decimal and 2 to the right.

In addition to variables, functions and commands, responses can be used in the message command. For example:

```
MG "Analog input is", @AN[1]
MG "The Gain of X is", _GNX
```

The message command normally sends a carriage return and line feed following the statement. The carriage return and the line feed may be suppressed by sending {N} at the end of the statement. This is useful when a text string needs to surround a numeric value.

Example:

#A	Label
JG 50000;BGX;ASX	Jog, Begin, After Speed
MG "The Speed is", _TVX {F5.1} {N}	Message
MG "counts/sec"	Message
EN	End Program

When #A is executed, the above example will appear on the screen as:

The speed is 50000 counts/sec

The MG command can also be used to configure terminals. Here, any character can be sent by using {^n} where n is any integer between 1 and 255.

Example:

MG {^07} {^255}

sends the ASCII characters represented by 7 and 255 to the bus.

Summary of Message Functions:

MG	Message command
" "	Surrounds text string
{Fn.m}	Formats numeric values in decimal n digits to the right of the decimal point and m digits to the left
{$n.m}	Formats numeric values in hexadecimal
{^n}	Sends ASCII character specified by integer n
{N}	Suppresses carriage return/line feed
{Sn}	Sends the first n characters of a string variable, where n is 1 through 6.

Variables may also be sent to the screen using the variable= format. Variable Name= returns the variable value. For example, V1= , returns the value of the variable V1.

Example - Printing a Variable

#DISPLAY	Label
PR 1000	Position Command
BGX	Begin
AMX	After Motion
V1=_TPX	Assign Variable V1
V1=	Print V1

Input of Data

The IN command is used to prompt the user to input numeric or string data. The input data is assigned to the specified variable or array element.

A message prompt may be sent to the user by specifying the message characters in quotes.

Example:

```
#A
IN "Enter Length", LENX
EN
```

The above program sends the message:

```
Enter Length
```

to the PC screen and waits for the operator to enter a value. The operator enters the numeric value which is assigned to the variable, LENX. String variables with up to six characters may also be input using the {S} specifier. For example, IN "Enter X,Y or Z", V{S} specifies a string variable to be input.

Cut-to-Length Example

In this example, a length of material is to be advanced a specified distance. When the motion is complete, a cutting head is activated to cut the material. The length is variable, and the operator is prompted to input it in inches. Motion starts with a start button which is connected to input 1.

The load is coupled with a 2 pitch lead screw. A 2000 count/rev encoder is on the motor, resulting in a resolution of 4000 counts/inch. The program below uses the variable LEN, to length. The IN command is used to prompt the operator to enter the length, and the entered value is assigned to the variable LEN.

#BEGIN	Label
AC 800000	Acceleration
DC 800000	Deceleration
SP 5000	Speed
LEN=3.4	Initial length in inches
#CUT	Cut routine
AI1	Wait for start signal
IN "enter Length(IN)", LEN	Prompt operator for length in inches
PR LEN *4000	Specify position in counts
BGX	Begin motion to move material
AMX	Wait for motion done
SB1	Set output to cut
WT100;CB1	Wait 100 msec, then turn off cutter
JP #CUT	Repeat process
EN	End program

Operator Data Entry Mode

The Operator Data Entry Mode permits data to be entered at anytime and it is not interpreted as a standard SMC-2000 command. This mode may only be used when executing an applications program. Normal SMC-2000 commands such as ST or JG will not be responded to in this mode. The mode may be exited with the <\> or <ESCAPE> key. The Operator Data Entry Mode may be specified for either Port 1 or Port 2 or both.

Note: This is not used for high rate data transfer.

For Port 1:

Use the third field of the CI command to set the Data Mode. A 1 specifies Operator Data Mode, a 0 disables the Data Mode.

For Port 2:

Use the third field of the CC command to set the Data Mode. A 0 configures P2 as a general port for the Operator Data Mode.

In the Operator Data Mode, the SMC-2000 provides a buffer for receiving characters.

To decode characters in the Operator Data Mode, the SMC-2000 provides four special keywords for Port 1 (P1) and Port (P2).

Keyword	**Function**
P1CH or P2CH	Contains the last character received
P1ST or P2ST	Contains the received string
P1NM or P2NM	Contains the received number
P1CD or P2CD	-1 - mode disabled 0 - nothing received 1 -received character, but NOT <ENTER>. 2 -received string, but NOT a number 3 -received number

Note: The value of P1CD and P2CD returns to zero after the corresponding string or number is read.

These keywords may be used in an applications program to decode data. They may be used in conditional statements with logical operators.

Examples:

JP #LOOP,P2CD<>3	Checks to see if status code is 3 (number received
JP #P,P1CH="V"	Checks if last character received was a V
PR P2NM	Assigns received number to position
JS #XAXIS,P1ST="X"	Checks to see if received string is X

The SMC-2000 provides a special interrupt for communication. The interrupt is enabled using the CI command, where CI n,m,o

n=0	Don't interrupt Port 1
1	Interrupt on <ENTER> Port 1
2	Interrupt on any character Port 1
-1	Clear any characters in buffer

m=0	Don’t interrupt Port 2
1	Interrupt on <ENTER> Port 2
2	Interrupt on any character Port 2
-1	Clear any characters in buffer
o=0	Disable operator data mode for Port 1
1	Enable operator data mode for Port 1

Formatting Data

Returned numeric values may be formatted in decimal or hexadecimal* with a specified number of digits to the right and left of the decimal point using the PF command.

The Position Format (PF) command formats motion values such as those returned by the Tell Position (TP), Speed? (SP?) and Tell Error (TE) commands.

Position Format is specified by:

PF m.n

where m is the number of digits to the left of the decimal point (0 through 10) and n is the number of digits to the right of the decimal point (0 through 4) A negative sign for m specifies hexadecimal format.

Hex values are returned preceded by a $ and in 2's complement. Hex values should be input as signed 2's complement, where negative numbers have a negative sign. The default format is PF 10.0.

Examples:

:DP21	Define position
:TPX	Tell position
0000000021	Default format
:PF4	Change format to 4 places
:TPX	Tell position
0021	New format
:PF-4	Change to hexadecimal format
:TPX	Tell Position
$0015	Hexadecimal value

The following interrogation commands are affected by the PF command:

DP
ER
PA
PR
TE
TP

If the number of decimal places specified by PF is less than the actual value, a nine appears in all the decimal places.

Example:

:PF2	Format 2 places
:TPX	Tell position
99	Returns 99 if actual position is more than allowed format

The Variable Format (VF) command is used to format variables and array elements. The VF command is specified by:

VF m.n

where m is the number of digits to the left of the decimal point (0 through 10) and n is the number of digits to the right of the decimal point (0 through 4).

A negative sign for m specifies hexadecimal format. The default format for VF is VF 10.4

Hex values are returned preceded by a $ and in 2's complement.

:V1=10	Assign V1
:V1=	Return V1
0000000010.0000	Default format
:VF2.2	Change format
:V1=	Return V1
10.00	New format
:VF-2.2	Specify hex format
:V1=	Return V1
$0A.00	Hex value
:VF1	Change format
:V1=	Return V1
9	Overflow

The variable format also affects returned values from internal variables such as _GNX.

PF and VF commands are global format commands. Parameters may also be formatted locally by using the {Fn.m} or {$n.m} specification following the variable = . For example:

V1={F4.2}	Specifies the variable V1 to be returned in a format of 4 digits to left of decimal and 2 to the right.

F specifies decimal and $ specifies hexadecimal. n is the number of digits to the left of the decimal, and m is the number of digits to the right of the decimal. The local format is also used with the MG* command.

Examples:

:V1=10	Assign V1
:V1=	Return V1
0000000010.0000	Default Format
:V1={F4.2}	Specify local format
0010.00	New format
:V1={$4.2}	Specify hex format
$000A.00	Hex value
:V1="ALPHA"	Assign string "ALPHA" to V1
:V1={S4}	Specify string format first 4 characters

ALPH	

User Units

Variables and arithmetic operations make it easy to input data in desired user units such as inches or RPM.

For example, an operator can be prompted to input a number in revolutions. The input number is converted into counts by multiplying it by the number of counts/revolution.

The SMC-2000 position parameters such as PR, PA and VP have units of quadrature counts. Speed parameters such as SP, JG and VS have units of counts/sec. Acceleration parameters such as AC, DC, VA and VD have units of counts/sec^2. All input parameters must be converted into these units.

Example:

#RUN	Label
IN "ENTER # OF REVOLUTIONS",N1	Prompt for revs
PR N1*2000	Convert to counts
IN "ENTER SPEED IN RPM",S1	Prompt for RPMs
SP S1*2000/60	Convert to counts/sec
IN "ENTER ACCEL IN RAD/SEC2",A1	Prompt for ACCEL
AC A1*2000/(2*3.14)	Convert to counts/sec2
BG	Begin motion
EN	End program

Programmable I/O

Digital Outputs

The SMC-2000-4 has an 8-bit uncommitted output port (I/O 1, pins 10-17) for controlling external events. The SMC-2000-8 has an additional eight output bits available at (I/O 2, pins 10-17). Each bit on the output port may be set and cleared with the software instructions SB (Set Bit) and CB(Clear Bit), or OB (define output bit).

For example:

Instruction	Function
SB6	Sets bit 6 of output port
CB4	Clears bit 4 of output port

The Output Bit (OB) instruction is useful for setting or clearing outputs depending on the value of a variable, array, input or expression. Any non-zero value results in a set bit.

Instruction	Function
OB1, POS	Set Output 1 if the variable POS is non-zero. Clear Output 1 if POS equals 0.
OB 2, @IN [1]	Set Output 2 if Input 1 is high. If Input 1 is low, clear Output 2.

OB 3, @IN [1]&@IN [2]	Set Output 3 only if Input 1 and Input 2 are high.
OB 4, COUNT [1]	Set Output 4 if element 1 in the array COUNT is non-zero.

The output port may also be written to as an 8-bit word using the instruction

OP (Output Port). This instruction allows a single command to define the state of the entire 8-bit output port, where 2^0 is output 1, 2^1 is output 2 and so on. A 1 designates that output is on.

For example:

Instruction	Function
OP6	Sets outputs 2 and 3 of output port to high. All other bits are 0. $(2^1 + 2^2 = 6)$
OP0	Clears all bits of output port to zero
OP 255	Sets all bits of output port to one. $(2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7)$

The output port is useful for firing relays or controlling external switches and events during a motion sequence.

Example - Turn on output after move

#OUTPUT	Label
PR 2000	Position Command
BG	Begin
AM	After move
SB1	Set Output 1
WT 1000	Wait 1000 msec
CB1	Clear Output 1
EN	End

Digital Inputs

The SMC-2000 has eight digital inputs (I/O 1, pins 18-25) for controlling motion by local switches. The @IN[n] function returns the logic level of the specified input 1 through 8. For example, a Jump on Condition instruction can be used to execute a sequence if a high condition is noted on an input 3. To halt program execution, the After Input (AI) instruction waits until the specified input has occurred.

Example:

JP #A,@IN[1]=0	Jump to A if input 1 is low
JP #B,@IN[2]=1	Jump to B if input 2 is high
AI 7	Wait until input 7 is high
AI -6	Wait until input 6 is low

The SMC-2000-8 provides an additional 16 inputs at I/O 2, pins 18-25, and pins 1-7 , and on D2, pin 24. These are returned with the function @IN[n] where n=9 through 24. Inputs 17 through 24 are TTL level. Inputs 9 through 16 are isolated.

Example - Start Motion on Switch

Motor X must turn at 4000 counts/sec when the user flips a panel switch to on. When panel switch is turned to off position, motor X must stop turning.

Solution: Connect panel switch to input 1 of SMC-2000. High on input 1 means switch is in on position.

Instruction	Function
#S;JG 4000	Set speed
AI 1;BGX	Begin after input 1 goes high
AI -1;STX	Stop after input 1 goes low
AMX;JP #S	After motion, repeat
EN;	

Input Interrupt Function

The SMC-2000 provides an input interrupt function which causes the program to automatically execute the instructions following the #ININT label. This function is enabled using the II m,n,o command. The m specifies the beginning input and n specifies the final input in the range. The parameter o is an interrupt mask. If m and n are unused, o contains a number with the mask. A 1 designates that input to be enabled for an interrupt, where 2^0 is bit 1, 2^1 is bit 2 and so on. For example, II,,5 enables inputs 1 and 3 ($2^0 + 2^2 = 5$).

A low input on any of the specified inputs will cause automatic execution of the #ININT subroutine. The Return from Interrupt (RI) command is used to return from this subroutine to the place in the program where the interrupt had occurred. If it is desired to return to somewhere else in the program after the execution of the #ININT subroutine, the Zero Stack (ZS) command is used followed by unconditional jump statements.

Important: Use the RI instruction (not EN) to return from the #ININT subroutine.

Examples - Input Interrupt

#A	Label #A
II 1	Enable input 1 for interrupt function
JG 30000,-20000	Set speeds on X and Y axes
BG XY	Begin motion on X and Y axes
#B	Label #B
TP XY	Report X and Y axes positions
WT 1000	Wait 1000 milliseconds
JP #B	Jump to #B
EN	End of program
#ININT	Interrupt subroutine
MG "Interrupt has occurred"	Displays the message
ST XY	Stops motion on X and Y axes
#LOOP;JP #LOOP,@IN[1]=0	Loop until Interrupt cleared
JG 15000,10000	Specify new speeds
WT 300	Wait 300 milliseconds
BG XY	Begin motion on X and Y axes
RI	Return from Interrupt subroutine

Analog Inputs

The SMC-2000 provides seven analog inputs. The value of these inputs in volts may be read using the @AN[n] function where n is the analog input 1 through 7. The resolution of the Analog-to-Digital conversion is 14 bits. Analog inputs are useful for reading special sensors such as temperature, tension or pressure.

The following examples show programs which cause the motor to follow an analog signal. The first example is a point-to-point move. The second example shows a continuous move.

Example - Position Follower (Point-to-Point)

Objective - The motor must follow an analog signal. When the analog signal varies by 10V, motor must move 10000 counts.

Method: Read the analog input and command X to move to that point.

Instruction	Interpretation
#POINTS	Label
SP 7000	Speed
AC 80000;DC 80000	Acceleration
#LOOP	Label
COMPP=@AN[1]*1000	Read analog input, and compute position
PA COMPP	Command position
BGX	Start motion
AMX	After completion
JP #LOOP	Repeat
EN	End

Example - Position Follower (Continuous Move)

Method: Read the analog input, compute the commanded position and the position error. Command the motor to run at a speed in proportions to the position error.

Instruction	Interpretation
#CONT	Label
AC 80000;DC 80000	Acceleration rate
JG 0	Start job mode
BGX	Start motion
#LOOP	
COMPP=@AN[1]*1000	Compute desired position
VE=COMPP-_TPX	Find position error
PVEL=VE*20	Compute velocity
JG PVEL	Change velocity
JP #LOOP	Repeat
EN	End

Example Applications

Wire Cutter

An operator activates a start switch. This causes a motor to advance the wire a distance of 10". When the motion stops, the controller generates an output signal which activates the cutter. Allowing 100 ms for the cutting completes the cycle.

Suppose that the motor drives the wire by a roller with a 2" diameter. Also assume that the encoder resolution is 1000 lines per revolution. Since the circumference of the roller equals 2π inches, and it corresponds to 4000 quadrature, one inch of travel equals:

$$4000/2\pi = 637 \text{ count/inch}$$

This implies that a distance of 10 inches equals 6370 counts, and a slew speed of 5 inches per second, for example, equals 3185 count/sec.

The input signal may be applied to I1, for example, and the output signal is chosen as output 1. The motor velocity profile and the related input and output signals are shown in Fig. 7.1.

The program starts at a state that we define as #A. Here the controller waits for the input pulse on I1. As soon as the pulse is given, the controller starts the forward motion.

Upon completion of the forward move, the controller outputs a pulse for 20 ms and then waits an additional 80 ms before returning to #A for a new cycle.

Instruction	Function
#A	Label
AI1	Wait for input 1
PR 6370	Distance
SP 3185	Speed
BGX	Start Motion
AMX	After motion is complete
SB1	Set output bit 1
WT 20	Wait 20 ms
CB1	Clear output bit 1
WT 80	Wait 80 ms
JP #A	Repeat the process

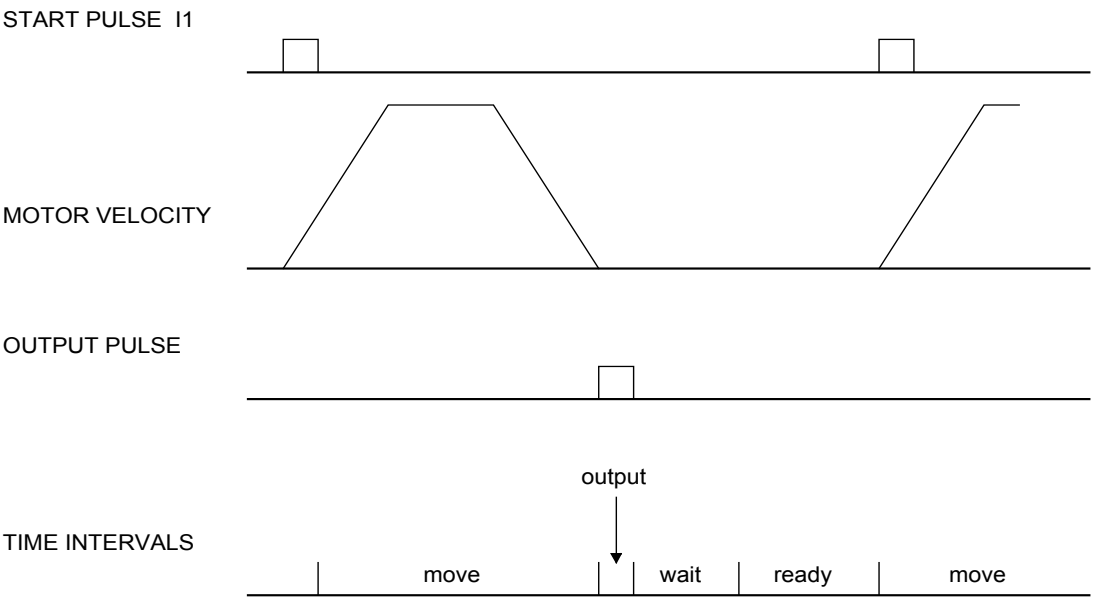


Figure 7.1 - Motor Velocity and the Associated Input/Output signals

X-Y Table Controller

An X-Y-Z system must cut the pattern shown in Fig. 7.2. The X-Y table moves the plate while the Z-axis raises and lowers the cutting tool.

The solid curves in Fig. 7.2 indicate sections where cutting takes place. Those must be performed at a feed rate of 1 inch per second. The dashed line corresponds to non-cutting moves and should be performed at 5 inch per second. The acceleration rate is 0.1 g.

The motion starts at point A, with the Z-axis raised. An X-Y motion to point B is followed by lowering the Z-axis and performing a cut along the circle. Once the circular motion is completed, the Z-axis is raised, and the motion continues to point C, etc.

Assume that all of the 3 axes are driven by lead screws with 10 turns-per-inch pitch. Also assume encoder resolution of 1000 lines per revolution. This results in the relationship:

1 inch = 40,000 counts

and the speeds of

1 in/sec = 40,000 count/sec

5 in/sec = 200,000 count/sec

an acceleration rate of 0.1g equals

$0.1g = 38.6\ in/s^2 = 1{,}544{,}000\ count/s^2$

Note that the circular path has a radius of 2" or 80000 counts, and the motion starts at the angle of 270° and traverses 360° in the CW (negative) direction. Such a path is specified with the instruction

CR 80000,270,-360

Further assume that the Z must move 2" at a linear speed of 2" per second. The required motion is performed by the following instructions:

Instruction	Function
#A	Label
VM XY	Circular interpolation for XY
VP 160000,160000	Positions
VE	End Vector Motion
VS 200000	Vector Speed
VA 1544000	Vector Acceleration
BGS	Start Motion
AMS	When motion is complete
PR,,-80000	Move Z down
SP,,80000	Z speed
BGZ	Start Z motion
AMZ	Wait for completion of Z motion
CR 80000,270,-360	Circle
VE	
VS 40000	Feed rate
BGS	Start circular move
AMS	Wait for completion
PR,,80000	Move Z up
BGZ	Start Z move
AMZ	Wait for Z completion
PR -21600	Move X
SP 20000	Speed X
BGX	Start X
AMX	Wait for X completion
PR,,-80000	Lower Z
BGZ	
AMZ	
CR 80000,270,-360	Z second circle move
VE	
VS 40000	
BGS	
AMS	
PR,,80000	Raise Z
BGZ	
AMZ	
VP -37600,-16000	Return XY to start
VE	
VS 200000	

BGS	
AMS	
EN	

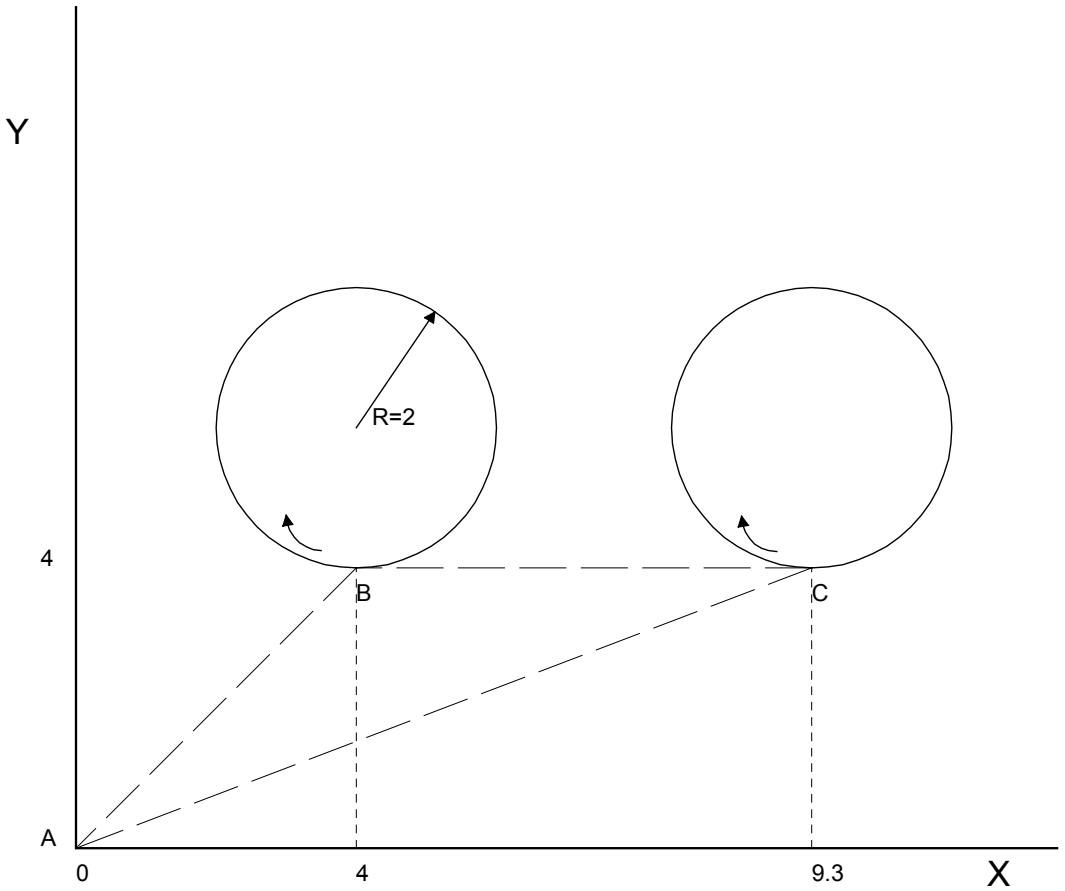


Figure 7.2 -The Required Path. All dimensions in inches.

Speed Control by Joystick

The speed of a motor is controlled by a joystick. The joystick produces a signal in the range between -10V and +10V. The objective is to drive the motor at a speed proportional to the input voltage.

Assume that a full voltage of 10 Volts must produce a motor speed of 3000 rpm with an encoder resolution of 1000 lines or 4000 count/rev. This speed equals:

3000 rpm = 50 rev/sec = 200000 count/sec

The program reads the input voltage periodically and assigns its value to the variable VIN. To get a speed of 200,000 ct/sec for 10 volts, we select the speed as

Speed = 20000 x VIN

The corresponding velocity for the motor is assigned to the VEL variable.

Instruction	Function
#A	Label
JG0	Set motor in jog mode speed zero
BGX	Start motion
#B	Label
VIN=@AN[1]	Read analog input
VEL=VIN*20000	Compute the desired velocity
JG VEL	Change the jog speed
JP #B	Repeat the process
EN	End

Position Control by Joystick

This system requires the position of the motor to be proportional to the joystick angle. Furthermore, the ratio between the two positions must be programmable. For example, if the control ratio is 5:1, it implies that when the joystick voltage is 5 Volts, corresponding to 4095 counts, the required motor position must be 20475 counts. The variable V3 changes the position ratio.

Instruction	Function
#A	Label
V3=5	Initial position ratio
DP0	Define the starting position
JG0	Set motor in jog mode as zero
BGX	Start
#B	
V1=@AN[1]	Read analog input
V2=V1*V3	Compute the desired position
V4=V2-_TPX-_TEX	Find the following error
V5=V4*20	Compute a proportional speed
JG V5	Change the speed
JP #B	Repeat the process
EN	End

Backlash Compensation by Dual-Loop

This design example addresses the basic problems of backlash in motion control systems. The objective is to control the position of a linear slide precisely. The slide is to be controlled by a rotary motor, which is coupled to the slide by a leadscrew. Such a leadscrew has a backlash of 4 micron, and the required position accuracy is for 0.5 micron.

The basic dilemma is where to mount the sensor. If you use a rotary sensor, you get a 4-micron backlash error. On the other hand, if you use a linear encoder, the backlash in the feedback loop will cause oscillations due to instability.

An alternative approach is the dual-loop, where we use two sensors, rotary and linear. The rotary sensor assures stability (because the position loop is closed before the backlash) whereas the linear sensor provides accurate load position information. The operation principle is to drive the motor to a given rotary position near the final

point. Once there, the load position is read to find the position error and the controller commands the motor to move to a new rotary position which eliminates the position error.

Since the required accuracy is 0.5 micron, the resolution of the linear sensor should preferably be twice finer. A linear sensor with a resolution of 0.25 micron allows a position error of +/-2 counts.

The dual-loop approach requires the resolution of the rotary sensor to be equal or better than that of the linear system. Assuming that the pitch of the lead screw is 2.5mm (approximately 10 turns per inch), a rotary encoder of 2500 lines per turn or 10,000 count per revolution results in a rotary resolution of 0.25 micron. This results in equal resolution on both linear and rotary sensors.

To illustrate the control method, assume that the rotary encoder is used as a feedback for the X-axis, and that the linear sensor is read and stored in the variable LINPOS. Further assume that at the start, both the position of X and the value of LINPOS are equal to zero. Now assume that the objective is to move the linear load to the position of 1000.

The first step is to command the X motor to move to the rotary position of 1000. Once it arrives we check the position of the load. If, for example, the load position is 980 counts, it implies that a correction of 20 counts must be made. However, when the X-axis is commanded to be at the position of 1000, suppose that the actual position is only 995, implying that X has a position error of 5 counts, which will be eliminated once the motor settles. This implies that the correction needs to be only 15 counts, since 5 counts out of the 20 would be corrected by the X-axis. Accordingly, the motion correction should be:

Correction = Load Position Error - Rotary Position Error

The correction can be performed a few times until the error drops below +/-2 counts. Often, this is performed in one correction cycle.

Example motion program:

Instruction	Function
#A	Label
DP0	Define starting positions as zero
LINPOS=0	
PR 1000	Required distance
BGX	Start motion
#B	
AMX	Wait for completion
WT 50	Wait 50 msec
LIN POS = _DEX	Read linear position
ER=1000-LINPOS-_TEX	Find the correction
JP #C,@ABS[ER]<2	Exit if error is small
PR ER	Command correction
BGX	
JP #B	Repeat the process
#C	
EN	

Error Handling

Introduction

The SMC-2000 provides several hardware and software features to check for error conditions and to inhibit the motor on error. These features help protect the various system components from damage.

Warning: Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machine. Since the SMC-2000 is an integral part of the machine, the engineer should design his overall system with protection against a possible component failure on the SMC-2000. Yaskawa shall not be liable or responsible for any incidental or consequential damages.

Hardware Protection

The SMC-2000 includes hardware input and output protection lines for various error and mechanical limit conditions. These include:

Output Protection Lines

Amp Enable - This signal goes low when the motor off command is given (MO), or when off-on-error condition is enabled (OE1) and, either the position error exceeds the value specified by the Error Limit (ER) or the abort command (AB) is given. Each axis amplifier has separate amplifier enable lines. This signal also goes low when the watch-dog timer is activated, or upon reset.

Input Protection Lines

Abort - A low input stops motion instantly without a controlled deceleration. Also aborts motion program.

Forward Limit Switch - Low input inhibits motion in forward direction. Also causes execution of limit switch subroutine, #LIMSWI, if the motor for that axis is moving in a forward direction.

Reverse Limit Switch - Low input inhibits motion in reverse direction. Also causes execution of limit switch subroutine, #LIMSWI, if the motor for that axis is moving in a reverse direction.

Note: The CN command can be used to change the polarity of the limit switches.

Software Protection

The SMC-2000 provides a programmable error limit. The error limit can be set for any number between 0 and 32767 using the ER n command. The default value for ER is 16384, and a value of 0 disables error monitoring.

Example:

ER 200,300,400,500	Set X-axis error limit for 200, Y-axis error limit to 300, Z-axis error limit to 400 counts, W-axis error limit to 500 counts
ER,1,,10	Set Y-axis error limit to 1 count, set W-axis error limit to 10 counts.

The units of the error limit are quadrature counts. The error is the difference between the command position and actual encoder position. If the absolute value of the error exceeds the value specified by ER, the SMC-2000 will generate several signals to warn the host system of the error condition. These signals include:

Signal or Function	State if Error Occurs
#POSERR	Jumps to automatic excess position error subroutine
Error Light	Turns on
OE Function	Shuts motor off if OE1
AEN Output Line	Goes low

The Jump on Condition statement is useful for branching on a given error within a program. The position error of X,Y,Z and W can be monitored during execution using the TE command.

Programmable Position Limits

The SMC-2000 provides programmable forward and reverse position limits. These are set by the BL and FL software commands. Once a position limit is specified, the SMC-2000 will not accept position commands beyond the limit. Motion beyond the limit is also prevented.

Example:

DP0,0,0	Define Position
BL -2000,-4000,-8000	Set Reverse position limit
FL 2000,4000,8000	Set Forward position limit
JG 2000,2000,2000	Jog
BG XYZ	Begin

(motion stops at forward limits)

Off-On-Error

The software command, Off-on-Error (OE1), turns the motor off when the position error exceeds the limit set by the ER command. To activate the OE function for each axis, specify 1 for X,Y,Z and W axis. 0 disables off-on-error.

Examples:

OE 1,1,1,1	Enable off-on-error for X,Y,Z and W
OE 0,1,0,1	Enable off-on-error for Y and W axes and disable off-on-error for W and Z axes

Automatic Error Routine

The #POSERR label causes the statements following to be automatically executed if error on any axis exceeds the error limit specified by ER. The error routine must be closed with the RE command. The RE command returns from the error subroutine to the main program.

NOTE: The Error Subroutine will be entered again unless the error condition is gone.

Example:

#A;JP #A;EN	"Dummy" program
#POSERR	Start error routine on error
MG "error"	Send message
SB 1	Fire relay
STX	Stop motor
AMX	After motor stops
SHX	Servo motor here to clear error
RE	Return to main program

Note: An applications program must be executing for the #POSERR routine to function.

Limit Switch Routine

The SMC-2000 provides forward and reverse limit switches which inhibit motion in the respective direction. There is also a special label for automatic execution of a limit switch subroutine. The #LIMSWI label specifies the start of the limit switch subroutine. This label causes the statements following to be automatically executed if any limit switch is activated and that axis motor is moving in that direction. The RE command ends the subroutine.

The state of the forward and reverse limit switches may also be tested during the jump-on-condition statement. The _LR condition specifies the reverse limit and _LF specifies the forward limit. X,Y,Z, or W following LR or LF specifies the axis.

Limit Switch Example:

#A;JP #A;EN	Dummy Program
#LIMSWI	Limit Switch Utility
V1=_LFX	Check if forward limit
V2=_LRX	Check if reverse limit
JP#LF,V1=0	Jump to #LF if forward
JP#LR,V2=0	Jump to #RF if reverse
JP#END	Jump to end
#LF	#LF
MG "FORWARD LIMIT"	Send message
STX;AMX	Stop motion

PR-1000;BGX;AMX	Move in reverse
JP#END	End
#LR	#LR
MG "REVERSE LIMIT"	Send message
STX;AMX	Stop motion
PR1000;BGX;AMX	Move forward
#END	End
RE	Return to main program

Note: An application program must be executing for #LIMSWI to function.

Chapter 9 Troubleshooting

Overview

The following discussion may help you get your system running if a problem is encountered.

Potential problems have been divided into groups as follows:

1. Installation
2. Communication
3. Stability and Compensation
4. Operation

The various symptoms along with the cause and the remedy are described in the following tables.

Installation

Symptom	Cause	Remedy
Motor runs away when connected to amplifier with no additional inputs.	Amplifier offset too large.	Adjust amplifier offset
Same as above, but offset adjustment does not stop the motor.	Damaged amplifier.	Replace amplifier.
Controller does not read changes in encoder position.	Wrong encoder connections.	Check encoder wiring.
Same as above	Bad encoder	Check the encoder signals. Replace encoder if necessary.
Same as above	Bad controller	Connect the encoder to different axis input. If it works, controller failure. Repair or replace.

Communication

Symptom	Cause	Remedy
Using DMCWIN, DMCDOS, or WSDK cannot communicate with the controller.	Plug and Play installation did not proceed properly	Check first that Dmc1802.INF was used to install the controller. Next check the controller registry to see if the controller was automatically added and an address selected.

Stability

Symptom	Cause	Remedy
Motor runs away when the loop is closed.	Wrong feedback polarity. (Positive Feedback)	Invert the polarity of the loop by inverting the motor leads (brush type) or the encoder (channel A+, B+ if single ended; channel A+, A- and B+, B- if differential)
Motor oscillates.	Too high gain or too little damping.	Decrease KI and KP. Increase KD.

Operation

Symptom	Cause	Remedy
Controller rejects command. Responded with a ?	Anything.	Interrogate the cause with TC or TC1.
Motor does not start or complete a move.	Noise on limit switches stops the motor. Noise on the abort line aborts the motion.	To check the cause, interrogate the stop code (SC). If caused by limit switch or abort line noise, reduce noise.
During a periodic operation, motor drifts slowly.	Encoder noise	Interrogate the position periodically. If controller states that the position is the same at different locations it implies encoder noise. Also use a scope to see the noise. Reduce noise. Use differential encoder inputs.
Same as above.	Programming error.	Avoid resetting position error at end of move with SH command.

THIS PAGE LEFT BLANK INTENTIONALLY

THIS PAGE LEFT BLANK INTENTIONALLY

Theory of Operation

Overview

The following discussion covers the operation of motion control systems. A typical motion control system consists of the elements shown in Fig 10.1.

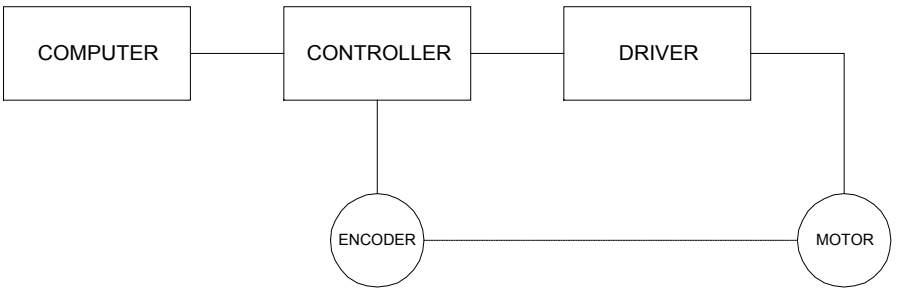


Figure 10.1 - Elements of Servo Systems

The operation of such a system can be divided into three levels, as illustrated in Fig. 10.2. The levels are:

1. Closing the Loop
2. Motion Profiling
3. Motion Programming

The first level, the closing of the loop, assures that the motor follows the commanded position. Closing the position loop using a sensor does this. The operation at the basic level of closing the loop involves the subjects of modeling, analysis, and design. These subjects will be covered in the following discussions.

The motion profiling is the generation of the desired position function. this function, R(t), describes where the motor should be at every sampling period. Note that the profiling and the closing of the loop are independent functions. The profiling function determines where the motor should be and the closing of the loop forces the motor to follow the commanded position

The highest level of control is the motion program. This can be stored in the host computer or in the controller. This program describes the tasks in terms of the motors that need to be controlled, the distances and the speed.

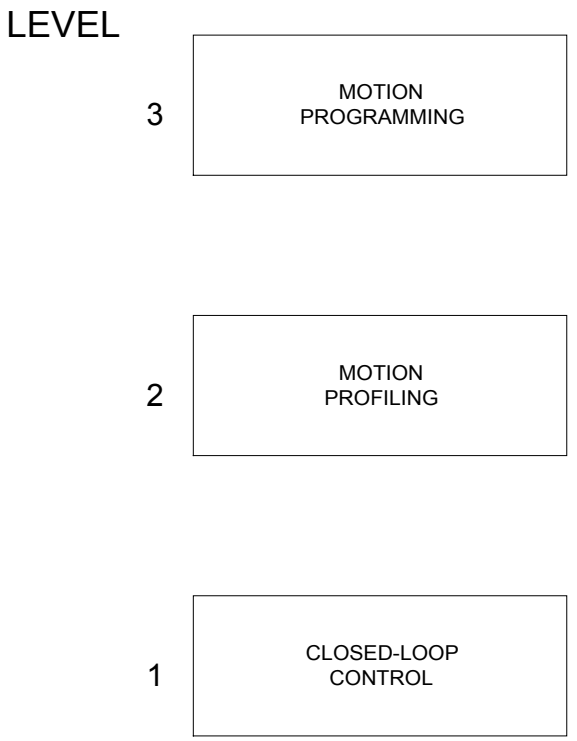


Figure 10.2 - Levels of Control Functions

The three levels of control may be viewed as different levels of management. The top manager, the motion program, may specify the following instruction, for example.

PR 6000,4000
SP 20000,20000
AC 200000,300000
BG X
AD 2000
BG Y
EN

This program corresponds to the velocity profiles shown in Fig. 10.3. Note that the profiled positions show where the motors must be at any instant of time.

Finally, it remains up to the servo system to verify that the motor follows the profiled position by closing the servo loop.

The operation of the servo system is done in two manners. First, it is explained qualitatively, in the following section. Later, the explanation is repeated using analytical tools for those who are more theoretically inclined.

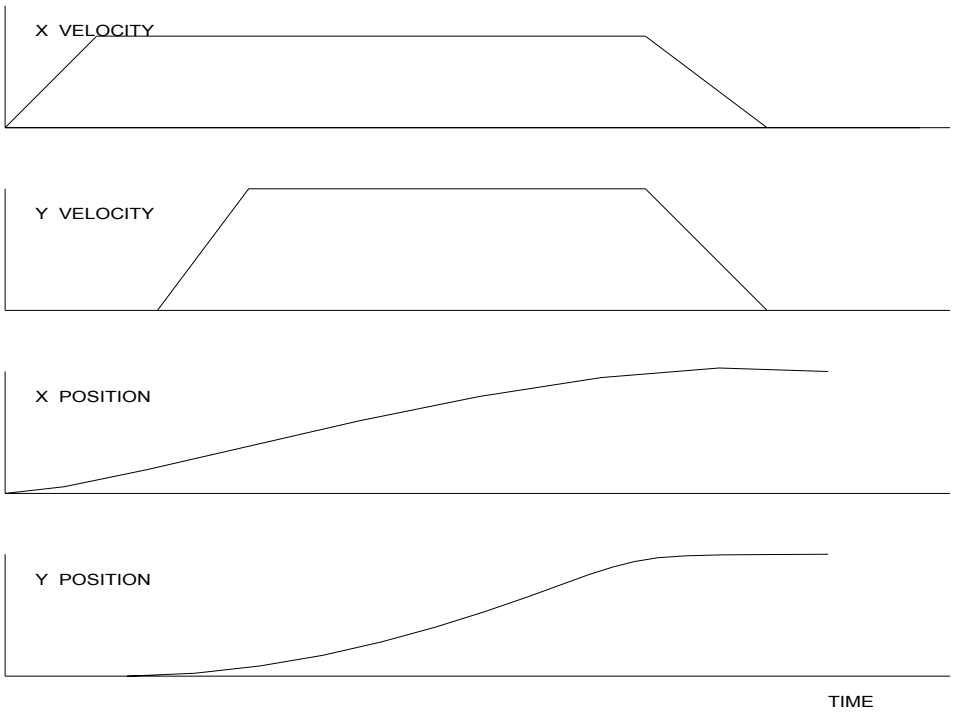


Figure 10.3 - Velocity and Position Profiles

Operation of Closed-Loop Systems

To understand the operation of a servo system, we may compare it to a familiar closed-loop operation, adjusting the water temperature in the shower. One control objective is to keep the temperature at a comfortable level, say 90 degrees F. To achieve that, our skin serves as a temperature sensor and reports to the brain (controller). The brain compares the actual temperature, which is called the feedback signal, with the desired level of 90 degrees F. The difference between the two levels is called the error signal. If the feedback temperature is too low, the error is positive, and it triggers an action which raises the water temperature until the temperature error is reduced sufficiently.

The closing of the servo loop is very similar. Suppose that we want the motor position to be at 90 degrees. A position sensor, often an encoder, measures the motor position and the position feedback is sent to the controller. Like the brain, the controller determines the position error, which is the difference between the commanded position of 90 degrees and the position feedback. The controller then outputs a signal that is proportional to the position error. This signal produces a proportional current in the motor, which causes a motion until the error is reduced. Once the error becomes small, the resulting current will be too small to overcome the friction, causing the motor to stop.

The analogy between adjusting the water temperature and closing the position loop carries further. We have all learned the hard way, that the hot water faucet should be turned at the "right" rate. If you turn it too slowly, the temperature response will be slow, causing discomfort. Such a slow reaction is called overdamped response.

The results may be worse if we turn the faucet too fast. The overreaction results in temperature oscillations. When the response of the system oscillates, we say that the system is unstable. Clearly, unstable responses are bad when we want a constant level.

What causes the oscillations? The basic cause for the instability is a combination of delayed reaction and high gain. In the case of the temperature control, the delay is due to the water flowing in the pipes. When the human reaction is too strong, the response becomes unstable.

Servo systems also become unstable if their gain is too high. The delay in servo systems is between the application of the current and its effect on the position. Note that the current must be applied long enough to cause a significant effect on the velocity, and the velocity change must last long enough to cause a position change. This delay, when coupled with high gain, causes instability.

This motion controller includes a special filter that is designed to help the stability and accuracy. Typically, such a filter produces, in addition to the proportional gain, damping and integrator. The combination of the three functions is referred to as a PID filter.

The filter parameters are represented by the three constants KP, KI and KD, which correspond to the proportional, integral and derivative term respectively.

The damping element of the filter acts as a predictor, thereby reducing the delay associated with the motor response.

The integrator function, represented by the parameter KI, improves the system accuracy. With the KI parameter, the motor does not stop until it reaches the desired position exactly, regardless of the level of friction or opposing torque.

The integrator also reduces the system stability. Therefore, it can be used only when the loop is stable and has a high gain.

The output of the filter is applied to a digital-to-analog converter (DAC). The resulting output signal in the range between +10 and -10 Volts is then applied to the amplifier and the motor.

The motor position, whether rotary or linear is measured by a sensor. The resulting signal, called position feedback, is returned to the controller for closing the loop.

The following section describes the operation in a detailed mathematical form, including modeling, analysis and design.

System Modeling

The elements of a servo system include the motor, driver, encoder and the controller. These elements are shown in Fig. 10.4. The mathematical model of the various components is given below.

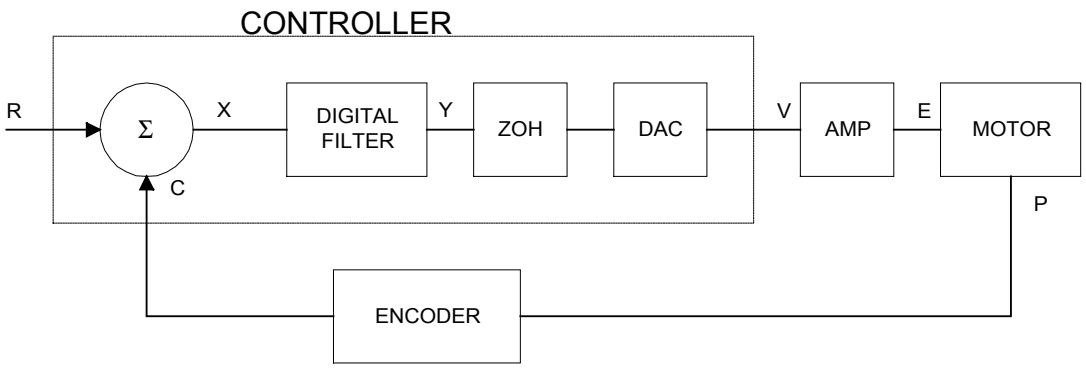


Figure 10.4 - Functional Elements of a Motion Control System

Motor-Amplifier

The motor amplifier may be configured in two modes:

1. Current Drive
2. Velocity Loop

The operation and modeling in the two modes is as follows:

Current Drive

The current drive generates a current I, which is proportional to the input voltage, V, with a gain of Ka, a torque constant of K_t, and inertia J. The resulting transfer function in this case is

$$P/V = K_a K_t / Js^2$$

For example, a current amplifier with K_a = 2 A/V with the motor described by the previous example will have the transfer function:

$$P/V = 1000/s^2 \qquad [rad/V]$$

If the motor is a DC Brushless motor, an amplifier that performs the commutation drives it. The combined transfer function of motor amplifier combination is the same as that of a similar brush motor, as described by the previous equations.

Velocity Loop

The motor driver system may include a velocity loop where the motor velocity is sensed by a tachometer and is fed back to the amplifier. Such a system is illustrated in Fig. 10.5. Note that the transfer function between the input voltage V and the velocity ω is:

$$\omega /V = [K_a K_t/Js]/[1+K_a K_t K_g/Js] = 1/[K_g(sT_1+1)]$$

where the velocity time constant, T1, equals

$$T1 = J/K_a K_t K_g$$

This leads to the transfer function

$$P/V = 1/[K_g s(sT1+1)]$$

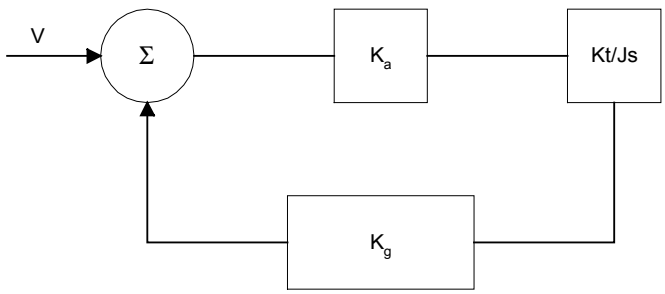


Figure 10.5 - Elements of velocity loops

The resulting functions derived above are illustrated by the block diagram of Fig. 10.6.

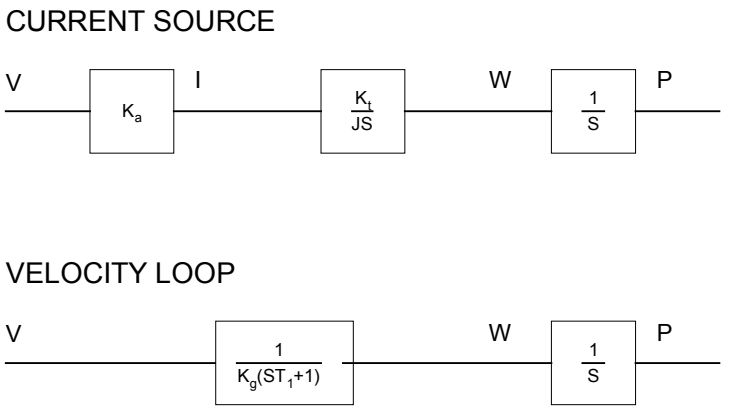


Figure 10.6 - Mathematical model of the motor and amplifier in two operational modes

Encoder

The encoder generates N pulses per revolution. It outputs two signals, Channel A and B, which are in quadrature. Due to the quadrature relationship between the encoder channels, the position resolution is increased to 4N quadrature counts/rev.

The model of the encoder can be represented by a gain of

$K_f = 4N/2\pi$ [count/rad]

For example, a 1000 lines/rev encoder is modeled as

$K_f = 638$

DAC

The DAC or D-to-A converter converts a 14-bit number to an analog voltage. The input range of the numbers is 16384 and the output voltage range is +/-10V or 20V. Therefore, the effective gain of the DAC is

K= 20/16384 = 0.0012 [V/count]

Digital Filter

The digital filter has a transfer function of D(z) = K(z-A)/z + Cz/z-1 and a sampling time of T.

The filter parameters, K, A and C are selected by the instructions KP, KD, KI or by GN, ZR and KI, respectively. The relationship between the filter coefficients and the instructions are:

K = KP + KD	or K = GN
A = KD/(KP + KD)	or A = ZR

C = KI/8	

This filter includes a lead compensation and an integrator. It is equivalent to a continuous PID filter with a transfer function G(s).

$G(s) = P + sD + I/s$

$P = K(1-A) = KP$

$D = T.K.A = T.KD$

$I = C/T = KI/8T$

For example, if the filter parameters are KP = 4

KD = 36

KI = 2

T = 0.001 s

the digital filter coefficients are

K = 40

A = 0.9

C = 0.25

and the equivalent continuous filter, G(s), is

$G(s) = 4 + 0.036s + 250/s$

ZOH

The ZOH, or zero-order-hold, represents the effect of the sampling process, where the motor command is updated once per sampling period. The effect of the ZOH can be modeled by the transfer function

$H(s) = 1/(1+sT/2)$

If the sampling period is T = 0.001, for example, H(s) becomes:

$H(s) = 2000/(s+2000)$

However, in most applications, H(s) may be approximated as one.

This completes the modeling of the system elements. Next, we discuss the system analysis.

System Analysis

To analyze the system, we start with a block diagram model of the system elements. The analysis procedure is illustrated in terms of the following example.

Consider a position control system with the SMC-2000 controller and the following parameters:

$K_t = 0.1$	Nm/A	Torque constant
$J = 2.10^{-4}$	$kg.m^2$	System moment of inertia
R = 2	Ω	Motor resistance
$K_a = 4$	Amp/Volt	Current amplifier gain
KP = 12.5		Digital filter gain
KD = 245		Digital filter zero

KI = 0		No integrator
N = 500	Counts/rev	Encoder line density
T = 1	ms	Sample period

The transfer function of the system elements are:

Motor

$M(s) = P/I = Kt/Js2 = 500/s^2$ [rad/A]

Amp

$K_a = 4$ [Amp/V]

DAC

$K_d = 0.0012$ [V/count]

Encoder

$K_f = 4N/2\pi = 318$ [count/rad]

ZOH

$2000/(s+2000)$

Digital Filter

KP = 12.5, KD = 245, T = 0.001

Therefore,

$D(z) = 12.5 + 245 (1-z-1)$

Accordingly, the coefficients of the continuous filter are:

P = 12.5

D = 0.245

The filter equation may be written in the continuous equivalent form:

$G(s) = 12.5 + 0.245s = 0.245(s+51)$

The system elements are shown in Fig. 10.7.

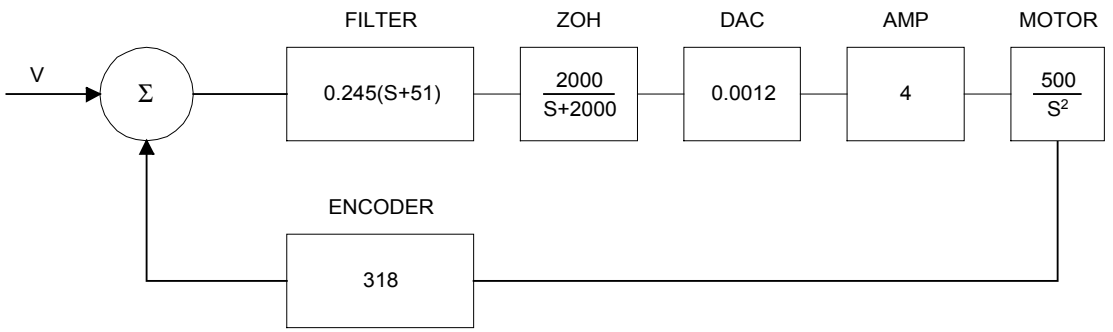


Figure 10.7 - Mathematical model of the control system

The open loop transfer function, A(s), is the product of all the elements in the loop.

$A = 390{,}000\ (s+51)/[s^2(s+2000)]$

To analyze the system stability, determine the crossover frequency, ω_c at which $A(j\ \omega_c)$ equals one. This can be done by the Bode plot of $A(j\ \omega_c)$, as shown in Fig. 10.8.

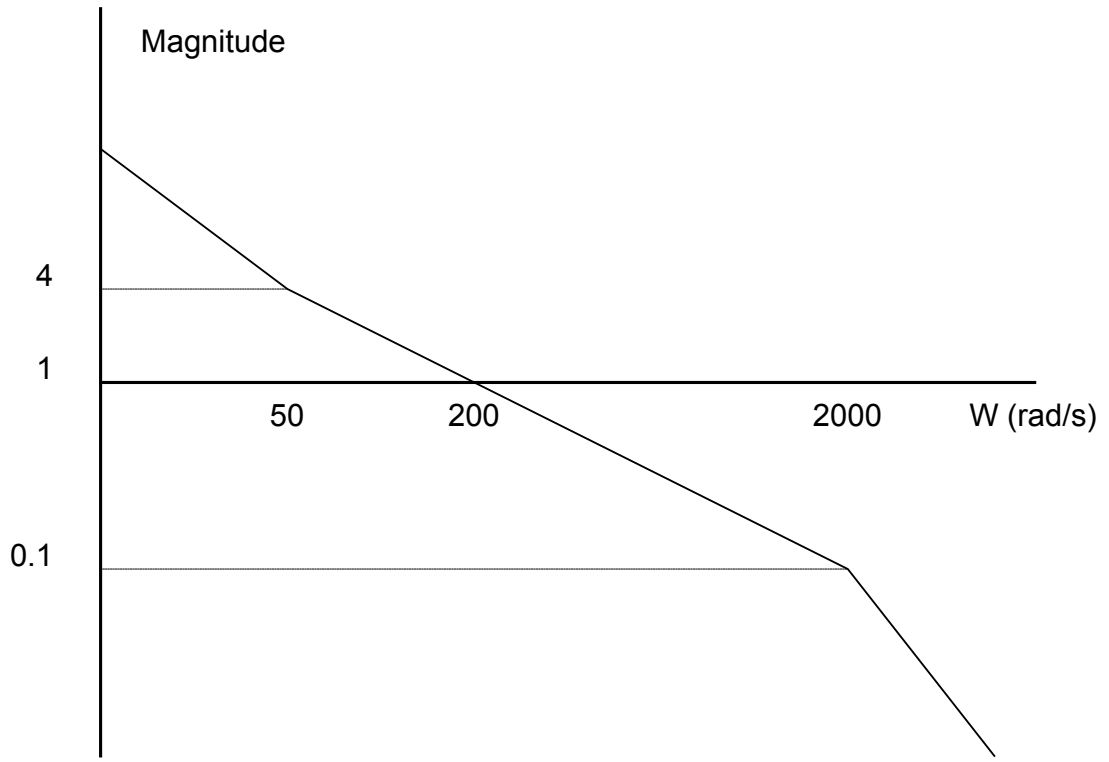


Figure 10.8 - Bode plot of the open loop transfer function

For the given example, the crossover frequency was computed numerically resulting in 200 rad/s.

Next, we determine the phase of A(s) at the crossover frequency.

$$A(j200) = 390{,}000\ (j200+51)/[(j200)^2\ .\ (j200 + 2000)]$$

$$\alpha = \text{Arg}[A(j200)] = \tan^{-1}(200/51) - 180° - \tan^{-1}(200/2000)$$

$$\alpha = 76° - 180° - 6° = -110°$$

Finally, the phase margin, PM, equals

$$PM = 180° + \alpha = 70°$$

As long as PM is positive, the system is stable. However, for a well damped system, PM should be between 30 degrees and 45 degrees. The phase margin of 70 degrees given above indicated overdamped response.

Next, we discuss the design of control systems.

System Design and Compensation

The closed-loop control system can be stabilized by a digital filter, which is preprogrammed in the SMC-2000 controller. The filter parameters can be selected by the user for the best compensation. The following discussion presents an analytical design method.

The Analytical Method

The analytical design method is aimed at closing the loop at a crossover frequency, ω_c, with a phase margin PM. The system parameters are assumed known. The design procedure is best illustrated by a design example.

Consider a system with the following parameters:

K_t	Nm/A	Torque constant
$J = 2.10^{-4}$	$kg.m^2$	System moment of inertia
$R = 2$	Ω	Motor resistance
$K_a = 2$	Amp/Volt	Current amplifier gain
$N = 1000$	Counts/rev	Encoder line density

The DAC of the SMC-2000 outputs +/-10V for a 14-bit command of +/-8192 counts.

The design objective is to select the filter parameters in order to close a position loop with a crossover frequency of $\omega_c = 500$ rad/s and a phase margin of 45 degrees.

The first step is to develop a mathematical model of the system, as discussed in the previous system.

Motor

$$M(s) = P/I = K_t/Js^2 = 1000/s^2$$

Amp

$$K_a = 2 \qquad [Amp/V]$$

DAC

$$K_d = 10/8192$$

Encoder

$$K_f = 4N/2\pi = 636$$

ZOH

$$H(s) = 2000/(s+2000)$$

Compensation Filter

$$G(s) = P + sD$$

The next step is to combine all the system elements, with the exception of G(s), into one function, L(s).

$$L(s) = M(s)\, K_a\, K_d\, K_f\, H(s) = 1.27*10^7/[s^2(s+2000)]$$

Then the open loop transfer function, A(s), is

$$A(s) = L(s)\, G(s)$$

Now, determine the magnitude and phase of L(s) at the frequency $\omega_c = 500$.

$$L(j500) = 1.27*10^7/[(j500)^2\ (j500+2000)]$$

This function has a magnitude of

$$|L(j500)| = 0.025$$

and a phase

$$\text{Arg}[L(j500)] = -180° - \tan^{-1}(500/2000) = -194°$$

G(s) is selected so that A(s) has a crossover frequency of 500 rad/s and a phase margin of 45 degrees. This requires that

$$|A(j500)| = 1$$

$$\text{Arg}\,[A(j500)] = -135°$$

However, since

$$A(s) = L(s)\,G(s)$$

then it follows that G(s) must have magnitude of

$$|G(j500)| = |A(j500)/L(j500)| = 40$$

and a phase

$$\arg\,[G(j500)] = \arg\,[A(j500)] - \arg\,[L(j500)] = -135° + 194° = 59°$$

In other words, we need to select a filter function G(s) of the form

$$G(s) = P + sD$$

so that at the frequency ω_c =500, the function would have a magnitude of 40 and a phase lead of 59 degrees.

These requirements may be expressed as:

$$|G(j500)| = |P + (j500D)| = 40$$

and

$$\arg\,[G(j500)] = \tan^{-1}[500D/P] = 59°$$

The solution of these equations leads to:

$$P = 40\cos 59° = 20.6$$

$$500D = 40\sin 59° = 34.3$$

Therefore,

$$D = 0.0686$$

and

$$G = 20.6 + 0.0686s$$

The function G is equivalent to a digital filter of the form:

$$D(z) = KP + KD(1-z^{-1})$$

where

$$KP = P$$

and

$$KD = D/T$$

Assuming a sampling period of T=1ms, the parameters of the digital filter are:

$$KP = 20.6$$

KD = 68.6

The SMC-2000 can be programmed with the instruction:

KP 20.6

KD 68.6

In a similar manner, other filters can be programmed. The procedure is simplified by the following table, which summarizes the relationship between the various filters.

Digital	$D(z) = K(z-A/z) + Cz/z-1$
Digital	$D(z) = KP + KD(1-z^{-1}) + KI/8(1-z^{-1})$
KP, KD, KI	$K = KP + KD$ $A = KD/(KP+KD)$ $C = KI/8$
Digital	$D(z) = GN(z-ZR)/z + KI\ z/8(z-1)$
GN, ZR, KI	$K = GN$ $A = ZR$ $C = KI/8$
Continuous	$G(s) = P + Ds + I/s$
PID, T	$P = K(1-A) = KP$ $D = K.A.T = T.KD$ $I = C/T = KI/8T$

Command Reference

Command Syntax

Each executable instruction is listed in the following section in alphabetical order.

The two letter op-code for each instruction is placed in the upper left corner. Below the op-code is a description of the command and required arguments. As arguments, some commands require actual values to be specified following the instruction. These commands are followed by lower case x, y, z, and w. Values may be specified for any axis separately or any combination of axes. Axis values are separated by commas. Examples of valid x ,y, z, w syntax are listed below. For the SMC-2000-8, the axis designators a,b,c,d,e,f,g,h are used where x,y,z,w can be used interchangeably with a,b,c,d.

Valid x,y,z,w syntax	
AC x	Specify x only
AC x,y	Specify x and y only
AC x,,z	Specify x and z only
AC x,y,z,w	Specify x,y,z,w
AC ,y	Specify y only
AC ,y,z	Specify y and z
AC ,,z	Specify z only
AC ,,,w	Specify w only
AC x,,,w	Specify x and w only
AC a,,,d,,f	Specify a,d and f only (SMC-2000-8)

Where x, y, z and w are replaced by actual values.

A ? returns the specified value for that axis. For example, AC ?,?,?,?, returns the acceleration of the X,Y,Z and W axes.

Other commands require action on the X,Y,Z or W axis to be specified. These commands are followed by uppercase X,Y,Z or W. Action for a particular axis or any combination is specified by writing X,Y,Z or W. No commas are needed. Valid XYZW syntax is listed below. The SMC-2000-8 uses ABCDEFGH axis designators where XYZW can be used interchangeably with ABCD.

Valid XYZW syntax	
SH X	Servo Here, X only
SH XYW	Servo Here, X,Y and W axes
SH XZW	Servo Here, X,Z and W axes
SH XYZW	Servo Here, X,Y,Z and W axes
SH Y	Servo Here, Y only
SH YZW	Servo Here, Y,Z and W axes
SH Z	Servo Here, Z only
SH	Servo Here, all axes
SH W	Servo Here, W only
SH ZW	Servo Here, Z and W axes
SH ABFG	Servo Here, A,B,F,G axes

Where X,Y,Z and W specify axes.

The usage "Description:" specifies the restrictions on allowable execution. "While Moving" states whether or not the command is valid while the controller is performing a previously defined motion. "In a program" states whether the command may be used as part of a user-defined program. "Not in a program" states whether the command may be used from the serial port.

"Can be Interrogated" states whether or not the command can be interrogated by using ? to return the specified value. "Used as an Operand" states whether a command can be used to generate a value for another command or variable (i.e. V=_TTX). "Default Format" defines the format of the value with number of digits before and after the decimal point. Finally, "Default Value" defines the values the instruction's parameters will have after a Master Reset.

AB (Abort)

DESCRIPTION:

AB (Abort) stops a motion instantly without a controlled deceleration. If there is a program executing, AB also aborts the program unless a 1 argument is specified. If the command Off-on-Error (OE1) has been activated, AB will shut off the motor enable signal (MO). On firmware 2.0g or 19n and higher, MG_AB returns the state of the abort input; 0=aborted, 1=not aborted (by input).

ARGUMENTS: AB n

where n = no argument or 1

1 aborts motion without aborting program, 0 aborts motion and program

AB aborts motion on all axes in motion and cannot stop individual axes.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

OPERAND USAGE:

If the controller has standard firmware 2.0g or higher, or special firmware d150N19n or greater, _AB returns the state of the abort input. Zero indicates aborted condition and one indicates a non aborted condition.

RELATED COMMANDS:

"SH"	Turns servos back on if they were shut-off by Abort and OE1.

EXAMPLES:

AB	Stops motion
OE 1,1,1,1	Enable off-on-error
AB	Shuts off motor command and stops motion
#A	Label - Start of program
JG 20000	Specify jog speed on X-axis
BGX	Begin jog on X-axis
WT 5000	Wait 5000 msec
AB1	Stop motion without aborting program
WT 5000	Wait 5000 milliseconds
SH	Servo Here
JP #A	Jump to Label A
EN	End of the routine

Hint: Remember to use the parameter 1 following AB if you only want the motion to be aborted. Otherwise, your application program will also be aborted.

@ABS (Absolute value function)

DESCRIPTION:

The Absolute Value (@ABS[n]) function returns the absolute value of a number or variable given in square brackets. Note that the @ABS command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @ABS [n]

where n is a number in the range of –2147483647.9999 to 2147483647.9999

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=-45.6	Set a variable equal to -45.6
MG @ABS[VAR1]	Display the absolute value of VAR1
VAR2=@ABS[VAR]+100.404	Perform calculation
EN	End of Program

AC (Acceleration)

DESCRIPTION:

The Acceleration (AC) command sets the linear acceleration rate of the motors for independent moves, such as PR, PA and JG moves. The parameters input will be rounded down to the nearest factor of 1024. The units of the parameters are counts per second squared. The acceleration rate may be changed during motion. The DC command is used to specify the deceleration rate.

ARGUMENTS: AC x,y,z,w AC a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range in the range 1024 to 67107840

?,?,?,? returns the value

USAGE:

While Moving	Yes	Default Value	256000
In a Program	Yes	Default Format	8.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

OPERAND USAGE:

_ACn returns the acceleration rate for an axis where n is the axis letter.

RELATED COMMANDS:

"DC"	Specifies deceleration rate.
"FA"	Feed forward Acceleration.
"IT"	Smoothing constant - S-curve

EXAMPLES:

AC 150000,200000,300000,400000	Set X-axis acceleration to 150000, Y-axis to 200000 , the Z-axis to 300000, and the W-axis to 400000 count/sec^2.
AC ?,?,?,?	Request the Acceleration
0149504,0199680,0299008,0399360	Return Acceleration (resolution, 1024)
V=_ACY	Assigns the Y acceleration to the variable V

Hints: Specify realistic acceleration rates based on your physical system such as motor torque rating, loads, and amplifier current rating. Specifying an excessive acceleration will cause large following error during acceleration and the motor may not be able to follow the commanded profile. The Acceleration Feed forward (FA) command can help minimize following error during acceleration and deceleration.

@ACOS (Arc Cosine Function)

DESCRIPTION:

The Arc Cosine (@ACOS[n]) function returns the arc cosine, in degrees, of a number or variable which is inserted in square brackets. Note that the @ACOS command is a function, which means that it does not follow the convention of other commands, and does not require the underscore when used as an operand. This function is available on firmware d15ON19L and higher

ARGUMENTS: @ACOS [n]

where n is a number in the range of –1 to 1.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=.707	Set variable
MG @ACOS[VAR1]	Display the arc cosine of .707
VAR2=@ACOS[VAR1]+5	Perform calculation
EN	End of Program

AD (After Distance)

DESCRIPTION:

The After Distance (AD) command is a trip-point used to control the timing of events. This command will hold up the execution of the following command until the position command has reached the specified relative distance from the start of the move. The units of the command are quadrature counts. Only one axis may be specified at a time.

ARGUMENTS: AD x or AD,y or AD,,z or AD,,,w ADX=x AD a,b,c,d,e,f,g,h

Where x,y,z,and w are unsigned integers in the range -2147483648 to 2147483647 decimal.

Only one axis may be specified at a time

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AR"	After relative distance for repetitive triggering
"AV"	After distance for vector moves

EXAMPLES:

#A;DP0,0,0,0	Begin Program
PR 10000,20000,30000,40000	Specify positions
BG	Begin motion
AD 5000	After X reaches 5000
MG "Halfway to X";TPX	Send message
AD ,10000	After Y reaches 10000
MG "Halfway to Y";TPY	Send message
AD ,,15000	After Z reaches 15000
MG "Halfway to Z";TPZ	Send message
AD ,,,20000	After W reaches 20000
MG "Halfway to W";TPW	Send message
EN	End Program

Hints: the AD command is accurate to the number of counts that occur in 2 msec. Multiply your speed by 2 msec to obtain the maximum position error in counts. Remember AD measures incremental distance from start of move on one axis.

AE (Absolute Encoder)

DESCRIPTION

The Absolute Encoder (AE) command reads the encoder data from a Yaskawa absolute encoder. The command automatically enters the absolute position received into the axes' position register. The motor must be off (MO) before using this command. This command is only valid for firmware version d150N19i and greater.

ARGUMENTS: AEX=32768 AE x,y,z,w AE a,b,c,d,e,f,g,h

where x,y,z,and w are signed numbers (encoder resolution, post quadrature) in the range +/- 1 to 100,000 decimal.

These values are the number of encoder pulses per revolution after quadrature.

TIP: If the rotation direction bit in the servo amplifier is set for reverse rotation, you will need to use a negative number with this command.

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

OPERAND USAGE:

_AE returns the number of the last encoder that was read, 0=X axis, 1=Y axis,…

RELATED COMMANDS:

TY	Tell Yaskawa Encoder
QY	Report Alarm code from Yaskawa encoder

EXAMPLES:

MOX	Motor Off
AEX=4096	Read X absolute encoder
DPX=_TPX+XhmOfs	Add offset variable to current absolute position
TPX	Tell position of X axis
SHX	Enable X servo amplifier

Hints: If an absolute encoder fails to respond, a command error will be generated. If you use the #CMDERR special label, you can write a routine to display the type of encoder problem using the "QY" and "_AE" commands. See the absolute encoder section in the options chapter for more information.

AF (Analog Feedback)

DESCRIPTION:

The Analog Feedback (AF) command is used to set an axis with analog feedback instead of digital feedback (quadrature/pulse dir.). Analog feedback uses analog inputs 1-7 as +/- 10 volt signals converted to +/- 2048 which can be read as position when examining _TPn.

ARGUMENTS: AF x, y, z, w, AFX=x AF a, b, c, d, e, f, g, h

where x, y, z, w or a, b, c, d, e, f, g, h are integers

1 = Enables analog feedback

0 = Disables analog feedback and switches to digital feedback

USAGE:

While Moving	No	Default Value	0,0,0,0
In a Program	Yes		
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

OPERAND USAGE:

_AFn returns the feedback type where n is the axis in question.

RELATED COMMANDS:

"CE"	Configure Encoder
"MT"	Motor Type

EXAMPLES:

AF 1, 0, 0, 1	Analog feedback on X and W axis
V1 = _AFX	Assign feedback type to variable
AF ?,?,?	Interrogate feedback types

NOTE: Selecting H-axis for analog feedback requires a small modification from the factory.

AI (After Input)

DESCRIPTION:

The After Input (AI) command is used to wait until after the specified input has occurred. If n is positive, it waits for the input to be high. If n is negative, it waits for n to be low. If you want to wait for a rising edge or falling edge, put two commands together as follows: AI –3; AI 3. This will cause the program to wait for a rising edge.

ARGUMENTS: AI +/-n

where n is an integer in the range 1 to 8 decimal

where n is an integer in the range 1 to 24 decimal for SMC-20008

where n is an integer in the range 1 to 64 decimal for SMC-2000xI (extended I/O or Network I/O options)

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

@IN[n]	Function to read input n
"II"	Input interrupt
#ININT	Special Label for input interrupt

EXAMPLES:

#A	Begin Program
AI 8	Wait until input 8 is high
SP 10000	Speed is 10000 counts/sec
AC 20000	Acceleration is 20000 counts/sec^2
PR 400	Specify position
BG X	Begin motion
EN	End Program

Hint: The AI command suspends execution of a program thread until the specified input is at the required logic level. Use the conditional Jump command (JP) or input interrupt (II) if you do not want the program sequence to suspend.

AL (Arm Latch)

DESCRIPTION:

The Arm Latch (AL) command enables the latching function (high-speed position capture) of the controller. When the ALXYZW command is used to arm the position latches, the encoder position will be captured upon a high or low going signal on Input 1 (X axis), Input 2 (Y axis), Input 3 (Z axis), Input 4 (W axis), Input 9 (E axis), Input 10 (F axis), Input 11 (G axis) and Input 12 (H axis). The RL command returns the captured position for the specified axes. When interrogated or used as an operand, the AL command will return a 1 if the latch for that axis is armed, or a zero after the latch has occurred. Use the CN command to change the active state of the latch.

ARGUMENTS: AL XYZW AL ABCDEFGH

where X,Y,Z,W specifies the X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

OPERAND USAGE:

_AL returns the state of the latch, one means still armed, zero if the latch has been tripped.

RELATED COMMANDS:

“RL”	Report Latch
“LT”	Latch Target

EXAMPLES:

#START	Start program
ALY	Arm Y-axis latch
JG,50	Set up jog at 50000 counts/sec
BGY	Begin the move
#LOOP	Loop until latch has occurred
JP #LOOP,_ALY=1	
RLY	Transmit the latched position
EN	End of program

Note: The Latch function will occur within 25 µSec only when used in the active low mode.

AM (After Motion)

DESCRIPTION:

The After Motion (AM) command is a trip-point used to control the timing of events. This command will hold up execution of the following commands until the current move on the specified axis or axes is completed. Any combination of axes or a motion sequence may be specified with the AM command. For example, AM XY waits for motion on both the X and Y axis to be complete. AM with no parameter specifies that motion on all axes is complete. AMS waits for a vector motion path to complete.

ARGUMENTS: AM XYZWS AM ABCDEFGH

where X,Y,Z,W,S specifies X,Y,Z or W axis or sequence. No argument specifies that motion on all axes is complete.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"BG"	(_BG returns a 0 if motion complete)
"MC"	Motion Complete – when encoder crosses the target position

EXAMPLES:

#MOVE	Program MOVE
PR 5000,5000,5000,5000	Position relative moves
BG X	Start the X-axis
AM X	After the move is complete on X,
BG Y	Start the Y-axis
AM Y	After the move is complete on Y,
BG Z	Start the Z-axis
AM Z	After the move is complete on Z
BG W	Start the W-axis
AM W	After the move is complete on W
EN	End of Program
#F;DP 0,0,0,0	Program F
PR 5000,6000,7000,8000	Position relative moves
BG	Start X,Y,Z and W axes
AM	After motion complete on all axes
MG "DONE";TP	Print message
EN	End of Program

Hint: AM is a very important command for controlling the timing between multiple move sequences. For example, if the X-axis is in the middle of a position relative move (PR) you cannot make a position absolute move (PAX; BGX) until the first move is complete. Use AMX to suspend the program sequences until the first motion is complete. AM tests for profile completion. The actual motor may still be moving, trying to settle on

the target position. Another method for testing if motion is complete is to check for the internal variable, _BG being equal to zero. To be sure the encoder actually crossed the final target, use the MC command.

@AN (Read Analog Input)

DESCRIPTION:

The Read Analog (@AN[n]) function returns the value of an analog input as a voltage (+/- 10 volt.) Note that the @AN command is a function, which means that it does not follow the convention of other commands, and does not require the underscore when used as an operand. The resolution is 14 bit, or 1.2 mV per bit.

ARGUMENTS: @AN[n]

where n is an unsigned integer in the range 1 to 7 decimal

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
MG @AN[2]	Display the value of analog input #2 as a voltage
JGX=@AN[2]*10000	Set jog speed according to analog input
BG X	Begin move
EN	End of Program

AP (After Absolute Position)

DESCRIPTION:

The AP command is a trip-point used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:

1) The commanded position crosses the absolute position specified.

2) The motor profiling on that axis is complete.

3) The commanded motion moves is in the direction which moves away from the specified absolute position.

The units of the command are in quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trip-point will automatically be cleared..

ARGUMENTS: APx or AP,y or AP,,z or AP,,,w APX=X AP abcdefgh

where x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

AD	After distance for relative distances
MF	Motion forward for general motion
MR	Motion reverse for general motion

EXAMPLES:

#TEST	Program TEST
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG X	Begin move
AP 2000	After passing the position 2000
V1=_TPX	Assign V1 X position
MG “Position is”,V1=	Print Message
ST	Stop
EN	End of Program

Hint: The accuracy of the AP command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. AP tests for absolute position. Use the AD command to measure incremental distances. Use the MF and MR commands for axes that are not under controlled motion.

AR (After Relative Distance)

DESCRIPTION:

The After Relative Distance command is a trip-point used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:

1) The commanded position crosses the specified relative distance from either the start of the move of the last AR or AD command.
2) The motor profiling on that axis is complete.
3) The commanded motion moves is in the direction which moves away from the specified absolute position.

The units of the command are in quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trip-point will automatically be cleared.

ARGUMENTS: ARx or AR,y or AR,,z or AR,,,w ARX=x AR abcdefgh

where x,y,z,w are unsigned integers in the range 0 to 2147483647 decimal.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AV"	After vector position for coordinated moves
"AP"	After absolute position

EXAMPLES:

#A;DP 0,0,0,0	Begin Program
JG 50000,,,7000	Specify speeds
BG XW	Begin motion
#B	Label
AR 25000	After passing 25000 counts of relative distance on X-axis
MG "Passed X";TPX	Send message on X-axis
JP #B	Jump to Label #B
EN	End Program

Hint: AR is used to specify incremental distance from last AR or AD command. Use AR if multiple position trip-points are needed in a single motion sequence.

AS (At Speed)

DESCRIPTION:

The AS command is a trip-point that occurs when the generated motion profile has reached the specified speed. This command will operate after either accelerating or decelerating. If the speed is not reached, the trip-point will be triggered after the motion is stopped (after deceleration).

ARGUMENTS: AS X or AS Y or AS Z or AS W or AS S AS ABCDEFGH

where XYZWS specifies X,Y,Z,W axis or sequence

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

#SPEED	Program SPEED
PR 100000	Specify position
SP 10000	Specify speed
BG X	Begin X
ASX	After speed is reached
MG “At Speed”	Print Message
EN	End of Program

Warning: The AS command applies to a trapezoidal velocity profile only with linear acceleration. AS used with S-curve profiling will be inaccurate.

@ASIN (Arc Sine Function)

DESCRIPTION:

The Arc Sine (@ASIN [n]) function returns the arc sine, in degrees, of a number or variable which is inserted in square brackets. Note that the @ASIN command is a function, which means that it does not follow the convention of other commands, and does not require the underscore when used as an operand. This function is available only in firmware d15ON19L and higher

ARGUMENTS: @ASIN [n]

where n is a number in the range of –1 to 1.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=.707	Set variable
MG @ASIN[VAR1]	Display the arc sine of .707
VAR2=@ASIN[VAR1]+5	Perform calculation
EN	End of Program

AT (At Time)

DESCRIPTION:

The AT command is a trip-point that is used to hold up execution of the next command until after the specified time has elapsed. The time is measured with respect to a defined reference time. AT 0 establishes the initial reference. AT n specifies n msec from the reference. AT -n specifies n msec from the reference and establishes a new reference after the elapsed time period.

ARGUMENTS: AT n

where n is a signed integer in the range 0 to 2147483647

n = 0 defines a reference time at current time

positive n waits n msec from reference

negative n waits n msec from reference and sets new reference after elapsed time period (AT -n is equivalent to AT n; AT 0)

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

The following commands are sent sequentially

AT 0	Establishes reference time 0 as current time
AT 50	Waits 50 msec from reference 0
AT 100	Waits 100 msec from reference 0
AT -150	Waits 150 msec from reference 0 and sets new reference at 150
AT 80	Waits 80 msec from new reference (total elapsed time is 230 msec)

AV (After Vector Distance)

DESCRIPTION:

The AV command is a trip-point that is used to hold up execution of the next command during coordinated moves such as VP,CR or LI. This trip-point occurs when the path distance of a sequence reaches the specified value. The distance is measured from the start of a coordinated move sequence or from the last AV command. The units of the command are quadrature counts.

When used as an operand, _AV returns the vector distance from the start of the sequence. _AV is valid in the linear mode, LM, and in the vector mode, VM

ARGUMENTS: AV n

where n is an unsigned integer in the range 0 to 2147483647 decimal

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#MOVE;DP 0,0	Label
LMXY	Linear move for X,Y
LI 1000,2000	Specify distance
LI 2000,3000	Specify distance
LE	
BGS	Begin
AV 500	After path distance = 500,
MG “Path>500”;TPXY	Print Message
EN	End Program

Hint: Remember AV is the vector distance along the path where $AV^2=X^2+Y^2+Z^2+W^2$

BG (Begin Motion)

DESCRIPTION:

The BG command starts a motion on the specified axis or sequence. When used in an operand, the BG command will return a 1 if the controller is performing a move of that axis.

ARGUMENTS: BG XYZWS BG ABCDEFGH

where XYZW are X,Y,Z,W axes and S is coordinated sequence

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"After Motion (AM"	After motion complete
"ST"	Stop motion

EXAMPLES:

PR 2000,3000,,5000	Set up for a relative move
BG XYW	Start the X,Y and W motors moving
HM	Set up for the homing
BGX	Start only the X-axis moving
JG 1000,4000	Set up for jog
BGY	Start only the Y-axis moving
YSTATE=_BGY	Assign a 1 to YSTATE if the Y-axis is performing a move
VP 1000,2000	Specify vector position
VS 20000	Specify vector velocity
BGS	Begin coordinated sequence
VMXY	Vector Mode
VP 4000,-1000	Specify vector position
VE	Vector End
PR ,,8000,5000	Specify Z and W position
BGSZW	Begin sequence and Z,W motion

Hint: You cannot give another BG command until current BG motion has been completed. Use the AM trip-point to wait for motion complete between moves. Another method for checking motion complete is to test for _BG being equal to 0.

BL (Reverse Software Limit)

DESCRIPTION:

The BL command sets the reverse software limit. If this limit is exceeded during motion, motion on that axis will decelerate to a stop. Reverse motion beyond this limit is not permitted. The reverse limit is activated at X-1, Y-1, Z-1, and W-1. To disable the reverse limit, set X,Y,Z,W to -2147483648. The units are in quadrature counts.

ARGUMENTS: BL x,y,z,w BLX=x BL a,b,c,d,e,f,g,h

where x,y,z,and w are signed integers in the range -2147483648 to 2147483647.

The value -2147483648 disables the reverse software limit.

USAGE:

While Moving	Yes	Default Value	-2147483648
In a Program	Yes	Default Format	Position format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"FL"	Forward Limit

EXAMPLES:

#TEST	Test Program
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
BL -15000	Set Reverse Limit
JG -5000	Jog Reverse
BGX	Begin Motion
AMX	After Motion (limit occurred)
TPX	Tell Position
EN	End Program

BN (Burn)

DESCRIPTION:

The BN command saves specific controller parameters shown below in non-volatile EEPROM memory. Programs are not saved by this command, use the BP command to burn programs. This command typically takes 1 second to execute and must not be interrupted. The controller returns a : when the Burn is complete.

Controller Parameters Saved During Burn:

AC	IT
AF	KD (ZR converted to KD)
BL	KI
CB	KP (GN converted to KP)
CC	LZ
CE	MO
CN	MOTOR State (Servo here or motor off)
CW	MT
DC	OE
DV	OF
EA	OP
EM	PF
EO	SB
EP	SP
ER	TL
ES	TM
ET[n] (Electronic Cam Table)	TR
FA	VA
FL	VD
FV	VF
GA	VS
GR	VT
IL	

ARGUMENTS: None

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#MAIN	Label
II1	Enable input interrupt on input 1
PR 5000	Move relative position of 5000 counts
SP 10000	Set speed

AC 200000	Set acceleration
BGX	Begin motion X-axis
AMX	After motion complete
WT 1000	Wait 1000 msec
JP #MAIN	Jump to MAIN label
#ININT	Input interrupt routine
AM	Wait for motion complete
WT 1000	Wait 1000 msec
BN	Burn values; may take up to 15 seconds
RI 1	Return to main program and restore trip-point

NOTE: Does not burn programs, variables, or arrays.

BP (Burn Program)

DESCRIPTION:

The BP command saves the application program in non-volatile EEPROM memory. This command typically takes up to 10 seconds to execute and must not be interrupted. The controller returns a : when the Burn is complete.

ARGUMENTS: None

USAGE:

While Moving	No	Default Value	---
In a Program	No		
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

NOTE: Does not burn variables or arrays.

BV (Burn Variables)

DESCRIPTION:

The BV command saves variables in non-volatile EEPROM memory. This command typically takes up to 2 seconds to execute and must not be interrupted. The controller returns a : when the Burn is complete.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes		
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

NOTE: BV does not save any array data unless you have an SMC with an "M" option for larger memory.

CB (Clear Bit)

DESCRIPTION:

The CB command clears an output bit. The CB and SB (Set Bit) instructions can be used to control the state of output lines.

ARGUMENTS: CB n

where n is an integer in the range 1 to 8 decimal for SMC-20001, SMC-20002, SMC-20004

where n is an integer in the range 1 to 16 decimal for SMC-20008

where n is an integer in the range 1 to 64 (Outputs 24,32,40,48, 56 and 64 do not physically exist) decimal for SMC-2000xI (extended I/O)

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"SB"	Set Bit
"OP"	Define all 16 outputs of an output port

EXAMPLES:

CB 1	Clear output bit 1
CB 7	Clear output bit 7
CB 16	Clear output bit 16 (SMC-2000-8 only)
CB 64	Reset a Network I/O option card

CC (Configure Communications Port 2)

DESCRIPTION:

The CC command allows the user to configure the RS232 port number 2. The m parameter specifies the baud rate. The n parameter specifies whether it is in handshake or non-handshake mode. The r parameter configures the port for either General port or for daisy chain. The p parameter specifies echo on or off; however, this parameter is only valid if the port is configured for General port. Port 2 must be configured before used.

ARGUMENTS: CC m,n,r,p

m - Baud rate – 300, 1200, 4800, 9600, 19200 or 38400

n - Handshake - n=0 means no handshake, n=1 means handshake

r - Mode; r=0 for general port at Com2, r=1 for daisy-chain at Com2 Active only if r=0 (general port at P2)

p - Echo; p=1 for echo on, p=0 for echo off

USAGE:

While Moving	Yes	Default Value	0,0,0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

CD (Contour Data)

DESCRIPTION:

The CD command specifies the incremental position on X,Y,Z and W axes. The units of the command are in quadrature counts. This command is used only in the Contour Mode (CM).

ARGUMENTS: CD x,y,z,w CDX=x CD a,b,c,d,e,f,g,h

where x,y,z,w are integers in the range of +/-32762

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"CM"	Contour Mode
"WC"	Wait for Contour
"DT"	Time Increment
"CS"	_CS is the Segment Counter

EXAMPLES:

CM XYZW	Specify Contour Mode
DT 4	Specify time increment for contour
CD 200,350,-150,500	Specify incremental positions on X,Y,Z and W axes X-axis moves 200 counts Y-axis moves 350 counts Z-axis moves -150 counts W-axis moves 500 counts
WC	Wait for complete
CD 100,200,300,400	New position data
WC	Wait for complete
DT0	Stop Contour
CD 0,0,0,0	Exit Mode

CE (Configure Encoder)

DESCRIPTION:

The CE command configures the encoder to quadrature type or to pulse and direction type. It also allows inverting the polarity of the encoders. The configuration applies independently to the four main axis encoders and the four auxiliary encoders.

ARGUMENTS: CE x,y,z,w CEX=x CE a,b,c,d,e,f,g,h

Where x,y,z,w are integers in the range of 0 to 15. Each integer is the sum of two integers n and m which configure the main and the auxiliary encoders. The values of m and n are

m =	Main encoder type	n =	Auxiliary encoder type
0	Normal quadrature	0	Normal quadrature
1	Normal pulse and direction	4	Normal pulse and direction
2	Reversed quadrature	8	Reversed quadrature
3	Reversed pulse and direction	12	Reversed pulse and direction

For example: x = 10 implies m = 2 and n = 8, both encoders are reversed quadrature.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	2.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

MT	Motor Type

EXAMPLES:

CE 0, 3, 6, 2	Configure encoders
CE ?,?,?,?	Interrogate configuration
V = _CEX	Assign configuration to a variable

Note: When using pulse and direction encoders, the pulse signal is connected to the B channel and the direction signal is connected to the A channel.

WARNING: This command works closely with the Motor Type (MT) command. Assuming your system is operating normally, but in the wrong direction, you must change both the (MT) and the (CE) commands under (MO) Motor Off conditions. Failure to perform this change correctly will lead to system run away!

CI (Communication Interrupt)

DESCRIPTION:

The CI command configures communication interrupt based on characters received either from Port 1 or Port 2. An interrupt causes program flow to jump to the #COMINT subroutine label. If multiple program threads are used, the #COMINT subroutine runs in thread zero, and threads one, two, and three continue to run without interruption. The characters received on the serial port are stored in internal variables such as P2CH.

ARGUMENTS: CI m,n,o

where:

m =	Port 1	n =	Port 2
0	Do not interrupt	0	Do not interrupt
1	Interrupt on <enter>	1	Interrupt on <enter>
2	Interrupt on any character	2	Interrupt on any character
-1	Clear buffer	-1	Clear buffer
o = 0	Disable live data for Port 1	o = 1	Enable live data for Port 1

USAGE:

While Moving	Yes	Default Value	
In a Program	Yes	Default Format	
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

CI 0,1,0	Set communications interrupt for Port 2 on <enter>
CI 1,2,0	Set Port 1 for interrupt on <enter>, interrupt Port 2 on any character.

NOTE: When using this command to jump to the #COMINT label, you must re-issue the CI command at the end of the #COMINT routine

CM (Contour Mode)

DESCRIPTION:

The Contour Mode is initiated by the instruction CM. This mode allows the generation of an arbitrary motion trajectory with any of the axes. The CD command specifies the position increment, and the DT command specifies the time interval. The CM? or _CM commands can be used to check the status of the Contour Buffer.

When CM? is sent and the returned value is 1, it indicates the Contour Buffer is full. If the returned value is 0, it indicates the Contour Buffer is empty, and a new CD command can be issued.

ARGUMENTS: CM XYZW CM ABCDEFGH

where XYZW specify the X,Y,Z,W axes

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"CD"	Contour Data
"WC"	Wait for Contour
"DT"	Time Increment

EXAMPLES:

V=_CM;V=	Return Contour Buffer Status
1	Contour Buffer is full
CM XZ	Specify X,Z axes for Contour Mode

CN (Configure)

DESCRIPTION:

The CN command configures the polarity of the limit switches, the home switch, and the latch input.

ARGUMENTS: CN m,n,o

where m,n,o are integers with values 1 or -1.

m =	1	Limit switches active high
	-1	Limit switches active low
n =	1	Home switch active high
	-1	Home switch active low
o =	1	Latch input active high
	-1	Latch input active low

USAGE:

While Moving	Yes	Default Value	-1.-1.-1
In a Program	Yes	Default Format	2.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"MT"	Motor Type

EXAMPLES:

CN 1,1	Sets limit and home switches to active high
CN,, -1	Sets input latch active low

@COM (2's Complement Function)

DESCRIPTION:

The 2's Complement (@COM[n]) function returns the complement of a number or variable given in square brackets. Note that the @COM command is a function, which means that it does not follow the convention of other commands, and does not require the underscore when used as an operand.

ARGUMENTS: @COM [n]

where n is a number in the range of –2147483647.9999 to 2147483647.9999

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=1234	Set variable
MG @COM[VAR1]	Display the complement of 1234
VAR2=@COM[VAR1]+99	Perform calculation
EN	End of Program

@COS (Cosine Function)

DESCRIPTION:

The Cosine (@COS[n]) function returns the cosine of a number or variable which is inserted in square brackets using units of degrees. Note that the @COS command is a function, which means that it does not follow the convention of other commands, and does not require the underscore when used as an operand.

ARGUMENTS: @COS [n]

where n is a number in the range of –32768 to 32768

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=45	Set variable
MG @COS[VAR1]	Display the cosine of 45
VAR2=@COS[VAR1]+99	Perform calculation
EN	End of Program

CR (Circle)

DESCRIPTION:

The CR command specifies a two-dimensional arc segment of radius, r, starting at angle, θ, and traversing over angle Δθ. A positive Δθ denotes counter-clockwise traverse, negative Δθ denotes clockwise. The VE command must be used to denote the end of the motion sequence after all CR and VP segments are specified. The BG (Begin Sequence) command is used to start the motion sequence. All parameters, r, θ, Δθ, must be specified. Radius units are in quadrature counts. θ and Δθ have units of degrees. The parameter n is optional and describes the vector speed that is attached to the motion segment. (Used when multiple segments are combined.)

ARGUMENTS: CR r,θ,Δθ <n

where r is an unsigned real number in the range 10 to 6000000 decimal

θ is an unsigned number in the range 0 to +/-32000 decimal

Δθ is a signed real number in the range 0.0001 to +/-32000 decimal

The product r * Δθ must be limited to +/-4.5 $E10^8$

N is an unsigned number between 0 and 8000000.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"VP"	Vector Position
"VS"	Vector Speed
"VD"	Vector Deceleration
"VA"	Vector Acceleration
"VM"	Vector Mode
"VE"	End Vector
"BG"	BGS - Begin Sequence

EXAMPLES:

VM XY	Specify motion plane
CR 1000,0,360	Generate circle with radius of 1000 counts, start at 0 degrees and complete one circle in counterclockwise direction.
VE	End Sequence
BGS	Start motion

CS (Clear Sequence)

DESCRIPTION:

The CS command will remove VP, CR or LI commands stored in a motion sequence. Please note that after a sequence has been run, the CS command is not necessary to enter a new sequence. This command is useful if you have incorrectly specified VP, CR or LI commands.

When used as an operand, _CS returns the number of the segment in the sequence, starting at zero. The instruction _CS is valid in the Linear Mode, LM, Vector Mode, VM, and Contour Mode, CM.

ARGUMENTS: None

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#CLEAR	Label
VP 1000,2000	Vector position
VP 4000,8000	Vector position
CS	Clear vectors
VP 1000,5000	New vector
VP 8000,9000	New vector
VE	End Sequence
BGS	Begin Motion sequence
EN	End of Program

CW (Copyright information / Data Adjustment bit on/off)

FUNCTION: DESCRIPTION:

The CW command has a dual usage. The CW command will return the copyright information when the argument, n is 0. Otherwise, the CW command is used as a communications enhancement. When turned on, the communication routine sets the MSB of unsolicited, returned ASCII characters to 1. Unsolicited ASCII characters are those characters which are returned from the controller without being directly queried from the terminal. This is the case when a program has a command that requires the controller to return a value or string.

ARGUMENTS: CWn where

n is a number, either 0,1 or 2:

0 Causes the controller to return the copyright information

1 Causes the controller to set the MSB of unsolicited returned characters to 1

2 Causes the controller to not set the MSB of unsolicited characters.

CW ? returns the copyright information for the controller.

USAGE:

While Moving	Yes	Default Value	2
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_CW contains the value of the data adjustment bit. 2 = off, 1 = on

Note: The CW command can cause garbled characters to be returned by the controller. The default state of the controller is to disable the CW command. The CW command status can be stored in EEPROM.

DA (De-allocate the Variables & Arrays)

DESCRIPTION:

The DA command frees the memory occupied by an array or variable. In this command, more than one array or variable can be specified for de-allocation of memories. A comma separates different arrays and variables when specified in one command (up to 80 characters). The * argument de-allocates all the variables, and *[0] de-allocates all the arrays.

ARGUMENTS: DA c[0],d, etc.

where c[0] - Defined array name

d - Defined variable name

* - De-allocates all the variables

*[0] - De-allocates all the arrays

_DA returns the number of available arrays

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"DM"	Dimension Array

EXAMPLES: Cars and Salesmen are arrays and Total is a variable.

DM CARS[400],SALESMEN[50]	Dimension 2 arrays
TOTAL=70	Assign 70 to the variable Total
DA CARS[0],SALESMEN[0],TOTAL	De-allocate the 2 arrays & variable
DA*[0]	De-allocate all arrays
DA *,*[0]	De-allocate all variables and all arrays

NOTE: Since this command de-allocates the spaces and compacts the array spaces in the memory, it is possible that execution of this command may take longer than 2 ms.

DC (Deceleration)

DESCRIPTION:

The Deceleration command (DC) sets the linear deceleration rate of the motors for independent moves such as PR, PA and JG moves. The parameters will be rounded down to the nearest factor of 1024, and have units of counts per second squared.

ARGUMENTS: DC x,y,z,w DCX=x DC a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range 1024 to 67107840

?,?,?,? returns the value

USAGE:

While Moving	Yes, only jog.	Default Value	256000
In a Program	Yes	Default Format	8.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AC"	Acceleration
"PR"	Position Relative
"PA"	Position Absolute
"SP"	Speed
"JG"	Jog
"BG"	Begin
"IT"	Smoothing constant - S-curve

EXAMPLES:

PR 10000	Specify position
AC 2000000	Specify acceleration rate
DC 1000000	Specify deceleration rate
SP 5000	Specify slew speed
BG	Begin motion

NOTE: The DC command may be changed during the move in JG mode, but not in PR or PA move.

DE (Dual (Auxiliary) Encoder Position)

DESCRIPTION:

The DE x,y,z,w command defines the position of the auxiliary encoders. The auxiliary encoders may be used for dual-loop applications. DE ? or _DEX returns the position of the auxiliary encoders.

The DE command defines the current motor position when used with stepper motors. DE? Returns the commanded reference position of the motor. The units are in steps.

ARGUMENTS: DE x,y,z,w DEX=x DE a,b,c,d,e,f,g,h

where x,y,z,w are signed integers in the range -2147483647 to 2147483648 decimal

USAGE:

While Moving	Yes	Default Value	0,0,0,0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

DE 0,100,200,400	Set the current auxiliary encoder position to 0,100,200,400 on X,Y,Z and W axes
DE?,?,?,?	Return auxiliary encoder positions
DUALX=_DEX	Assign auxiliary encoder position of X-axis to the variable DUALX

Hint: Dual encoders are useful when encoders for both the motor and the load are needed. The encoder on the load can be either the auxiliary encoder or main encoder and is used to verify the true load position. Any error in load position is used to correct the motor position.

DL (Download)

DESCRIPTION:

The DL command allows transfer of a data file from the host computer to the SMC-2000. The file will be accepted as a data stream (program listing) without line numbers. The file can be terminated using <control> Z, <control> Q, <control> D, or \.

If no parameter is specified, downloading a data file will clear any programs in the SMC-2000 memory. The data is entered beginning at line 0. If there are too many lines or too many characters per line, the SMC-2000 will return a ?. To begin the download following a label, that label may be specified following DL. Or, the # argument may be used with DL to append a file at the end of the SMC-2000 program in memory. DO NOT insert any spaces before each command.

When used as an operand, _DL gives the number of available labels. The total number of labels is 254 for the SMC-2000.

ARGUMENTS: DL n

n = no argument	Downloads program beginning at line 0. Overwrites existing programs.
n = #Label valid program label.	Begins download at line following #Label where label may be any
n = #	Begins download at end of program in memory.

USAGE:

While Moving	Yes	Default Value	---
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"UL"	Upload

EXAMPLES:

DL;	Begin download
#A;PR 4000;BGX	Data
AMX;MG DONE	Data
EN	Data
<control> Z	End download

DM (Dimension)

DESCRIPTION:

The DM command defines a single dimensional array with a name and n total elements. The maximum number of arrays, which the user can define, is limited to thirty. The maximum number of total elements within all arrays is limited to 8000. The first element of the defined array starts with element number 0 and the last element is at n-1.

ARGUMENTS: DM c[n]

where c is a name of up to eight characters, starting with an uppercase alphabetic character. n is the number of entries from 1 to 8000.

_DM returns the number of available arrays elements

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"DA"	De-allocate Array

EXAMPLES:

DM PETS[5],DOGS[2],CATS[3]	Define dimension of arrays, pets with 5 elements; Dogs with 2 elements; Cats with 3 elements
DM TESTS[1600]	Define dimension of array Tests with 1600 elements

DP (Define Position)

DESCRIPTION:

The DP command sets the current motor position and current command positions to a user specified value. If a ? is used, the controller returns the current position of the motor. The units are in quadrature counts.

ARGUMENTS: DP x,y,z,w DPX=x DP a,b,c,d,e,f,g,h

where x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	No	Default Value	0,0,0,0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

DP 0,100,200,400	Sets the current position of the X axis to 0, the Y axis to 100, the Z axis to 200, and the W axis to 400
DP ,-50000	Sets the current position of Y-axis to -50000. The Y,Z and W axes remain unchanged.
DP ?,?,?,?	Returns all the motor positions
0000000,-0050000,0000200,0000400	
DP ?	Returns the X-axis motor position
0000000	

Hint: The DP command is useful to redefine the absolute position. For example, you can manually position the motor by hand using the Motor Off command, MO. Turn the servo motors back on with SH and then use DP0 to redefine the new position as your absolute zero.

DT (Delta Time)

DESCRIPTION:

The DT command sets the time interval for Contouring Mode. Sending the DT command once will set the time interval for all following contour data until a new DT command is sent. 2^n milliseconds is the time interval. Sending DT0 followed by CD0 command terminates the Contour Mode.

ARGUMENTS: DT n

where n is an integer in the range 0 to 8. 0 terminates the Contour Mode. n=1 through 8 specifies the time interval of 2^n samples.

The default time interval is n=1 for a 2 msec sample period.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"CM"	Contour Mode
"CD"	Contour Data
"WC"	Wait for next data

EXAMPLES:

DT 4	Specifies time interval to be 16 msec
DT 7	Specifies time interval to be 128 msec
#CONTOUR	Begin
CMXY	Enter Contour Mode
DT 4	Set time interval
CD 1000,2000	Specify data
WC	Wait for contour
CD 2000,4000	New data
WC	Wait
DT0	Stop contour
CD0	Exit Contour Mode
EN	End

DV (Dual Velocity (Dual Loop))

DESCRIPTION:

The DV function changes the operation of the PID loop. It causes the KD (derivative) term to operate on the dual encoder instead of the main encoder. This results in improved stability in the cases where there is a backlash between the motor and the main encoder, and where the dual encoder is mounted on the load.

ARGUMENTS: DV x,y,z,w DVX=x DV a,b,c,d,e,f,g,h

where x,y,z,w may be 0 or 1. 0 disables the function. 1 enables the dual loop.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"KD"	Damping constant
"FV"	Velocity feed forward

EXAMPLES:

DV 1,1,1,1	Enables dual loop on all axes
DV 0	Disables DV on X axis
DV,,11	Enables dual loop on Z-axis and WX axis. Other axes remain unchanged.
DV 1,0,1,0	Enables dual loop on X and Z-axis. Disables dual loop on Y and W axis.

Hint: The DV command is useful in backlash and resonance compensation.

EA (ECAM Master Axis)

DESCRIPTION:

The EA command selects the master axis for the Electric Cam mode. Any main axis may be chosen.

ARGUMENTS: EA p

Where p is X,Y,Z,W,E,F,G, or H for main encoder as ECAM Master

p is DX,DY,DZ,DW,DE,DF,DG,or DH for auxiliary encoder as ECAM Master.

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"EM"	Define CAM cycles for each axis
"EP"	Define CAM Table intervals and start point
"ET"	CAM Table entries for slave axes
"EB"	Enable CAM mode

EXAMPLES:

EAY	Select Y as the ECAM Master
EADX	Select X Auxiliary Encoder as ECAM Master

EB (Enable ECAM Mode)

DESCRIPTION:

The EB function enables or disables the cam mode. In this mode, the starting position of the master is specified within the cycle. When the EB command is given, the master axis is modularized.

ARGUMENTS: EB n

Where n = 1 starts cam mode; n = 0 stops cam mode.

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Can be Interrogated	No
Used as an Operand	Yes

RELATED COMMANDS:

"EM"	Define CAM cycles for each axis
"EP"	Define CAM Table intervals and start point

EXAMPLES:

EB 1	Starts ECAM mode
EB 0	Stops ECAM mode
B = _EB	Return status of cam mode

EG (ECAM Engage)

DESCRIPTION:

The EG command engages an ECAM slave axis at a specified position of the master. _EGX returns 1 if the X-axis is engaged. If a value is specified outside of the master's range, the slave will engage immediately. Once a slave motor is engaged, its position is redefined to fit within the cycle.

ARGUMENTS: EG x,y,z,w EGX=x EG a,b,c,d,e,f,g,h

Where x, y, z, w are the masters positions at which the X, Y, Z, W axis must be engaged.

USAGE:

While Moving	Yes
In a Program	No
Can be Interrogated	Yes
Not in a Program	Yes
Used as an Operand	Yes

RELATED COMMANDS:

"EB"	Enable CAM mode
"EQ"	Stop CAM motion for slaves

EXAMPLES:

EG 700, 1300	Engages the X and Y axes at the master position 700 and 1300 respectively.
B= _EGY	Returns the status of Y axis, 1 if engaged.

NOTE: This command is not a trip-point. This command will not hold the execution of the program flow. If the execution needs to be held until the master position is reached, use MF or MR command.

EM (ECAM Cycle)

DESCRIPTION:

The EM command is part of the ECAM mode. It is used to define the change in position over one complete cycle of the master. The field for the master axis is the cycle of the master position. For the slaves, the field defines the net change in one cycle. If a slave will return to its original position at the end of the cycle, the change is zero. If the change is negative, specify the absolute position.

NOTE: This command is not valid for setting the master's modulus if the controller has d150N19p or greater. Refer to the MM command in this case.

ARGUMENTS: EM x,y,z,w EMX=x EM a,b,c,d,e,f,g,h

where the parameters are positive integers in the range between 1 and 8,388,607 for the master axis and between 1 and 2,147,483,647 for the slave axis

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"EA"	Select master CAM axis
"EP"	Define CAM table intervals and start point
"ET"	CAM Table entries for the slave axes
"EB"	Enable CAM mode

EXAMPLES:

EAZ	Select Z axis as master for ECAM
EM 0,3000,2000	Define the changes in X and Y to be 0 and 3000 respectively. Define master cycle as 2000.
V=_EMX	Return cycle of X

EN (End)

DESCRIPTION:

The EN command is used to designate the end of a program or subroutine. If a subroutine has been called by the JS instruction, the EN command ends the subroutine and returns program flow to the line just after the JS command. The EN command is used to end the automatic subroutines #MCTIME, #CMDERR, and #COMINT. When the EN command is used to terminate the #COMINT communications interrupt subroutine, there are two arguments; the first determines whether trip-points will be restored upon completion of the subroutine and the second determines whether the communication interrupt will be re-enabled.

ARGUMENTS: EN m, n

m=0 Return from subroutine without restoring trip-point

m=1 Return from subroutine and restore trip-point

n=0 Return from #COMINT without re-enabling interrupt

n=1 Return from #COMINT and re-enable interrupt

Note 1: The default values for the argument are 0. For example, EN,1 and EN0,1 have the same effect

Note 2: Trip-points cause a program to wait for a particular event. The AM command, for example, waits for motion on all axes to complete. If the #COMINT subroutine is executed due to a communication interrupt while the program is waiting for a trip-point, the #COMINT can end by continuing to wait for the trip-point as if nothing happened, or clear the trip-point and continue executing the program at the command just after the trip-point. The EN arguments will specify how the #COMINT routine handles trippoints.

Note 3: Use the RE command to return from the interrupt handling subroutines #LIMSWI and #POSERR. Use the RI command to return from the #ININT subroutine.

USAGE:

While Moving	Yes	Default Value	n=0, m=0
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"RE"	Return from error subroutine
"RI"	Return from interrupt subroutine

EXAMPLES:

#A	Program A
PR 500	Relative position move
BGX	Begin the move
AMX	After the move is complete
PR 1000	Set another Position Relative move
BGX	Begin the move
EN	End of Program

NOTE: Instead of EN, use the RE command to end the #POSERR subroutine and #LIMSWI subroutine. Use the RI command to end the input interrupt (#ININT) subroutine.

EO (Echo)

DESCRIPTION:

The EO command turns the echo ON or OFF. If the echo is OFF, characters input over the bus will not be echoed back.

ARGUMENTS: EO n

where n=0 or 1. 0 turns echo off, 1 turns echo on.

USAGE:

While Moving	Yes	Default Value	1
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

EO 0	Turns echo off
EO 1	Turns echo on

EP (Cam table intervals and starting point)

DESCRIPTION:

The EP command defines the ECAM table intervals and offset. The offset is the master position of the first ECAM table entry. The interval is the difference of the master position between two consecutive table entries. This command effectively defines the size of the ECAM table. The parameter m is the interval and n is the starting point.

ARGUMENTS: EP m,n

Where m is a positive integer in the range between 1 and 32,767 and n is an integer between -2,147,483,648 and 2,147,483,647.

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Can be Interrogated	No
Used as an Operand	Yes (m only)

RELATED COMMANDS:

"EM"	Define CAM cycles for each axis
"ET"	CAM Table entries for each axis

EXAMPLE:

EP 20, 100	Sets the cam master points to 100, 120, 140....
D = _EP	Returns the interval (m)

EQ (ECAM quit (disengage))

DESCRIPTION:

The EQ command disengages an electric cam slave axis at the specified master position. Separate points can be specified for each axis. If a value is specified outside of the master's range, the slave will disengage immediately. _EQX returns 1 if axis is waiting to start, 2 if axis is waiting to stop, 3 if both waiting to start and stop and 0 if ECAM engaged or already stopped.

ARGUMENTS: EQ x,y,z,w EQX=x EQ a,b,c,d,e,f,g,h

Where x, y, z, w are the master positions at which the XYZW axes are to be disengaged.

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Can be Interrogated	No
Used as an Operand	Yes

RELATED COMMANDS:

"EG"	Start CAM motion for slaves

EXAMPLES:

EQ 300, 700	Disengages the X and Y motors at master positions 300 and 700, respectively

NOTE: This command is not a trip-point. This command will not hold the execution of the program flow. If the execution needs to be held until master position is reached, use MF or MR command.

ER (Error Limit)

DESCRIPTION:

The ER command sets the magnitude of the X,Y,Z and W-axis position errors that will trigger an error condition. When the limit is exceeded, the alarm light will come on. If the Off On Error (OE1) command is active, the motors will be disabled (MO). With firmware 2.0g or d15ON19n or greater, ER0 will disable the error limit for that axis. The alarm light will not come on, the #POSERR will not execute, and the motor will not disable if the OE is set to one. This can be useful if an encoder is not a servo, but an external master. The units of ER are quadrature counts.

ARGUMENTS: ER x,y,z,w ERX=x ER a,b,c,d,e,f,g,h

x,y,z,w are unsigned numbers in the range 0 to 32767

USAGE:

While Moving	Yes	Default Value	16384
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"OE"	Off-On Error
#POSERR	Automatic Error Subroutine

EXAMPLES:

ER 200,300,400,600	Set the X-axis error limit to 200, the Y-axis error limit to 300, the Z-axis error limit to 400, and the W-axis error limit to 600.
ER ,1000	Sets the Y-axis error limit to 1000, leave the X-axis error limit unchanged.
ER ?,?,?,?	Return X,Y,Z and W values
00200,01000,00400,00600	
ER ?	Return X value
00200	
V1=_ERX	Assigns V1 value of ERX
V1=	Returns V1
00200	

Hint: The error limit specified by ER should be high enough as not to be reached during normal operation. Examples of exceeding the error limit would be a mechanical jam, or a fault in a system component such as encoder or amplifier.

ES (Ellipse Scale)

DESCRIPTION:

The ES command divides the resolution of one of the axes in a vector mode. This allows the generation of an ellipse instead of a circle, or a circle path with ball screws of different pitch.

The command has two parameters, m and n, (ES m,n), and it applies to the axes designated by the VM command (VMXY, for example). When m>n, the resolution of the first axis (X in the example), will be divided by the ratio m/n. When m<n, the resolution of the second axis (Y in the example), will be divided by n/m. The resolution change applies for the purpose of generating the VP and CR commands. Note that this command results in one axis moving a distance specified by the CR and VP commands, while the other one moves a larger distance.

ARGUMENTS: ES m,n

where m and n are positive integers in the range between 1 and 65,535.

USAGE:

While Moving	Yes	Default Value	1,1
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"VM"	Vector Mode
"CR"	Circle move
"VP"	Vector position

EXAMPLES:

VMXY;ES3,4	Divide Y resolution by 4/3
VMZX;ES3,2	Divide Z resolution by 3/2

Hint: The ES command is useful for generating "true" circles or vectors when using ballscrews with various ratios.

ET (Electric cam table)

DESCRIPTION:

The ET command sets the ECAM table entries for the slave axes. The values of the master axes are not required. The slave entry (n) is the position of the slave axes when the master is at the point n * i + o, where i and o are the interval and offset as determined by the EP command.

ARGUMENTS: ET [n] = x, y, z, w ET [n] = a,b,c,d,e,f,g,h

where n is an integer between 0 and 256 and the parameters x, y, z, w are integers in the range between -2,147,438,648 and 2,147,438,647.

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	No

RELATED COMMANDS:

"EA"	Select master CAM axis
"EM"	Define CAM cycles for each axis
"EB"	Enable CAM mode
"EP"	Define CAM table intervals and start point

EXAMPLES:

ET [7] = 1000,300,500	Specifies the position of the slave axes X, Y, and Z that must be synchronized with the eighth increment of the master

NOTE: The table entry generated with the ET [n] command can be stored on the non-volatile memory of the controller. The BN command stores this table into the EEPROM.

FA (Acceleration Feed Forward)

DESCRIPTION:

The FA command sets the acceleration feed forward coefficient, or returns the previously set value. This coefficient, when scaled by the acceleration, adds a torque bias voltage during the acceleration phase and subtracts the bias during the deceleration phase of a motion. FA will only be operational during independent moves.

Acceleration Feed forward Bias = $FA \cdot AC \cdot 1.5 \cdot 10^{-7}$

Deceleration Feed forward Bias = $FA \cdot DC \cdot 1.5 \cdot 10^{-7}$

ARGUMENTS: FA x,y,z,w

where x,y,z,w are unsigned numbers in the range 0 to 4097 decimal

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	4.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"FV"	Velocity feed forward

EXAMPLES:

AC 500000,1000000	Specify acceleration rates
FA 10,15	Set feed forward coefficient to 10 for the X-axis and 15 for the Y-axis. The effective bias will be 0.75V for X and 2.25V for Y.
FA ?,?	Return X and Y values
010,015	

NOTE: If the feed forward coefficient is changed during a move, then the change will not take effect until the next move.

Hint: This command is useful for systems that require high acceleration in short time periods.

FE (Find Edge)

DESCRIPTION:

The FE command moves a motor until a transition is seen on the homing input for that axis. The direction of motion depends on the initial state of the homing input (use the CN command to configure the polarity of the home input). Once the transition is detected, the motor decelerates to a stop.

This command is useful for creating your own homing sequences.

ARGUMENTS: FE XYZW FE ABCDEFGH

where X,Y,Z,W specify XYZ or W axis. No argument specifies all axes.

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"FI"	Find Index
"HM"	Home
"BG"	Begin
"AC"	Acceleration Rate
"DC"	Deceleration Rate
"SP"	Speed for search

EXAMPLES:

FE	Set find edge mode
BG	Begin all axes
FEX	Only find edge on X
BGX	
FEY	Only find edge on Y
BGY	
FEZW	Find edge on Z and W
BGZW	

Hint: Find Edge only searches for a change in state on the Home Input. Use FI (Find Index) to search for the encoder index pulse. Use HM (Home) to search for both the Home input and the Index. Remember to specify BG after each of these commands.

FI (Find Index)

DESCRIPTION:

The FI and BG commands move the motor until an encoder index pulse is detected. The controller looks for a transition from low to high. When the transition is detected, motion stops and the position is defined as zero. To improve accuracy, the speed during the search should be specified as 1000 counts/s or less. The FI command is useful in custom homing sequences. The sign of the JG command specifies the direction of motion.

ARGUMENTS: FI XYZW FI ABCDEFGH

where X,Y,Z,W specify XYZ or W axis. No argument specifies all axes.

USAGE:

While Moving	**No**	**Default Value**	**---**
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"FE"	Find Edge
"HM"	Home
"BG"	Begin
"AC"	Acceleration Rate
"DC"	Deceleration Rate
"SP"	Search Speed

EXAMPLES:

#HOME	Home Routine
JG 500	Set speed and forward direction
FIX	Find index
BGX	Begin motion
AMX	After motion
MG "FOUND INDEX"	

Hint: Find Index only searches for a change in state on the index pulse. Use FE to search for the Home input. Use HM (Home) to search for both the Home input and the Index pulse. Remember to specify BG after each of these commands.

FL (Forward Software Limit)

DESCRIPTION:

The FL command sets the forward software position limit. When the command position exceeds the forward limit, motion on that axis will decelerate to a stop. Forward motion beyond this limit is not permitted. The forward limit is activated at x+1, y+1, z+1, w+1. The forward limit is disabled at 2147483647. The units are in quadrature counts.

ARGUMENTS: FL x,y,z,w FLX=x FL a,b,c,d,e,f,g,h

where x,y,z,w are signed integers in the range -2147483648 to 2147483647

2147483647 turns off the forward limit

USAGE:

While Moving	Yes	Default Value	2147483647
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"BL"	Reverse Limit

EXAMPLES:

FL 150000	Set forward limit to 150000 counts on the X-axis
#TEST	Test Program
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
FL 15000	Forward Limit
JG 5000	Jog Forward
BGX	Begin
AMX	After Limit
TPX	Tell Position
EN	End

@FRAC (Fraction function)

DESCRIPTION:

The Fraction (@FRAC[n]) function returns only the decimal portion of a number or variable given in square brackets. Note that the @FRAC command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @FRAC [n]

where n is a number in the range of –2147483647.9999 to 2147483647.9999

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=123.456	Set variable
MG @FRAC[VAR1]	Display only the fractional portion of VAR1
VAR2=@FRAC[VAR1]+.5	Perform calculation
EN	End of Program

FV (Velocity Feed Forward)

DESCRIPTION:

The FV command sets the velocity feed forward coefficient, or returns the previously set value. This coefficient, generates an output bias signal in proportion to the commanded velocity.

Velocity feed forward bias = $1.22 \cdot 10^{-6} \cdot FV \cdot Velocity$ [cts./sec.].

For example, if FV=10 and the velocity is 200,000 count/s, the velocity feed forward bias equals 2.44 volts.

ARGUMENTS: FV x,y,z,w FVX=x FV a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range 0 to 8191 decimal

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"FA"	Acceleration feed forward

EXAMPLES:

FV 10,20	Set feed forward coefficients to 10 and 20 for X and Y respectively. This produces 0.366 volts for X and 1.95 volts for Y.
JG 30000,80000	Set jog speeds
FV ?,?	Return the x and y values.
010,020	

GA (Master Axis for Gearing)

DESCRIPTION:

The GA command specifies the master axis for electronic gearing. Only one master may be specified. The master may be the main encoder input, auxiliary encoder input, or the commanded position of any axis. The master may also be the commanded vector move in a coordinated motion of LM or VM type. When the master is a simple axis, it may move in any direction and the slave follows. When the master is a commanded vector move, the vector move is considered positive and the slave will move forward if the gear ratio is positive, and backward if the gear ratio is negative. The slave axes and ratios are specified with the GR command.

ARGUMENTS: GA n

where n = X or Y or Z or W or A,B,C,D,E,F,G,H for main encoder as axis master

n = CX or CY or CZ or CW or CA,CB,CC,CD,CE,CF,CG,CH for command position as master axis

n = S for vector motion as master

n = DX or DY or DZ or DW or DA,DB,DC,DD,DE,DF,DG,DH for auxiliary encoder as master

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"GR"	Gear Ratio

EXAMPLES:

#GEAR	Gear program
GAX	Specify X axis as master
GR ,.5,-2.5	Specify Y and Z ratios
JG 5000	Specify master jog speed
BGX	Begin motion
WT 10000	Wait 10000 msec
STX	Stop

Hint: Using the commanded position as the master axis is useful for gantry applications. Using the vector motion as master is useful in generating helical motion.

GN (Gain)

DESCRIPTION:

The GN command sets the gain of the control loop or returns the previously set value. It fits in the z-transform control equation as follows:

$$D(z) = GN(z-ZR)/z$$

ARGUMENTS: GN x,y,z,w GNX=x GN a,b,c,d,e,f,g,h

where x,y,z,w are unsigned integers in the range 0 to 2047 decimal.

USAGE:

While Moving	Yes	Default Value	70
In a Program	Yes	Default Format	4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"ZR"	Zero
"KI"	Integrator
"KP"	Proportional
"KD"	Derivative

EXAMPLES:

GN 12,14,15,20	Set X-axis gain to 12 Set Y-axis gain to 14 Set Z-axis gain to 15 Set W-axis gain to 20
GN 6	Set X-axis gain to 6 Leave other gains unchanged
GN ,8	Set Y-axis gain to 8 Leave other gains unchanged
GN ?,?,?,?	Returns X,Y,Z,W gains
0006,0008,0015,0020	
GN ?	Returns X gain
0006	
GN ,?	Returns Y gain
0008	

GR (Gear Ratio)

DESCRIPTION:

GR specifies the Gear Ratios for the geared axes in the electronic gearing mode. The GAX, GAY, GAZ, or GAW command defines the master axis. The gear ratio may be different for each geared axis and range between +/-127.9999. The slave axis will be geared to the actual position of the master. The master can go in both directions. GR 0,0,0,0 disables gearing for each axis. A limit switch also disables the gearing.

ARGUMENTS: GR x,y,z,w GRX=x GR a,b,c,d,e,f,g,h

where x,y,z,w are signed numbers in the range +/-127, with a fractional resolution of 1/65535.

0 disables the gearing

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"GA"	Master Axis

EXAMPLES:

#GEAR	
MOY	Turn off servo to Y motor
GAY	Specify master axis as Y
GR .25,,-5	Specify X and Z gear ratios
EN	End program

When the Y motor is rotated by hand, the X will rotate at 1/4th the speed and Z will rotate 5 times the speed in the opposite direction.

Note: More gearing resolution can be achieved by using a calculation such as GRY=1/3 rather than GRY=0.3333. This is because there is actually a resolution of 1/65535 in the fraction portion of a number.

HM (Home)

DESCRIPTION:

The HM command performs a three-stage homing sequence for servo systems and a two stage sequence for stepper motor operation.

For servo motor operation:

The first stage consists of the motor moving at the user-programmed speed until it sees a transition on the homing input for that axis. The direction for this first stage is determined by the initial state of the Homing Input. Once the homing input changes state, the motor decelerates to a stop. The state of the homing input can be configured using the CN command.

The second state consists of the motor changing directions and slowly approaching the transition again. When it detects the transition, it stops.

The third stage consists of the motor moving forward at 256 cts/sec until it detects an index pulse from the encoder. It stops at this point and defines it as position 0.

For stepper motor operation:

The sequence consists only of the first two stages shown above. The frequency of the motion in stage 2 is 256 cts./sec.

ARGUMENTS: HM XYZW HM ABCDEFGH

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

When the HM command is used in an operand (_HMX), it returns the state of the home switch.

RELATED COMMANDS:

"CN"	Configure Home
"FI"	Find Index Only
"FE"	Find Home Only

EXAMPLES:

HM	Set Homing Mode for all axes
BG	Home all axes
BGX	Home only the X-axis
BGY	Home only the Y-axis
BGZ	Home only the Z-axis
BGW	Home only the W-axis

It is possible to customize the speed of the FI command, so if the HM command is too slow, use the FE and FI commands to make your own homing routines.

Hint: You can create your own custom homing sequence by using the FE (Find Home Sensor only) and FI (Find Index only) commands.

Note: _HM = 0 Always indicates that the home switch is active regardless of the state of the CN command.

HX (Halt Execution)

DESCRIPTION:

The HX command halts the execution of any of the four programs that may be running independently in multitasking. The parameter n specifies the program to be halted. If no parameter is specified, all threads will halt. If an HX command is given while moving, the motion will not stop (use the Abort command (AB) to stop motion).

When used as an operand, _HXn returns the running status of thread n with:

0 Thread not running

1 Thread is running

2 Thread has stopped at trip-point

ARGUMENTS: HXn

where n is an integer in the range of 0 to 3 to indicate the thread number.

USAGE:

While Moving	Yes	Default Value	n = 0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"XQ"	Execute program

EXAMPLES:

XQ #A	Execute program #A, thread zero
XQ #B,3	Execute program #B, thread three
HX0	Halt thread zero
HX3	Halt thread three

II (Input Interrupt)

DESCRIPTION:

The II command enables the interrupt function for the specified inputs. m specifies the beginning input and n specifies the final input in the range. For example, II 2,4 specifies interrupts occurring for Input 2, Input 3 and Input 4. m=0 disables the Input Interrupts. If only the m parameter is given, only that input will generate an interrupt.

The parameter o is an interrupt mask for all eight inputs. If m and n are unused, o contains a number with the mask. A 1 designates that input to be enabled for an interrupt.

Example: II,,5 enables inputs 1 and 3

If any of the specified inputs go low during program execution, the program will jump to the subroutine with label #ININT. The RI command is used to return from the #ININT routine. The RI command also re-enables input interrupts. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack.

ARGUMENTS: II m,n,o

where:

m is an integer in the range 0 to 8 decimal

n is an integer in the range 1 to 8 decimal

o is an integer in the range 0 to 255 decimal

with firmware type n19K or higher, the interrupts can be either active high or low:

m is an integer in the range -8 to 8 decimal

n is an integer in the range -8 to –1or 1 to 8 decimal

o is an integer in the range -255 to 255 decimal

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0 (mask only)
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

“RI”	Return from Interrupt
#ININT	Interrupt Subroutine
“After Input (AI”	After Input

EXAMPLES:

#A	Program A
II 4	Specify interrupt on input 4
JG 5000	Specify jog speed
BGX	Begin motion
#LOOP;JP #LOOP	Loop
EN	End Program

#ININT	Interrupt subroutine
STX;MG “INTERRUPT”	Stop X, print message
AMX	After stopped
AI 4	Check for interrupt clear (input 4 goes high)
BGX	Begin motion
RI	Return to main program

IL (Integrator Limit)

DESCRIPTION:

The IL command limits the effect of the integrator function in the filter to a certain voltage. For example, IL 2 limits the output of the integrator of the X-axis to the +/-2 Volt range.

A negative parameter also freezes the effect of the integrator during the move. For example, IL -3 limits the integrator output to +/-3V. If, at the start of the motion, the integrator output is 1.6 Volts, that level will be maintained through the move. Please note, however, that the KD and KP terms remain active in any case.

ARGUMENTS: IL x,y,z,w ILX=x IL a,b,c,d,e,f,g,h

where x,y,z,w are numbers in the range –9.9988 to 9.9988 Volts.

USAGE:

While Moving	Yes	Default Value	9.9988
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"KI"	Integrator

EXAMPLES:

KI 2,3,5,8	Integrator constants
IL 3,2,7,2	Integrator limits
IL ?	Returns the X-axis limit
3.0000	
IL*=2	Set all axes

IN (Input Variable)

DESCRIPTION:

The IN command allows data to be input from a serial port. When the IN command is executed in a program, the prompt message is displayed. The operator then enters the variable value followed by a carriage return. The entered value is assigned to the specified variable name.

The IN command suspends execution of following commands in program thread zero until a carriage return or semicolon is received. If no value is given prior to a semicolon or carriage return, the previous variable value is kept. Input Interrupts, Error Interrupts and Limit Switch Interrupts will remain active.

ARGUMENTS: IN {P2}"m",n

where:

m is an optional prompt message

n is the variable name (n cannot be an array location)

{P2} is only required when interfacing to port 2. The default is port 1

{So} specifies string data and o is the number of characters from 1 to 6.

Note 1: The limit on the number of characters for m is 60 characters or less per line.

Note 2: Configure Port 2 communications with the CC command before using IN command with Port 2.

Note 3: IN command can only be used in thread 0.

USAGE:

While Moving	Yes	Default Value	{P1}
In a Program	Yes	Default Format	Position Format
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

Operator specifies length of material to be cut in inches and speed in inches/sec (2-pitch lead screw, 2000 counts/rev encoder).

#A	Program A
IN "Enter Speed(in/sec)",V1	Prompt operator for speed
IN "Enter Length(in)",V2	Prompt for length
V3=V1*4000	Convert units to counts/sec
V4=V2*4000	Convert units to counts
SP V3	Speed command
PR V4	Position command
BGX	Begin motion
AMX	Wait for motion complete
MG "MOVE DONE"	Print Message
EN	End Program

Note: It is good practice to clear the input buffer before executing the IN command.

@IN (Status of Digital Input Function)

DESCRIPTION:

The Digital Input(@IN[n]) function returns the status of the digital input number or variable given in square brackets. Note that the @IN command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @IN [n]

where n is an unsigned integer in the range 1 to 8 decimal

where n is an unsigned integer in the range 1 to 24 for SMC-20008

where n is an unsigned integer in the range 1 to 64 for an SMC-2000 with the "I", "D", "P", or "S" option.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=2	Set variable
MG @IN[VAR1]	Display the status of digital input 2
VAR2=@IN[VAR1]+1	Perform calculation
EN	End of Program

@INT (Integer function)

DESCRIPTION:

The Integer (@INT[n]) function returns only the whole number part of a number or variable given in square brackets. Note that the @INT command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @INT [n]

where n is a number in the range of –2147483648.9999 to 2147483647.9999

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=123.456	Set variable
MG @INT[VAR1]	Display only the whole number portion of VAR1
VAR2=@INT[VAR1]+25	Perform calculation
EN	End of Program

IP (Increment Position)

DESCRIPTION:

The IP command allows for a change in the command position while the motor is moving. This command does not require a BG (Begin Command). The command has three effects depending on the motion being executed. The units of this command are quadrature.

Case 1: Motor is standing still (no move profile in progress)

An IP x,y,z,w command is equivalent to a PR x,y,z,w and BG command. The motor will move to the specified position at the requested slew speed and acceleration.

Case 2: Motor is moving towards specified position

An IP x,y,z,w command will cause the motor to move to a new position target, which is the old target plus x,y,z,w.

x,y,z,w must be in the same direction as the existing motion, or a command error will result.

Case 3: Motor is in the Jog Mode

An IP x,y,z,w command will cause the motor to instantly try to servo to a position x,y,z,w from the present instantaneous position. The SP and AC parameters have no effect. This command is useful when synchronizing 2 axes in which one of the axis' speed is indeterminate due to a variable diameter pulley.

ARGUMENTS: IP x,y,z,w IPX = x IP a,b,c,d,e,f,g,h

x,y,z,w are signed numbers in the range -2147483648 to 2147483647 decimal.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	7.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

IP 50	50 counts with set acceleration and speed
#CORRECT	Label
AC 100000	Set acceleration
JG 10000;BGX	Jog at 10000 counts/sec rate
WT 1000	Wait 1000 msec
IP 10	Move the motor 10 counts instantaneously
STX	Stop Motion

IT (Independent Time Constant - Smoothing Function)

DESCRIPTION:

The IT command filters the acceleration and deceleration functions in independent moves of JG, PR, PA type to produce a smooth velocity profile. The resulting profile, known as S-curve, has continuous acceleration and results in reduced mechanical vibrations. IT sets the bandwidth of the filter where 1 means no filtering and 0.004 means maximum filtering. NOTE that the filtering results in longer motion time.

ARGUMENTS: IT x,y,z,w ITX=x IT a,b,c,d,e,f,g,h

where x,y,z,w are positive numbers in the range between 0.004 and 1.0 with a resolution of 1/256

USAGE:

While Moving	Yes	Default Value	1.0
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VT"	Vector Time Constant for smoothing vector moves

EXAMPLES:

IT 0.8, 0.6, 0.9, 0.1	Set independent time constants for x,y,z,w axes
IT ?	Return independent time constant for X-axis
0.8	

JG (Jog)

DESCRIPTION:

The JG command sets the jog mode. The parameters following the JG set the slew speed of the axes. Use of the question mark returns the previously entered value or default value. The units of this are counts per second.

ARGUMENTS: JG x,y,z,w JGX=x JG a,b,c,d,e,f,g,h

where: x,y,z,w are signed numbers in the range 0 to +/-8,000,000 decimal

USAGE:

While Moving	Yes	Default Value	25000
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"BG"	Begin
"ST"	Stop
"AC"	Acceleration
"DC"	Deceleration
"IP"	Increment Position
"TV"	Tell Velocity

EXAMPLES:

JG 100,500,2000,5000	Set for jog mode with a slew speed of 100 counts/sec for the X-axis, 500 counts/sec for the Y-axis, 2000 counts/sec for the Z-axis, and 5000 counts/sec for W-axis.
BG	Begin Motion
JG ,,-2000	Change the Z-axis to slew in the negative direction at -2000 counts/sec.

JP (Jump to Program Location)

DESCRIPTION:

The JP command causes a jump to a program location on a specified condition. The program location may be any program line number or label. The condition is a conditional statement that uses a logical operator such as equal to or less than. A jump occurs if the specified condition is true. **WITH FIRMWARE VERSIONS 2.0C AND HIGHER ONLY** you may combine conditions with logical operators such as AND (&) , and OR (|) by enclosing the individual conditions within parentheses.

Example JP #ERROR, (SPEED<0)|(SPEED>450)

ARGUMENTS: JP location, condition

where: location is a program line number or label

condition is a conditional statement using a logical operator (optional)

The logical operators are:

< less than

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal to

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	No
Can be Interrogated	No
Used in an Operand	No

EXAMPLES:

JP #POS1,V1<5	Jump to label #POS1 if variable V1 is less than 5
JP #A,V7*V8=0	Jump to #A if V7 times V8 equals 0
JP #B	Jump to #B (no condition)

Hint: JP is similar to an IF, THEN command. Text to the right of the comma is the condition that must be met for a jump to occur. The destination is the specified label before the comma.

JS (Jump to Subroutine)

DESCRIPTION:

The JS command will change the sequential order of execution of commands in a program. If the jump is taken, program execution will continue at the line specified by the destination parameter, which can be either a line number or label. The line number of the JS command is saved and after the next EN command is encountered (End of subroutine), program execution will continue with the instruction following the JS command. There can be a JS command within a subroutine. These can be nested 16 deep in the SMC-2000. **WITH FIRMWARE VERSIONS 2.0C AND HIGHER ONLY** you may combine conditions with logical operators such as AND (&) , and OR (|) by enclosing the individual conditions within parentheses.

A jump is taken if the specified condition is true. Conditions are tested with logical operators. The logical operators are:

< less than or equal to

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal

ARGUMENTS: JS destination, condition

where destination is a line number or label

condition is a conditional statement using a logical operator (optional)

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"EN"	End

EXAMPLES:

JS #SQUARE,V1<5	Jump to subroutine #SQUARE if V1 is less than 5
JS #LOOP,V1<>0	Jump to #LOOP if V1 is not equal to 0
JS #A	Jump to subroutine #A (no condition)

KD (Derivative Constant)

DESCRIPTION:

KD designates the derivative constant in the controller filter. The filter transfer function is

D(z) = KP + KD(z-1)/z + KI . z/(z-1)

For further details on the filter see the section Theory of Operation.

ARGUMENTS: KD x,y,z,w KDX=x KD a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range 0 to 4095.875 with a resolution of 1/8.

USAGE:

While Moving	Yes	Default Value	64
In a Program	Yes	Default Format	4.2
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"KP"	Proportional Constant
"KI"	Integral

EXAMPLES:

KD 100,200,300,400.25	Specify KD
KD ?,?,?,?	Return KD
0100.00,0200.00,0300.00,0400.25	

KI (Integrator)

DESCRIPTION:

The KI command sets the integral gain of the control loop. It fits in the control equation as follows:

D(z) = KP + KD(z-1)/z + KI z/(z-1)

The integrator term will reduce the position error at rest to zero.

ARGUMENTS: KI x,y,z,w KIX=x KI a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range 0 to 2047.875 with a resolution of 1/8

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	4.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"GN"	Gain
"KP"	Proportional Gain
"KD"	Derivative
"ZR"	Zero
"IL"	Integrator Limit

EXAMPLES:

KI 12,14,16,20	Specify x,y,z,w-axis integral
KI 7	Specify x-axis only
KI ,,8	Specify z-axis only
KI ?,?,?,?	Return X,Y,Z,W
0007,0014,0008,0020	KI values

KP (Proportional Constant)

DESCRIPTION:

KP designates the proportional constant in the controller filter. The filter transfer function is

$$D(z) = KP + KD(z-1)/z + KIz/(z-1)$$

For further details see the section Theory of Operation.

ARGUMENTS: KP x,y,z,w KPX=x KP a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range 0 to 1023.875 with a resolution of 1/8.

USAGE:

While Moving	Yes	Default Value	6
In a Program	Yes	Default Format	4.2
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"KP"	Proportional Gain
"KI"	Integral constant

KS (Stepper Motor Smoothing)

DESCRITPION

KS parameter smoothes the frequency of the step motor pulses. Larger values of KS provide greater smoothness. This parameter will also increase the motion time by 3*KS sampling periods. KS adds a single pole low pass filter onto the output of the motion profiler. This function smoothes out the generation of step pulses and is most useful when operating in full or half step mode.

Note: The KS function will cause the step output to be delayed.

ARGUMENTS: KS x,y,z,w, KSX=x KS a,b,c,d,e,f,g,h

where x,y,z,w are positive integers in the range between 0.5 and 16 with a resolution of 1.

USAGE:

While Moving	Yes	Default Value	2
In a Program	Yes	Default Format	4.0
Command Line	Yes		
Can be Interrogated	Yes	KS ?,?,?,?	
Used in an Operand	Yes		

OPERAND USAGE:

_KSn contains the value of the step motor smoothing constant for the specified axis ‘n’.

RELATED COMMANDS:

“MT”	Motor Type

EXAMPLES:

KS 2, 4, 8	Specify x,y,z axes
KS 5	Specify x-axis only
KS ,,15	Specify z-axis only

Note: KS is valid for step motors only

LA (List Arrays)

DESCRITPION

The LA command returns a list of all arrays in memory. The listing will be in alphabetical order. The size of each array will be included next to each array name in square brackets. This command is valid only with firmware 2.0g and higher, or d15ON19o and higher.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"LL"	List Labels
"LS"	List Program
"LV"	List Variable

EXAMPLES:

:LA	List Arrays
CA[10]	
LA [5]	
NY[25]	
VA[17]	

LC (Lock Controller)

DESCRIPTION:

The (LC) Lock Controller command is used to prohibit the execution of certain commands from the serial port by setting a security password. See the table below for a list of commands that are disabled in the "Locked" mode.

ARGUMENTS: LC p,l

where p is the password as previously established with the "PW" command.

"l" is the Lock setting, 0=Unlock, 1=Lock commands (see table), 2=Lock commands and prohibit setting any commands from the serial port.

USAGE:

While Moving	Yes	Default Value	0
In a Program	No	Default Format	8.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	No		

RELATED COMMANDS:

"PW"	PassWord

COMMANDS DISABLED UNDER LOCK STATUS 1 & 2:

BN (Burn Parameters)	TR (Trace Mode)
BP (Burn Program)	DL (DownLoad)
BV (Burn Variables)	LS (List Program)
UL (Upload)	

EXAMPLES:

PW MOTION,MOTION	Set a new password "MOTION"
LC MOTION,1	Lock controller
LC MOTION,0	Unlock controller
MG _LC	Returns Lock State (0,1,2)

LE (Linear Interpolation End)

DESCRIPTION:

LE signifies the end of a linear interpolation sequence. It follows the last LI specification in a linear sequence. The controller will calculate the linear move profile such that the motors will decelerate to a stop by the end of the last LI segment before the LE. LE? or _LE returns the length of the vector in counts. The VE command is interchangeable with the LE command.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"LI"	Linear Distance
"BG"	BGS - Begin Sequence
"LM"	Linear Interpolation Mode
"VS"	Vector Speed
"VA"	Vector Acceleration
"VD"	Vector Deceleration

EXAMPLES:

LM ZW	Specify linear interpolation mode
LI ,,100,200	Specify linear distance
LE	End linear move
BGS	Begin motion

LF (Forward Limit)

DESCRIPTION:

The _LF operand contains the state of the forward limit switch for the specified axis. A zero always means the limit is activated no matter what setting (active low or active high). This is not a command, but a status bit.

ARGUMENTS: _LFn

Where n is the axis letter

EXAMPLES:

MG _LFX	Display the status of the X axis forward limit switch
JP#A,_LFZ=0	Jump to label #A if Z axis forward limit switch is activated

LI (Linear Interpolation Distance)

DESCRIPTION:

The LI x,y,z,w command specifies the incremental distance of travel for each axis in the Linear Interpolation (LM) mode. LI parameters are relative distances given with respect to the current axis positions. Up to 511 LI specifications may be given ahead of the Begin Sequence (BGS) command. Additional LI commands may be sent during motion when the SMC-2000 sequence buffer frees additional spaces for new vector segments. The Linear End (LE) command must be given after the last LI specification in a sequence. This command tells the controller to decelerate to a stop at the last LI command.

It is the responsibility of the user to keep enough LI segments in the SMC-2000 sequence buffer to ensure continuous motion. LM? or _LM returns the available spaces for LI segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 LI segments can be sent. A zero means the buffer is full and no additional segments can be sent. It should be noted that the SMC-2000 computes the vector speed based on the axes specified in the LM mode. For example, LM XYZ designates linear interpolation for the X,Y and Z-axes. The speed of these axes will be computed from $VS^2=XS^2+YS^2+ZS^2$ where XS, YS and ZS are the speed of the X,Y and Z axes. If the LI command specifies only X and Y, the speed of Z will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed.

ARGUMENTS: LI x,y,z,w LI a,b,c,d,e,f,g,h

x,y,z,w are signed integers in the range -8,388,607 to 8,388,607 and represent incremental move distance

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"BGS"	BGS - Begin sequence
"LM"	Linear Interpolation Mode
"LE"	Linear end
"CS"	Clear Sequence
"VS"	Vector Speed
"VA"	Vector Acceleration
"VD"	Vector Deceleration

EXAMPLES:

LM XYZ	Specify linear interpolation mode
LI 1000,2000,3000	Specify distance
LE	Last segment
BGS	Begin sequence

LL (List Labels)

DESCRITPION

The LL command returns a list of all program labels in memory. The listing will be in alphabetical order. This command is valid only with firmware 2.0g and higher, or d15ON19o and higher.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

“LA”	List Arrays
“LS”	List Program
“LV”	List Variable

EXAMPLES:

:LL	List Labels
# FIVE	
# FOUR	
# ONE	
# THREE	
# TWO	

LM (Linear Interpolation Mode)

DESCRIPTION:

The LM XYZW command specifies the linear interpolation mode where XYZW denote the axes for linear interpolation. Any set of axes may be used for linear interpolation. LI x,y,z,w commands are used to specify the travel distances for linear interpolation. The LE command specifies the end of the linear interpolation sequence. Several LI commands may be given as long as the SMC-2000 sequence buffer has room for additional segments. _LM or LM? may be used to return the number of spaces available in the sequence buffer for additional LI commands. Once the LM command has been given, it does not need to be given again unless the VM command has been used.

It should be noted that the SMC-2000 computes the vector speed based on the axes specified in the LM mode. For example, LM XYZ designates linear interpolation for the X,Y and Z-axes. The speed of these axes will be computed from $VS^2=XS^2+YS^2+ZS^2$, where XS, YS and ZS are the speed of the X,Y and Z-axes. If the LI command specifies only X and Y, the speed of Z will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed.

ARGUMENTS: LM XYZW LM ABCDEFGH

XYZW denote X,Y,Z or W axes

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

“LE”	Linear end
“LI”	Linear distances
“BG”	BGS - Begin sequence
“VA”	Vector acceleration
“VS”	Vector Speed
“VD”	Vector deceleration
“AV”	Vector distance
“CS”	_CS - Sequence counter

EXAMPLES:

LM XYZW	Specify linear interpolation mode
VS 10000; VA 100000;VD 1000000	Specify vector speed, acceleration and deceleration
LI 100,200,300,400	Specify linear distance
LI 200,300,400,500	Specify linear distance
LE; BGS	Last vector, then begin motion

LR (Reverse Limit)

DESCRIPTION:

The _LR operand contains the state of the reverse limit switch for the specified axis. A zero always means the limit is activated no matter what setting (active high or low). This is not a command, but a status bit.

ARGUMENTS: _LRn

Where n is the axis letter

EXAMPLES:

MG _LRX	Display the status of the X axis reverse limit switch
JP#A,_LRZ=0	Jump to label #A if Z axis reverse limit switch is activated

LS (List Program)

DESCRIPTION:

The LS command sends a listing of the programs in memory to the PC bus. The listing will start with the line pointed to by the first parameter, which can be either a line number or a label. If no parameter is specified, it will start with line 0. The listing will end with the line pointed to by the second parameter--either a line number or label. If no parameter is specified, the listing will go to the last line of the program.

ARGUMENTS: LS n,m

where n,m are valid numbers from 0 to 999, or labels. n is the first line to be listed, m is the last.

USAGE:

While Moving	Yes	Default Value	0,Last Line
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

:LS #A,6	List program starting at #A through line 6
002 #A	
003 PR 500	
004 BGX	
005 AM	
006 WT 200	

LT (Latch Target)

DESCRIPTION

The LT command is used for stopping an axis a defined distance after a registration mark (latch) input. The distance specified by the LT command is in encoder counts. The distance must be sufficiently large for the controller to decelerate normally at the specified deceleration rate. A stop code will be generated if the distance is too small to stop for the deceleration rate or if the speed is too high.

ARGUMENTS: LTX=x LTx,y,z,w LTa,b,c,d,e,f,g,h

POSSIBLE STOP CODES

1	Motors stopped at commanded independent position (Latch input not received)
40	Stopped at Latch Target
41	Latched target overrun due to limit switch or stop comment
42	Latched target overrun due to insufficient distance

USAGE:

While Moving	Yes	Default Value	2,147,483,647
In a Program	Yes	Default Format	10.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

AL	Arm Latch
RL	Report Latch

EXAMPLES:

ALX	Set latch function
LTX=25000	Set Latch Target to stop 25000 counts after registration
PRX=100000	Position Relative Move
BGX	Begin Motion
AMX	After Motion Trip-point
JP #NOMARK_SCX=1	Jump to #NOMARK routine if did not receive a registration mark

Note: The LT command must be issued before each move requiring a stop upon registration.

LV (List Variables)

DESCRITPION

The LV command returns a list of all program variables in memory. The listing will be in alphabetical order. This command is valid only with firmware 2.0g and higher, or d15ON19o and higher.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"LA"	List Arrays
"LS"	List Program
"LL"	List Labels

EXAMPLES:

:LV	List Variables
APPLE = 60.0000	
BOY = 25.0000	
ZEBRA= 37.0000	

LZ (Leading Zero)

DESCRIPTION:

The LZ command is used for formatting the values returned from the interrogation commands or interrogation of variables and arrays. By enabling the LZ function, all leading zeros of returned values will be removed.

ARGUMENTS: LZ n

where n=1 suppresses and n=0 does not.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

OPERAND USAGE:

_LZ will report the current setting of the Leading Zero (LZ) command.

Note: LZ is only available for firmware versions 1.1h and higher, 2.0 or higher, and all D15ON19 versions.

MC (Motion Complete - "In Position")

DESCRIPTION:

The MC command is a trip-point used to control the timing of events. This command will delay execution of the following commands until the current move on the specified axis or axes is completed and the encoder reaches or passes the specified position. Any combination of axes or a motion sequence may be specified with the MC command. For example, MC XY waits for motion on both the X and Y-axis to be complete. MC with no parameter specifies that motion on all axes is complete. TW x, y, z, w sets the timeout if the encoder is not in position within the specified time. If a timeout occurs, the trip-point will clear and the stop code will be set to 99. The program will jump to the special label #MCTIME if it is included in the program.

When used in stepper mode, the controller will hold up execution of the proceeding commands until the controller has generated the same number of steps as specified in the commanded position. The actual number of steps that have been generated can be monitored by using the interrogation command TD. Note: the MC command is useful when operating with stepper motors since the step pulses can be delayed from the commanded position due to the stepper motor smoothing function, KS.

ARGUMENTS: MC XYZWS MC ABCDEFGH

Where X, Y, Z, W are XYZW axes. No argument specifies that motion on all axis is complete.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

BG	_BGx (returns a 0 if motion complete)
AM	After Motion
TW	Timeout for in Position

EXAMPLES:

#MOVE	Program MOVE
PR 5000,5000,5000,5000	Position relative moves
BG X	Start the X-axis
MC X	After the move is complete on X
BG Y	Start the Y-axis
MC Y	After the move is complete on Y
BG Z	Start the Z-Axis
MC Z	After the move is complete on Z
BG W	Start the W-Axis
MC W	After the move is complete on W
EN	End of Program
#F; DP 0,0,0,0	Program F Position
PR 5000,6000,7000,8000	Position relative moves
BG	Start X, Y, Z, W axes
MC	After motion complete on all axes
MG "DONE"; TP	Print message

EN	End of Program

Hint: MC can be used to verify that the actual motion has been completed.

MF (Forward Motion to Position)

DESCRIPTION:

The MF command is a trip-point used to control the timing of events. This command will hold up the execution of the following command until the specified motor moves forward and crosses the position specified. The units of the command are in quadrature counts. Only one axis may be specified at a time. The MF command can also be used when the encoder is the master and not under servo control.

ARGUMENTS: MFx or MF,y or MF,,z or MF,,,w MFX = X MF abcdefgh

Where x, y, z, w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

AD	After Distance
AP	After Absolute Position
MR	Reverse Motion To Position

EXAMPLES:

#TEST	Test Program
DP0	Define zero
JG 1000	Jog Mode (speed of 1000 counts/sec)
BG X	Begin Move
MF 2000	Wait for forward position
V1=_TPX	Assign V1 X position
MG “Position is”,V1	Print Message
ST	Stop
EN	End Program

Hint: The accuracy of the MF command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. MF tests for absolute position. The MF command can also be used when the specified motor is driven independently by an external device.

MG (Message)

DESCRIPTION:

The MG command sends data out a serial port. This can be used to alert an operator, send data or return a variable value.

ARGUMENTS: MG {P2}"m",V{Fm.n} {N}

{P2} is only required when interfacing to port 2. The default is port 1

"m" is a text message including letters, numbers, symbols or <ctrl>G (up to76 characters).

V is a variable name.

{Fm.n} designates decimal format with m digits to left of decimal, and n to the right. Hex format is specified by {$m.n}.

{N} suppresses carriage return line feed.

USAGE:

While Moving	Yes	Default Value	{P2}
In a Program	Yes	Default Format	Variable Format
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

Case 1: Message command displays ASCII strings

MG "Good Morning"	Displays the string

Case 2: Message command displays variables or arrays with specified format

MG "The Answer is", Total {F4.2}	Displays the string with the content of variable TOTAL in local format of 4 digits before and 2 digits after the decimal point.

Case 3: Message command sends any ASCII characters to the port.

MG {^13}, {^30}, {^37}, {N}	Sends carriage return, characters 0 and 7 followed by no carriage return line feed command to the port.

Hint: Message commands must be sent to an actual device. If an MG command is sent to a port with no device attached, the program will stop execution when the message buffer is full (256 bytes).

MM (Master's Modulus)

DESCRIPTION:

The (MM) Master's Modulus command is part of the ECAM mode. It is used to define the change in position over one complete cycle of the master. This command is only valid on firmware versions d150N19p and greater. For this firmware, this command replaces the master moodulus setting with the EM command. This is because the firmware was enhanced to allow camming with the auxiliary encoder as the master. In this case, it is possible for the AUX X axis to be the master, and the MAIN X to be the slave.

ARGUMENTS: MM v

where v is the value of the masters modulus.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	8.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"EA"	Select master cam axis
"EP"	Define cam table intervals and start point
"ET"	Cam table entries for the slave axes
"EB"	Enable ECAM mode

EXAMPLES:

EADX	Select Auxiliary X axis as ECAM master
MM 30500	Set master modulus
EM 20000	Set main X axis slave modulus
MG _MM	Return master modulus

MO (Motor Off)

DESCRIPTION:

The MO command shuts off the control algorithm and the servo enable signal. The controller will continue to monitor the motor position. To turn the motor back on use the Servo Here command (SH).

ARGUMENTS: MO XYZW MO ABCDEFGH

where X,Y,Z,W are XYZW axes

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

OPERAND USAGE:

_MOX will report the current status of the servo enable signal. Zero means the axis is enabled and one means the axis is disabled.

RELATED COMMANDS:

"SH"	Servo Here

EXAMPLES:

MO	Turn off all motors
MOX	Turn off the X motor. Leave the other motors unchanged
MOY	Turn off the Y motor. Leave the other motors unchanged
MOZX	Turn off the Z and X motors. Leave the other motors unchanged
SH	Turn all motors on
BOB=_MOX	Sets BOB equal to the X-axis servo status
BOB=	Return value of BOB. If 1, in motor off mode, If 0, in servo mode

MR (Reverse Motion to Position)

DESCRIPTION:

The MR command is a trip-point used to control the timing of events. This command will hold up the execution of the following command until the specified motor moves backward and crosses the position specified. The units of the command are in quadrature counts. Only one axis may be specified at a time. The MR command can also be used when the encoder is the master and not under servo control.

ARGUMENTS: MRx or MR,y or MR,,z or MR,,,w MRX=X MR abcdefgh

where x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AD"	After Distance
"AP"	After Position
"MF"	Forward motion to position

EXAMPLES:

#TEST	Program TEST
DP0	Define zero
JG -1000	Jog mode (speed of -1000 counts/sec)
BG X	Begin move
MR -3000	After passing the position -3000
V1=_TPX	Assign V1 X position
MG "Position is",V1	Print Message
ST	Stop
EN	End of Program

Hint: The accuracy of the MR command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. MR tests for absolute position. The MR command can also be used when the specified motor is driven independently by an external device.

MT (Motor Type)

DESCRIPTION:

The MT command selects the type of the motor and the polarity of the drive signal. Motor type includes standard servomotors that require a voltage in the range of +/- 10 Volts. The polarity reversal inverts the analog signals for servomotors, and inverts logic level of the pulse train for stepper motors.

ARGUMENTS: MT x,y,z,w MTX=x MT a,b,c,d,e,f,g,h

where x,y,z,w are integers with

1 - Servo motor

-1 - Servomotor reversed polarity

2 - Stepper motor

-2 - Stepper motor with active low pulses

2.5 - Stepper motor with active low pulses and reversed polarity

-2.5 – Stepper motor with active high pulses and reversed polarity

USAGE:

While Moving	Yes	Default Value	1,1,1,1
In a Program	Yes	Default Format	1
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

OPERAND USAGE:

_MTx reports the value of the motor type setting.

EXAMPLES:

MT 1,-1	Configure X as servo, and Y as reverse servo
MT ?,?	Interrogate motor type
V=_MTX	Assign motor type to variable

WARNING: This command works closely with the Configure Encoder (CE) command. Assuming your system is operating normally, but in the wrong direction, you must change both the (MT) and the (CE) commands under (MO) conditions. Failure to perform this change correctly will lead to system run away!

NO (No Operation)

DESCRIPTION:

The NO command performs no action in a sequence, but can be used as a comment in a program. After the NO, up to 78 characters can be given to form a program comment. This helps to document a program.

ARGUMENTS: NO m

where m is any group of letters, numbers, symbols or <control>G except the semicolon, ;. The semicolon delimits commands, therefore, a NO command must be given prior to any commands that are on the same line and are delimited by the semicolon.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

#A	Program A
NO	No Operation
NO This Program	No Operation
NO Does Absolutely	No Operation
NO Nothing	No Operation
EN	End of Program

OB (Output Bit)

DESCRIPTION:

The (OB n, logical expression) command sets an output bit as either 0 or 1 depending on the result from the logical expression. Any non-zero value of the expression results in a one on the output. **WITH FIRMWARE VERSIONS 2.0C AND HIGHER ONLY** you may combine conditions with logical operators such as AND (&) , and OR (|) by enclosing the individual conditions within parentheses.

Example OB2, (_TTX>0)&(_TTX<7)

ARGUMENTS: OB n, expression

where n is 1 through 8 for SMC-20004

where n is 1 through 16 for SMC-20008

where n is an integer in the range 1 to 64 (Outputs 24,32,40,48, 56 and 64 do not physically exist) decimal for SMC-2000xI (extended I/O)

expression is any valid logical expression, variable or array element.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

OB 1, POS1	If POS1 is non-zero, Bit 1 is high.
	If POS1 is zero, Bit 1 is low
OB 2, @IN[1]&@IN[2]	If Input 1 and Input 2 are both high, then
	Output 2 is set high
OB 3, COUNT[1]	If the element 1 in the array is zero, clear bit 3, otherwise set bit 3
OB N, COUNT[1]	If element 1 in the array is zero, clear bit N

OE (Off on Error)

DESCRIPTION:

The OE command enables/disables the "Off-On-Error" function. When enabled, the SMC-2000 will shut off the motor command under the following error conditions: if a position error exceeds the error limit specified by the ER command. An Abort either from the Abort input or the Abort command will shut off the motor.

If a position error is detected on an axis, and the axis is currently making an independent move, only that axis will be shut off. However, if the motion is a coordinated mode of the types VM, LM, or CM, all the participating axes will be stopped.

ARGUMENTS: OE x,y,z,w

where x,y,z,w may be 0 or 1; 0 disables function, 1 enables "Off on Error."

USAGE

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AB"	Abort
"ER"	Error limit
"SH"	Servo Here
#POSERR	Error Subroutine

EXAMPLES:

OE 1,1,1,1	Enable OE on all axes
OE 0	Disable OE on X-axis other axes remain unchanged
OE ,,1,1	Enable OE on Z-axis and W-axis other axes remain unchanged
OE 1,0,1,0	Enable OE on X and Z-axis Disable OE on Y and W axis

Hint: The OE command is useful for preventing system damage on excessive error.

OF (Offset)

DESCRIPTION:

The OF command sets a bias voltage in the motor command output or returns a previously set value. This can be used to counteract gravity or an offset in an amplifier.

ARGUMENTS: OF x,y,z,w OFX=x OF a,b,c,d,e,f,g,h

where x,y,z,w are signed numbers in the range -9.998 to 9.998 volts with resolution of .001.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

OF 1,-2,3,5	Set X-axis offset to 1, the Y-axis offset to -2, the Z-axis to 3, and the W-axis to 5
OF -3	Set X-axis offset to -3 Leave other axes unchanged
OF ,0	Set Y-axis offset to 0 Leave other axes unchanged
OF ?,?,?,?	Return offsets
-3.0000,0.0000,3.0000,5.0000	
OF ?	Return X offset
-3.0000	
OF ,?	Return Y offset
0.0000	

OP (Output Port)

DESCRIPTION:

The OP command sets an entire output port on the controller. The first field controls the standard output ports and the remaining fields set the output bits for the Extended I/O [I], DeviceNet [D], Profibus [P], and Interbus-S [S].

ARGUMENTS: OP m,n,o,p

where m is in the range 0 to 255 decimal, or $0 to $FF hexadecimal for SMC-20001, SMC-20002 or SMC-20004 and in the range 0 to 65,535 decimal, or $0 to $FFFF hexadecimal for SMC-20008

where n,o, and p are in the range 0 to 65,535 decimal, or $0 to $FFFF hexadecimal for SMC-2000xI (extended I/O)

NOTE : Outputs 24,32,40,48, 56 and 64 do not physically exist on the extended I/O option

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"SB"	Set output bit
"CB"	Clear output bit

EXAMPLES:

OP 0	Clear Output Port -- all bits
OP $85	Set outputs 1,3,8
OP $5	Set outputs 1,3
OP ,$05	Set Outputs 17 and 19
OP ,$F000	Set Outputs 29, 30 and 31 (32 does not exist)
MG _OP or MG _OP0	Return value of Outputs 1 through 16
MG _OP1	Return value of Outputs 17 through 32

@OUT (Status of Digital Output Function)

DESCRIPTION:

The Digital Output (@OUT[n]) function returns the status of the digital output number or variable given in square brackets. Note that the @OUT command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @OUT [n]

where n is an unsigned integer in the range 1 to 8 decimal on SMC-20001, SMC-20002, and SMC-20004

where n is an unsigned integer in the range 1 to 16 on SMC-20008

where n is an unsigned integer in the range 1 to 64 on SMC-2000 units with "I" (Extended I/O), "D" (DeviceNet), "P" (Profibus), and "S" (Interbus-S) options.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=3	Set variable
MG @OUT[VAR1]	Display the status of output 3
VAR2=@OUT[VAR1]+4	Perform calculation
EN	End of Program

PA (Position Absolute)

DESCRIPTION:

The PA command will set the final destination of the next move. The position is referenced with respect to absolute zero. If a ? is used, then the current destination (current command position if not moving, destination if in a move) is returned. For each single move, the largest position move possible is +/-2147483647. Units are in quadrature counts.

ARGUMENTS: PA x,y,z,w PAX=x PA a,b,c,d,e,f,g,h

where x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"PR"	Position relative
"SP"	Speed
"AC"	Acceleration
"DC"	Deceleration
"BG"	Begin

EXAMPLES:

PA 400,-600,500,200	X-axis will go to 400 counts Y-axis will go to -600 counts Z-axis will go to 500 counts W-axis will go to 200 counts
PA ?,?,?,?	Returns the current commanded position
0000000,0000000,0000000,0000000	
BG	Start the move
PA 700	X-axis will go to 700 on the next move while the Y,Z and W-axis will travel the previously set relative distance if the preceding move was a PR move, or will not move if the preceding move was a PA move.
BG	Start the move
PA*=0	All axes to zero
PAY=10000	Absolute position of Y at 10000

PF (Position Format)

DESCRIPTION:

The PF command allows the user to format the position numbers such as those returned by TP. The number of integer digits and the number of fractional digits can be selected with this command. An extra digit for sign and a digit for decimal point will be added to the total number of digits. If PF is minus, the format will be hexadecimal and a dollar sign will precede the characters. Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: PF m.n

where m is an integer between -8 and 10

n is an integer between 0 and 4

The negative sign for m specifies hexadecimal representation.

USAGE:

While Moving	Yes	Default Value	10.0
In a Program	Yes	Default Format	10.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

:TPX	Tell position of X
0000000021	Default format
:PF 5.2	Change format to 5 digits of integers and 2 of fractions
:TPX	Tell Position
00021.00	
PF-5.2	New format Output changed to hexadecimal
:TPX	Tell Position
$00015.00	Report in hex

PR (Position Relative)

DESCRIPTION:

The PR command sets the incremental distance and direction of the next move. The move is referenced with respect to the current position. If a ? is used, then the current incremental distance is returned (even if it was set by a PA command). Units are in quadrature counts.

ARGUMENTS: PR x,y,z,w PRX=x PR a,b,c,d,e,f,g,h

where x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"PA"	Position Absolute
"BG"	Begin
"AC"	Acceleration
"DC"	Deceleration
"SP"	Speed
"IP"	Increment Position

EXAMPLES:

PR 100,200,300,400	On the next move the X-axis will go 100 counts, the Y-axis will go to 200 counts forward, Z-axis will go 300 counts and the W-axis will go 400 counts.
BG	Start the move
PR ?,?,?,?	Return relative distances
0000000100,0000000200,0000000300,0000000400	
PR 500	
BGXY	The X-axis will go 500 counts on the next move while the Y-axis will go its previously set relative distance.
PRH=2000	Set H axis at 2000
PR*=10000	Specify all command positions at 10000

PW (PassWord)

DESCRIPTION:

The (PW) PassWord command is used to set or change the controller's security password. The command requires two parameters; p,p. Both parameters are the new password up to 8 characters in length. Both parameters must be identical for the new password to be accepted. The password can only be set or changed while the controller is in the "Unlocked" mode, (see the LC command) or a command error will result.

ARGUMENTS: PW p,p

where p,p are identical passwords up to 8 characters in length.

All characters can be alphabetic or numeric.

USAGE:

While Moving	Yes	Default Value	00000000
In a Program	No	Default Format	8.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"LC"	Lock Controller

EXAMPLES:

PW MOTION,MOTION	Set a new password "MOTION"
LC MOTION,1	Lock controller
LC MOTION,0	Unlock controller

QD (Download Array)

DESCRIPTION:

The QD command transfers array data from the host computer to the SMC-2000. QD array[], start, end requires that the array name be specified along with the first element of the array and last element of the array. After the QD command is entered in the terminal window, data can be sequentially entered one element at a time. The downloaded array is terminated by a <control>Z, <control>Q, <control>D or \.

ARGUMENTS: QD array[], start, end

array[] is any valid array name

start is first element of array (default=0)

end is last element of array (default = last element)

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"QU"	Upload Array

EXAMPLES

(In Terminal mode)

:DM DATA[5]	Define DATA array with 5 elements
:QD DATA[]	Begin download to array
:5	Set DATA[0]=5
:4	Set DATA[1]=4
:3	Set DATA[2]=3
:2	Set DATA[3]=2
:1	Set DATA[4]=1
:<control> Z	Terminate download to the array

QU (Upload Array)

DESCRIPTION:

The QU command transfers array data from the SMC-2000 to a host computer. QU array[], start, end, comma requires that the array name be specified along with the first element of the array and last element of the array. If comma is 1, then a comma will separate the array elements. Otherwise, the elements will be separated by a carriage return. A <control> Z as an end of text marker will follow the uploaded array.

ARGUMENTS: QU array[], start, end, comma

array[] is any valid array name

start is first element of array (default=0)

end is last element of array.(default=last element)

comma -- if it is a 1, then elements are separated by a comma, else a carriage return

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"QD"	Download Array

QY (Query Yaskawa Absolute Encoder Alarm)

DESCRIPTION:

The Query Yaskawa Absolute Encoder Alarm (QY) displays the serial data that was received from an absolute encoder when the position was requested using the (AE) command. Usually, the data in the QY command is the number of revolutions of the servo from the absolute zero point. If there was an encoder alarm, one of the following codes will be stored in the QY command:

ALMRMOA=Backup Alarm

ALARMOB=Checksum Error

ALARMOD=Battery Alarm

ALARMOE=Battery/Backup Combination Error

ALARMOH=Absolute Error

ALARMOP=Overspeed

ARGUMENTS: None

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	No		

RELATED COMMANDS

AE	Absolute Encoder
TY	Tell Yaskawa absolute encoder position (when last read)

Hint: Read the Absolute Encoder (AE) section of this manual for a detailed example of the absolute encoder commands. Also, see the Tell Error Code (TC) section of this manual or the Yaskawa SIGMA Servomotor manual for a complete listing of absolute encoder alarms.

RA (Record Array)

DESCRIPTION:

The Record Array (RA) command uses up to four arrays for automatic data capture. The arrays must be dimensioned by the Dimension (DM) command. The data to be captured is specified by the Record Data (RD) command and time interval by the Record (RC) command.

ARGUMENTS: RA n[],m[],o[],p[]

where n,m,o, and p are dimensioned arrays as defined by DM command. The [] contains nothing.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"DM"	Dimension Array
"Record Data (RD"	Record Data
"RC"	Record Interval

EXAMPLES:

#RECORD	Label
DM POS[100]	Define array
RA POS[]	Specify Record Mode
RD _TPX	Specify data type for record
RC 1	Begin recording at 2 msec intervals
PR 1000;BG	Start motion
EN	End

Hint: The record array mode is useful for recording the real-time motor position during motion. The data is automatically captured in the background and does not interrupt the program sequencer. The record mode can also be used to teach or learn a motion path.

RC (Record)

DESCRIPTION:

The RC command begins recording for the Automatic Record Array Mode (RA).

ARGUMENTS: RC n,m

where n is an integer 1 through 8 and specifies 2^n milliseconds between samples. RC 0 stops recording.

m is optional and specifies the number of records to be recorded. If m is not specified, the DM number will be used.

RC? or V=_RC

returns a 1 if recording

returns a 0 if not recording

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"DM"	Dimension Array
"Record Data (RD"	Record Data
"Record Array (RA"	Record Array Mode

EXAMPLES:

#RECORD	Record
DM Torque[1000]	Define Array
RA Torque[]	Specify Record Mode
RD _TTX	Specify Data Type
RC 2	Begin recording and set 4 msec between records
JG 1000;BG	Begin motion
#A;JP #A,_RC=1	Loop until done
MG "DONE RECORDING"	Print message
EN	End program

RD (Record Data)

DESCRIPTION:

The Record Data (RD) command specifies the data type to be captured for the Record Array (RA) mode. The command type includes:

_DEn	2nd encoder position (dual loop)
_TPn	Position
_TEn	Position error
_SHn	Commanded position
_RLn	Latched position
_TI	Inputs
_OP	Outputs
_TSn	Switches, only 0-4 bits valid
_SCn	Stop code
_TTn	Tell torque

where X,Y,Z or W may be specified.

ARGUMENTS: RD x,x,x,x

where x specifies the data type to be captured. The order is important. Each of the four data types correspond with the array specified in the RA command.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

“Record Array (RA”	Record Array
“RC”	Record Interval
“DM”	Dimension Array

EXAMPLES:

DM ERRORX[50],ERRORY[50]	Define array
RA ERRORX[],ERRORY[ ]	Specify record mode
RD _TEX,_TEY	Specify data type
RC1	Begin record
JG 1000;BG	Begin motion

RE (Return from Error Routine)

DESCRIPTION:

The Return from Error (RE) command is used to end a Special Label routine such as position error handling or limit switch subroutine. The Position Error handling subroutine begins with the #POSERR label. The limit switch handling subroutine begins with #LIMSWI. An RE at the end of these routines causes a return to the command that was executing in the main program when the error was generated. Care should be taken to be sure the error or limit switch conditions no longer occur to avoid re-entering the subroutines. If the program sequencer was waiting for a trip-point to occur, prior to the error interrupt, the trip-point condition is preserved on the return to the program if RE1 is used. RE0 clears the trip-point. To avoid returning to the main program after handling a special label event, use the ZS command to zero the subroutine stack, then jump to the program label of your choice.

ARGUMENTS: RE n

where n = 0 or 1

0 clears the interrupted trip-point

1 restores state of trip-point

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

#LIMSWI	Limit Subroutine
#POSERR	Error Subroutine

EXAMPLES:

#A;JP #A;EN	Label for main program
#POSERR	Begin Error Handling Subroutine
MG “ERROR”	Print message
SB1	Set output bit 1
RE	Return to main program and clear trip-point

Hint: An applications program must be executing for the #LIMSWI and #POSERR subroutines to function.

RI (Return from Interrupt Routine)

DESCRIPTION:

The RI command is used to end the interrupt subroutine beginning with the label #ININT. An RI at the end of this routine causes a return to the main program. The RI command also re-enables input interrupts. If the program sequencer was interrupted while waiting for a trip-point, such as WT, RI1 restores the trip-point on the return to the program. RI0 clears the trip-point. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack. After doing this, reissue the II command and jump to the label of your choice.

ARGUMENTS: RI n

where n = 0 or 1

0 clears interrupt trip-point

1 restores trip-point

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

#ININT	Input interrupt subroutine
"II"	Enable input interrupts

EXAMPLES:

#A;II1;JP #A;EN	Program label
#ININT	Begin interrupt subroutine
MG "INPUT INTERRUPT"	Print Message
SB 1	Set output line 1
RI 1	Return to the main program and restore trip-point

Hint: An applications program must be executing for the #ININT subroutine to function.

RL (Report Latched Position)

DESCRIPTION:

The RL command will return the last position captured by the latch. The latch must first be armed by the AL command. The activated state of the latch can be configured using the CN command.

ARGUMENTS: RL XYZW RL ABCDEFGH

where X,Y,Z,W are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMAND:

"AL"	Arm Latch

EXAMPLES:

JG ,5000	Set up to jog the Y-axis
BGY	Begin jog
ALY	Arm the Y latch; assume that after about 2 seconds, input goes low
WT 2000	Wait 2 seconds
RLY	Report the latch
10000	

@RND (Round Function)

DESCRIPTION:

The Round (@RND[n]) function rounds a number or variable given in square brackets to the nearest integer. Note that the @RND command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @RND [n]

where n is a number in the range of –2147483648.9999 to 2147483647.9999

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=123.456	Set variable
MG @RND[VAR1]	Display the value of VAR1 rounded to the nearest integer
VAR2=@RND[VAR1]+25	Perform calculation
EN	End of Program

RP (Reference Position)

DESCRIPTION:

This command returns the commanded reference position of the motor(s).

ARGUMENTS: RP XYZW RP ABCDEFGH

where XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"TP"	Tell Position
"TE"	Tell Error

EXAMPLES: Assume that XYZ and W axes are commanded to be at the positions 200, -10, 0, -110 respectively. The returned units are in quadrature counts.

PF 7	Position format of 7
RP	
0000200,-0000010,0000000,-0000110	Return X,Y,Z,W reference positions
RPX	
0000200	Return the X motor reference position
RPY	
-0000010	Return the Y motor reference position
PF-6.0	Change to hex format
RP	
$0000C8,$FFFFF6,$000000,$FFFF93	Return X,Y,Z,W in hex
Position=_RPX	Assign the variable, Position, the value of RPX

Note: The relationship between RP, TP and TE is that the position error, _TEX, equals the difference between the reference position, _RPX and the actual position, _TPX. _TEX is a positive number when the servo position is lagging behind the commanded position.

RS (Reset)

DESCRIPTION:

The RS command resets the state of the processor to its power-on condition. The previously saved state of the controller, along with parameter values, and saved sequences are restored. If the program has an #AUTO label, it will begin execution at that label.

USAGE:

While Moving	Yes	Default Value	---
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

RS	Reset the controller
RS1	Restore parameters only
RS2	Clear App program

<control>R <control>S (Master Reset)

DESCRIPTION:

The Master Reset command resets the SMC-2000 to factory default settings and erases EEPROM.

While Moving	Yes	Default Value	---
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

SB (Set Bit)

DESCRIPTION:

The SB command sets one of the bits on an output port.

ARGUMENTS: SB n

where n is an integer in the range 1 to 8 for SMC-20001, SMC-20002, and SMC-20004.

where n is an integer in the range 1 to 16 for SMC-20008

where n is an integer in the range 1 to 64 for extended I/'O options "D", "I", "P", and "S" (Outputs 24,32,40,48, 56 and 64 do not physically exist)

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS

"OP"	Configure output port
"CB"	Clear Bit

EXAMPLES:

SB 5	Set output line 5
SB 1	Set output line 1

SC (Stop Code)

DESCRIPTION:

The SC command allows the user to determine why a motor stopped. The controller responds with the stop code as follows:

CODE	MEANING
0	Motors are running, independent mode
1	Motors stopped at commanded independent position
2	Decelerating or stopped by FWD limit switches
3	Decelerating or stopped by REV limit switches
4	Decelerating or stopped by Stop Command (ST)
6	Stopped by Abort input
7	Stopped by Abort command (AB)
8	Decelerating or stopped by Off-on-Error (OE1)
9	Stopped after Finding Edge (FE)
10	Stopped after Homing (HM)
40	Stopped at Latched Target (LT)
41	Latched target overrun due to limit switch or stop command
42	Latched target overrun due to insufficient distance
50	Contour running
51	Contour Stop
99	Timeout for in-position (MC)
100	Motors are running, vector sequence
101	Motors stopped at commanded vector

ARGUMENTS: SC XYZW SC ABCDEFGH

where XYZW or ABCDEFGH are the axes

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

TOM=_SCW	Assign the Stop Code of W to variable Tom

SH (Servo Here)

DESCRIPTION:

The SH command tells the controller to use the current motor position as the commanded position, and to enable the amplifies.

ARGUMENTS: SH XYZW SH ABCDEFGH

where XYZW are X,Y,Z,W axes

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"MO"	Motor-off

EXAMPLES:

SH	Servo X,Y,Z,W motors
SHX	Only servo the X motor, the Y,Z and W motors remain in its previous state.
SHY	Servo the Y motor; leave the X,Z and W motors unchanged
SHZ	Servo the Z motor; leave the X,Y and W motors unchanged
SHW	Servo the W motor; leave the X,Y and Z motors unchanged

@SIN (Sin Function)

DESCRIPTION:

The Sine (@SIN[n]) function returns the sine of a number or variable which is inserted in square brackets using units of degrees. Note that the @SIN command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @SIN [n]

where n is a number in the range of –32768 to 32768

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=60	Set variable
MG @SIN[VAR1]	Display the value of the sine of VAR1
VAR2=@SIN[VAR1]+9	Perform calculation
EN	End of Program

SP (Speed)

DESCRIPTION:

This command sets the slew speed of any or all axes for independent moves, or it will return the previously set value. The parameters input will be rounded down to the nearest factor of 2. The units of the parameter are in counts per second. The maximum value for speed in 2,000,000 cts./sec. when using stepper motors.

ARGUMENTS: SP x,y,z,w SPX=x SP a,b,c,d,e,f,g,h

where x, y, z, are unsigned numbers in the range 0 to 8,000,000

USAGE:

While Moving	Yes	Default Value	25000
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AC"	Acceleration
"DC"	Deceleration
"PR"	Position Relation
"PA"	Position Absolute
"BG"	Begin

EXAMPLES:

PR 2000,3000,4000,5000	Specify x,y,z,w parameter
SP 5000,6000,7000,8000	Specify x,y,z,w speeds
BG	Begin motion of all axes
AM Z	After Z motion is complete
SP*=5000	Set all speeds at 5000
SPH=10000	Set speed of H axis at 10000

NOTE: For vector moves, use the vector speed command (VS) to change the speed. SP is not a "mode" of motion like JOG (JG).

@SQR (Square Root Function)

DESCRIPTION:

The Square Root (@SQR[n]) function returns the square root of a number or variable which is inserted in square brackets. Note that the @SQR command is a function, which means that it does not follow the convention of the commands, and does not require the underscore when used as an operand.

ARGUMENTS: @SQR [n]

where n is a number in the range of 0 to 2147483647.9999

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

#TEST	Program TEST
VAR1=49	Set variable
MG @SQR[VAR1]	Display the square root of VAR1
VAR2=@SQR[VAR1]+7	Perform calculation
EN	End of program

ST (Stop)

DESCRIPTION:

The ST command stops motion on the specified axis. Motors will decelerate to a stop. If ST is given without an axis specification (from serial port), program execution will stop in addition to XYZW. XYZW specification will not halt program execution.

ARGUMENTS: ST XYZW ST ABCDEFGH

where XYZW are X,Y,Z,W axes. No parameters will stop motion on all axes and stop program.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"BG"	Begin Motion
"AB"	Abort Motion
"After Motion (AM"	Wait for motion end
"DC"	Deceleration rate
"HX"	Halt program execution

EXAMPLES:

ST X	Stop X-axis motion
ST S	Stop coordinated sequence
ST XYZW	Stop X,Y,Z,W motion
ST	Stop program and XYZW motion
ST SZW	Stop coordinated XY sequence, and Z and W motion

Hint: Use the after motion complete command, AM, to wait for motion to decelerate to a stop.

TB (Tell Status Byte)

DESCRIPTION:

The TB command returns status information from the controller.

Bit	Status when high
Bit 7	Controller addressed
Bit 6	Executing program
Bit 5	Contouring
Bit 4	Executing error or limit switch routine
Bit 3	Input interrupt enabled
Bit 2	Executing input interrupt routine
Bit 1	0 (Reserved)
Bit 0	Echo on

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

TB	Tell status information from the controller
65	Executing program and echo on ($2^6 + 2^0 = 64 + 1 = 65$)

TC (Tell Error Code)

DESCRIPTION:

The TC command returns a number between 1 and 255. This number is a code that reflects why a command was not accepted by the controller. This command is useful when the controller halts execution of a program at a command or when the response to a command is a question mark. Entering the TC command will provide the user with a code as to the reason. After TC has been read, it is set to zero. TC 1 returns the text message as well as the numeric code.

Error Codes:

1	Unrecognized command
2	Command only valid from program
3	Command not valid in program
4	Operand error
5	Input buffer full
6	Number out of range
7	Command not valid while running
8	Command not valid while not running
9	Variable error
10	Empty program line or undefined label
11	Invalid label or line number
12	Subroutine more than 16 deep
13	JG only valid when running in jog mode
14	EEPROM check sum error
15	EEPROM write error
16	IP incorrect sign during position move or IP given during forced deceleration
17	ED, BN and DL not valid while program running
18	Command not valid when contouring
19	Application program/strand already executed
20	Begin not valid with motor off
21	Begin not valid while running
22	Begin not possible due to Limit Switch
24	Begin not valid because no sequence defined
25	Variable not given in IN command
28	S operand not valid
29	Not valid during coordinated move
30	Sequence segment too short
31	Total move distance in a sequence > 2 billion
32	More than 511 segments in a sequence
41	Contouring record range error
42	Contour data being sent too slowly

46	Gear axis both master and follower
50	Not enough fields
51	Question mark not valid
52	Missing " or string too long
53	Error in {}
54	Question mark part of string
55	Missing [or []
56	Array index invalid or out of range
57	Bad function or array
58	Unrecognized command in a command response (i.e._TPQ)
59	Mismatched parentheses
60	Download error - line too long or too many lines
61	Duplicate or bad label
62	Too many labels
65	IN command must have a comma
66	Array space full
67	Too many arrays or variables
71	IN only valid in task #0
80	Record mode already running
81	No array or source specified
82	Undefined array
83	Not a valid number
84	Too many elements
90	Only X,Y,Z,W or A,B,C,D,E,F,G,H valid operand
96	SM jumper needs to be installed for stepper motor operation
100	Not valid when running ECAM
101	Improper index to ET (must be 0-256)
102	No master axis for ECAM
103	Master axis modulus greater than 256*EP value
104	Not valid when axis performing ECAM
105	EB1 command must be given first
114	Absolute Encoder option not installed
115	Motor must be in MO for this comment
116	Absolute Encoder responded with an alarm
117	Absolute Encoder did not respond
118	Controller has GL1600, not GL1800

ARGUMENTS: TC n

n=0 returns code only

n=1 returns code and message

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

MG _ED	The last line that had an error

EXAMPLES:

GF32	Bad command
?TC	Tell error code
001	Unrecognized command

TD (Tell Dual Encoder)

DESCRIPTION:

This command returns the current position of the dual (auxiliary) encoder(s). When operating with stepper motors, the TD command returns the number of counts that have been output by the controller.

ARGUMENTS: TD XYZW TD ABCDEFGH

where XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"DE"	Dual Encoder

EXAMPLES:

PF 7	Position format of 7
TD	Return X,Y,Z,W Dual encoders
0000200,-0000010,0000000,-0000110	
TDX	Return the X motor Dual encoder
0000200	
DUAL=_TDX	Assign the variable, DUAL, the value of TDX

TE (Tell Error)

DESCRIPTION:

This command returns the current position error of the motor(s). The range of possible error is –2147483648 to 2147483647. The Tell Error command is not valid for step motors since they operate open loop.

ARGUMENTS: TE XYZW TE ABCDEFGH

Where XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"OE"	Off On Error
"ER"	Error Limit
#POSERR	Error Subroutine

EXAMPLES:

TE	Return all position errors
00005,-00002,00000,00006	
TEX	Return the X motor position error
00005	
TEY	Return the Y motor position error
-00002	
ERROR=_TEX	Sets the variable, ERROR, with the X-axis position error

Hint: Under normal operating conditions with servo control, the position error should be small. The position error is typically largest during acceleration. The TE value is positive when the servo position is lagging the commanded position.

TI (Tell Inputs)

DESCRIPTION:

This command returns the state of the general inputs. TI or TI0 return inputs I1 through I8, TI1 returns I9 through I16 and TI2 returns I17 through I24. For motion controllers with extended I/O 40 additional inputs are available using TI3 through TI7.

	TI	TI1	TI2	TI3	TI4	TI5	TI6	TI7
Bit 7 (MSB)	Input 8	Input 16	Input 24	Input 32	Input 40	Input 48	Input 56	Input 64
Bit 6	Input 7	Input 15	Input 23	Input 31	Input 39	Input 47	Input 55	Input 63
Bit 5	Input 6	Input 14	Input 22	Input 30	Input 38	Input 46	Input 54	Input 62
Bit 4	Input 5	Input 13	Input 21	Input 29	Input 37	Input 45	Input 53	Input 61
Bit 3	Input 4	Input 12	Input 20	Input 28	Input 36	Input 44	Input 52	Input 60
Bit 2	Input 3	Input 11	Input 19	Input 27	Input 35	Input 43	Input 51	Input 59
Bit 1	Input 2	Input 10	Input 18	Input 26	Input 34	Input 42	Input 50	Input 58
Bit 0 (LSB)	Input 1	Input 9	Input 17	Input 25	Input 33	Input 41	Input 49	Input 57

Shaded inputs available with extended I/O or a network option card

ARGUMENTS: TIn

where n equals 0 for SMC-20002 and SMC-20004

n equals 0, 1 or 2 for SMC-20008

n equals 1 to 7 for SMC-2000xI (extended I/0)

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

TI	
08	Input 4 is high, others low
TI	
00	All inputs low
Input=_TI	Sets the variable, Input, with the TI value
TI	
255	All inputs high
TI1	
04	Input 11 high, others low

TIME

DESCRIPTION:

The TIME operand contains the value of the internal free running, real time clock. The returned value represents the number of servo loop updates and is based on the TM command. The default value for the TM command is 1000. With this update rate, the operand TIME will increase by one count every millisecond. Note that a value of 1000 for the update rate (TM command) will actually set an update rate of 1/1024 seconds. Thus the value returned by the time operand will be off by 2.4% of the actual time.

USAGE:

While Moving		Default Value	
In a Program		Default Format	TIME
Not in a Program			
Can be Interrogated			
Used in an Operand	Yes		

EXAMPLES:

MG TIME	Display the value of the internal clock
START=TIME	Set variable
PRX=10000	Position relative move
BGX	Begin motion
AMX	After Motion
DONE=TIME	Set variable
MG"Complete in",DONE-START,"mSec"	

Note: This is an operand, not a command

TL (Torque Limit)

DESCRIPTION:

The TL command sets the limit on the motor command output. For example, TL of 5 limits the motor command output to 5 volts. Maximum output of the motor command is 9.998 volts.

ARGUMENTS: TL x,y,z,w TLX=x TL a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range 0 to 9.998 volts with a resolution of .001 volt.

USAGE:

While Moving	Yes	Default Value	9.9988
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

TL 1,5,9,7.5	Limit X-axis to 1volt Limit Y-axis to 5 volts Limit Z-axis to 9 volts Limit W-axis to 7.5 volts
TL ?,?,?,?	Return limits
1.0000,5.0000,9.0000,7.5000	
TL ?	Return X-axis limit
1.0000	
TLZ=5	Set torque limit of Z to 5 volts
TL*=9	Set all torque limits at 9 volts

TM (Time Command)

DESCRIPTION:

The TM command sets the sampling period of the control loop. Changing the sampling period will uncalibrate the speed and acceleration parameters (Actually, all time units are counts/servo cycle, etc.). A negative number turns off the internal clock allowing for an external source to be used as the time base. The units of this command are μsec.

ARGUMENTS: TM n

where n is an integer in the range 250 to 20000 decimal with resolution of 125 microseconds. The minimum sample time for the SMC-2000-2 is 375 μsec; and 500 μsec for the SMC-2000-4.

USAGE:

While Moving	Yes	Default Value	1000
In a Program	Yes	Default Format	5.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

TM -1000	Turn off internal clock
TM 2000	Set sample rate to 2000 microseconds (This will cut all speeds in half and all acceleration in fourths)
TM 1000	Return to default sample rate

TN (Tangent)

DESCRIPTION:

The TN m,n command describes the tangent axis to the coordinated motion path. m is the scale factor in counts/degree of the tangent axis. n is the absolute position of the tangent axis, at which the resulting angle of the tangent axis equals zero in the coordinated motion plane. The tangent axis is specified with the VM n,m,p command where p is the tangent axis. _TN gives you the position of the first tangent point. Tangent is useful for cutting applications where a cutting tool must remain tangent to the part.

ARGUMENTS: TN m,n

where m is the scale factor in counts/degree

n is the absolute position at which the tangent angle is zero

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VM"	Vector mode

EXAMPLES:

VM X,Y,Z	Specify coordinated mode for X and Y-axis; Z-axis is tangent to the motion path
TN 100,50	Specify scale factor as 100 counts/degree and 50 counts at which tangent angle is zero
VP 1000,2000	Specify vector position X,Y
VE	End Vector
BGS	Begin coordinated motion with tangent axis

TP (Tell Position)

DESCRIPTION:

This command returns the current position of the motor(s).

ARGUMENTS: TP XYZW TP ABCDEFGH

where XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

Assume the X-axis is at the position 200 (decimal), the Y-axis is at the position -10 (decimal), the Z-axis is at position 0, and the W-axis is at -110 (decimal). The returned parameter units are in quadrature counts.

PF 7	Position format of 7
TP	Return X,Y,Z,W positions
0000200,-0000010,0000000,-0000110	
TPX	Return the X motor position
0000200	
TPY	Return the Y motor position
-0000010	
PF-6.0	Change to hex format
TP	Return X,Y,Z,W in hex
$0000C8,$FFFFF6,$000000,$FFFF93	
POSITION=_TPX	Assign the variable, POSITION, the value of TPX

TR (Trace)

DESCRIPTION:

The TR command causes each instruction in a program to be sent out the communications port prior to execution. The trace command is useful in debugging programs.

ARGUMENTS: TR n

where n=0 or 1

0 disables function

1 enables function

USAGE:

While Moving	Yes	Default Value	TR0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

Note: It may be helpful to start the logging feature of YTERM before starting a trace. The logging feature writes all information that is transmitted on the serial port to a file called SMCLOG.LOG.

TS (Tell Switches)

DESCRIPTION:

TS returns the state of the Home, Forward Limit and Reverse Limits for each axis. TS also returns error, motion and motor status.

Bit	Status if high
Bit 7	Axis in motion
Bit 6	Axis error exceeds error limit
Bit 5	X motor off
Bit 4	Undefined
Bit 3	Forward Limit inactive
Bit 2	Reverse Limit inactive
Bit 1	Home X
Bit 0	Latch not armed

ARGUMENTS: TS XYZW TS ABCDEFGH

where XYZW designate X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

V1=_TSY	Assigns value of TSY to the variable V1

TT (Tell Torque)

DESCRIPTION:

The TT command reports the value of the analog output signal, which is a number between -9.998 and 9.998 volts.

ARGUMENTS: TT XYZW TT ABCDEFGH

where XYZW specify X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"TL"	Torque Limit

EXAMPLES:

V1=_TTX	Assigns value of TTX to variable, V1
TTX	Report torque on X
-0.2843	Torque is -.2843 volts

TV (Tell Velocity)

DESCRIPTION:

The TV command returns the actual velocity of the axes in units of quadrature count/s.

ARGUMENTS: TV XYZW TV ABCDEFGH

where XYZW specifies X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	7.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

VELX=_TVX	Assigns value of X-axis velocity to the variable VELX
TVY	Returns the Y-axis velocity
0003420	

NOTE: The TV command is computed using a special averaging filter (over approximately .25 sec). Therefore, TV will return average velocity, not instantaneous velocity.

TW (Timeout for In Position (MC))

DESCRIPTION:

The TW x,y,z,w command sets the timeout in msec to declare an error if the MC command is active and the motor is not at or beyond the actual position within n msec after the completion of the motion profile. If a timeout occurs, then the MC trip-point will clear and the stop code will be set to 99. An application program will jump to the special label #MCTIME, if it exists . The RE command should be used to return from the #MCTIME subroutine.

ARGUMENTS: TW x,y,z,w TW a,b,c,d,e,f,g,h

where x,y,z,w specifies timeout in msec range 0 to 32767 msec -1 disables the timeout

EXAMPLES:

USAGE:

While Moving	Yes	Default Value	32766
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"MC"	Motion Complete

TY (Tell Yaskawa Absolute Encoder)

DESCRIPTION

TY (Tell Yaskawa Absolute Encoder) reports the position that was read when the absolute encoder data was requested using the AE command.

ARGUMENTS: TY XYZW TP ABCDEFGH

USAGE:

While Moving	Yes	Default Value	-2,147,483,648
In a Program	Yes	Default Format	10.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AE"	Absolute
"QY"	Report alarm code from Yaskawa encoder

EXAMPLES:

MOX	Motor Off
AEX=4096	Read X absolute encoder
DPX=_TPX=XHmOfs	Add offset variable to current absolute position
TPX	Tell current defined Position
TYX	Tell absolute encoder position (when it was read)
SHX	Enable servo (Servo Here)

UL (Upload)

DESCRIPTION:

The UL command transfers data from the SMC-2000 to a host computer through port 1. Programs are sent without line numbers. A <control> Z as an end of text marker will follow the Uploaded program. When used as an operand, _UL gives the number of available variables. The total number of variables is 254 for the SMC-2000.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMAND:

"DL"	Download

EXAMPLES:

UL	Begin upload
#A	Line 0
NO This is an Example	Line 1
NO Program	Line 2
EN	Line 3
<control>Z	Terminator

VA (Vector Acceleration)

DESCRIPTION:

This command sets the acceleration rate of the vector in a coordinated motion sequence. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

ARGUMENTS: VA n

where n is an unsigned number in the range 1024 to 68,431,360 decimal.

USAGE:

While Moving	Yes	Default Value	262144
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VS"	Vector Speed
"VP"	Vector Position
"VE"	End Vector
"CR"	Circle
"VM"	Vector Mode
"BG"	Begin Sequence
"VD"	Vector Deceleration
"VT"	Vector smoothing constant - S-curve

EXAMPLES:

VA 1024	Set vector acceleration to 1024 counts/sec2
VA ?	Return vector acceleration
00001024	
VA 20000	Set vector acceleration
VA ?	
0019456	Return vector acceleration
ACCEL=_VA	Assign variable, ACCEL, the value of VA

VD (Vector Deceleration)

DESCRIPTION:

This command sets the deceleration rate of the vector in a coordinated motion sequence. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

ARGUMENTS: VD n

where n is an unsigned number in the range 1024 to 68,431,360 decimal.

USAGE:

While Moving	No	Default Value	262144
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VA"	Vector Acceleration
"VS"	Vector Speed
"VP"	Vector Position
"CR"	Circle
"VE"	Vector End
"VM"	Vector Mode
"BG"	Begin Sequence
"VT"	Smoothing constant - S-curve

EXAMPLES:

#VECTOR	Vector Program Label
VMXY	Specify plane of motion
VA1000000	Vector Acceleration
VD 5000000	Vector Deceleration
VS 2000	Vector Speed
VP 10000, 20000	Vector Position
VE	End Vector
BGS	Begin Sequence

VE (Vector Sequence End)

DESCRIPTION:

VE is required to specify the end segment of a coordinated move sequence. VE would follow the final VP or CR command in a sequence. VE ? or _VE returns the length of the vector in counts. VE is equivalent to the LE command.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

“VM”	Vector Mode
“VS”	Vector Speed
“VA”	Vector Acceleration
“VD”	Vector Deceleration
“CR”	Circle
“VP”	Vector Position
“BG”	Begin Sequence
“CS”	Clear Sequence

EXAMPLES:

VM XY	Vector move in XY
VP 1000,2000	Linear segment
CR 0,90,180	Arc segment
VP 0,0	Linear segment
VE	End sequence
BGS	Begin motion

VF (Variable Format)

DESCRIPTION:

The VF command allows the variables and arrays to be formatted for number of digits before and after the decimal point. When displayed, the value m represents the number of digits before the decimal point, and the value n represents the number of digits after the decimal point. When in hexadecimal, the string will be preceded by a $. Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: VF m.n

where m and n are unsigned numbers in the range 0<m<10 and 0<n<4. A negative m specifies hexadecimal format

USAGE:

While Moving	Yes	Default Value	10.4
In a Program	Yes	Default Format	2.1
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

VF 5.3	Sets 5 digits of integers and 3 digits after the decimal point
VF 8.0	Sets 8 digits of integers and no fractions
VF -4.0	Specify hexadecimal format with 4 bytes to the left of the decimal

VM (Coordinated Motion Mode)

DESCRIPTION:

The VM command specifies the coordinated motion mode and the plane of motion. This mode may be specified for motion on any set of two axes.

The motion is specified by the instructions VP and CR, which specify linear and circular segments. Up to 511 segments may be given before the Begin Sequence (BGS) command. Additional segments may be given during the motion when the SMC-2000 buffer frees additional spaces for new segments.

The Vector End (VE) command must be given after the last segment. This tells the controller to decelerate to a stop during the last segment.

It is the responsibility of the user to keep enough motion segments in the buffer to ensure continuous motion. VM ? or _VM returns the available spaces for motion segments that can be sent to the buffer.

511 returns means that the buffer is empty and 511 segments may be sent. A zero means that the buffer is full and no additional segments may be sent.

ARGUMENTS: VM nmp

where n and m is the plane of motion of any two axes of X,Y,Z,W or A,B,C,D,E,F,G,H

p is the tangent axis X,Y,Z,W or A,B,C,D,E,F,G,H. N turns off tangent.

USAGE:

While Moving	No	Default Value	X,Y
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

“VP”	Vector Position
“VS”	Vector Speed
“VA”	Vector Acceleration
“VD”	Vector Deceleration
“CR”	Circle
“VE”	End Vector Sequence
“BG”	Begin Sequence
“CS”	Clear Sequence
“CS”	_CS - Segment counter
“VT”	Vector smoothing constant -- S-curve
“AV”	Vector distance

EXAMPLES:

VM X,Y	Specify coordinated mode for X,Y
CR 500,0,180	Specify arc segment
VP 100,200	Specify linear segment
VE	End vector
BGS	Begin sequence

VP (Vector Position)

DESCRIPTION:

The VP command defines the target coordinates of a straight line segment in a 2 axis motion sequence. The axes are chosen by the VM command. The motion starts with the Begin sequence command. The units are in quadrature counts, and are a function of the vector scale factor. For three or four axis linear interpolation, use the LI command. When used as an operand, _VPX, _VPY, _VPZ, _VPW return the absolute coordinate of the axes at the last intersection along the sequence. For example, during the first motion segment, this instruction returns the coordinate at the start of the sequence. The use as an operand is valid in the linear mode, LM, and in the Vector mode, VM. The parameter N is optional and can be used to define the vector speed that is attached to the motion segment.

ARGUMENTS: VP n,m < N

where n,m are signed integers in the range -2147483648 to 2147483647. The length of each segment must be limited to $8 \cdot 10^6$. N is an unsigned even integer between 0 and 8,000,000

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"CR"	Circle
"VM"	Vector Mode
"VA"	Vector Acceleration
"VD"	Vector Deceleration
"VE"	Vector End
"VS"	Vector Speed
"BG"	Begin Sequence
"VT"	Vector smoothing constant - S-curve

EXAMPLES:

#A	Program A
VM X,Y	Specify motion plane
VP 1000,2000	Specify vector position X,Y
CR 1000,0,360	Specify arc
VE	Vector end
VS 2000	Specify vector speed
VA 400000	Specify vector acceleration
BGS	Begin motion sequence
EN	End Program

Hint: The first vector in a coordinated motion sequence defines the origin for that sequence. All other vectors in the sequence are defined by their endpoints with respect to the start of the move sequence.

VR (Vector Speed Ratio)

DESCRIPTION:

The VR r command multiplies the vector speed specifications given by VS or < by the value specified by r. r is between 0 and 10 with a resolution of .0001. VR takes effect immediately and will ratio all the following VS commands and any <n specifications used on VP, CR or LI segments. VR doesn't ratio the accelerations. VR is useful for feed rate override.

ARGUMENTS: VR r

where r is between 0 and 10 with a resolution of .0001

USAGE:

While Moving	Yes	Default Value	1
In a Program	Yes	Default Format	
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VS"	Vector speed

EXAMPLES:

#A	Vector Program
VMXY	Vector Mode
VP 1000,2000	Vector Position
CR 1000,0,360	Specify Arc
VE	End Sequence
VS 2000	Vector Speed
BGS; AMS	Begin Sequence After Motion
JP#A	Repeat Move
#SPEED	Speed Override
VR@AN[1]*.1	Read analog input, compute ratio
JP#SPEED	Loop
XQ#A,0	Execute task 0 and 1 simultaneously
XQ#SPEED,1	

The above two programs #A and #SPEED are executed at the same time. #SPEED reads the analog input continuously and sets the speed ratio accordingly.

VS (Vector Speed)

DESCRIPTION:

The VS command specifies the speed of the vector in a coordinated motion sequence in either the LM or VM modes. The parameter input is rounded down to the nearest factor of 2. The units are counts per second. VS may be changed during motion.

ARGUMENTS: VS n

where n is an unsigned number in the range 2 to 8000000 decimal

USAGE:

While Moving	Yes	Default Value	8192
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VA"	Vector Acceleration
"VP"	Vector Position
"CR"	Circle
"LM"	Linear Interpolation
"VM"	Vector Mode
"BG"	Begin Sequence
"VE"	Vector End
"VR"	Vector Speed Ratio

EXAMPLES:

VS 2000	Define vector speed as 2000 counts/sec
VS ?	Return vector speed
002000	

VT (Vector Time Constant - S curve)

DESCRIPTION:

The VT command filters the acceleration and deceleration functions in vector moves of VM or LM type to produce a smooth velocity profile. The resulting profile, known as S-curve, has continuous acceleration and results in reduced mechanical vibrations. VT sets the bandwidth of the filter, where 1 means no filtering and 0.004 means maximum filtering. NOTE that the filtering results in longer motion time.

ARGUMENTS: VT n

where n is a positive number in the range between 0.004 and 1.0, with a resolution of 1/256

USAGE:

While Moving	Yes	Default Value	1.0
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"IT"	Independent Time Constant for smoothing independent moves

EXAMPLES:

VT 0.8	Set vector time constant
VT ?	Return vector time constant
0.8	

WC (Wait for Contour Data)

DESCRIPTION:

The WC command acts as a flag in the Contour Mode. After this command is executed, the controller does not receive any new data until the internal contour data buffer is ready to accept new commands. This command prevents the contour data from overwriting on itself in the contour data buffer.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"CM"	Contour Mode
"CD"	Contour Data
"DT"	Contour Time

EXAMPLES:

CM XYZW	Specify contour mode
DT 4	Specify time increment for contour
CD 200,300,-150,500	Specify incremental position on X,Y,Z and W X-axis moves 200 counts Y-axis moves 300 counts Z-axis moves -150 counts W-axis moves 500 counts
WC	Wait for contour data to complete
CD 100,200,300,400	
WC	Wait for contour data to complete
DT 0	Stop contour
CD 0,0,0,0	Exit mode

WT (Wait)

DESCRIPTION:

The WT command is a trip-point used to time events. After this command is executed, the controller will wait for the number of samples specified before executing the next command. If the TM command has not been used to change the sample rate from 1 msec, then the units of the Wait command are milliseconds.

ARGUMENTS: WT n

where n is an integer in the range 0 to 2 Billion decimal

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES: Assume that 10 seconds after a move is over a relay must be closed.

#A	Program A
PR 50000	Position relative move
BGX	Begin the move
AMX	After the move is over
WT 10000	Wait 10 seconds
SB 5	Turn on relay
EN	End Program

XQ (Execute Program)

DESCRIPTION:

The XQ command starts a previously entered program. Execution will start at the label or line number specified. Up to four programs may be executed simultaneously to perform multitasking.

The function can be used as an operand where _XQn returns the line number for thread n, and -1 if thread n is not running.

ARGUMENTS: XQ #A,n XQm,n

where A is a program name of up to seven characters

where m is a line number

where n is the thread number (0,1,2 or 3) for multitasking

USAGE:

While Moving	Yes	Default Value	n = 0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"HX"	Halt execution

EXAMPLES:

XQ #APPLE,0	Start execution at label APPLE, thread zero
XQ #DATA,2	Start execution at label DATA, thread two
XQ 0	Start execution at line 0

ZR (Zero)

DESCRIPTION:

The ZR command sets the compensating zero in the control loop or returns the previously set value. It fits in the control equation as follows:

D(z) = GN(z-ZR/z)

ARGUMENTS: ZR x,y,z,w ZRX=x ZR a,b,c,d,e,f,g,h

where x,y,z,w are unsigned numbers in the range 0 to 1 decimal with a resolution of 1/256

USAGE:

While Moving	Yes	Default Value	.9143
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"GN"	Gain
"KD"	Derivative
"KP"	Proportional
"KI"	Integral Gain

EXAMPLES:

ZR .95,.9,.8,.822	Set X-axis zero to 0.95, Y-axis to 0.9, Z-axis to 0.8, W-axis zero to 0.822
ZR ?,?,?,?	Return all zeroes
0.9527,0.8997,0.7994,0.8244	
ZR ?	Return X zero only
0.9527	
ZR ,?	Return Y zero only
0.8997	
ZRY=0.9	Y zero as .9
ZR*=0.89	All zeros as .89

ZS (Zero Subroutine Stack)

DESCRIPTION:

The ZS command is only valid in an application program and is used to avoid returning from an interrupt (either input or error). ZS alone returns the stack to its original condition. ZS1 adjusts the stack to eliminate one return. This turns the jump to subroutine into a jump. Do not use RI (Return from Interrupt) when using ZS. To re-enable interrupts, you must use II command again.

The status of the stack can be interrogated with the instruction _ZS. The response, an integer between zero and fifteen, indicates zero for beginning condition and fifteen for the deepest value.

ARGUMENTS: ZS n

where 0 returns stack to original condition

1 eliminates one return on stack

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

II1	Input Interrupt on 1
#A;JP #A;EN	Main program
#ININT	Input Interrupt
MG “INTERRUPT”	Print message
S=_ZS	Interrogate stack
S=	Print stack
ZS	Zero stack
S=_ZS	Interrogate stack
S=	Print stack
EN	End

Additional Program Examples

Homing Sequence

This example demonstrates how to home servos with a home sensor in the middle of a slide where it is possible for the servo to be on either side of the home sensor at power up. If the servo is already past the sensor, it will hit a limit switch first, and the #LIMSWI special label subroutine will reverse the CN command and turn the servo around. The #BACKUP subroutine is used to make the servo come back to the home input and go a small distance past it, so the #HOMING routine can always hit the same side of the home sensor.

Ideally, the home sensor is a photo device. If there is a white and black strip along the slide, the photo eye will see either light or dark, and the value of _HMX will be "1" or "0". Under this design, the FEX command can automatically determine the direction to find the transition point of the black and white strip. You should not need to account for the limit switch or the #BACKUP routine in the case.

```
#TEST
   SPX=10000
   ACX=1000000
   DCX=1000000
   CN,1
   TRUE=1
   FALSE=0
   HOMING=FALSE
   SHX; WT 2000
#HOMING
   MG "Attempting to find home "{N}
   MG "(normal direction)"
   FLX=200000000
   BLX=-200000000
   HOMING=TRUE
   JGX=8192; FEX; BGX; AMX
   HOMING=FALSE
   JG*=500; FIX; BGX; MCX      (MCX because controller will automatically  define the position as zero when index
                               found).
   MG "Homed O.K!" EN


#BACKUP
```

```
    MG "Going back to the "{N}
    MG "home input.."
    CN,-1
    FEX; BGX; AMX
    IPX=-20000; AMX                 (May need to  adjust number based on distance)
    CN,1
    JP #HOMING
EN

#LIMSWI; AB1                        (AB1 optional, to instantly stop all servos)
    JP #REALPRB,HOMING=FALSE        ( Do next part if limit during homing)
    ZS; WT 1000; JP #BACKUP                 (Next part handles a real limit event)
    #REALPRB
    MG "Limit Hit!"
RE1

#CMDERR                             (Command error handler special label)
    AB1; ZS
    JP #LIMSWI,_TC=22               (Refer to #LIMSWI handler if try to begin or motor off)
    MG "Error "{N};TC1{N}
    MG " on line",_ED{F3.0}
    MG "Program halted!"
    AB
EN
```

Absolute Encoder

This program can be used as a simple diagnostic tool for absolute encoder alarms. The variable XHmOfs is used as an offset to fine tune the zero point of the absolute encoder. If an absolute encoder alarm is encountered, the axis on which the error occurred is displayed. If a command error is encountered, the error message is displayed along with the line number. The program is also aborted in this case.

```
#ABSOLUT
XHmOfs=2344                         (A user defined variable to "fine tune" the zero point)
AEX=4096                    (Read Absolute Encoder with 4096 pulses per rev)
DPX=_TPX+XHmOfs             (Adds user defined offset to number received from encoder)
MG "POSITION=",_TPX
EN



#CMDERR                             (Command error handler special label )
    AB1; ZS
    JP #ABS_ALM,_TC=116             (116 is an absolute encoder alarm)
    MG "Error "{N};TC1{N}
    MG " on line",_ED{F3.0}
    MG "Program halted!"
    AB
EN

#ABS_ALM
```

```
  QY; MG " <= Encoder Alarm Code"
  Axis=_AE+65*$1000000
  MG "Absolute Encoder Error on -",Axis{S1},"- Axis."
EN
```

Port Two Interface

This program shows how to use an operator interface on port 2. The thread zero is used for gathering the data from the operator. Thread one is used for controlling the motion of the indexer. Input #1 makes one index per at every falling edge. Inputs #2 & #3 are used for manual jogging.

```
#AUTO
#CUT2LEN
    CC 9600,0,0,0; CI,1                    (The following are parameter settings)
    ACX=3000000; DCX=_ACX
    Response=0; Dwell=0; Speed=68266.6626          (The following are variables)
    Dist=2048; Cycles=0; X=0; Quit=0; Option=0
    StartInt=1; Cut=1
    JogXP=2; JogXN=3
    JogSpeed=50000
    MainCnt=0
    BSpace=""; Min=0; Max=0; ASpace=""
    DM MINCOL[4], MAXCOL[4]
    MINCOL[1]=7;  MAXCOL[1]=12
    MINCOL[2]=6;  MAXCOL[2]=13
    MINCOL[3]=7;  MAXCOL[3]=12
    TRUE=1;  FALSE=0                       (The following are constants)
    SConv=(8192/60)*30                     (Converts a percentage to 100% = 3000 R.P.M).
    DConv=8192
    MaxSpeed=3000                          (Max Speed in R.P.M.)
#MAIN                                      (This is the #MAIN loop of the program)
    JS #SHOMENU
    #LOOP
        JP #ASKSPD,P2CH="e"
        JP #ASKDIST,P2CH="f"
        JP #ASKCYC,P2CH="g"
        JP #SHOWCNT,P2CH="h"
        JS #START,_HX1=0
    JP #LOOP,Quit=0
    JS #CLRSCR
    Row=2; Col=2; JS #SETXY
    MG {P2} "Program Completed"
    HX1
EN


#START; SHX; WT 500; XQ #MOTION,1; EN      (This subroutine executes the #MOTION program)


#SHOMENU                                   (Display the Main Menu )
```

```
   JS #CLRSCR
   MG {P2} "F1) Set Speed      " {N}
   MG {P2} "F2) Set Distance   " {N}
   MG {P2} "F3) Set Quantity   " {N}
   MG {P2} "F4) View Batch Cnt" {N}
EN

#GETANS                                    (Used for getting all responses from port #2)
   CI,-1; WT 50; CC 9600,0,0,1
   IN{P2},Response
   CI,-1; WT 50; CC 9600,0,0,0
EN


#CLRSCR; MG {P2} {^27},"E" {N}; EN         (Clears the screen of the QSI terminal )



#SETXY                                     (Sets the X,Y position on the QSI terminal .  Just before calling this routine,
                                           set "Col" to 1 through 20.  Just before calling this routine, set "Row" to 1
                                           through 3.  Col and Row start at 1,1 in the upper left corner.)
   hRow=Row+63*$100
   hCol=Col+63
   SetXYStr=$1B490000 | hRow | hCol   //  QSI Code ESC "I" row col
   MG {P2} SetXYStr {S} {N}
EN


#ASKSPD                                    (Ask user for speed value )
   Option=1
   Response=Speed/SConv
   JS #CLRSCR
   MG {P2} "Speed (%) =",Response{F4.2}
   MG {P2} "Enter index speed   "{N}
   MG {P2} "in % of full-> " {N}
   Min="0"; Max="100"
   JS #GETANS
   JP #ERROR,Response<1
   JP #ERROR,Response>100
   Speed=Response*SConv
   JP #MAIN
EN


#ASKDIST                                   (Ask user for distance value)
   Option=2
   Response=Dist/DConv
   JS #CLRSCR
   MG {P2} "Distance =",Response{F3.2}," ft"{N}
   MG {P2} "Enter index distance"{N}
   MG {P2} "in feet. -> " {N}
   Min="0.1"; Max="4000"
   JS #GETANS
   JP #ERROR,Response<.1
```

```
   JP #ERROR,Response>4000
   Dist=Response*DConv
   JP #MAIN
EN


#ASKCYC                                    (Ask user for number of cycles )
   Option=3
   Response=Cycles
   JS #CLRSCR
   Row=1; Col=2; JS #SETXY
   MG {P2} "Set Indexes=",Response{F4.0} {N}
   Row=3; Col=2; JS #SETXY
   MG {P2} "How many indexes?"{N}
   Row=4; Col=1; JS #SETXY
   MG {P2} "(0 TO Quit) " {N}
   Min="0"; Max="1000"
   JS #GETANS
   JP #ERROR,Response<0
   JP #ERROR,Response>1000
   Cycles=@INT[Response]
   X=0
   JS #SETQUIT,Response=0
   JP #MAIN
EN


#SETQUIT; Quit=1; EN                       (Set a flag if number of cycles is 0)


#SHOWCNT                                   (The subroutine displays the batch count )
   JS #CLRSCR
   Row=1; Col=2; JS #SETXY
   MG {P2} "Run Speed =",Speed/SConv{F3.1},"%" {N}
   Row=3; Col=1; JS #SETXY
   MG {P2} "Batch Count =" {N}
   Row=4; Col=2; JS #SETXY
   MG {P2} "Press F1 to cancel" {N}
   Row=3; Col=14
   #UPDATE
      JS #SETXY;  WT 50
      MG {P2} MainCnt{F6.0}{N}
   JP #UPDATE,P2CH<>"e"
   CI,-1
   JP #MAIN
EN

#ERROR                                     (Subroutine to display all data entry errors)
   JS #CLRSCR
   MG {P2} "ERROR! NOT IN RANGE!"{N}
   Row=2; Col=MINCOL[Option]; JS #SETXY
   MG {P2} Min {S}," to " {N}
   Col=MAXCOL[Option]; JS #SETXY
   MG {P2} Max {S} {N}
```

```
    Row=3; Col=5; JS #SETXY
    MG {P2} "PRESS <ENTER>"{N}
    Row=4; Col=6; JS #SETXY
    MG {P2} "TO CONTINUE"{N}
    JS #GETANS
    JP #ASKSPD,Option=1
    JP #ASKDIST,Option=2
    JP #ASKCYC,Option=3
EN

#MOTION                                                    (Program that runs the motion)

    JS #MANUAL,@IN[JogXP] | @IN[JogXN] = TRUE
    JP #MOTION,X>=Cycles
    SPX=Speed
    PRX=Dist
    AI StartInt; AI -StartInt
    BGX; X=X+1; MainCnt=MainCnt+1; AMX
    SB Cut; WT 250; CB Cut
    JP #MOTION
EN



#MANUAL
    JP #JOGXP,@IN[JogXP]=TRUE
    JP #JOGXN,@IN[JogXN]=TRUE
    JP #XSTOP,@IN[JogXP] | @IN[JogXN] = FALSE
    JP #XSTOP,@IN[JogXP] & @IN[JogXN] = TRUE
    JP #MANUAL,@IN[JogXP] | @IN[JogXN] = TRUE
    STX; AMX
EN

#JOGXP; JGX=JogSpeed; BGX; JP #MANUAL; EN
#JOGXN; JGX=-JogSpeed; BGX; JP #MANUAL; EN
#XSTOP; STX; AMX; EN
```

Engineering Units

This program demonstrates how the SMC-2000 can be used with engineering units from an operator point of view. This program asks the operator to enter a speed in R.P.M. until they enter a zero. Everywhere in the program a speed or distance is required, use the engineering unit multiplied by the conversion factor.

```
#RPM
    SHX; ACX=1000000; DCX=_ACX
    RPM=8192/60                          (Convert RPM to counts per second)
    INCHES=8192*5/0.2                          (Pulses Per Rev * Gear Box / Ball Screw Pitch)

#LOOP
    IN "Enter the speed in R.P.M. ",Speed
```

```
    IN "Enter the distance in inches. ",Dist
    JP #QUIT,Speed<0.1
    JP #QUIT,Speed>4500
    SPX=Speed*RPM                    (Use engineering units to convert speed)
    PRX=Dist*INCHES                  (Use engineering units to convert distance)
    BGX
    AMX
    JP #LOOP
EN

#QUIT
    STX; AMX
    MG "Program STOPPED."
EN
```

Special Labels

This program demonstrates 5 of the 7 SPECIAL LABELS as part of an SMC-2000 Application Program. #AUTO is usually the first line of a program. When this program is burned into the SMC using the BP command, the program will begin executing when the power is turned ON or after the RS command is given, or the RESET button on the front of the SMC is pressed.

```
#AUTO
    ERX=150; OEX=1; II3
    SHX; WT 500
#BUSY
    JGX=@AN[1]*10000
    BGX
    MG "BUSY..."
    WT 500
    JP #BUSY
EN
```

#POSERR -- This special label is used to handle the situation when a servo is not able to remain in position. The special label works with the ER command. When the value of the ER command is exceeded, thread zero automatically jumps to the #POSERR label. In this program example, ERX=150 counts. If you have low gains or a small motor, you should be able to cause more than 150 counts of error by hand, causing the #POSERR label to execute. In the following example, the program displays a message and waits for input #1 to go low (falling edge). The servo is then re-energized.

There are 3 ways to return from a special label like this. The example below uses RE1. This means to return from the error routine to the line in thread zero that was being executed when the #POSERR occurred. The "1" means to restore a trippoint if one was in progress, such as WT, AI, AM, AT, etc.

The second way is to simply do an RE, which means that any trippoints that were in progress are cleared. This means that if thread zero was waiting for a AM command, it would continue as if the profiler had completed the path.

The third way is to use the ZS command, which clears the subroutine stack, and the SMC forgets that it is in the middle of an error routine. After the ZS is given, you can do a JP to anywhere in the program that is convenient. Typically there would be a jump back to a main loop where manual jogging can take place.

```
#POSERR
   SB1
   MG "FOLLOWING ERROR IS HIGH!"
   MG "TOGGLE INPUT #1 TO CONTINUE"
   AI1; AI-1
   CB1; SHX; WT 500
RE1
```

The following is the special label that is automatically executed when there is a programming error, a command given where it cannot be used, or a number out of range for a command. The example below includes a jump to the #LIMSWI label if the _TC code is 22, which is "Begin not valid due to limit switch." This is considered a command error, but is easier to treat as a limit switch error. Similar conditions could be handleed by checking other _TC code and reacting accordingly. If the error is anything other than 22, motion is aborted without aborting the program (AB1), then a message is printed indicating the type of error and what line number it happened on. NOTE: _ED reports the last line that had an error. The #CMDERR routine can be finished just like the #POSERR special label but this is not recommended because usually there is very little reason to continue execution of the program if there are serious errors in it. This routine is very useful in two ways:

#1 - During program design when there will be many programming mistakes, it is convenient to have the program display the error and line number automatically.

#2 - It is safer to abort motion if there is a program fault. Without the AB1 command, the motors will continue doing whatever they were doing before the fault. For example, if they were jogging, they would continue jogging.

```
#CMDERR
   JP #LIMSWI,_TC=22
   AB1
   MG "Error "{N};TC1{N}
   MG " on line",_ED{F3.0}
   MG "Program Halted!"
   AB
EN
```

The following is the #LIMSWI special label for handling situations where limit switches are hit during motion. This label automatically executes if an axis is in motion and a limit switch in the direction of motion is hit, or a software limit is exceeded. Without this special label, if a limit switch is hit during motion, such as a position absolute move, the motor will decelerate to a stop with NO ERROR. If a AM command was used, it would be cleared. The example as shown does not recover from the limit switch error, but a recovery method that works well is the use of a status flag variable. For example if the machine was in a manual jog operation, a variable could be used to indicate that it was in jog mode (JOGMODE=1) The first line in the #LIMSWI could jump to #PROBLEM if JOGMODE<>1, otherwise return from the error. The two commented lines below demonstrate this. (The JOGMODE variable would be set to "1" in your JOG routine and set back to "0" at the end of your jog routine)

```
#LIMSWI
   Limit="+"
   Axis="X";  JS #HARD,_LFX=0;  JS #SOFT,_FLX<_TPX
   Axis="Y";  JS #HARD,_LFY=0;  JS #SOFT,_FLY<_TPY
   Limit="-"
   Axis="X";  JS #HARD,_LRX=0;  JS #SOFT,_BLX>_TPX
   Axis="Y";  JS #HARD,_LRY=0;  JS #SOFT,_BLY>_TPY
        (JP #PROBLEM,JOGMODE=0; RE1)
        (#PROBLEM)
```

```
   AB1; HX1; HX2; HX3
   ZS
   MG "PROGRAM HALTED! (LIMSWI)"
EN

#HARD; MG Limit{S}," ",Axis," HARDWARE LIMIT HIT!"; EN
#SOFT; MG Limit{S}," ",Axis," SOFTWARE LIMIT HIT!"; EN
```

The following is the special label to handle input interrupts. Inputs 1-8 can be used as interrupts. This example uses an input to tell the SMC that the system is under an E-STOP condition. This input may come from a contact that also removes power from the amplifiers. Notice that the interrupt command II is used at the beginning of the program to designate input #3 as an interrupt. When this input goes low, thread zero automatically jumps to #ININT if it is included in the program. Notice that the example assumes that if an E-STOP occurs, the current operation has been scrapped. The ZS (Zero Subroutine Stack) command is used which allows the program to jump anywhere. Usually it is easiest to jump back to a main loop which handles the different modes of operation of the machine. Also note that if the ZS is used, the interrupt must be re enabled for next time.

```
#ININT
   AB1; HX1; HX2; HX3
   SB3
   MG "ESTOPPED!"
   AI-3; AI3                    (Wait for e-stop input to go high (re-enabled))
   CB3
   MG "RE-ENABLED.."
   SHX
   WT 2000
   ZS
   II3                          (Re-enable input interrupt for next time)
   JP #BUSY
EN
```

Options

Absolute Encoder (W-Option)

The SMC-2000 has the ability to interface to Yaskawa motors with absolute encoders. Absolute encoders allow the system to "remember" its position during power loss, even if the motor is moved while power is off.

To use the absolute encoder with the SMC 2000, there are several servo parameters that need to be set, and there are several special commands you must place in a program initialization sequence.

Setting servo(s) parameters

Yaskawa servo amplifier models **SGD**, **SGDA**, **SGDB** need to be set up to operate in a Torque Mode, as described in the parameter setting section of chapter 2 "tuning the servo system". The following parameters must also be set.

Parameter (SGD, SGDA)	Function	Setting
Cn-01, bit 1 , E	Absolute Encoder Selection	0,1
Cn-0A	Dividing Ratio setting	1024
Cn-11	Number of Encoder Pulses	1024

Parameter (SGDB)	Encoder Type	Function	Setting
Cn-01, bit 1 , E	W,S	Absolute Encoder Selection	0,1
Cn-0A	W	Dividing Ratio setting	1024
	S		8192
Cn-11	W	Number of Encoder Pulses	1024
	S		8192

Absolute Encoder Commands

This SMC-2000 has new firmware with an improved method of reading absolute encoders. The new firmware can be identified by entering <control>R <control>V in the terminal window of YTerm. This will return the firmware revision level. Firmware D150N19I and higher includes the following absolute encoder commands.

AE (Absolute Encoder)

Reads the absolute encoder for specified axis(axes).

Examples:

AEX=4096; Where 4096 is the encoder resolution after quadrature.

AE4096,4096,,,4096

The SMC-2000 will read the encoder and automatically Define the Position (DP) to the current value.

If an encoder had an alarm, a Command Error will result. If the #CMDERR special label is used, you can use _TC to determine what type of encoder problem was encountered. Possible errors are listed below.

114 Absolute encoder option not installed

115 Motor must be in MO for this command

116 Absolute encoder responded with an alarm

117 Absolute encoder did not respond

Note: _AE will report the last encoder read that was attempted. 0=X, 1=Y, 2=Z, and so on.

TY (Tell Yaskawa Encoder)

This command can be used to indicate where the absolute encoder was when an absolute read was done. If there was an alarm, this number will be 2147483647 (default).

TYX for one axis or

TY all axes or

_TYX to use as an operand

QY (Absolute Encoder Alarm Received Serially from Encoder)

This will return the serial string that was received from the encoder. The following is typical of what may be received. (These are on page 159 in the *SGDB Manual*.)

ALRMOA	Backup Alarm
ALARMOD	Battery Alarm
ALARMOB	Checksum Error
ALARMOP	Over Speed
ALARMOH	Absolute Error

ALARMOE Battery/Backup Combination Error

Extended I/O (I-Option)

The SMC-2000 has optional extended I/O that contains an additional 40 inputs and 42 outputs. All inputs are grouped into banks of eight, and work the same as standard inputs. All outputs are grouped into banks of seven with a separate common for each bank. However, the software will treat outputs as banks of eight, just like standard outputs, for simpler programming.

Extended I/O is available only through a Yaskawa factory modification due to required hardware change.

Output Specification (Extended I/O only)

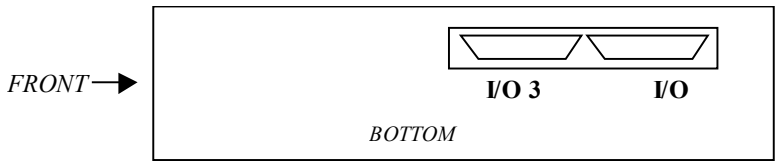
Switching Power:	Max. 10 Watt
Switching Voltage:	Max. 200 Vdc
Switching Current:	Max. 0.5 Amp.
Carrying Current:	Max. 1.25 Amp.

Part Number

SMC2000 X → I: Extended I/O Option.

Location of Connectors

The connectors for the Extended I/O option are located on the bottom side of the SMC-2000. There are (2) 50-pin connectors used for the Extended I/O option, as shown below.



Terminal Block

The standard 50 pin terminal block for the SGDB can be used for extended I/O. Yaskawa part number: JUSP-TA50P

Command Reference

	INPUTS	OUTPUTS
TI3	Input [32 25]	Lower Byte of OP,$FFFF Output [23:17]
TI4	Input [40:33]	Upper Byte of OP,$FFFF Output [31:25]
TI5	Input [48:41]	Lower Byte of OP,,$FFFF Output [39:33]
TI6	Input [56:49]	Upper Byte of OP,,$FFFF Output [47:41]
TI7	Input [64:57]	Lower Byte of OP,,,$FFFF Output [55:49]
		Upper Byte of OP,,,$FFFF Output [63:57]

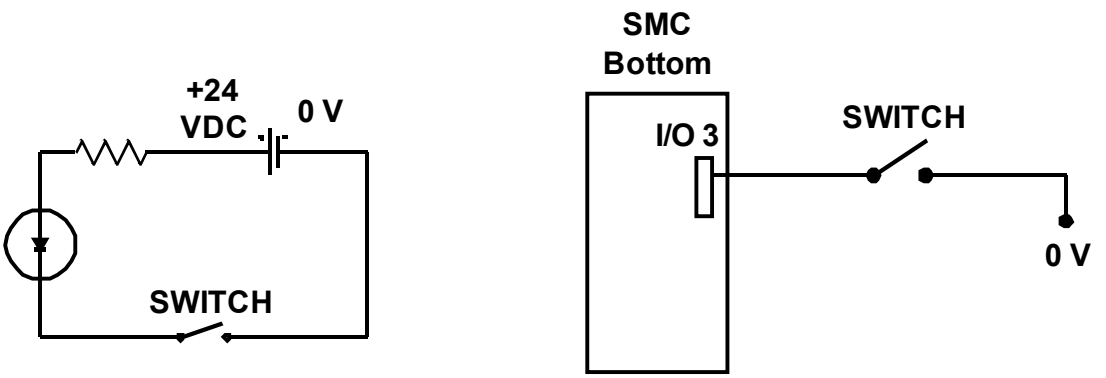
Note: Outputs 24, 32, 40, 48, 56, and 64 do not physically exist.

_OP# can be used to determine the port's current setting.

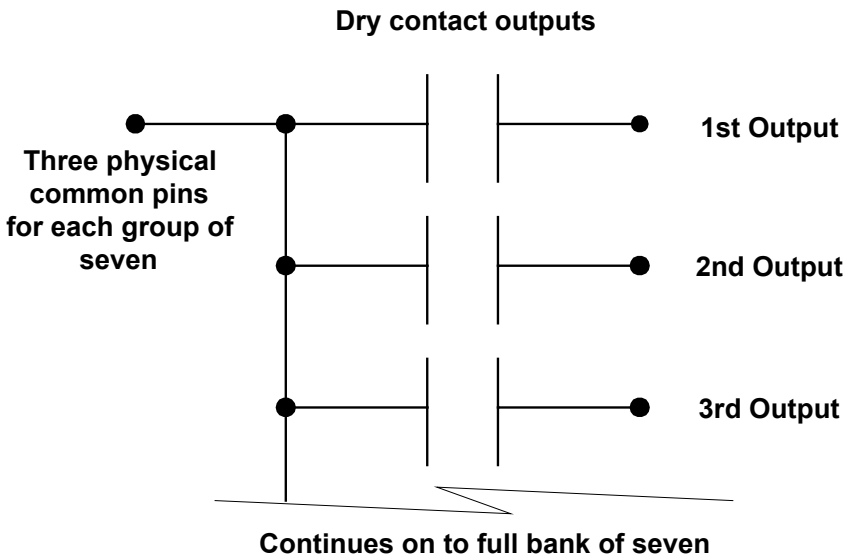
For example:

_OP, or _OP0	will report settings for Outputs	1 - 16
_OP1	will report settings for Outputs	17 - 32
_OP2	will report settings for Outputs	33 - 48
_OP3	will report settings for Outputs	49 - 64

Extended Input Diagram



Extended Output Diagram



I/O Port Lay Out

I/O 3 - 50 pin

Pin Number	Function	Pin Number	Function
1	Output 17-23 Common	26	Output 35
2	Output 17-23 Common	27	Output 36
3	Output 17-23 Common	28	Output 37
4	Output 17	29	Output 38
5	Output 18	30	Output 39
6	Output 19	31	Input 25
7	Output 20	32	Input 26
8	Output 21	33	Input 27
9	Output 22	34	Input 28
10	Output 23	35	Input 29
11	Output 25-31 Common	36	Input 30
12	Output 25-31 Common	37	Input 31
13	Output 25-31 Common	38	Input 32
14	Output 25	39	Input 33
15	Output 26	40	Input 34
16	Output 27	41	Input 35
17	Output 28	42	Input 36
18	Output 29	43	Input 37
19	Output 30	44	Input 38
20	Output 31	45	Input 39
21	Output 33-39 Common	46	Input 40
22	Output 33-39 Common	47	Input 41
23	Output 33-39 Common	48	Input 42
24	Output 33	49	Input 43
25	Output 34	50	Input 44

I/O 4 - 50 pin

Pin Number	Function	Pin Number	Function

1	Output 41-47 Common	26	Output 59
2	Output 41-47 Common	27	Output 60
3	Output 41-47 Common	28	Output 61
4	Output 41	29	Output 62
5	Output 42	30	Output 63
6	Output 43	31	Input 45
7	Output 44	32	Input 46
8	Output 45	33	Input 47
9	Output 46	34	Input 48
10	Output 47	35	Input 49
11	Output 49-55 Common	36	Input 50
12	Output 49-55 Common	37	Input 51
13	Output 49-55 Common	38	Input 52
14	Output 49	39	Input 53
15	Output 50	40	Input 54
16	Output 51	41	Input 55
17	Output 52	42	Input 56
18	Output 53	43	Input 57
19	Output 54	44	Input 58
20	Output 55	45	Input 59
21	Output 57-63 Common	46	Input 60
22	Output 57-63 Common	47	Input 61
23	Output 57-63 Common	48	Input 62
24	Output 57	49	Input 63
25	Output 58	50	Input 64

Industrial I/O Networks

As an alternative to the Extended I/O option [I], Yaskawa has added options to the SMC-2000 which make it possible to use the extended I/O of the SMC2000 as a slave node on a network. These networks are DeviceNet, ProfiBus, and InterBus-S, and have been designed to allow an I/O interface of 32 inputs and 32 outputs. Commands cannot be sent as they would on the RS-232 ports.

Note: All network I/O interface modules are supplied by Hassbjer Micro Systems of Halmstad, Sweden.

Part Numbers

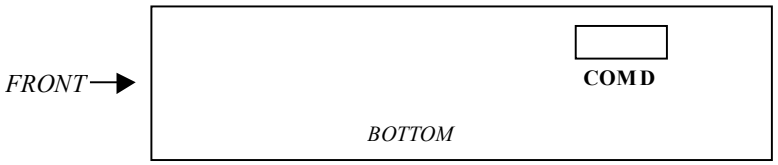
SMC2000 X
- D: DeviceNet Option
- P: ProfiBus Option
- S: InterBus-S Option

DeviceNet I/O (D-Option)

This option will provide 32 inputs and 32 outputs of extended I/O as a slave node on a DeviceNet network. The DeviceNet module resides inside the SMC-2000, and the network connection is made on the bottom panel of the unit, labeled COM-D. SMC commands are used to set the address and baud rate, as well as to reset the controller. SMC commands can also be used to monitor network status.

Location of Connectors

The connector for the DeviceNet I/O option are located on the bottom side of the SMC-2000. This connector is a 5-wire Phoenix type connector, as shown below.



Specifications and Technical Features

The media for the fieldbus is a shielded copper cable composed of one twisted pair and two wires for the external power supply. The baud rate can be changed between 125K, 250K, and 500kbit/s. This can be done in one of two different ways; first is simply by the DIP switches, the second way is via SMC commands.

Cable Specifications		
baud	**thin**	**thick**
125k	100m	500m
250k	100m	250m
500k	100m	100m

Configuration

The DeviceNet module is a slave node that can be read and written to only by a DeviceNet Master. The DeviceNet module will not initiate communication with other nodes, it will only respond to incoming commands. The DeviceNet for SMC2000 modules are Group 2 only servers using the pre-defined master/slave connection set for poll and bit strobing.

1) DeviceNet Connector

COM-D (Phoenix Type) Connector		
1	V-	Black
2	CAN_L	Blue
3	SHIELD	Bare
4	CAN_H	White
5	V+	Red

2) Terminating the last module

Termination of the fieldbus requires a terminating resistor at each end of the fieldbus. These resistors should have a value of 121 ohms. The resistor must be connected from CAN_L to CAN_H.

2) **Address Setting**
The SMC-2000 supports node address setting through software. The SMC-2000 dedicated outputs 49-54 (OP3) are used for this. See the example below.

3) **Baud rate**

There are three different baud rates for DeviceNet, 125k, 250k, or 500kbit/s. This is also settable through the SMC program using dedicated outputs 55 & 56 (OP3)

3) LED indicators

The table below shows the function of the LED indicators.

Comment	LED Color	Function	SMC Input
Internal Power	Green	Off = Network card power is off or reset	63 High
		On = Network card power is ON.	63 Low
Net Status	Red	Flashing = Recoverable fault	61 Toggle
		Solid = Critical module fault (no 24 VDC etc.)	61 Low
	Green	Flashing = Online but not connected	64 Toggle
		Solid = On-line, link OK, connected	64 Low
Address Overwritten	Red	Off = Address DIP switch is valid	62 High
		On = DIP switch not valid	62 Low

Sample program for SMC-2000 with DeviceNet option

This example simply shows how the baud rate and slave address can be set using SMC language. This program also monitors the connection, and upon connection to the network shows the value of all inputs once per second and sends a test pattern on the network outputs.

```
#DEV_NET
     OP 0,0,0,0
     NetOK=$3F          (Hex Status of inputs TI7 from DeviceNet card)
     OnLineNC=$BF       (Hex Status of inputs TI7 if Online but Not
                        Connected)
     ByteVal=$00
     VF -3.0                  (Format to show data as hexadecimal)
     NetReset=64              (Bit to RESET the DeviceNet card)
     IN "Enter DeviceNet Address (0-63). ",Address
     MG "Select the Baud Rate"
     MG "1) 128K"
     MG "2) 256K"
     MG "3) 512K",{^13}
     IN ,Baud                 (Get selection number for baud rate.)
     JP #LOW,Baud=1; JP #MED,Baud=2; Baud=$80  ; JP #CONFIG
     #LOW; Baud=$00; JP #CONFIG
     #MED; Baud=$40
#CONFIG
     Config=Baud | Address          (Combine values to make complete byte.)
     OP,,,Config                    (Set Address and baud rate)
     CB NetReset; WT 200; SB NetReset      (Reset DeviceNet card.)
     WT 1000; MG "Attempting to Connect..."{N}
     Attempts=0; AT0
#CONNECT
     NetStat=@IN[64]; AT -150
     NetStat=(NetStat | @IN[64]); AT -150
     NetStat=(NetStat | @IN[64]); AT -150
     NetStat=(NetStat | @IN[64]); AT -150
     NetStat=(NetStat | @IN[64]); AT -150
```

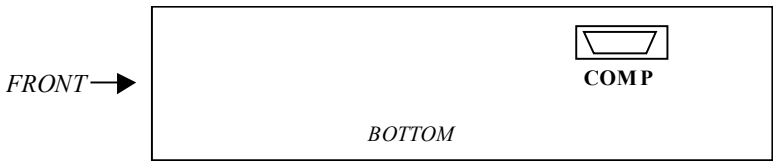
```
     Attempts=Attempts+1
     MG "."{N}
     JP #NETBAD,Attempts>25
     JP #CONNECT,NetStat<>0
     MG {^13}, "CONNECTED!"
#LOOP
     WT 1000
     MG "DATA BYTES = ",_TI3,"    ",_TI4,"    ",_TI5,"    ",_TI6 {N}
     MG "--     STATUS BYTE = ",_TI7
     MG "- - - - - - - - "
     OP,ByteVal*$100+ByteVal,ByteVal*$100+ByteVal
     ByteVal=ByteVal+1
     JP #LOOP,ByteVal<256
     ByteVal=$00
     JP #LOOP
EN


#NETBAD;
     MG "DeviceNet card is not working correctly!"
     AB
EN
```

ProfiBus (P-Option)

This option will provide 32 inputs and 32 outputs of extended I/O as a node on a ProfiBus DP network. The ProfiBus DP module resides inside the SMC-2000, and the network connection is made on the bottom panel of the unit, labeled COM-P. The address is set through dedicated SMC outputs 49-56. The network card can be reset via an SMC command and network status can also be monitored. ProfiBus-DP is used normally in industrial automation to transfer data for motor controllers, MMIs, I/O units and various other industrial applications. The Profibus-DP module is a slave node that can be read and written to by a Profibus-DP master. The Profibus-DP for SMC-2000 will not initiate communication to other nodes, it will only respond to incoming commands. This module does not support the ProfiBus-DP diagnostic functions.

Location of Connectors

The connector for the ProfiBus option are located on the bottom side of the SMC-2000. This connector is a 9-pin female DSUB connector, located as shown below.



Specifications and Technical Features

The media for the fieldbus is a shielded copper cable composed of a twisted pair. The baud rate for the bus is between 9.6kbaud and 12Mbaud. The ProfiBus-DP network consists of up to 32 different modules (126 with a repeater), and the total amount of data transfer is 246 Byte out/module and 246 Byte in/module.

Several different ProfiBus-DP Masters are available on the market, for both PLC-systems and PC computers.

Configuration

1) Memory Map

The Memory Map consists of the Process Data for the ProfiBus Network. This data is updated every ProfiBus cycle.

Address	Memory Map	Size	Description
000h - 03Fh	Input Data	64 bytes	Default no. of input bytes that can be addressed by all supported fieldbus systems
040h - 0C7h	Additional Input Data	136 bytes	Area of free bytes that can be assigned to input data during initialization of the ProfiBus-DP module
0C8h - 1DFh	Not Used	280 bytes	Not used in ProfiBus-DP SMC-2000
1E0h - 21Fh	Output Data	64 bytes	Default no. of output bytes that can be addressed by all supported fieldbus systems
220h - 2A7h	Additional Output Data	136 bytes	Area of free bytes that can be assigned to output data during initialization of the ProfiBus-DP module
2A8h - 3BFh	Not Used	280 bytes	Not used in ProfiBus-DP SMC-2000
3C0h - 3DFh	Fieldbus Specific Data	32 bytes	Fieldbus specific area, part of the control register
3E0h - 3FFh	Control Register	32 bytes	Control and status information of the fieldbus and the module

2) Setting up the module

SMC-2000 outputs 49-56 are used for setting the address of the ProfiBus Slave. The following chart describes the addresses. It is BCD format, 0-99. Note that the logic state is reversed.

	B3 (MSD)	B2(MSD)	B1 (MSD)	B0 (MSD)	B3 (LSD)	B2(LSD)	B1 (LSD)	B0 (LSD)
Address	OUT 56	OUT 55	OUT 54	OUT 53	OUT 52	OUT 51	OUT 50	OUT 49
01	1	1	1	1	1	1	1	0
02	1	1	1	1	1	1	0	1
…								
98	0	1	1	0	0	1	1	1
99	0	1	1	0	0	1	1	0

3) Baud rate

No setting of the Baud rate has to be done for the ProfiBus-DP module. The module has auto-baud rate setting from 9.6 kbaud - 12Mbaud.

4) Terminating the last module

The first and the last ProfiBus module must be terminated with a resistor for the bus to work properly. To do this on the PROFIBUS-DP module, just click down the terminating switch. This is the blue, two-position DIP switch near the COM-P connector. The termination switches should be turned off for all other modules in the network.

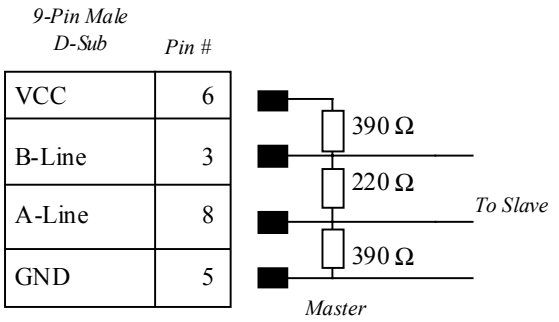
5) LED indicators

The table below shows the function of the LED indicators. The physical layout for the module is shown on the next page.

Comment	LED Color	Function	SMC Input
Bus Error	Red	LED Off = Normal, LED On = Bus Error	61 = HIGH = Normal 61 = LOW = Bus Error
Running	Green	LED Off = Not Running LED On = Running	62 = HIGH = Not Running 62 = LOW = Running
Power	Green	LED Off = No Power LED On = Slave has Power	63 = HIGH = No Power 63 = LOW = Slave has Power

6) Fieldbus connectors

The picture below shows both the pin function of the fieldbus connectors and the necessary resistors within the fieldbus cable.

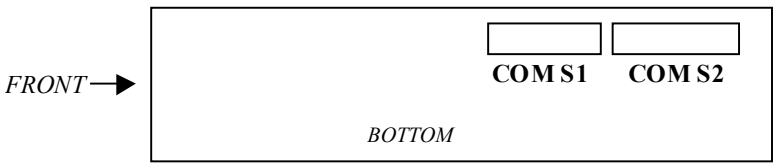


InterBus-S (S-Option)

This option will provide 32 inputs and 32 outputs of extended I/O as a node on an InterBus-S network. The InterBus-S module resides inside the SMC-2000, and the network connection is made on the bottom panel of the unit, labeled COM-S1 and COM-S2. The network card can be reset via an SMC command, and network status can be monitored via an SMC command.

Location of Connectors

The connectors for the InterBus-S option are located on the bottom side of the SMC-2000. COM S1 is a 6-wire Phoenix connector, and COM S2 is a 7-wire Phoenix connector, located as shown below.



Specifications and Technical Features

The media for the fieldbus is a shielded copper cable composed of three twisted pairs. Two of these pairs are used for the Bus connection and in the last pair only one wire is used. This wire is for the ground connection of the Bus. The only baud rate for the bus is 500 kbit/s. Total amount of data for InterBus-S is 4096 I/O points.

Configuration

1) Terminating the last module

If this is the last module on the network, your must connect RBST [pin 6] to GND [pin7] on the COM S2 connector.

2) Fieldbus connectors COM S1 and COM S2

COM-S1 Connector		COM-S2 Connector	
Phoenix Type		Phoenix Type	
1	DO1	1	DO2
2	/DO1	2	/DO2
3	DI1	3	DI2
4	/DI1	4	/DI2
5	GND	5	GND
6	PE	6	RBST
		7	PE

If you are connecting to an InterBus-S master that has a 9 pin female connector, note the pin connections below.

Master Cable Connection	
9 Pin D-Sub	Description
1	DO1
6	/DO1
2	DI1
7	/DI1
3	GND

3) LED Indicators

Regarding the fieldbus dependent parts on the SMC / IBS module, the modules are fully compatible with the InterBus-S specifications. They include the ID code 03 Hex (General I/O module). The table below shows the function of the LED indicators.

LED	Color	Function	SMC Input
RC	Green	LED Off = Incoming Remote Bus Not active	62 High
		LED On = Normal Operation	62 Low
BA	Green	LED Off = Bus not active	64 High
		LED On = Normal operation	64 Low
ERR	Red	LED Off = Normal operation	61 High
		LED On = Outgoing Remote Bus Not enabled	61 Low

Appendices

Appendix A - Electrical Specifications

Servo Control

ACMD Amplifier Command:	+/-10 Volts analog signal. Resolution 16-bit DAC or .0003 Volts. 3 mA maximum
A+, A-, B+, B-, C+, C- Encoder and Auxiliary	TTL compatible but can accept up to +/-12 Volts. Quadrature phase on CHA, CHB. Can accept single-ended (A+, B+ only) or differential (A+, A-, B+, B-). Maximum A, B edge rate: 8 MHz. Minimum C pulse width: 120nsec.

Input/Output

Uncommitted Inputs, Limits, Home Abort Inputs:	2.2K ohm in series with optoisolator. Requires at least 1 mA for on. Can accept up to 28 Volts without additional series resistor. Above 28 Volts requires additional resistor.
AN[1] through AN[7] Analog Inputs:	Standard configuration is +/-10 Volt. 14-Bit Analog-to-Digital converter.
OUT[1] through OUT[8] Outputs:	24 Volts, 600mA max. per point, not to exceed 800mA per group of 8
OUT[9] through OUT [16] Outputs	24 Volts 600mA max. per point, not to exceed 800mA per group of 8 (available on SMC-2000-8 only)
IN[17] through IN[24] Inputs	TTL (available on SMC-2000-8 only)

Power

+5V available power output	750 mA
+12V available power output	40 mA
-12V available power output	40 mA
AC input power requirement	15 W
DC input power requirement	100 mA per output point

Appendix B - Performance Specifications

Minimum Servo Loop Update Time:	SMC-2000-1 -- 250 µsec
	SMC-2000-2 -- 375 µsec
	SMC-2000-4 -- 500 µsec
	SMC-2000-8 – 875 µsec
Position Accuracy:	+/-1 quadrature count
Velocity Accuracy:	
Long Term	Phase-locked, better than .003%
Short Term	System dependent
Position Range:	+/-2147483647 counts per move
Velocity Range:	Up to 8,000,000 counts/sec
Velocity Resolution:	2 counts/sec
Motor Command Resolution:	16 Bits or .0003V
Variable Range:	+/-2 billion
Variable Resolution:	$1 \cdot 10^{-4}$
Array Size:	8000 elements - 30 arrays
Program Size:	1000 lines x 80 characters

Appendix C - Connectors

X (Y, Z, ... , G, H) - 20 pin

1 A (+) Encoder Pulse	11 Servo On (Amp Enable)
2 A (-) Encoder Pulse	12 STEP
3 B (+) Encoder Pulse	13 SEN/DIR
4 B (-) Encoder Pulse	14 5V Ground
5 C (+) Encoder Pulse	15 ALM
6 C (-) Encoder Pulse	16 +5V
7 Torque/Speed Reference Command	17 24V Ground
8 5V Ground	18 -BA
9 5V Ground	19 +24 Volts
10 5V Ground	20 +BA

D1 - 15 pin

1 Positive Overtravel X	9 Home Z
2 Negative Overtravel X	10 Positive Overtravel W
3 Home X	11 Negative Overtravel W
4 Positive Overtravel Y	12 Home W
5 Negative Overtravel Y	13 Abort input
6 Home Y	14 +12 Volts (for analog inputs)
7 Positive Overtravel Z	15 -12 Volts (for analog inputs)
8 Negative Overtravel Z	

D2 - 15 pin

1 Positive Overtravel E	9 Home G
2 Negative Overtravel E	10 Positive Overtravel H
3 Home E	11 Negative Overtravel H
4 Positive Overtravel F	12 Home H
5 Negative Overtravel F	13 Input 24 (TTL)
6 Home F	14 +12 Volts (for analog inputs)
7 Positive Overtravel G	15 -12 Volts (for analog inputs)
8 Negative Overtravel G	

I/O 1 - 25 pin

1 Analog 1	14 Output 5
2 Analog 2	15 Output 6
3 Analog 3	16 Output 7
4 Analog 4	17 Output 8
5 Analog 5	18 Input 8
6 Analog 6	19 Input 7
7 Analog 7	20 Input 6
8 24V Ground	21 Input 5
9 5 Volts	22 Input 4 (Latch W)
10 Output 1	23 Input 3 (Latch Z)
11 Output 2	24 Input 2 (Latch Y)
12 Output 3	25 Input 1 (Latch X)
13 Output 4	

I/O 2 - 25 pin

1 Input 17 (TTL)	14 Output 13
2 Input 18 (TTL)	15 Output 14
3 Input 19 (TTL)	16 Output 15
4 Input 20 (TTL)	17 Output 16
5 Input 21 (TTL)	18 Input 16
6 Input 22 (TTL)	19 Input 15
7 Input 23 (TTL)	20 Input 14
8 24V Ground	21 Input 13
9 5 Volts	22 Input 12 (Latch H)
10 Output 9	23 Input 11 (Latch G)
11 Output 10	24 Input 10 (Latch F)
12 Output 11	25 Input 9 (Latch E)
13 Output 12	

AE 1- 25 pin

1 Sample clock	14 Synch
2 W Aux. B-	15 W Aux. B+
3 W Aux. A-	16 W Aux. A+
4 Z Aux. B-	17 Z Aux. B+
5 Z Aux. A-	18 Z Aux. A+
6 Y Aux. B-	19 Y Aux. B+
7 Y Aux. A-	20 Y Aux. A+
8 X Aux. B-	21 X Aux. B+
9 X Aux. A-	22 X Aux. A+
10 5 Volt	23 5V Ground
11 X-axis Stepper mode jumper	24 Y-axis Stepper mode jumper
12 Z-axis Stepper mode jumper	25 W-axis Stepper mode jumper
13 No Connection	

AE 2- 25 pin

1 N.C.	14 N.C.
2 H Aux. B-	15 H Aux. B+
3 H Aux. A-	16 H Aux. A+
4 G Aux. B-	17 G Aux. B+
5 G Aux. A-	18 G Aux. A+
6 F Aux. B-	19 F Aux. B+
7 F Aux. A-	20 F Aux. A+
8 E Aux. B-	21 E Aux. B+
9 E Aux. A-	22 E Aux. A+
10 5 Volt	23 5V Ground
11 E-axis Stepper mode jumper	24 F-axis Stepper mode jumper
12 G-axis Stepper mode jumper	25 H-axis Stepper mode jumper
13 No Connection	

Note: The ABCD axes and other I/O are located on the main SMC-2000-4 card.

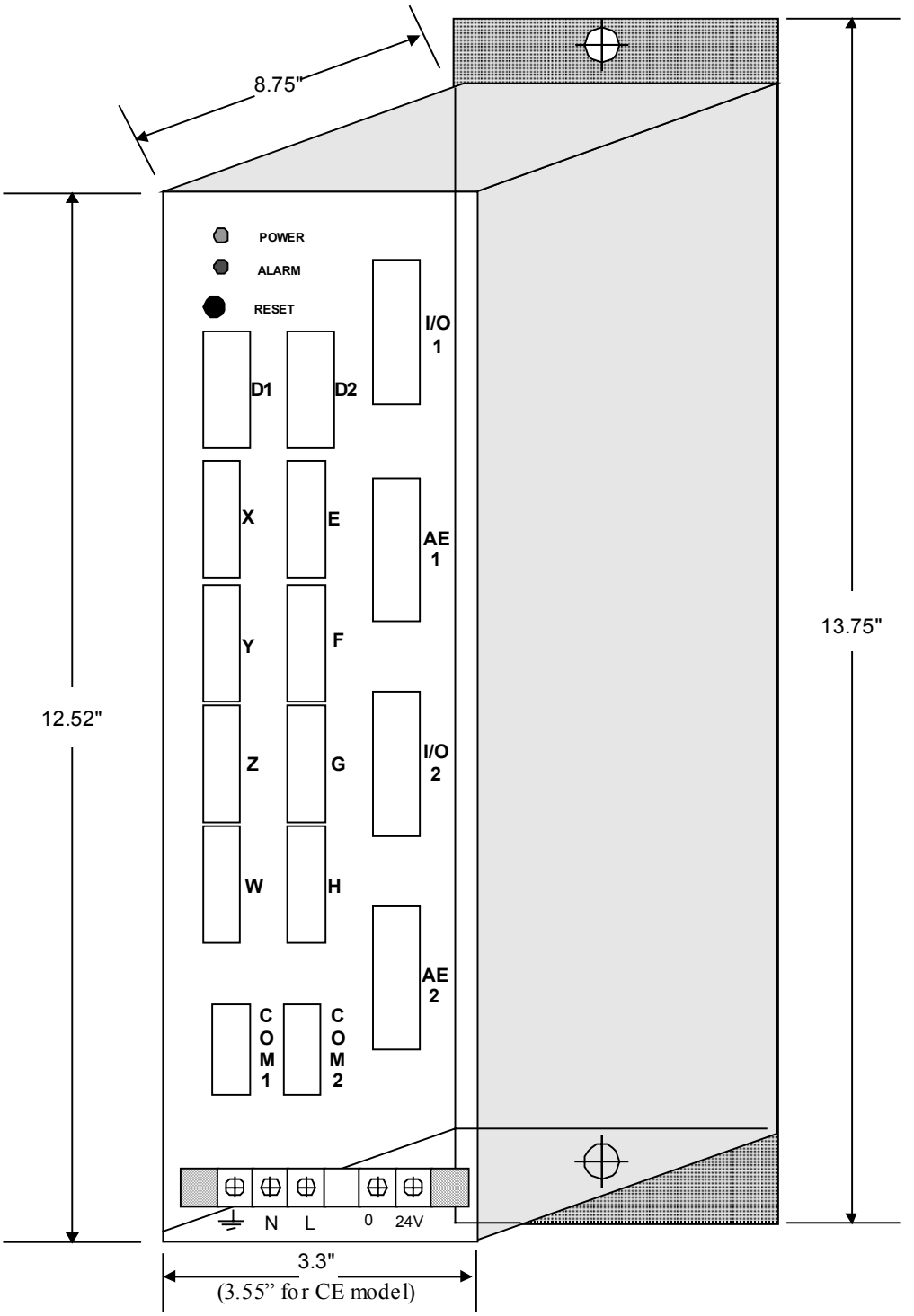
Appendix D - Pin-Out Description

Outputs	
Analog Motor Command	+/- 10 Volt range signal for driving amplifier. In servo mode, motor command output is updated at the controller sample rate. In the motor off mode, this output is held at the OF command level.
Servo On (Amp Enable)	Signal to disable and enable an amplifier. Amp Enable goes low on Abort and OE1.
Error	The signal goes low when the position error on any axis exceeds the value specified by the error limit command, ER.
Output 1-Output 8 Output 9-Output 16 (SMC-2000-8 only)	24 Volt open collector outputs (600mA max. per point; not to exceed 800mA per group of 8) are uncommitted and may be designated by the user to toggle relays and trigger external events. The output lines are toggled by Set Bit, SB, and Clear Bit, CB, instructions. The OP instruction is used to define the state of all the bits of the Output port.

Inputs	
Encoder, A+, B+	Position feedback from incremental encoder with two channels in quadrature, CHA and CHB. The encoder may be analog or TTL. Any resolution encoder may be used as long as the maximum frequency does not exceed 8,000,000 quadrature states/sec. The controller performs quadrature decoding of the encoder signals resulting in a resolution of quadrature counts (4 x encoder cycles). Note: Encoders that produce outputs in the format of pulses and direction may also be used by inputting the pulses into CHA and direction into Channel B and using the CE command to configure this mode.
Encoder Index, I+	Once-Per-Revolution encoder pulse. Used in Homing sequence or Find Index command to define home on an encoder index.
Encoder, A-, B-, I-	Differential inputs from encoder. May be input along with CHA, CHB for noise immunity of encoder signals. The CHA- and CHB- inputs are optional.
Auxiliary Encoder, Aux A+, Aux B+, Aux I+, Aux A-, Aux B-, Aux I-	Inputs for additional encoder. Used when an encoder on both the motor and the load is required.
Abort	A low input stops commanded motion instantly without a controlled deceleration. Also aborts motion program.
Reset	A low input resets the state of the processor to its power-on condition. The previously saved state of the controller, along with parameter values, and saved sequences are restored.
Positive Overtravel Limit Switch	When active, inhibits motion in forward direction. Also causes execution of limit switch subroutine, #LIMSWI. The polarity of the limit switch may be set with the CN command.

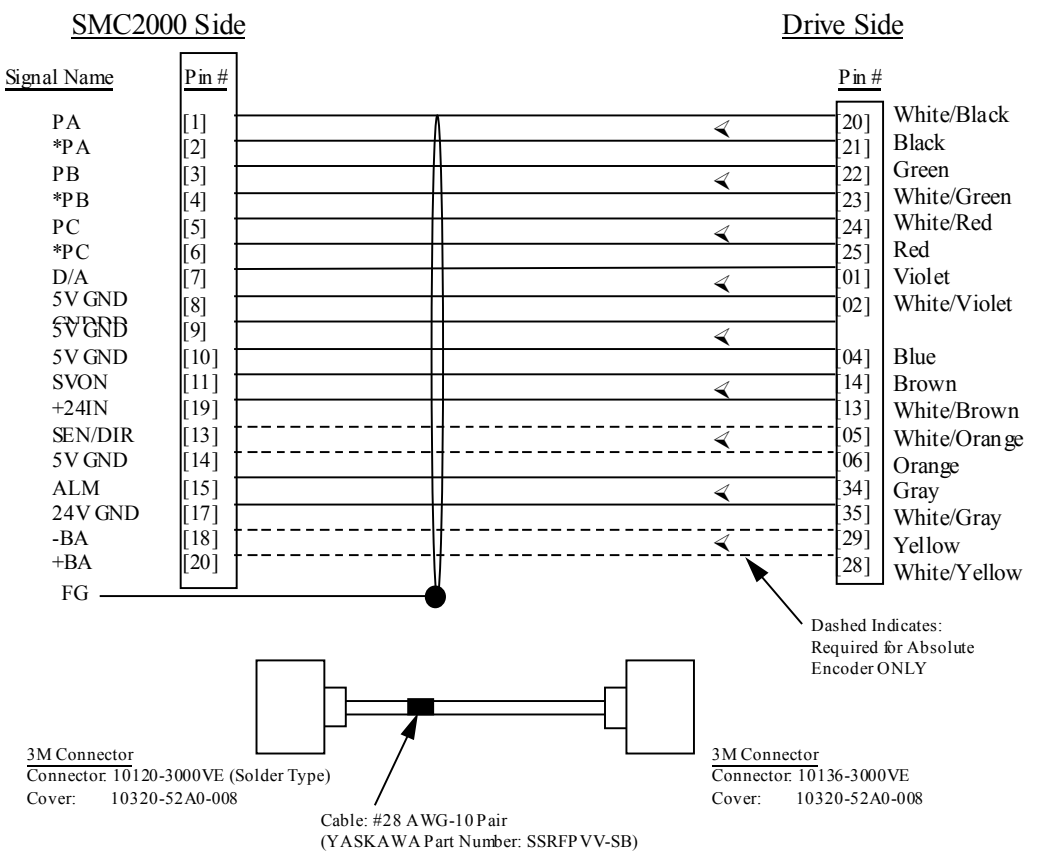
Negative Overtravel Limit Switch	When active, inhibits motion in reverse direction. Also causes execution of limit switch subroutine, #LIMSWI. The polarity of the limit switch may be set with the CN command.
Home (Zero Return) Switch	Input for Homing (HM) and Find Edge (FE) instructions. Upon BG following HM or FE, the motor accelerates to slew speed. A transition on this input will cause the motor to decelerate to a stop. The polarity of the Home Switch may be set with the CN command.
Input 1 - Input 8 *Input 9 - Input 16 isolated* *Input 17 - Input 23 - TTL*	Uncommitted inputs. May be defined by the user to trigger events. Inputs are checked with the Conditional Jump instruction and After Input instruction or Input Interrupt. Input 1 is latch X, Input 2 is latch F, Input 3 is latch Z and Input 4 is latch W if the high speed position latch function is enabled.
Latch	High speed position latch to capture axis position within 25 micro seconds on occurrence of latch signal. AL command arms latch. Input 1 is latch X, Input 2 is latch Y, Input 3 is latch Z and Input 4 is latch W. *Input 9 is latch E, Input 10 is latch F, Input 11 is latch G, Input 12 is latch H.*

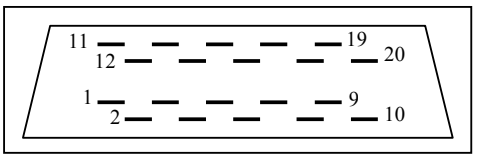
Appendix E - SMC-2000 Dimensions



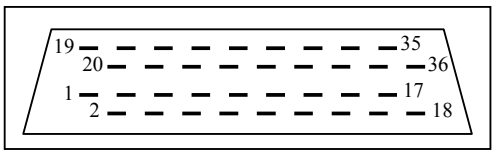
Appendix F - SMC-2000 Cable Layouts

Σ SGD/SGDA (Cable # SMCCBL1XX)





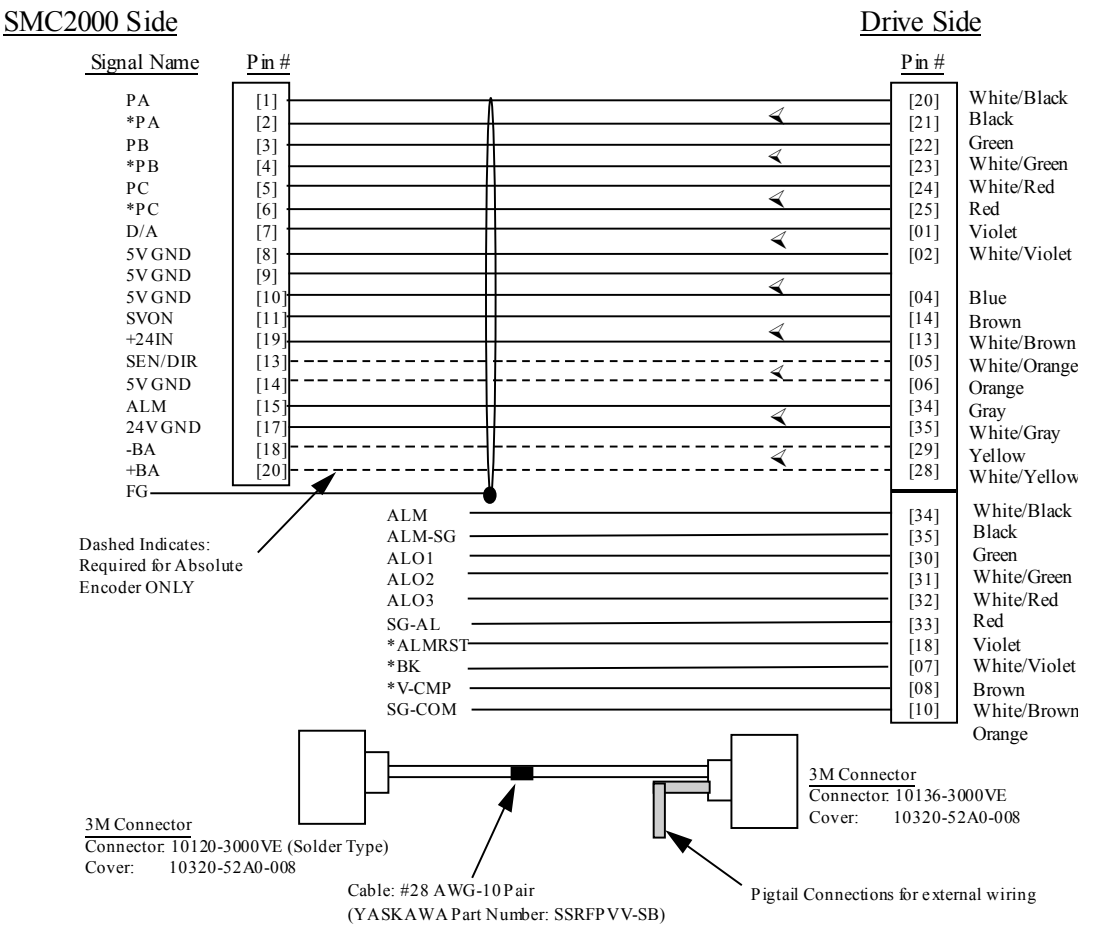
PIN NUMBER LAYOUT
Solder Side View
3M Connector 10120-3000VE
(Solder Type)
Cover: 10320-52A0-008

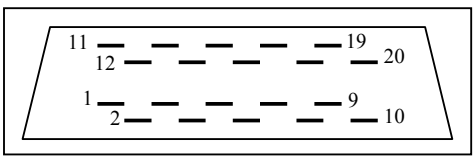


PIN NUMBER LAYOUT
Solder Side View
3M Connector 10136-3000VE

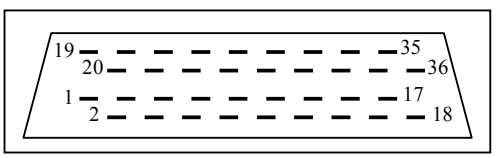
Cover: 10336-52A0-008

Σ SGD/SGDA w/ Breakout (Cable # SMCCBLAXX)





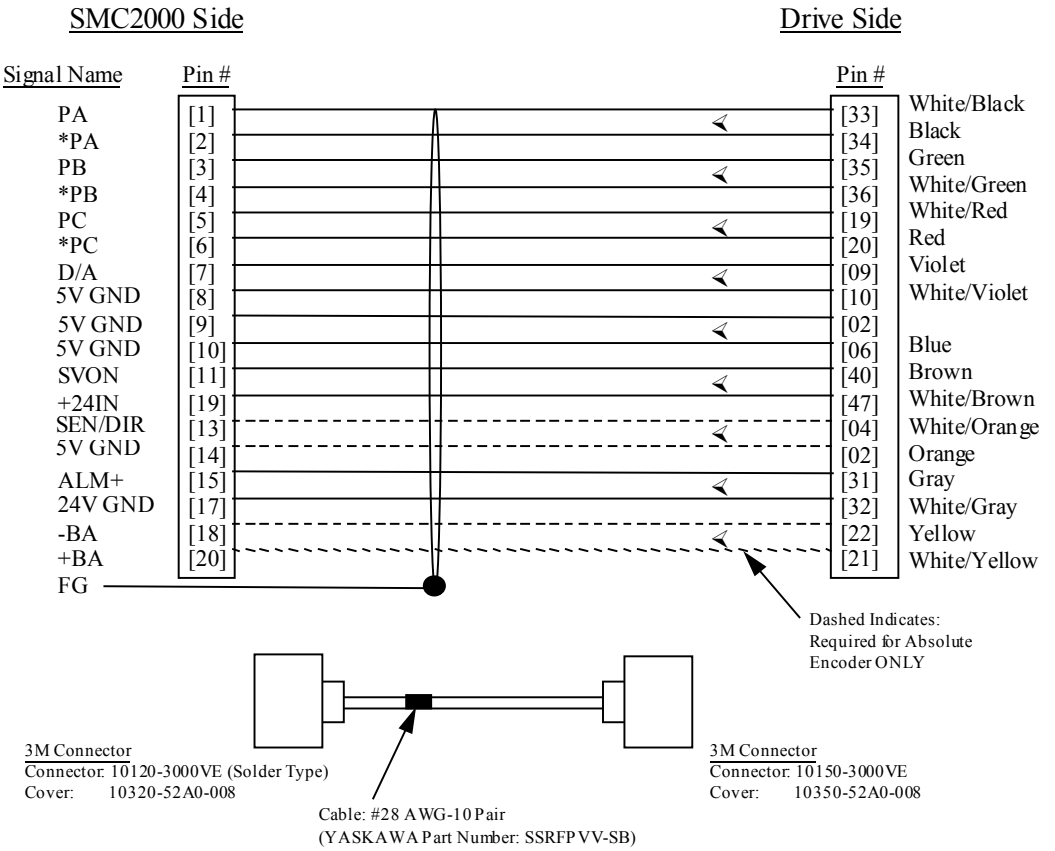
PIN NUMBER LAYOUT
Solder Side View
3M Connector 10120-3000VE
(Solder Type)
Cover: 10320-52A0-008

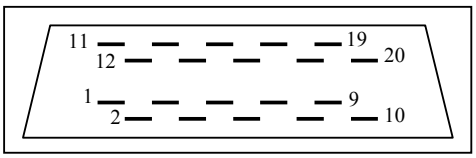


PIN NUMBER LAYOUT
Solder Side View
3M Connector 10136-3000VE

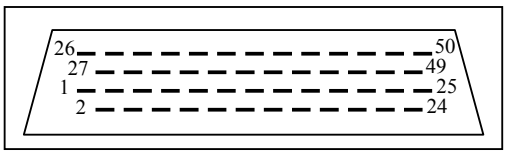
Cover: 10336-52A0-008

Σ SGDB (Cable # SMCCBL2XX)





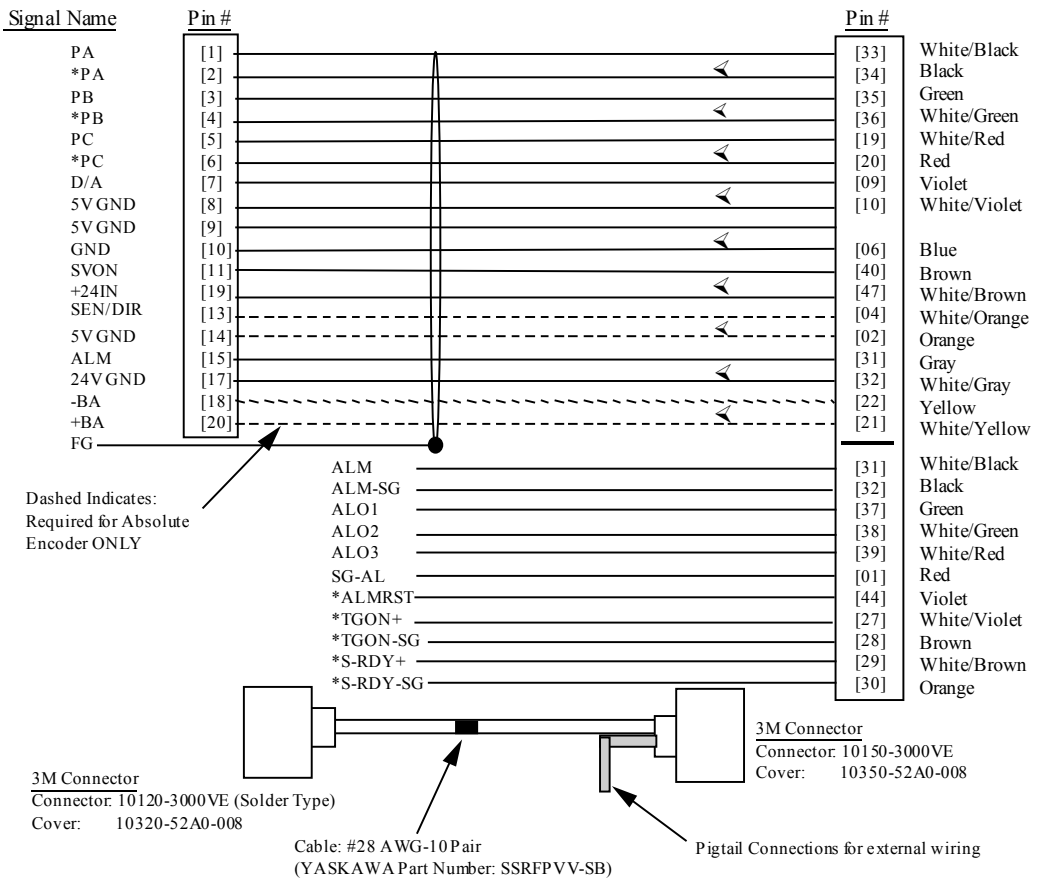
PIN NUMBER LAYOUT
Solder Side View
3M Connector 10120-3000VE
(Solder Type)
Cover: 10320-52A0-008



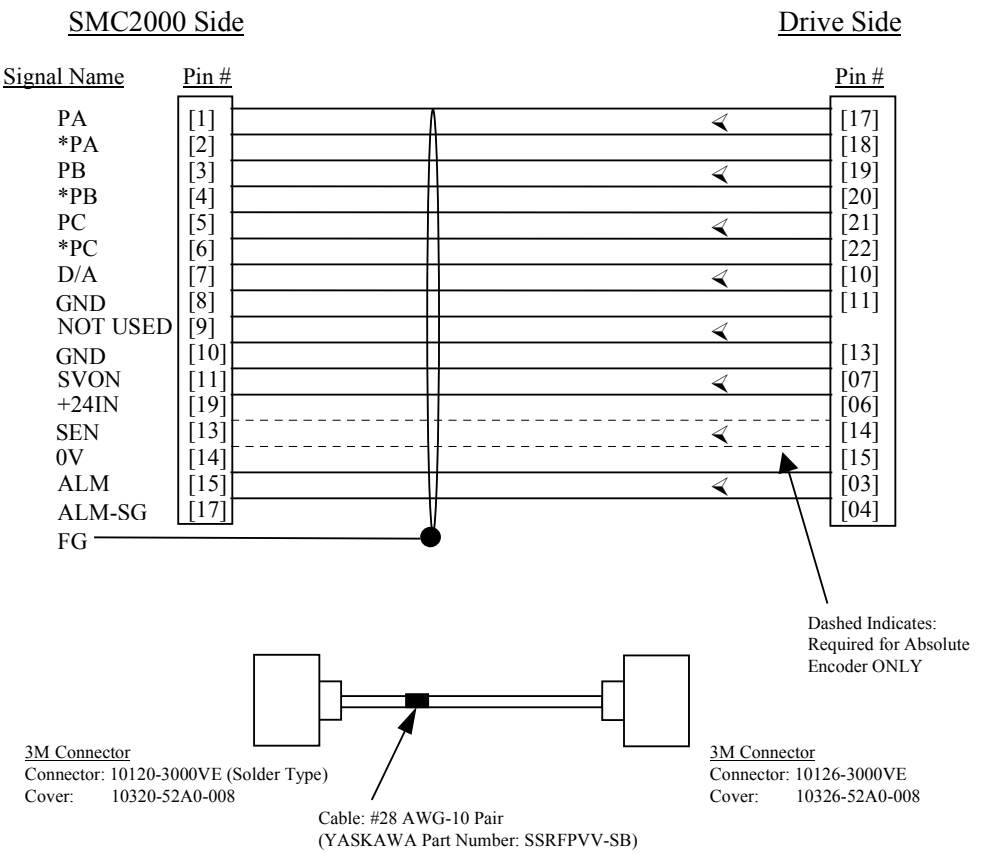
PIN NUMBER LAYOUT
Solder Side View
3M Connector 10150-3000VE

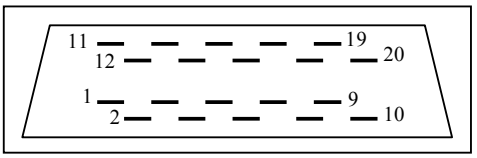
Cover: 10350-52A0-008

Σ SGD/SGDB w/ Breakout (Cable # SMCCBLBXX)

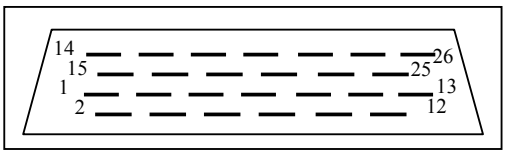


Σ SGDC (Cable # SMCCBL)





PIN NUMBER LAYOUT
Solder Side View
3M Connector 10120-3000VE
(Solder Type)
Cover: 10320-52A0-008

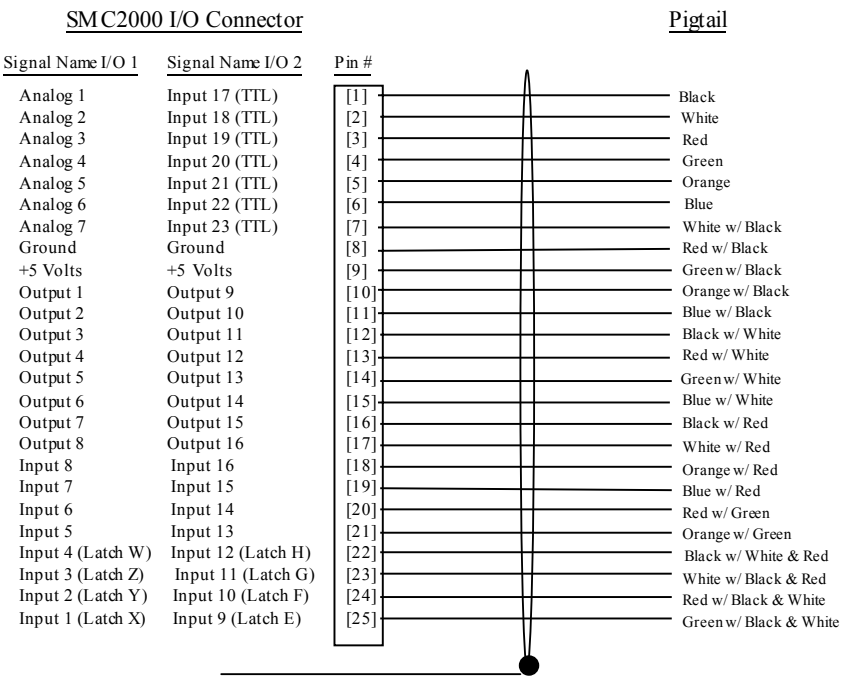


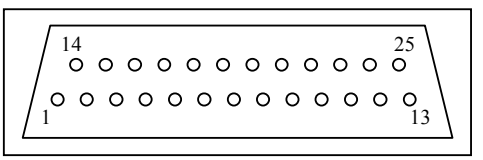
PIN NUMBER LAYOUT
Solder Side View
3M Connector 10126-3000VE

Cover: 10326-52A0-008

Σ MINI (Cable # SMCCBLMXX)

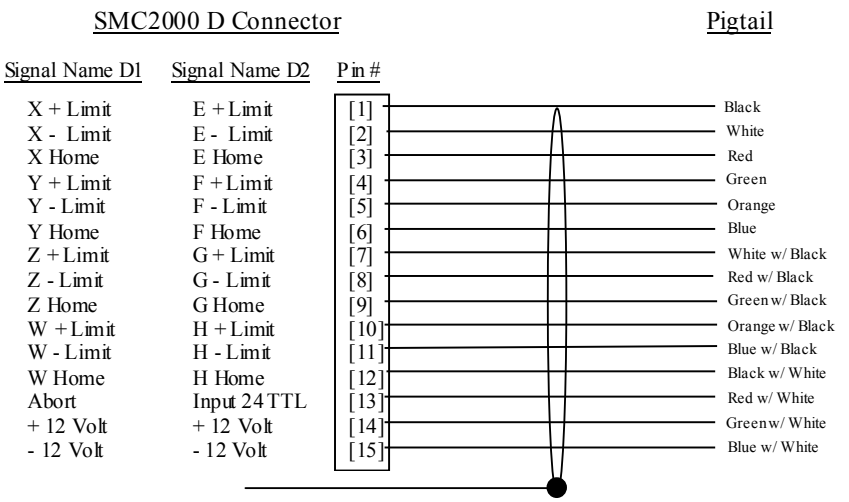
I/O (Cable # SMCCBL5XX)

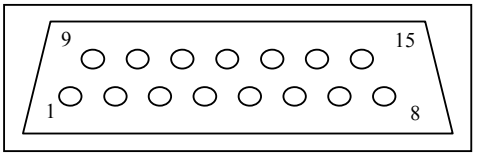




PIN NUMBER LAYOUT
Solder Side View
25D Male Connector

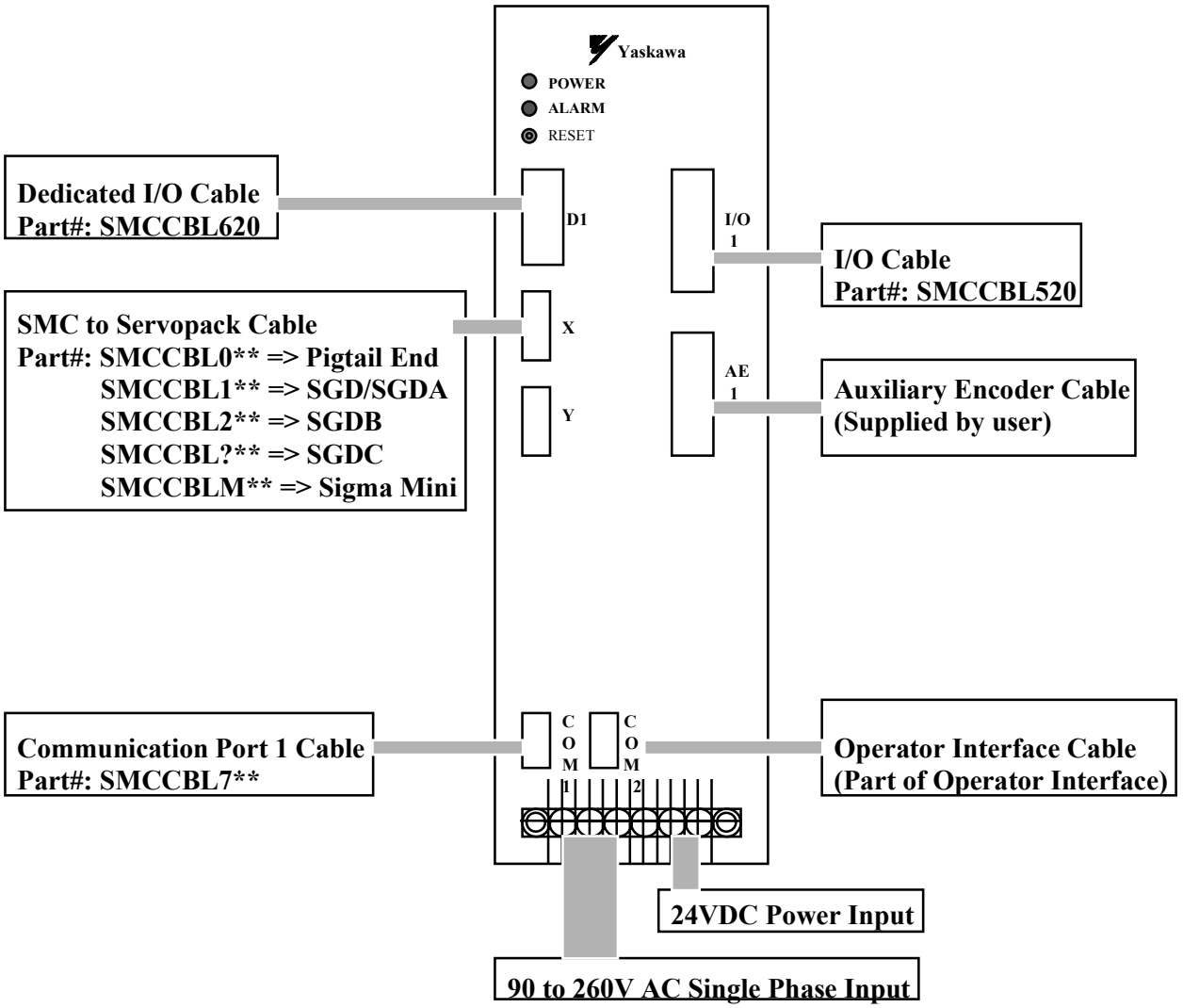
D (Cable # SMCCBL6XX)





PIN NUMBER LAYOUT
Solder Side View
15D Male Connector

Appendix G - Connection Diagram



**Length of the cable 05: 5 feet
10: 10 feet
15: 15 feet

Glossary of Terms

Absolute Position

Position referenced to a fixed zero position.

Amplifier

Electronics that convert low level command signals to high power voltages and currents in order to operate a servomotor.

ASCII

Abbreviation for American Standard Code for Information Interchange. This code assigns a seven (7) bit number to each numeral and letter of the alphabet. In this manner, information can be transmitted between machines as a series of binary numbers.

Axis

This is the principal direction along which movements of the tool or work piece occurs. The term also refers to one of the reference lines of a coordinate system.

Bandwidth

Term used for measurement of system response. Bandwidth is the frequency range that a control system can follow.

Baud Rate

The number of binary bits transmitted per second on a serial communications link, i.e. RS232.

Example: 19200 baud is approximately one (1) character every 520 µs (in 8+2 bit communication).

Bode Plot

A graphic recording of the magnitude of system gain in dB, and the phase of system gain in degrees versus the sinusoidal input signal frequency in logarithmic scale.

Closed-Loop

A system that has a velocity and/or position transducer to generate correction signals resulting from actual data compared to desired parameters.

Compensation

Corrective or control action in a feedback system, used to improve system performance characteristics, i.e. accuracy and response time.

Crossover Frequency

Frequency at which the gain intercepts the 0 dB point on a Bode. Used in reference to the open-loop gain plot.

Damping

An indication of the signal's rate of change compared to its steady state value. Related to settling time.

Damping Ratio

Comparison of actual damping to critical damping. A damping ratio of less then 1 is an underdamped system, and a value of over 1 is an overdamped system.

Duty Cycle

The ratio of on-time to total cycle time for a repeating cycle.

Efficiency

The ratio of power output to power input.

Encoder

A type of feedback device which converts mechanical motion into electrical signals to indicate actuator position. Typical encoders are designed using a printed disc and a light source. As the disc turns with the actuator shaft, the light source shines through the disc's printed pattern onto an optical sensor. The transmitted light is interrupted by the patterns on the disc. These interruptions are sensed and converted to electrical pulses. By counting these pulses, the position of the actuator shaft is calculated.

Filter

A transfer function used to modify the frequency or time response of a control system.

Gains

The ratio of a system's output signal to input signal. The control loop parameter that determines system performance characteristics.

Holding Torque

The maximum external force applied to a stopped, energized motor without causing the motor to continuously rotate. Also called static torque.

Home

A reference position in a motion control system derived from mechanical specification or external switch. Often used as the Zero position.

I/O

Abbreviation of Input/Output term. Input(s) refers to input signal(s) from switches or sensors. Output(s) refers to output signal(s) to relays, solenoids, etc.

Inertia

The unit measure of an object's resistance to change in speed. As an object's inertia increases, larger torque is needed to accelerate (increase) or decelerate (decrease) it's speed.

Jogging

A means of accomplishing incremental motor movement by repeated closing and opening of a circuit.

Optoisolated

A method of sending a signal from one piece of equipment to another without the usual requirements of common ground potentials. The signal is transmitted optically with a light source (i.e. light emitting diode) and a light sensor (i.e. photosensitive transistor) These optical components provide electrical isolation.

Position Error

The positional error sensed during motion determined by position control loops. It is the difference between the actual motor position and where it should be.

Ramping

Term to define the acceleration and deceleration of a motor.

Rated Torque

The torque producing capacity of a motor at a given speed. This is also the maximum torque that the motor can deliver to a load, and is usually specified with a torque/speed curve.

Regeneration

Usually refers to a circuit in a drive amplifier that accepts and drains energy produced by a rotating motor either during deceleration or free-wheel shutdown.

RS232

An EIA standard that specifies characteristics for serial binary communication circuits in a point-to-point link. This single ended hardware configuration is used extensively in office computer equipment.

RS422

Serial data communications hardware standard similar to RS232. The major difference is that RS422 is a differential hardware configuration providing improved noise immunity for use in more hostile environments like industrial conditions. RS422 provides longer communication distances and several devices can be placed on one link in a multidrop configuration. A simple converter is available to connect RS232 devices to RS422 devices.

Serial port

A digital data communications port configured with a minimum number of signal lines. This is achieved by passing binary information signals as a timed series of "1"s and "0"s on a single line.

Servo amplifier

An electronic device that produces the winding currents for a servo motor.

Servo System

Automatic feedback control system for mechanical motion in which the controlled or output quantity is position, velocity or acceleration. Servo systems are closed-loop systems.

Sigma

Sigma Servo System is a compact, high performance combination of brushless servomotors that match with flexible (all digital) amplifiers manufactured by Yaskawa. The Sigma series of motors provide high torque to inertia, higher torque per overall physical length and higher torque per dollar ratios then previous generations of brushless servomotors.

Slew speed

The maximum velocity at which an encoder will be required to perform consistently.

Speed

The linear or rotational velocity of a motor (or any other object in motion).

Torque

A measure of angular force which produces motion. This force is defined by a linear force applied at a distance e.g. lb./inch or oz/inch. Torque is one of the most important performance parameters for any motion control system.

Transfer Function

A mathematical means of expressing the output to input relationship of a system. Expressed as a function of frequency.

Yaskawa

Yaskawa, founded in 1905, is a global leader in the research, design, development and manufacturing of industrial and commercial electrical products. Yaskawa is the first manufacturer specializing in industrial electronic equipment to win the prestigious Deming Application Award.

YTerm-2000

YTerm-2000 is a front end to the SMC-2000 Multi-Axis Motion Controller. YTerm offers the user a Windows interface for design, development, and control of motion control applications

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Y

Z