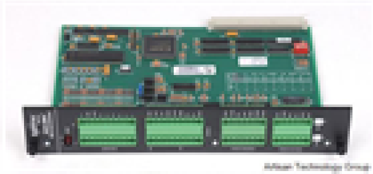
Superior Electric SLO-SYN 221797-001

Dual Axis Interface Module



PRICE: $25.00

INSTRUCTIONS
for
SUPERIOR ELECTRIC
SLO-SYN® MODEL MX-2000
PROGRAMMABLE MULTI-AXIS
MOTION CONTROLLER

INSPECTION

When unpacking the control, examine it carefully for any shipping damage. The "Damage and Shortage" instruction packed with the unit outlines the procedure to follow if any parts are missing or damaged.

QUALITY ASSURED FIRM
ISO
9002
SGS
Manufactured under an ISO 9002
compliant manufacturing system

WARNER ELECTRIC

DANA

ENGINEERING CHANGES

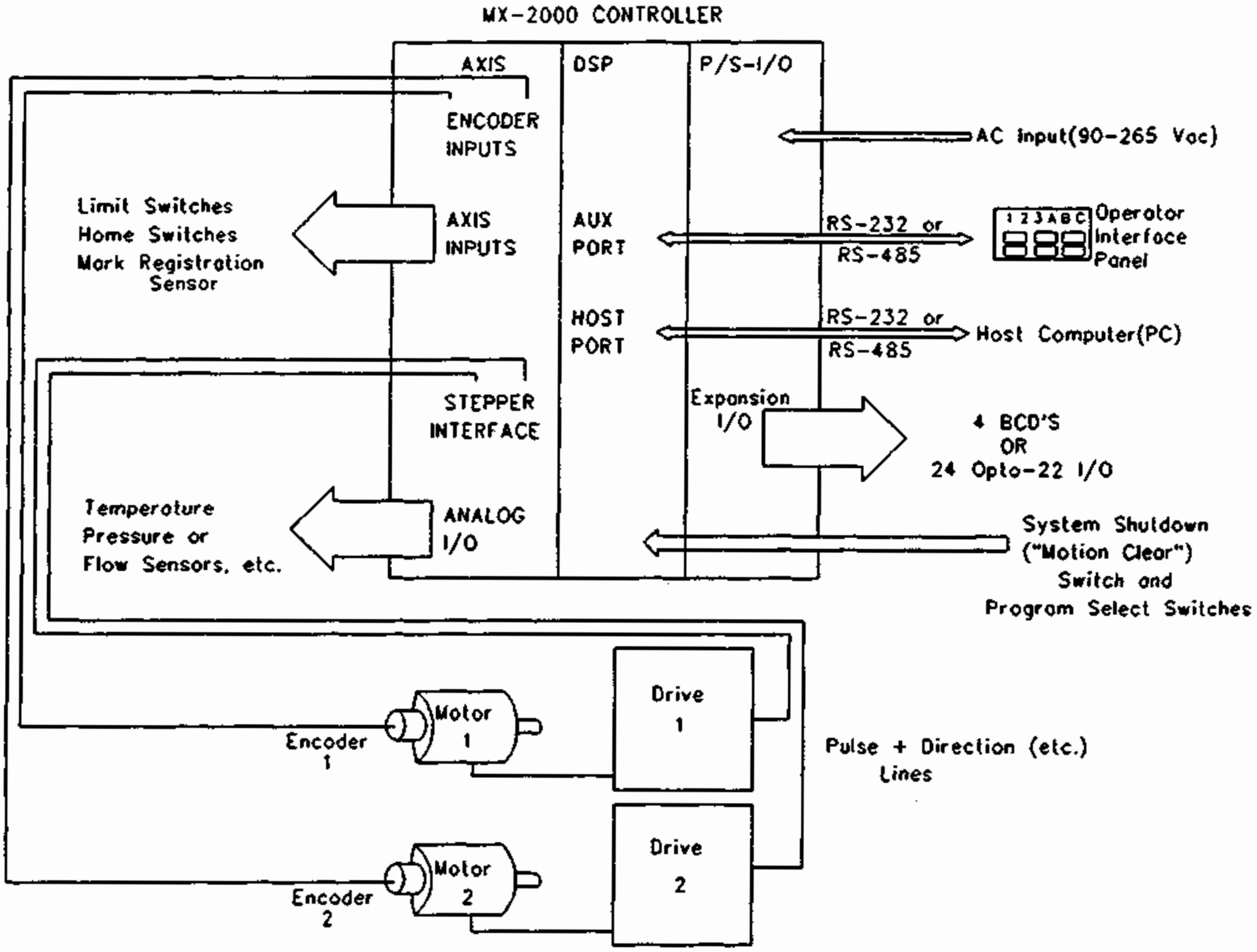
Superior Electric reserves the right to make engineering refinements on all its products. Such refinements may affect information given in instructions. Therefore, **USE ONLY THE INSTRUCTIONS THAT ARE PACKED WITH THE PRODUCT.**

The MX2000-2 and MX2000-6 are UL Recognized components, File No. E146240.

Table of Contents

SECTION & TITLE PAGE

Figure 4.1, General Application Overview



4.2 - Use of the Serial Ports, "HOST" and "AUXILIARY"

The MX2000 controller has two serial ports, which are identified as "HOST" and "AUXILIARY". The "HOST" port, as it's name implies, is typically connected to a host computer such as an IBM PC or compatible. The "AUXILIARY" port is intended for use with an operator interface panel such as Superior Electric's IWS series product line

The "HOST" port is used for downloading the user's application program and for direct control of the MX2000. When using either the DOS-based MX2000 program or the Windows-based Motion Workbench™ program all communication with the MX2000 controller is via the "HOST" port. In addition, all on-line debugging is accomplished using this port. The "HOST" port also has the capability to "DAISY CHAIN" to other controllers; this requires only one serial port on a user's host computer to communicate to multiple controllers. While the user's program could use the "HOST" port for communication with any device that has a serial port, it is recommended that the "HOST" port be reserved for debugging the user's program and for communication with the host computer.

The "AUXILIARY" port, while intended for use with an operator interface panel (O.I.P.), can in fact communicate with any device that has a serial port, such as counter units, etc. The "AUXILIARY" port can send and receive standard ASCII characters. The user's application program can transmit a prompt or message using the "PRINT" statement and wait for a response using the "INPUT" statement.

Example:

```
PRINT #2,"Enter 6 digit part number"
INPUT #2, PART$
```

A message is displayed on the OIP screen prompting the machine operator to enter a part number. The string variable PART$ can now be examined (by the controller program) to determine what type of process to perform. The information provided by the operator can then be used to control the process flow, ie. move distance, velocity, dwell, etc., for the desired part number that the machine is processing.

While the process is in operation, messages can also be sent back to the OIP, telling the operator the status of the process. For example,

```
PRINT #2, "Coarse grind"
PRINT #2, "Finish grind"
```

will display the indicated messages on the OIP regarding the grinding operation that is occurring.

(this page intentionally left blank)

SECTION 5 - SETUP AND INSTALLATION

5.1 - SWITCH SETTINGS AND INDICATOR LIGHTS

5.1.1. - *Switch and Jumper Settings*

Before mounting and installing the Programmable Motion Controller, it is best to set the switches and internal jumpers that govern various operating features.

Serial Communications Parameter Switches

1. The "BAUD" DIP switch located with the "HOST" serial port connectors needs to be set to match the Controller's baud rate with that of the host computer or terminal to which it is connected. The factory default is 9600 baud; if this is not what is desired, then set the switches toward one of the appropriate values shown on the label. Valid selections are "9.6" (9600), "19.2" (19,200), and "38.4" (38,400). If all switches are "off" (toward the right), then the baud rate is set to 4800. These switches are only read at power-up, so changing the baud rate requires a power-down, power-up cycle before the change takes effect.

Although the MX2000 serial ports are configurable for up to 38.4 Kbaud the serial communications may be limited when communicating with a PC. A PC may not be able to receive data from the MX2000 indexer at a baud rate above 9600. This limitation is due to the PC's inability, at the higher baud rates, to read the received character in time, ie. before another character is received. If this happens, an OVERRUN ERROR will occur. This problem will not exist if the serial port's UART has hardware buffering. The following is a list of UARTs commonly used on PC serial port cards. The UARTs marked with an * are buffered.

UARTs: 8250, 16450, 16451,
16452, *16550,*16552

The following is a table of indexer operation vs. Max. receive baud rate.

OPERATION	UART (unbuffered)	UART (buffered)
Load operating system	38400*	38400
Load user program	38400*	38400
Extract user source code	9600	38400
Host commands	9600	38400

* The unbuffered UART will perform the first two operations at the higher baud rate, since during these operations, the indexer does not transmit multiple characters in succession.

This DIP switch also needs to be set to match the type of host, RS-232 or RS-485, to which the Controller is connected. The factory default is for "232" (RS-232); if RS-485 is desired, set the switch toward "485".

Communications protocol (RS232 or RS485) for the Auxiliary Serial Port on the DSP Card is selected via a jumper on the card, immediately behind the port connector. The factory default is for RS485; if RS232 is desired, the DSP Card must be removed and the jumper setting changed to the RS232 setting. Baud rate for this port is set as a factory default at 9600; if another value is desired, select it via software from the Host port, using the "SETCOM" command. See Section 6 for further details on this command.

Serial communications format for both ports is "N-8-1", or No parity, 8 data bits, and 1 stop bit.

2. The Controller is capable of being operated in "daisy-chain" fashion, with up to 9 units connected to a single host. This is described in further detail in Sections 5.3.2 and 6.2. Each unit in such a chain needs a unique identification number (ID #); this value is entered via the "UNIT ID" select switch on the unit's front panel. The unit is shipped with a factory default ID # of 1 (for the first unit in the chain). If needed, set the ID # select switch to a different value by using a small screwdriver. Set the pointer on the switch to the desired value of ID #, from 1 to 9.

Axis Card Inputs - Sink/Source Polarity Jumper

Your Programmable Motion Controller offers the optimum in flexibility when using the inputs on the Axis Card. A jumper allows selection of either "sinking" or "sourcing" modes for the "axis inputs" on the Axis Card: +Lim, -Lim (limit switch inputs), and EVNT 1 and EVNT 2 ("event" inputs for homing or mark registration).

To access this jumper, you must first disconnect power from the unit, then remove the Axis Card (see Cautions in Section 2.3). This is done by loosening the thumbscrews at the top and bottom of the card's faceplate, then pulling it straight out to disengage the DIN connector at the lower rear edge of the card from the passive backplane at the rear of the enclosure.

The Sink/Source jumper can be found in the upper center of the card (near the Encoder connector), and is clearly marked **"Sink/Source"**on the circuit card silkscreen label. If set toward "Sink" (the factory default), all of these inputs will be configured for sinking mode. If source mode is desired, set the switch toward the word "Source".

(Note: this jumper is located inside the unit to prevent inadvertent change once the unit is set up; this helps avoid "nuisance" field problems that could cause the I/O not to work because of an external switch being accidentally moved.)

5.1.2 - *Indicator Lights*

The separate cards that comprise the MX2000 unit have one or several indicator lights (red LED's) as follows:

Power Supply Card: "POWER" indicator. When lit, signifies the unit power supply is energized.

DSP Controller Card: "FAULT" indicator. When lit, signifies a programming error, processor error, or motion error has occurred. The Fault LED flashes a code to signify the particular error that occurred. See Section 6 - Programming ("ERROR") and Section 8 - Troubleshooting, for further details.

Axis Card: "BUSY" indicators, one for each axis, "A" and "B". When lit, these signify that motion is occurring on the indicated axis.

5.2 - INSTALLATION

It is important to select a mounting location for you controller that will meet the environmental specifications listed in Section 10. Avoid locations that expose the unit to extremes of temperature, humidity, dirt/dust, or vibration.

Also, it is best to avoid areas with high "electrical noise." This will help to prevent misoperation due to electromagnetic interference. Please refer to Section 5.3.1 for general guidelines on selecting a location for your controller where it will be less susceptible to EMI/RFI problems.

When mounting the unit near other apparatus, such as inside an electrical cabinet or enclosure, please leave at least 2 inches of space on all sides for proper cooling. Mounting brackets are supplied to attach the controller to a vertical surface. The MX2000-8 can also be mounted in a standard 19 inch rack configuration by removing the mounting brackets and rotating them 180°. Please refer to Figures 5.1, 5-2, and 5-3 for overall dimensions and mounting hole locations for the MX2000-2, -6, and -8 respectively.

5.3 - WIRING THE CONTROLLER FOR OPERATION

The following pages show how to wire up the controller. Equivalent circuits are shown for the inputs and outputs, in both sinking and sourcing modes. All of the terminal strip connectors have their terminals clearly labeled. Please see Section 10 - Specifications for the pin-outs of the all the connectors and the ratings, signal characteristics, etc. Be sure to observe the listed electrical ratings of the ac input and the various I/O circuits; this will ensure proper, reliable operation of your controller. Please see Appendix A, Optional Cards, for additional information concerning the optional cards available for use in the MX2000-6 and MX2000-8 configurations.

Figure 5.1, MX2000-2 Outline Drawing

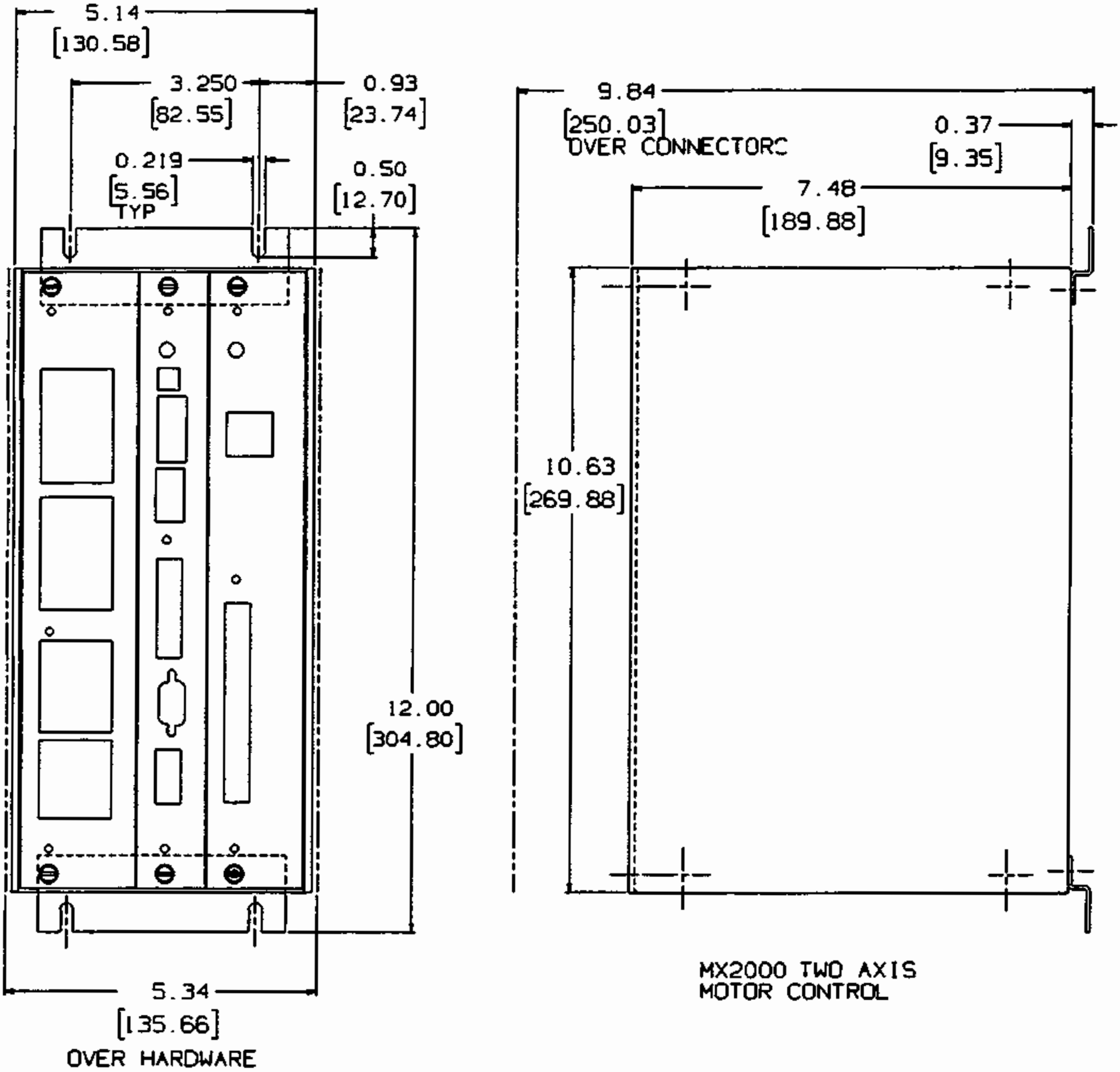
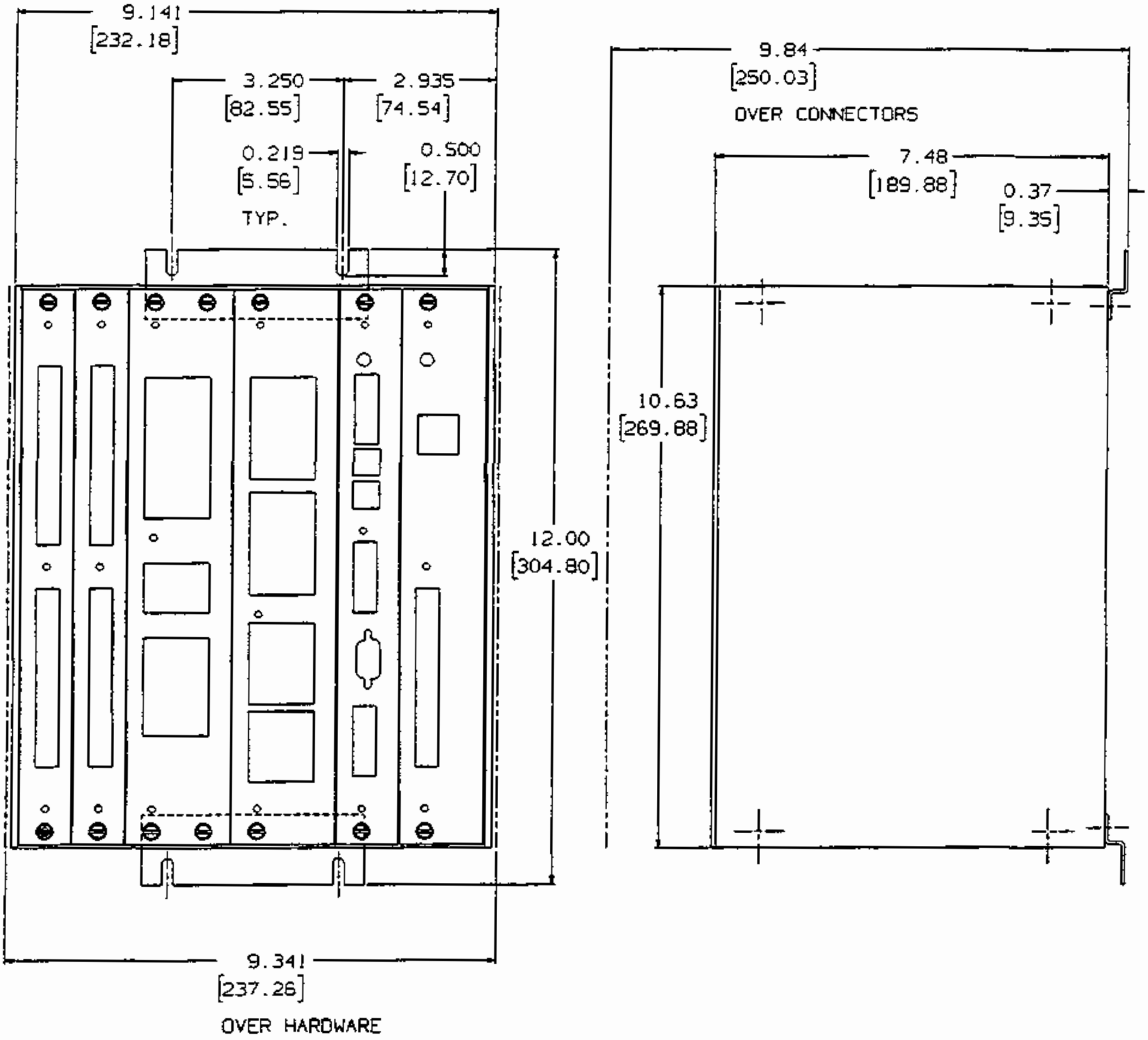
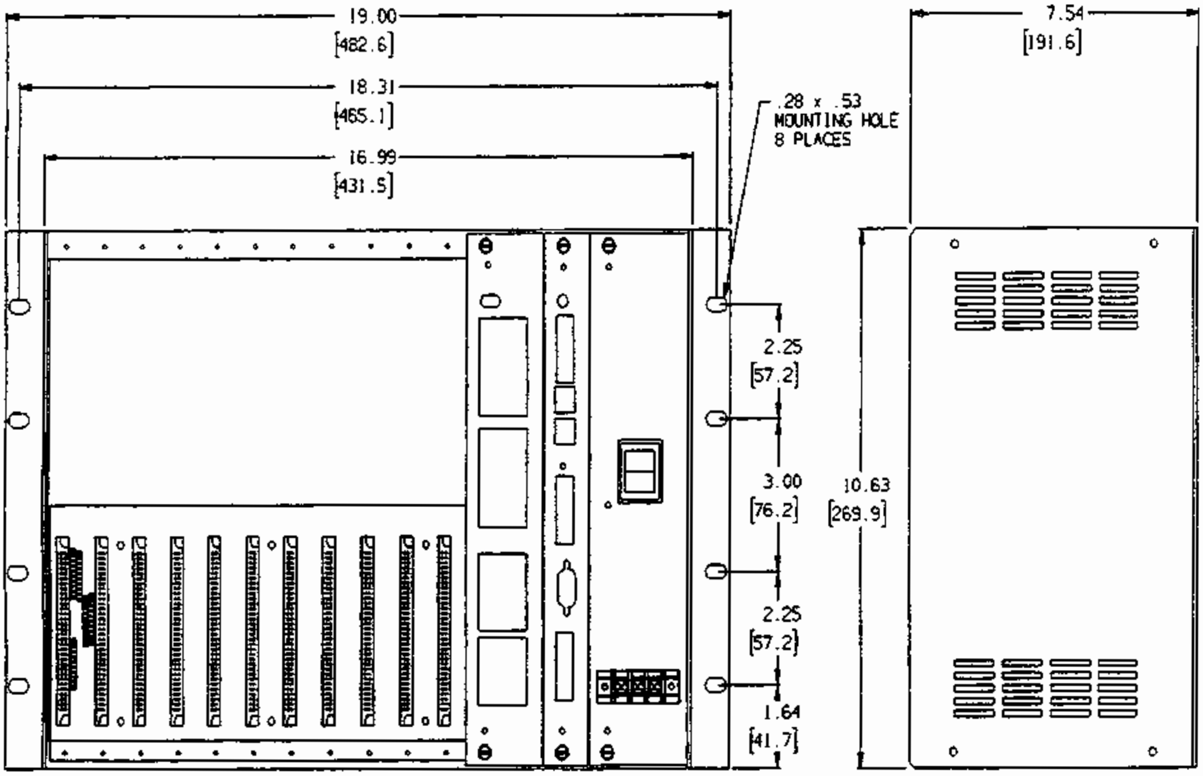


Figure 5.2, MX2000-6 Outline Drawing



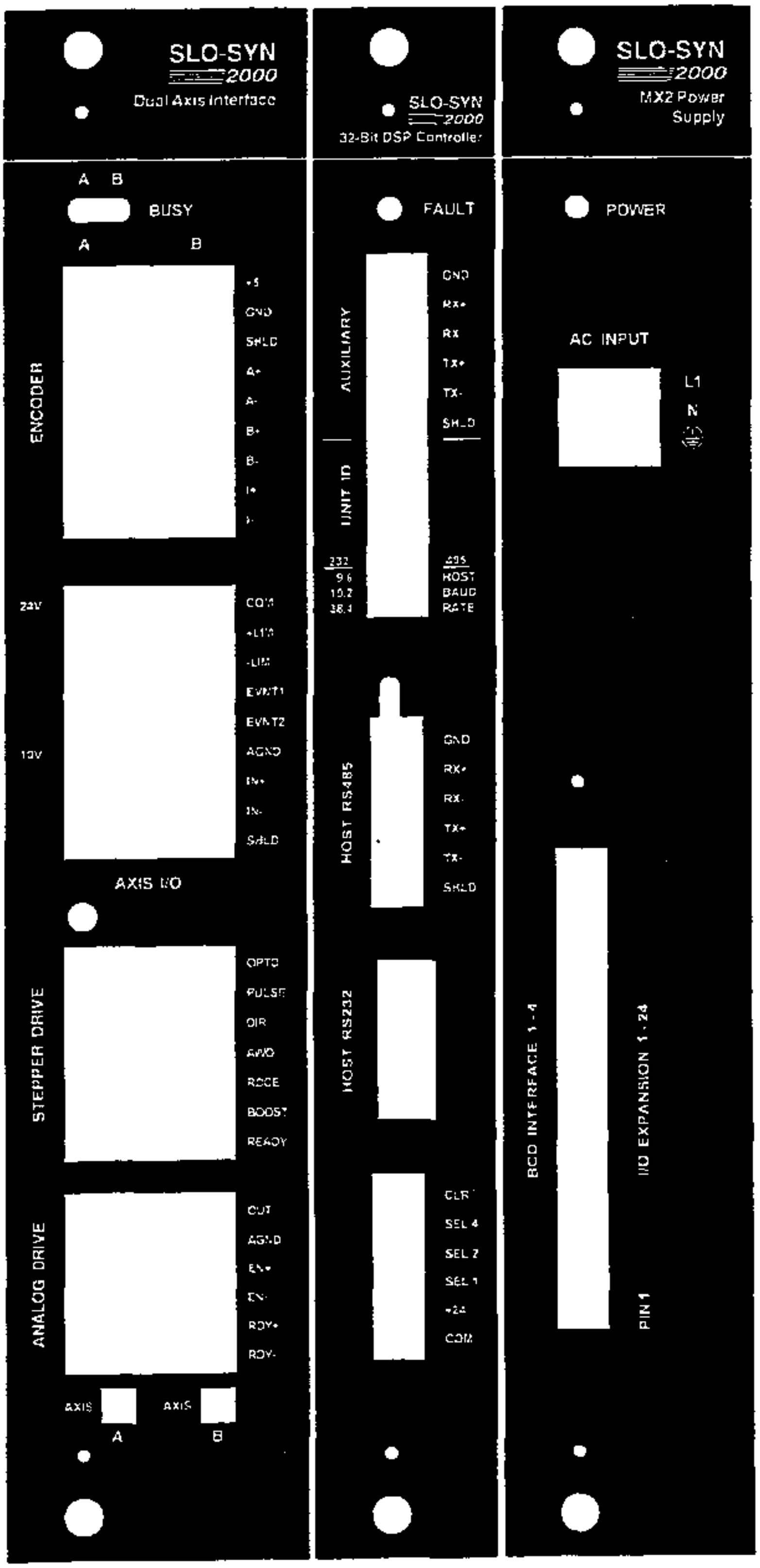
Note: Shown with optional I/O cards installed.

Figure 5.3, MX2000-8 Outline Drawing



Note: Shown with standard cards installed; optional Dual Axis and I/O cards may be installed to fill the card cage as required for your application.

Figure 5.4, MX2000-2 and MX2000-6 (Base) Standard Front Panel Details



5.3.1 - GENERAL WIRING GUIDELINES

SLO-SYN 2000 controls and drives use modern solid-state digital electronics to provide the features needed for advanced motion control applications. Some user equipment may produce electromagnetic interference (EMI, or electrical "noise") that can cause inappropriate operation of the digital logic used in the control, drive, or other computer-type equipment in the user's system.

In general, any equipment that causes arcs or sparks or that switches voltage or current at high frequencies can cause interference. In addition, ac utility lines are often "polluted" with electrical noise from sources outside a user's control (such as equipment in the factory next door). Some of the more common causes of electrical interference are:

- power from the utility ac line
- relays, contactors and solenoids
- light dimmers
- arc welders
- motors and motor starters
- induction heaters
- radio controls or transmitters
- switch-mode power supplies
- computer-based equipment
- high frequency lighting equipment
- dc servo and stepper motors and drives

The following wiring practices should be used to reduce noise interference.

Solid grounding of the system is essential. Be sure that there is a solid connection to the ac system earth ground. Bond the drive case to the system enclosure. Use a single-point grounding system for all related components of a system (a "hub and spokes" arrangement). Keep the ground connection short and direct.

Keep signal and power wiring well separated. If possible, use separate conduit or ducts for each. If the wires must cross, they should do so at right angles to minimize coupling.

Note: Power wiring includes ac wiring, motor wires, etc. Signal wiring is inputs and outputs (I/O), encoder wiring, serial communications (RS232 lines), etc.

Use shielded, twisted-pair cables for the drive to motor wiring. BE SURE TO GROUND THE SHIELD AT THE DRIVE END.

Suppress all relays to prevent noise generation. Typical suppressors are capacitors or MOV's. (See manufacturer's literature for complete information). Whenever possible, use solid-state relays instead of mechanical contact types to minimize noise generation.

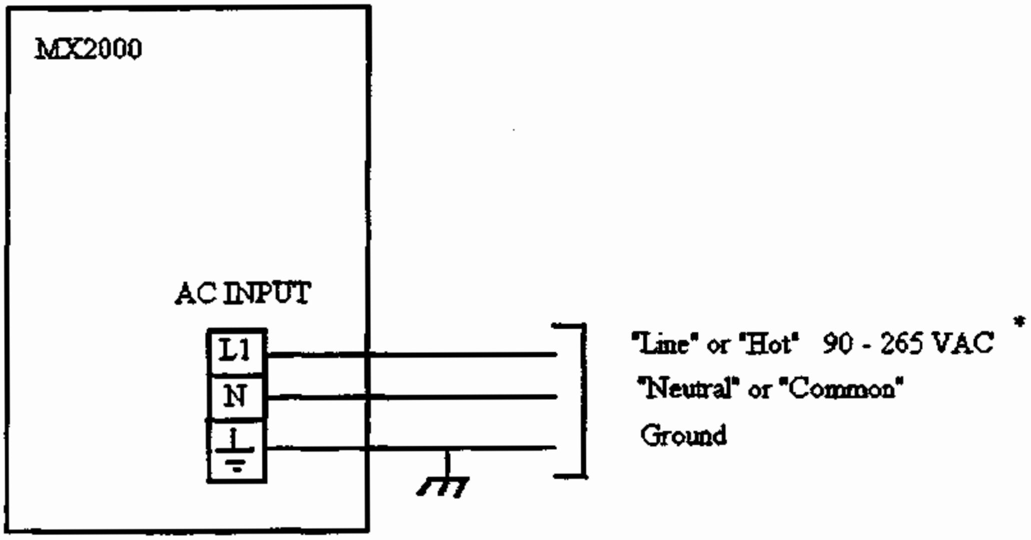
In some extreme cases of interference, it may be necessary to **add external filtering** to the ac line(s) feeding affected equipment, or to **use isolation transformers** to supply their ac power.

NOTE: Superior Electric makes a wide range of ac power line conditioners that can help solve electrical interference problems. Contact 1-800-SUP-ELEC for further assistance.

5.3.2.1 - AC INPUT

The AC input is connected to a 3-screw terminal strip. The terminals are labeled as follows:

Terminal	Lead Color North American Standard	Lead Color European Standard (CEE)
"L1" for line or "Hot"	Black	Brown
"N" for Common or Neutral	White	Blue
⏚ for Ground	Green	Green with Yellow Stripe

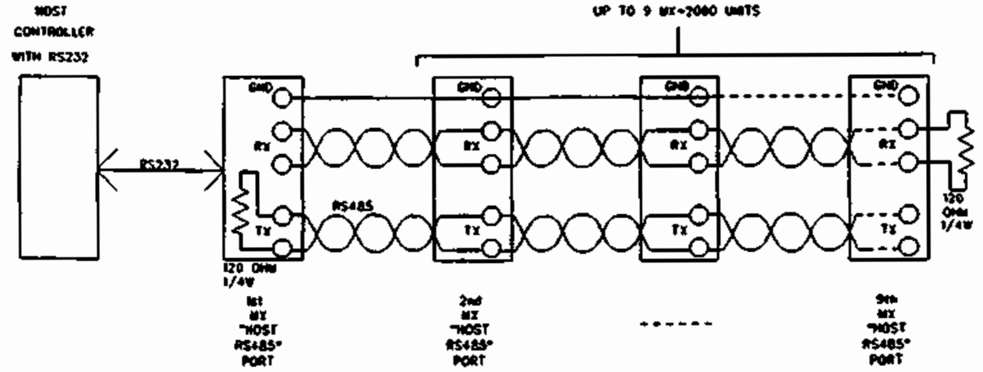


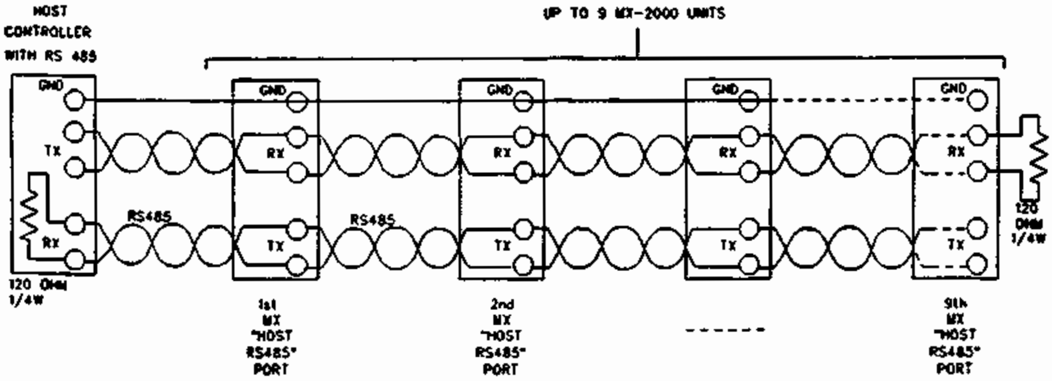
* For the MX2000-8 the input voltage to be applied has to be 90 - 132 VAC or 175 - 264 VAC. The MX2000-8 will not operate correctly if the input voltage is not within these ranges. No operator action is required. The MX2000-8 automatically senses the input voltage and configures itself to operate at either AC input voltage.

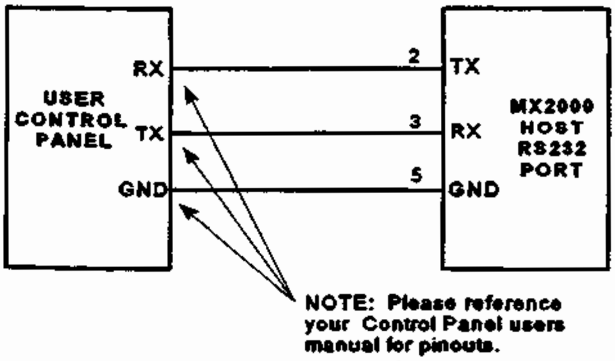
5.3.2.2 - HOST SERIAL PORT

The SLO-SYN controller can execute an internal program or it can respond to commands received over the host serial port, which is located on the DSP Card. Programs for the controller are developed on a PC and then down- loaded to the controller via the HOST RS232 or HOST RS485 port. The HOST RS232 port is primarily intended for this purpose, since RS232 ports are more prevalent on PCs than are RS485 ports. The selection of RS232 or RS485 is made with a dip switch on the DSP Card front panel, as is the baud rate. See also sections 4.2 and 5.1.1.

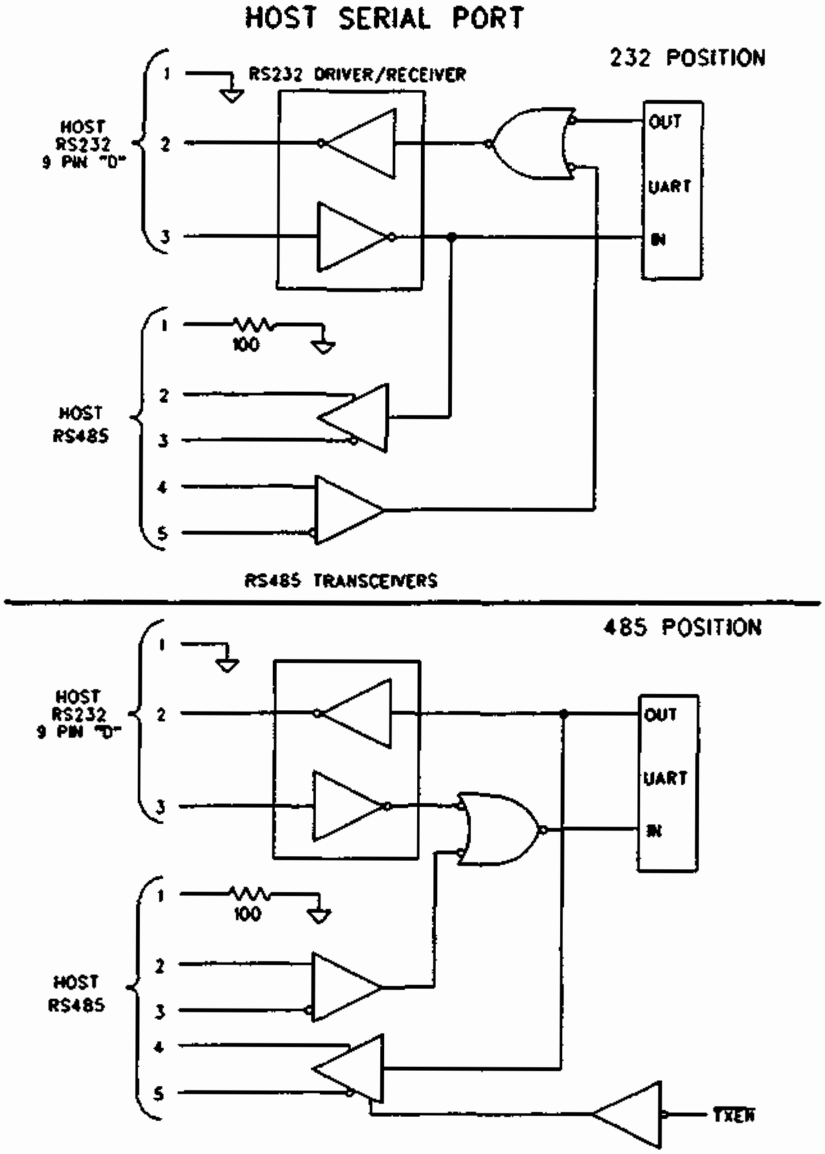
When a host communicates to more than one SLO-SYN controller, the units are wired together as shown below. The host can communicate to the first unit in the chain via RS232 or RS485. Subsequent units communicate via the HOST RS485 port. Use twisted pair wires to connect the RS485 ports together. A single serial port can communicate with up to 9 units. Each unit in the chain must have its UNIT ID switch set to a unique number; the rotary Unit ID switch on the front of the DSP Card is used for this purpose. See also section 5.1.1.







HOST SERIAL PORT EQUIVALENT CIRCUITS



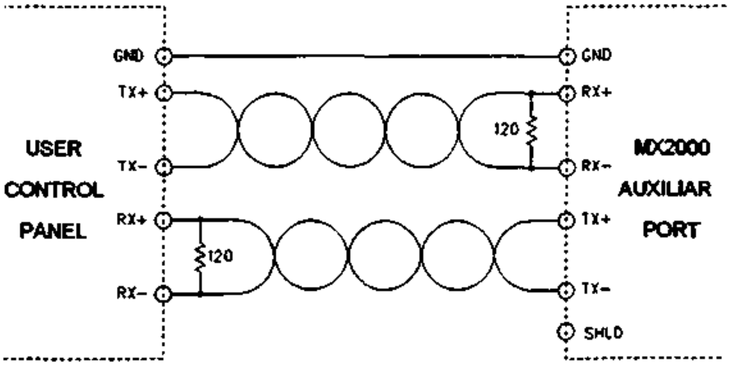
THE OUTPUT TRANSMITTER IS ONLY ENABLED WHEN THE UNIT'S ID MATCHES THE ACTIVE ID.

5.3.2.3 - AUXILIARY SERIAL PORT

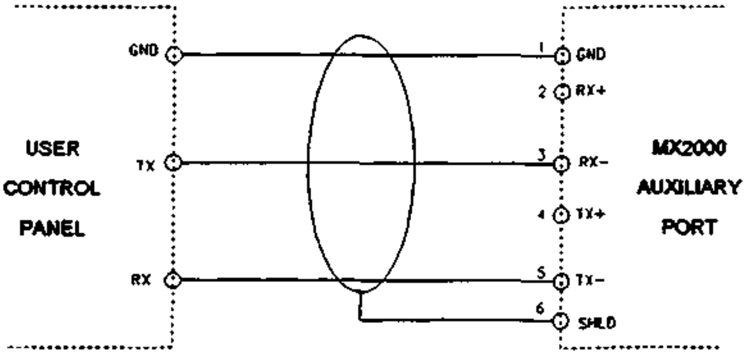
Also located on the DSP Card is an Auxiliary Serial Port, which can be used to communicate with a user's control panel. It can be configured for either RS232 or RS485 operation via a jumper on the DSP Card, behind the port connector (see also section 5.1.1). Messages prompting an operator can be output to a control panel, and keyboard responses can be read. The signal characteristics are in accordance with the RS232 or RS485 standard, depending on how the port is set up. Baud rate for this port is set via the Host port, using the SETCOM#2 command (see Section 6.4.1).

Below are wiring diagrams showing the connections from a user's control panel to the MX2000's Auxiliary serial port, for both RS485 and RS232 operation. Use twisted pair wires to make the connections and use termination resistors as shown when using RS485. The internal equivalent circuit of this port is also illustrated.

a) Wiring with the Auxiliary Serial Port set for RS485:

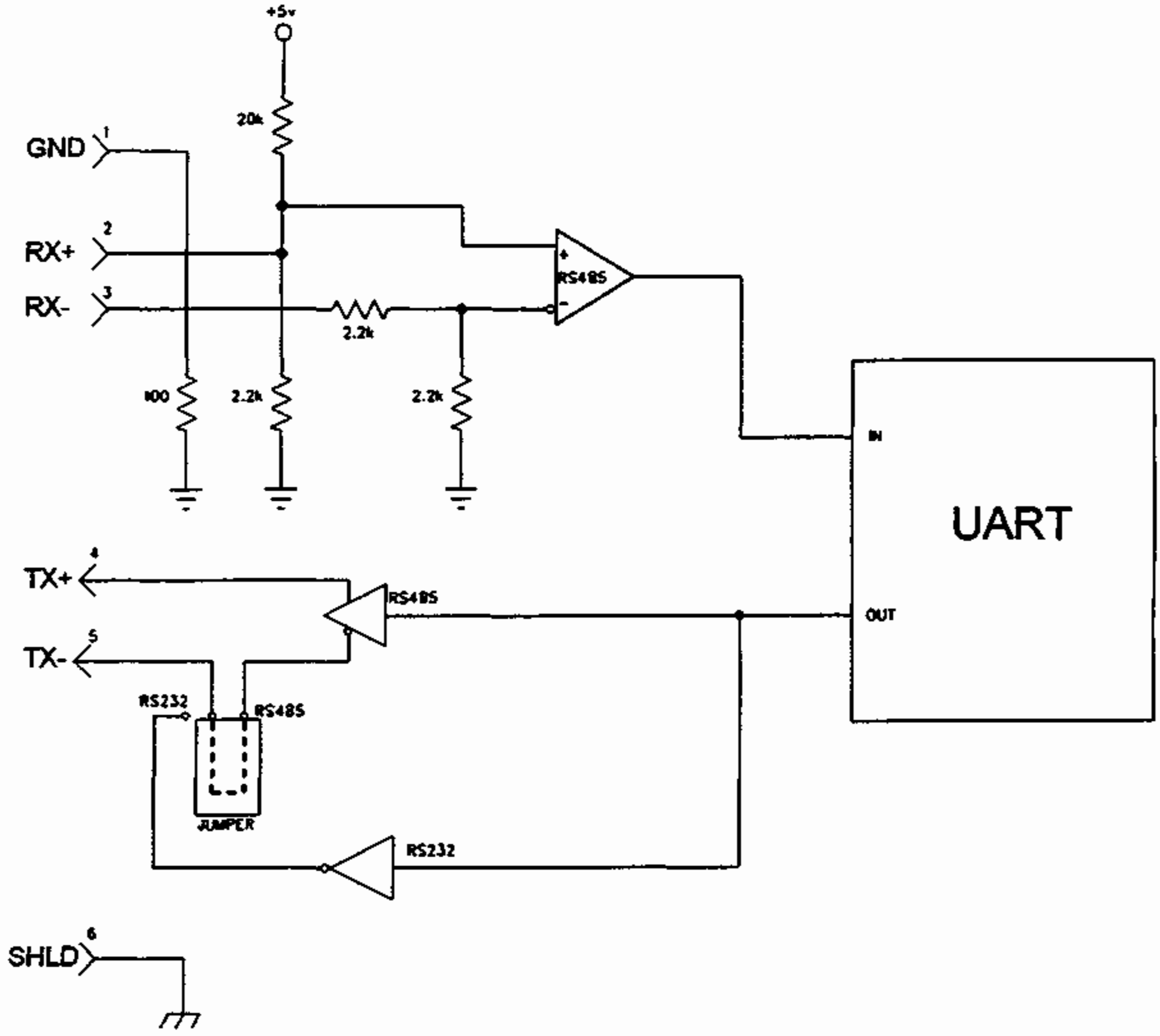


b) Wiring with the Auxiliary Serial Port set for RS232:



AUXILIARY SERIAL PORT, continued

c) Equivalent circuit for Auxiliary Serial Port:



5.3.2.4 - DSP CARD INPUTS

These inputs can be used in either sinking or sourcing mode; their operating mode is selected as a group with the Sink/Source Jumper on the DSP Card. The factory default setting is for sinking mode.

The CLR input is used to allow motion and program execution if the input is active, or stop motion and terminate program execution if the input is inactive. This input must be active to allow auto execution of a selected project. Auto execution occurs following power turn-on, or when issuing the RESET command through a serial port.

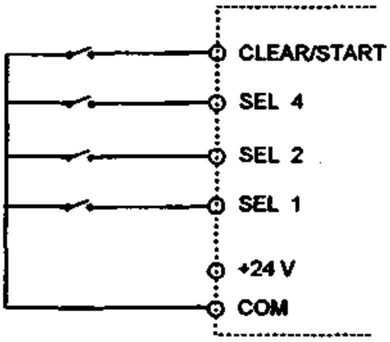
The SEL 4, SEL 2 and SEL 1 inputs are used to select the project to be auto-executed at power turn-on or after a RESET command. Any one of 7 projects can be selected. The inputs are binary weighted with SEL 1 having a weight of 1, SEL 2 having a weight of 2, and SEL 4 having a weight of 4. Projects are loaded sequentially into user memory after the ERASE DIR command is issued. On power-up, the selected project is loaded into the DSP memory if the CIR input on the DSP card is active and the selected project exists. If a project has been loaded into the DSP memory it will auto start. The following table shows how to connect the Select inputs to auto start particular projects in the DSP Card user memory. To run a particular project after a power-up or commanded Reset use the Host "LOAD" and "RUN" commands. The LOAD command requires the project name.

To start:	SEL 4	SEL 2	SEL 1
1st Project (Project 0)	0	0	0
2nd Project (Project 1)	0	0	1
3rd Project (Project 2)	0	1	0
4th Project (Project 3)	0	1	1
5th Project (Project 4)	1	0	0
6th Project (Project 5)	1	0	1
7th Project (Project 6)	1	1	0
1st Project (Project 0)	1	1	1

0 = select switch "off" 1 = select switch "on"

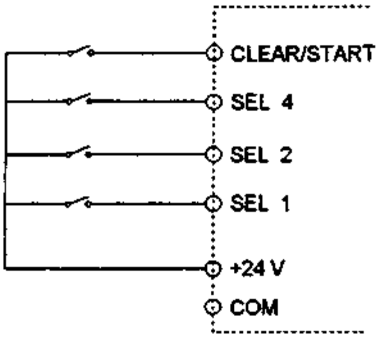
Below are the user connections for sinking or sourcing modes and the internal equivalent circuit for these inputs.

b) User input connections for Sinking Mode. Jumper must be in SINK position, factory default.

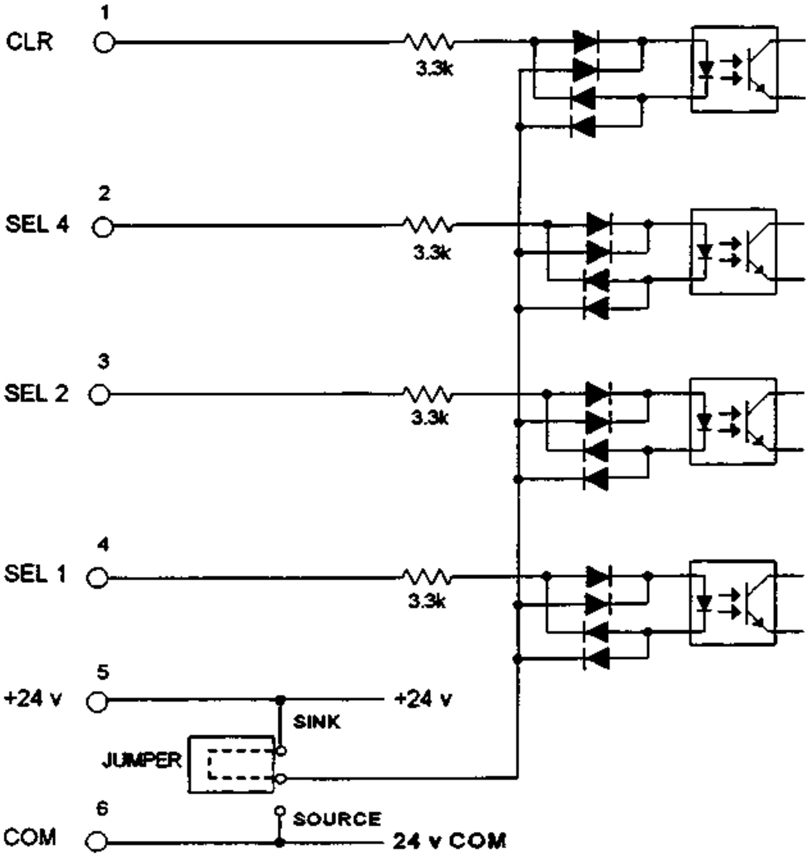


DSP CARD INPUTS, continued

b) User input connections for Sourcing Mode. Jumper must be in SOURCE position.



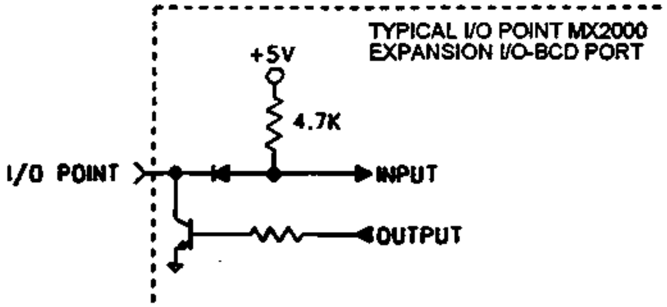
c) Input equivalent circuit.



5.3.2.5 - EXPANSION I/O - BCD PORT

The EXPANSION I/O - BCD port (located on the Power Supply Card) has been designed to interface to BCD switches (to read in move parameters such as velocity or distance, etc.) and/or to industry-standard "OPTO 22" input/output module boards (see figure on next page for I/O module manufacturers and part numbers). The port consists of 24 bi-directional I/O points contained on a standard 50 pin header. Odd pins 1 - 47 are signal pins, even pins 2 - 50 are ground and pin 49 is not used. The equivalent circuit of an I/O point is shown below. The internal signal "input" reads the state of the I/O point and the internal signal "output" controls the output transistor. When an I/O point is used as an input, "output" is set low to turn the transistor off.

The figures on the following pages show how to connect up expansion I/O and BCD switches to this port.



This port can accommodate up to 24 "OPTO 22" I/O modules and up to 4 BCD switch banks (each bank 7 digits with sign). The interface capacity is listed below.

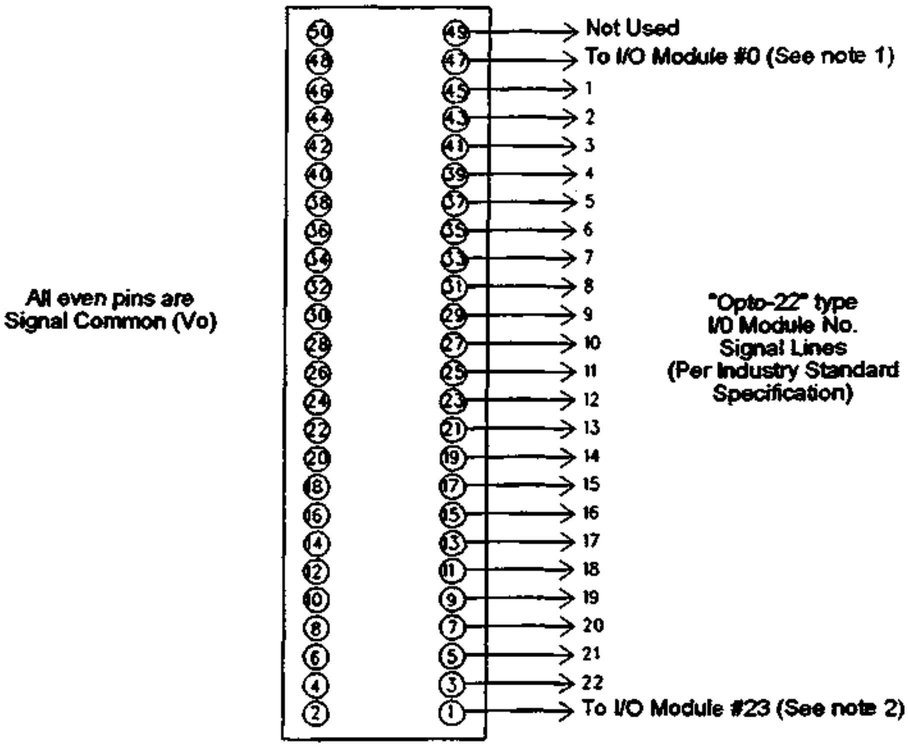
BCD switch banks	"OPTO 22" I/O modules
0	24
1	12
2	8
3	4
4	0

As shown above, BCD switch banks and "OPTO 22" I/O modules can be used simultaneously, however care must be exercised to avoid conflicts. Below is a list of pins used by each BCD switch bank. Any unused pins can be used to interface to "OPTO 22" I/O modules.

BCD switch bank	port pins used
1	1,3,5,7,9,11,13,15, 17,19,21,23
2	1,3,5,7,9,11,13,15, 25,27,29,31
3	1,3,5,7,9,11,13,15, 33,35,37,39
4	1,3,5,7,9,11,13,15, 41,43,45,47

EXPANSION I/O - BCD PORT, continued

Expansion I/O-BCD Connector Pinout



<u>Notes</u> 1. Using this point as an input, command syntax is EXIN(100)
Using this point as an output, command syntax is EXOUT(100)

2. Using this point as an input, command syntax is EXIN(123)
Using this point as an output, command syntax is EXOUT(123)

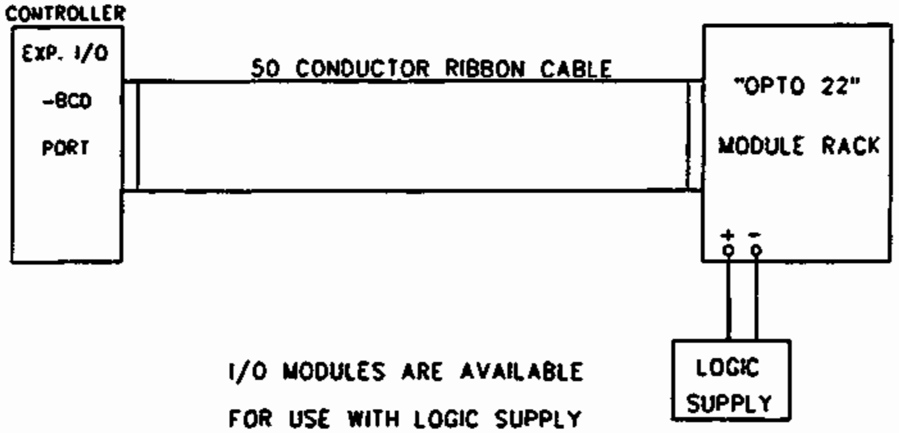
3. Digital I/O modules in a wide range of voltages and currents, plus their required mounting boards, are available from several sources. Below is a list of some of the sources available with part numbers:

MANUFACTURER	PART NUMBER FOR 24-MODULE MOUNTING BOARD*
Crydom	PB-24
Gordos	PB-24
Grayhill	7ORCK24, 7OMRCQ24 series
Potter & Brumfield	ZIO24, ZIOM24 series
OPTO-22	PB-24, PB-24Q, PB-24HQ

*Smaller boards (fewer I/O modules) are also available.

EXPANSION I/O – BCD PORT

CONNECTION TO "OPTO 22" MODULE RACK

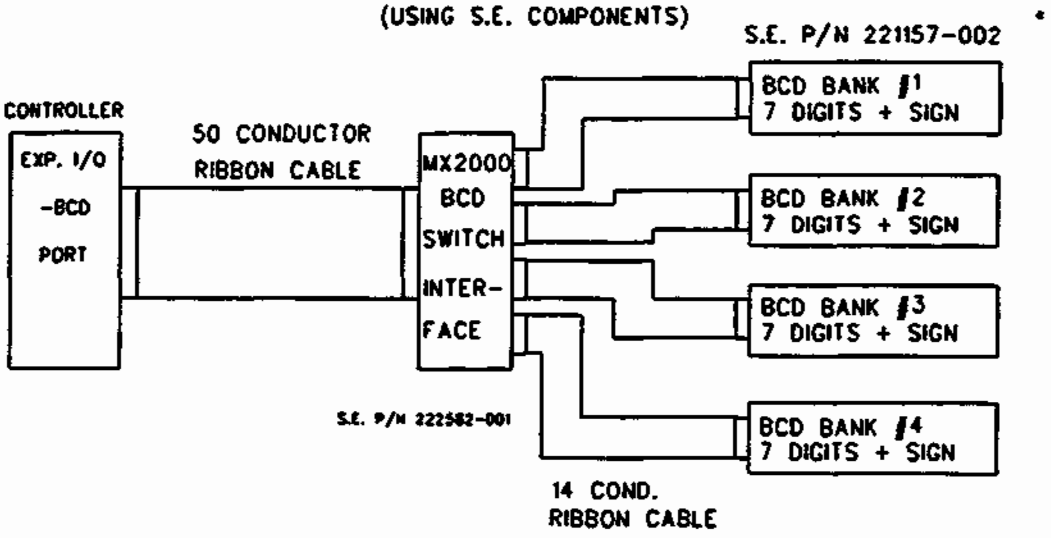


I/O MODULES ARE AVAILABLE
FOR USE WITH LOGIC SUPPLY
VOLTAGES 5, 15, AND 24V.
THE LOGIC SUPPLY (-) TERMINAL IS CONNECTED
TO EVEN PINS 2-50, WHICH ARE GROUND

EXPANSION I/O – BCD PORT

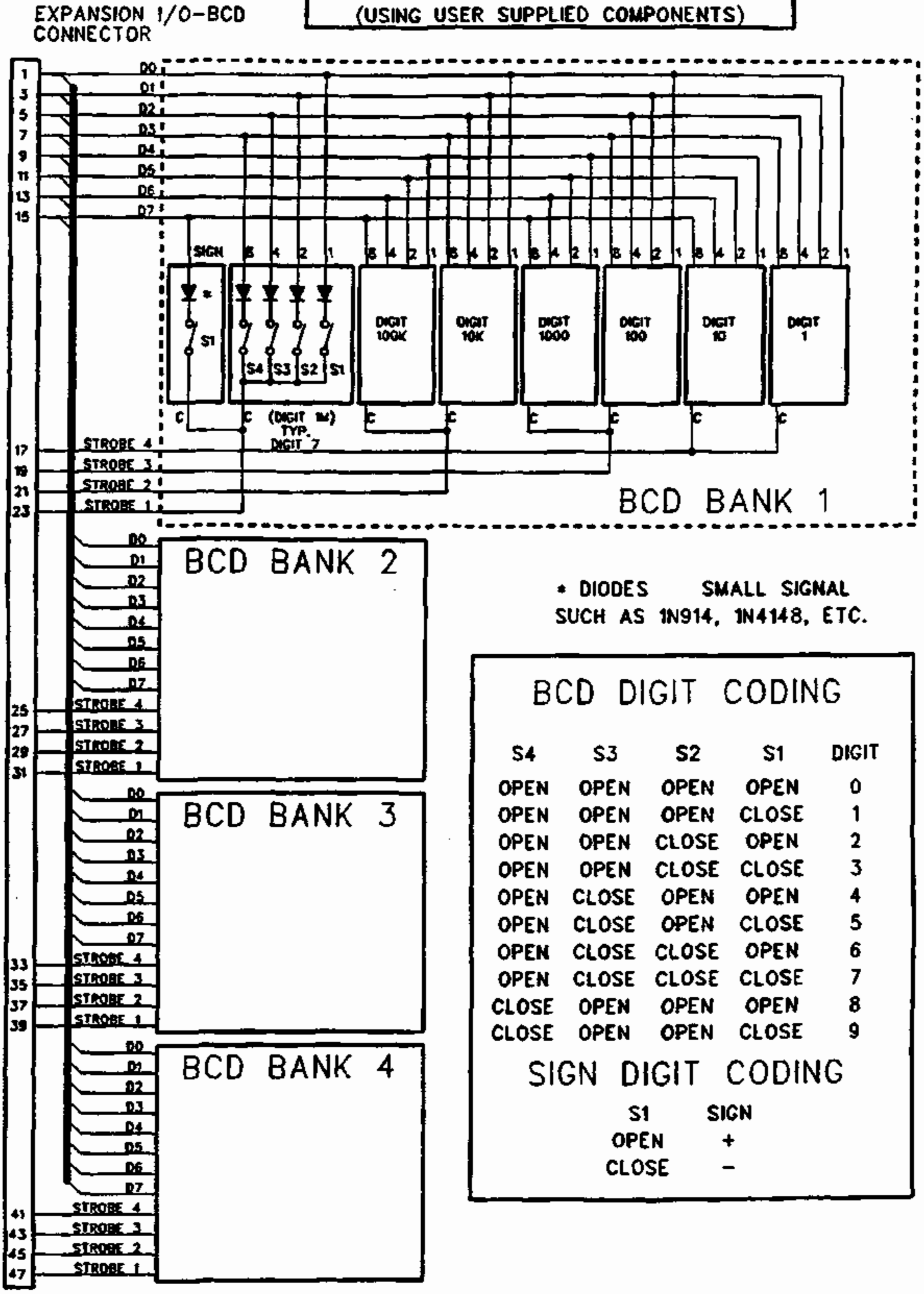
CONNECTION TO BCD SWITCH BANKS

(USING S.E. COMPONENTS)



* (AN 18 INCH CABLE IS INCLUDED WITH 221157-002)

EXPANSION I/O-BCD PORT
CONNECTION TO BCD SWITCH BANKS
(USING USER SUPPLIED COMPONENTS)



* DIODES SMALL SIGNAL SUCH AS 1N914, 1N4148, ETC.

BCD DIGIT CODING

S4	S3	S2	S1	DIGIT
OPEN	OPEN	OPEN	OPEN	0
OPEN	OPEN	OPEN	CLOSE	1
OPEN	OPEN	CLOSE	OPEN	2
OPEN	OPEN	CLOSE	CLOSE	3
OPEN	CLOSE	OPEN	OPEN	4
OPEN	CLOSE	OPEN	CLOSE	5
OPEN	CLOSE	CLOSE	OPEN	6
OPEN	CLOSE	CLOSE	CLOSE	7
CLOSE	OPEN	OPEN	OPEN	8
CLOSE	OPEN	OPEN	CLOSE	9

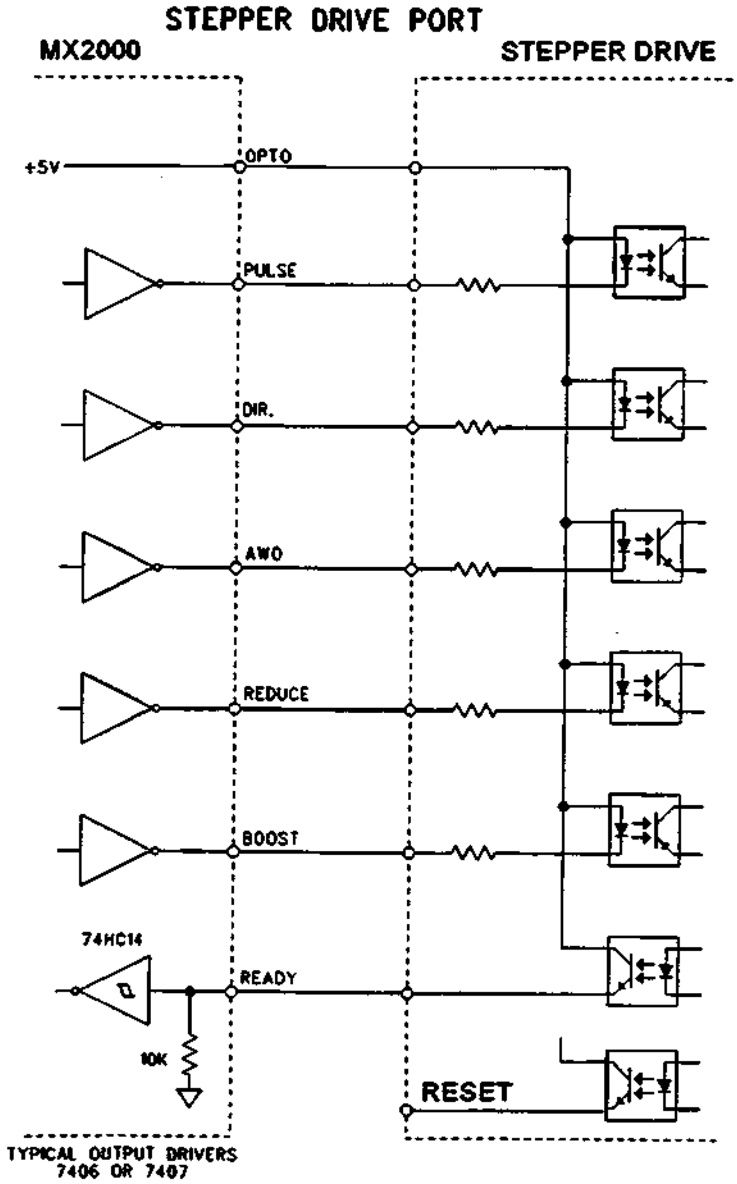
SIGN DIGIT CODING

S1	SIGN
OPEN	+
CLOSE	-

5.3.2.6 - STEPPER DRIVE PORT

There are two stepper drive interface ports located on the Dual Axis Card. Each stepper drive port has seven signals, 6 outputs and 1 input. The 3 primary outputs, OPTO, PULSE and DIR, are all that is needed to run a stepper drive. The 3 remaining outputs (AWO, REDUCE and BOOST) are used to control the drive's motor current. AWO shuts the current off, REDUCE cuts the motor current in half (to reduce heating), and BOOST increases the motor current (to deliver more torque). The drive signals that it is ready to accept pulses via the controller's READY input.

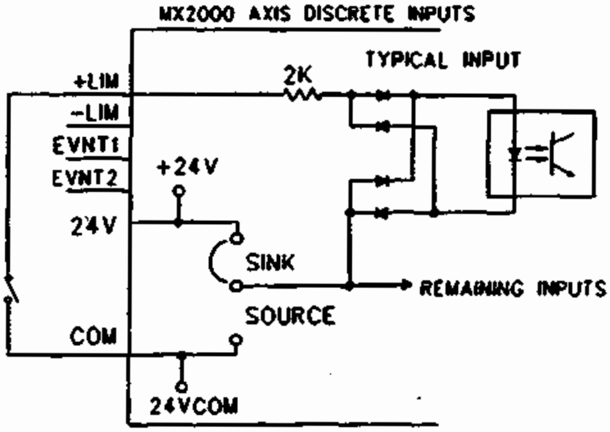
The operation of signals AWO, REDUCE, BOOST, and READY pertains to Superior Electric drives. If the current control functions of AWO, REDUCE, and BOOST are not required, leave these pins disconnected. If the drive does not have a "ready" output, simply connect a jumper from OPTO to READY on the controller's drive port. Following is an illustration of the connections to a stepper drive as well as the equivalent circuit of the STEPPER DRIVE PORT. The equivalent circuit for a Superior Electric drive's signal interface is also shown. Note: The MX2000 has no "Reset" output on the stepper drive ports; leave this pin unterminated at the drive.



5.3.2.7 - AXIS I/O and ANALOG INPUTS

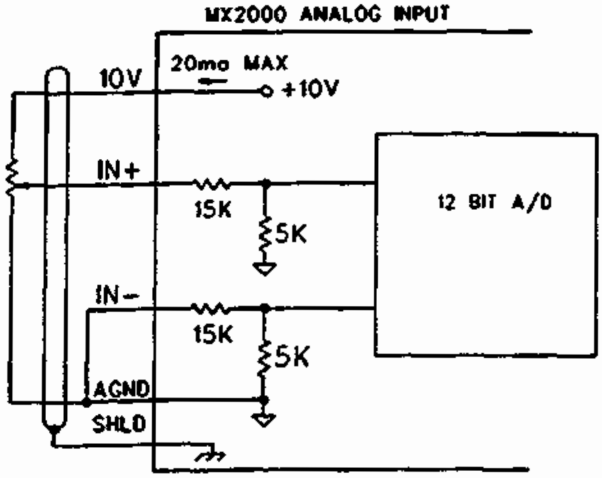
Axis I/O is located on the Dual Axis Card. Each axis has 5 inputs and 1 output associated with it. There is one analog input and one analog output, plus 4 discrete digital inputs which are typically wired to switches.

The discrete inputs consist of +LIM, -LIM, EVNT1 and EVNT2. Inputs +LIM and -LIM are travel limit inputs. EVNT1 and EVNT2 inputs are used for homing and mark registration. All of the discrete inputs (8) on the Dual Axis Board, can be collectively configured for sink or source operation (see Section 5.1.1 for configuration). The 24V and COM outputs are provided for wiring switches to the discrete inputs. Below is an example of wiring a switch to a discrete input, in this example the +LIM input. The equivalent circuit of a typical input is also shown. Note that if the limit and event inputs are not used for their primary intended functions (e.g. if limit switches, homing, and mark registration are not used), then they may be used as general-purpose program-testable inputs.



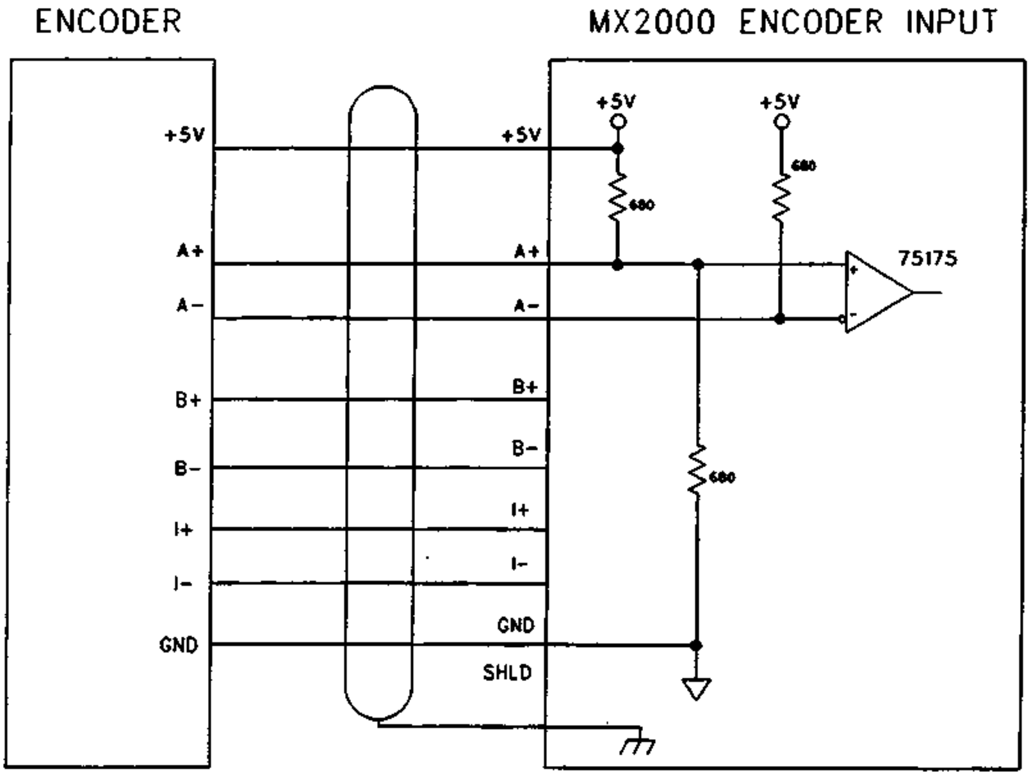
The analog I/O are located on the ANALOG connector. Inputs IN+ and IN- are analog inputs. These inputs can be configured as one differential input or two single ended inputs in the Project/ Setup and Configuration folder. A differential input will measure the difference (IN+ minus IN-) between the inputs. When configured as single ended each input (IN+, IN-) will be measured with respect to AGND. The 10V, AGND and SHLD pins are intended to be used with the analog input. The 10V output can be used to power a potentiometer. The equivalent circuit of the analog input as well as a typical connection to a potentiometer is shown below. The A/D services two such inputs although only one is shown. The circuit shown below will measure 0-10Vdc depending on the position of the potentiometer (The inputs can be configured differential or single ended). If IN- was connected to +5Vdc, instead of AGND and configured as differential, +5 to -5Vdc would be measured depending on the potentiometer position. The analog output is on pin OUT; pin AGND is its ground reference.

Note: See Section 7.5 for details on Joystick connections to the analog inputs.



5.3.2.8 - ENCODER INPUT PORT

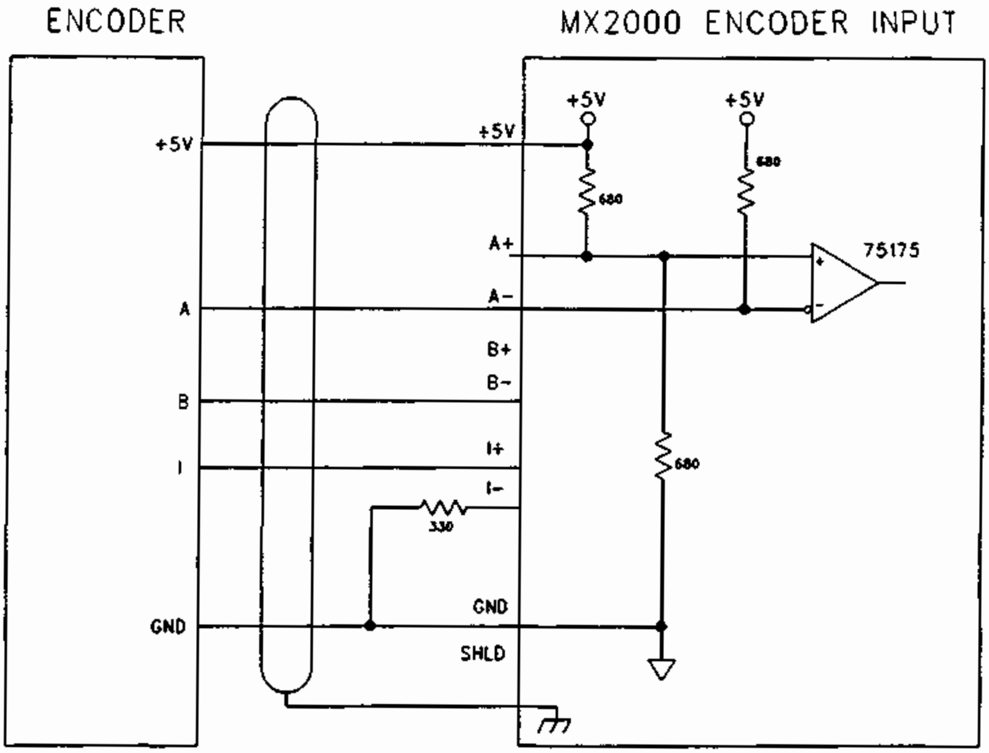
Two encoder input connectors are located on the Dual Axis Card. Each encoder interface can accommodate a single-ended encoder, differential encoder or a pulse and direction input. The pulse and direction configuration is useful in some digital following applications. 5V power is available for powering the encoder, up to 100 ma per encoder is allowed. The following diagrams illustrates the wiring for the three interfaces. The electrical circuit for input A is shown; inputs B and I are identical. We highly recommend the use of twisted-pair (approximately 6 per foot) shielded cable for all encoder wiring to minimize interference problems.



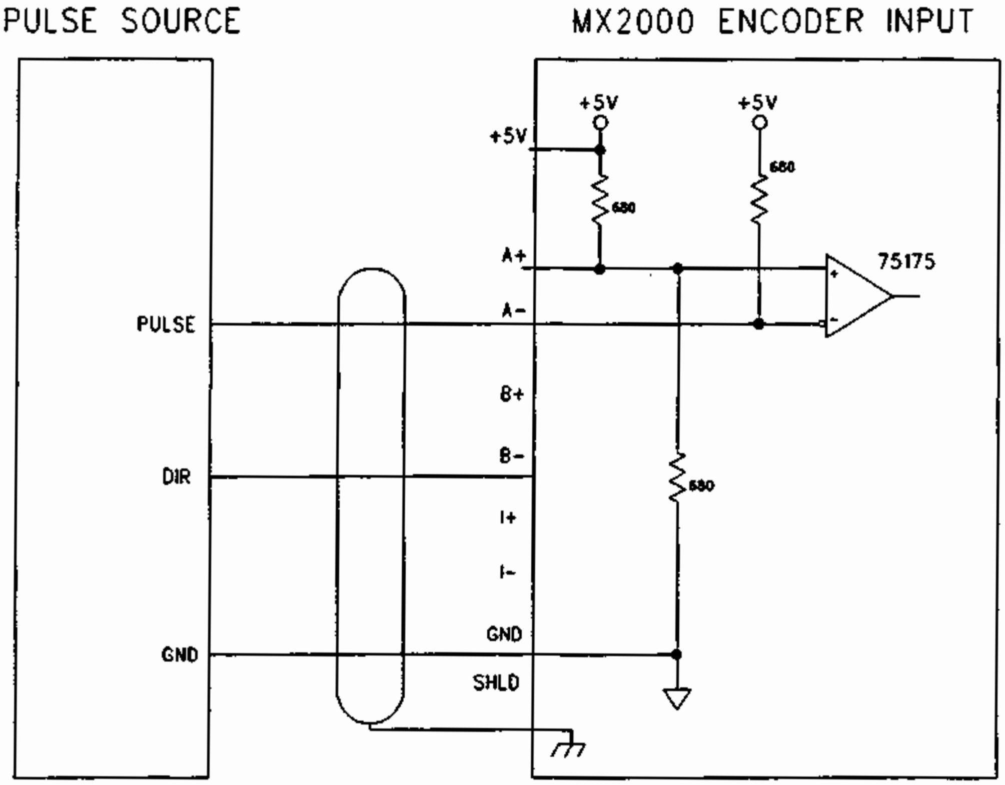
Differential Encoder

Note: The following signals should be twisted together i.e. use the same twisted, shielded pair:

A+ with A-, B+ with B-, I+ with I-, and +5 Vdc with GND.



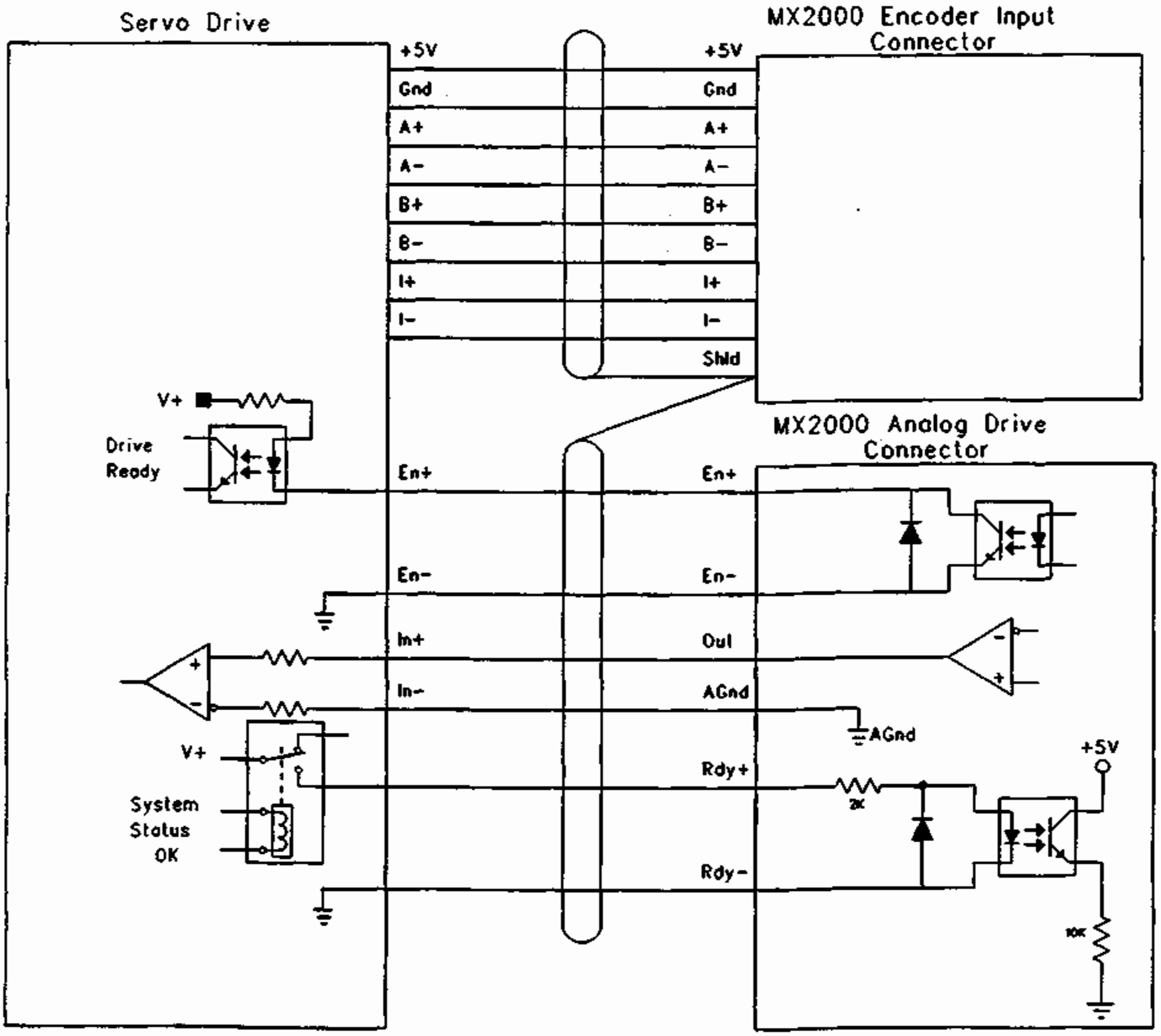
SINGLE ENDED ENCODER



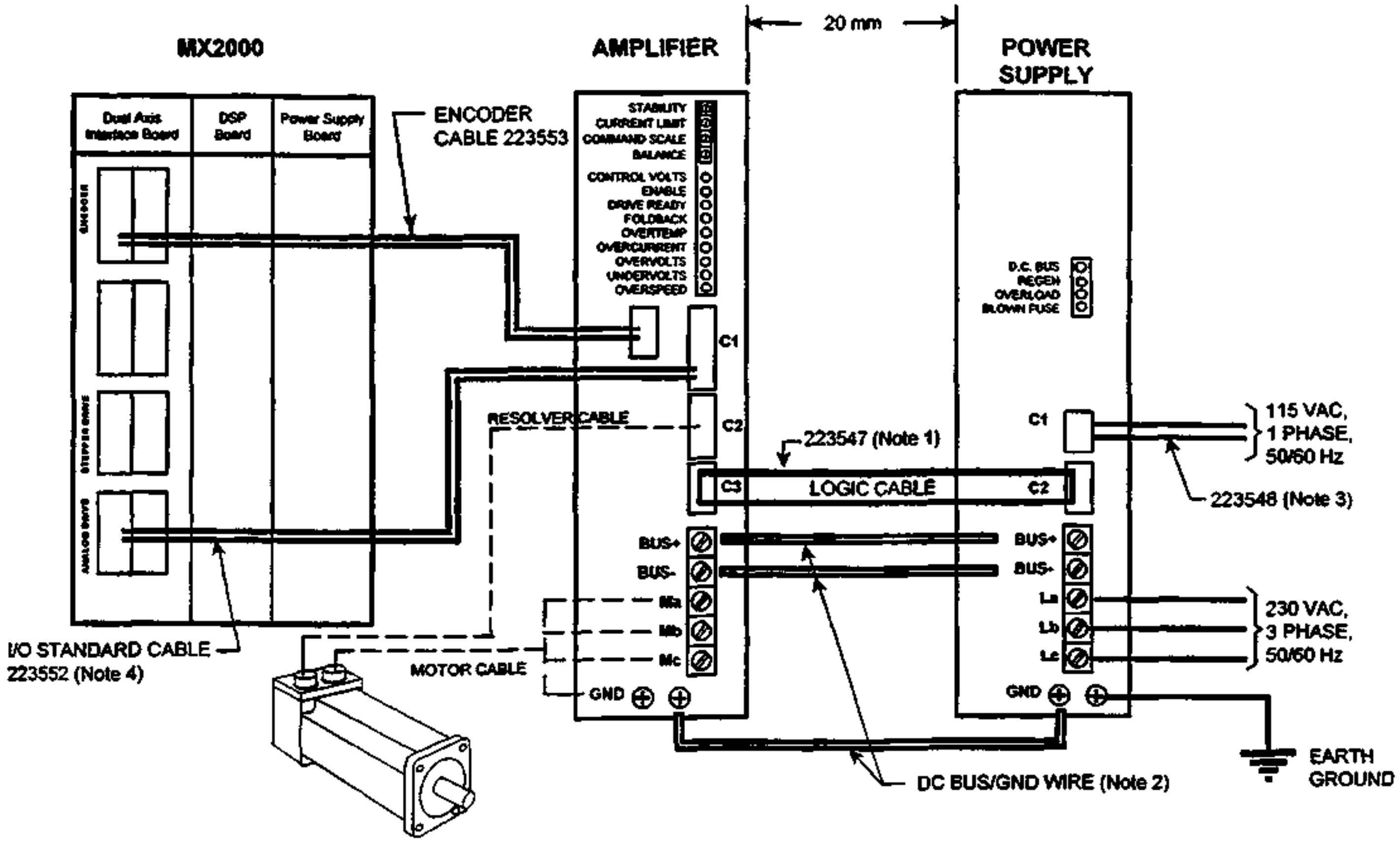
PULSE and DIRECTION

5.3.2.9 - Analog Drive Port - for Servo I/O

The Servo I/O is located on the Dual Axis Card. Each axis has 6 connections for the ANALOG DRIVE and 9 connections for the Servo ENCODER. The Encoder and Servo Drive connections of a typical servo drive are depicted below. The EN+ and EN- signals are used to enable or disable the servo drive and is controlled by the WNDGS command. The command WNDGS(axis)=1 enables the servo drive. The RDY+ and RDY- signals are used by the MX2000 for detecting fault conditions or can be used by the servo drive to indicate a ready condition. This signal can be monitored using the DRVREADY command.



MX2000 & SERVO AMPLIFIER CONNECTION DIAGRAM



SECTION 6 - PROGRAMMING & SOFTWARE REFERENCE

6.1 - GENERAL DESCRIPTION OF PROGRAMMING

This section provides an overview to the process of programming a Controller. Once the "logic" behind the various commands are understood, and the mix of numbers and letters are explained, programming your Controller will be seen as a straightforward process.

Programming of any sort requires planning and forethought. Programming your Controller is no exception. This section will provide aids to facilitate your planning process. Be patient! Allow time for mistakes, adjustments ("debugging"), and experimentation.

WHAT IS PROGRAMMING?

At its most basic level, a computer program is a means of using electronic digital signals (simple ON and OFF) to produce certain results from a machine. A line of code, or "command string," is built up from the presence (On) or absence (Off) of electrical signals. On or Off signals, called "Bits," are bunched together to form "Bytes", or groupings that are coded into what we recognize as alphabetical characters or numbers. (This character coding is accomplished via the ASCII code - see Section 9-Glossary for further details).

A program is a list of discrete lines or command strings that, taken together in sequence, provide the information needed to get a machine to perform your predetermined sequence of instructions. These instructions can, in the case of a Programmable Motion Controller, cause the motor to move at certain speeds and for given distances, read various inputs or set outputs, or send and receive messages from an operator interface panel, all used to accomplish different machine-related tasks.

WHAT'S IN A PROGRAM?

A program consists of many individual lines organized in a prescribed sequence. The MX2000 system uses an English language, BASIC-type computer programming language ("SEBASIC"). This makes it easy and intuitive to write and read machine control programs. The language we have designed supports many higher-level-language features, such as statement labels, subroutines, for-next and do-while loops for program flow control. This makes it easy to write concise, well-organized, easily-debugged programs. Also, there are built-in mathematical, Boolean, array, and trigonometry functions to perform complex calculations. The rich string-handling functions allow easy data input and message-writing when using external operator interface panels. Finally, the motion, I/O, and timing commands are easy to understand, remember, and apply.

In addition to program lines, the controller needs and stores (separate from the commands) a series of set-up parameters in a "header file". This file is automatically created by the MX2000 compiler program, or by the Motion Workbench program for Windows.

HOW IS THE CONTROLLER PROGRAMMED?

There are two primary ways to set up and program your MX2000 controller. Both involve the use of a personal computer (PC). One is a programming environment called "MX2000", and is supplied on diskette with your unit. Section 6.3 of this manual gives detailed instructions on installing and using this tool to develop your application. A second programming aid, is available to allow graphical entry of the controller program. It is called the Superior Electric Motion Workbench programming system. A third way is to create your program using any standard text editor or word processor. Write the SEBASIC commands, save the file as an ASCII format then use the MX2000 PC program to compile your code and download it.

The types of commands your Controller can accept are pre-set. Thus each command is assigned a "name". These commands are explained in detail in the Software Reference Section of this Manual.

Commands are performed via the statement lines in your program. The program is a sequence of commands that control the motor and motion-related events you want to happen in a particular period of time. Thus, the sequence of commands is critical to the proper operation of your system.

WHAT ARE "HOST COMMANDS"?

There are also "Host Commands" available for certain programming needs. These commands go straight from your input device (computer or terminal, for example), to the Controller, and override the normal sequence of operation directed by your program. These are useful for manually controlling a machine that normally operates under program control.

MEMORY TYPES AND USAGE

A program is stored in Memory. There are two kinds of memory. RAM (Random Access Memory) is called "Volatile Memory" because when power is removed from the Controller, all the electrical signals in that memory are lost, and accordingly, the information stored in that memory is lost. The Controller, for example, stores some transient information in RAM.

The second kind of memory is "Non-Volatile Memory", such as Flash memory, EEPROM (Electrically Programmable Read-Only Memory), or a BBRAM (Battery Backed RAM). The electrical codes stored in this type of memory are not lost when external power is removed from the Controller. The Controller uses a battery backed RAM for storing NVR variables (1-2048). The Flash memory is used by the controller for storing the operating system as well as user programs. This memory is located on the DSP Controller card.

A program in your Controller can have hundreds , or even thousands of program lines. Because of the wide variety of program commands, and the variable line lengths allowed, it is impossible to state how many lines of code can be stored in the controller. However, the user memory available is 2044 sectors of 128 bytes per sector, for a total of 261,632 bytes of program space. The FREE command may be used to determine how much memory is available; see Section 6.4.2 for details on using this

REFERENCES

Newcomers to programming are encouraged to obtain a copy of an elementary text on computer programming. Since your MX2000 controller uses a modified form of the familiar "BASIC" computer language, you may refer to a book on using BASIC. There are a great number of such books available in the technical or computer section of your local library or bookstore. We have found that books by SAM's, Microsoft Press particularly "Running MS DOS QBASIC," by Michael Haverson & David Rygmyr, and those by the Waite Group are among the most helpful.

SECTION 6.2 - MULTI-TASKING OPERATION

A single computer can only do one thing at a time. However, a complex motion control system needs to have many tasks done, all at once. An effective way to do this is with a very fast microprocessor (or DSP), running a preemptive multi-tasking operating system. This causes a single computer to appear to be doing several things simultaneously. The computer works on one program for a while, then switches to another program for a while, and after all programs have been serviced, goes back to the first, and repeats the cycle. With a fast computer, the time slice for each program can be small and the outward appearance is that a separate computer is running each program.

The MX2000 uses this approach to give the user up to 7 "virtual" motion controllers in a single package. An additional advantage of multi-tasking is that information can be easily shared among the 7 virtual controllers. The MX2000 runs 1 system task and up to 7 user tasks. Task 0 is a system task, which always runs. It processes commands received over the Host serial port. Up to 7 user, SEBASIC, BASIC programs (Task 1 - Task 7) may be running in addition to Task 0.

Every 256 microseconds, task execution is interrupted in order to perform the time-critical functions associated with motion control. Execution of the next task is resumed upon completion of the interrupt routine. The execution sequences for a 1-user-task system and for a 7-user-task system are shown below.

If an application uses all 7 tasks, then each task will be serviced once every 2.048 ms. If fewer tasks are used, then the service time decreases. A one-task system would be serviced every 512 us. The service time can be calculated by the following formula:

$T_{service} = (n + 1) \times 256$ microseconds
where n is the number of user tasks.

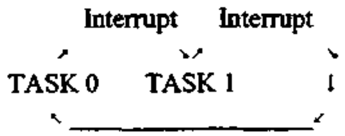
Large, complicated applications typically consist of several independent operations occurring simultaneously. Multi-tasking allows the user to program the application as a collection of several smaller and hence simpler applications.

A typical example of the use of tasks is to break up the system functions into logical groups. For example, control of a large machine might assign functions to tasks as follows:

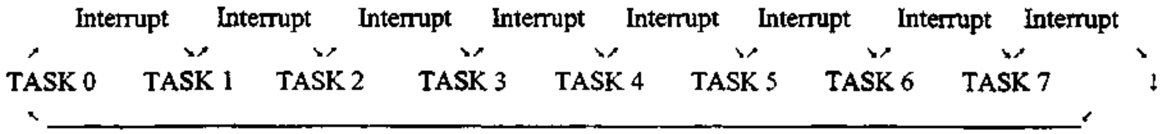
Task 1 - Motion on axes 1 and 2
Task 2 - Handling all inputs and outputs
Task 3 - Communicating with operator interface panel

Using tasks and multi-tasking allows programs to be more modular, hence they are easier to write, debug, and maintain.

Interrupt routine (executed every 256 us.) with only one user task running (Task 1) plus the system task (Task 0):



Interrupt routine (executed every 256 us.) with all 7 user tasks (Tasks 1 - 7) running, plus the system task (Task 0):



SECTION 6.3 - MX PROGRAMMING ENVIRONMENT

6.3.1 - Software Installation & Computer Setup

The MX2000 programming environment provides the means by which an application can be fully developed and the controller can be operated using a personal computer (PC). The application can be written, compiled and downloaded to the MX2000 controller, using the MX2000 PC program. In addition, a "Terminal Mode" is provided for operating the MX2000 controller from your computer.

Installation:

1. If Windows 3.1(or 3.11) is not already running, type **WIN** at the DOS prompt, and press **ENTER**.
2. Insert the **MX2000** Program Disk into drive A: (or B:).
3. Click on the **FILE** menu in the Program Manager.
4. Select **RUN**... to display the Run Dialog box.
5. Type **A:setup** (or B:setup) and click OK.
6. The installation program will display the MX2000 File Manager Setup screen. Follow the prompts on the screen to complete the installation.
7. After the program files have been installed, the installation will create a new Windows group called MX2000.
8. Remove the MX2000 installation disk. This concludes the MX2000 installation.

Starting the MX programming environment:

1. If Windows 3.1(or later) is not already running, type **WIN** at the DOS prompt, and press **ENTER**.
2. Double click on the MX2000 Icon.

Setting communication parameters:

The MX2000 PC program uses the computer's serial port to communicate with the MX2000 controller. The MX program supports the use of four serial ports, (Com 1 ,Com 2, Com 3 or Com 4). To communicate, an XON - XOFF protocol is used. This protocol needs only three wires to establish a communication link between the computer and the MX2000 controller. These wires should be connected to transmit (TX), receive (RX) and common (V0) as follows:

Computer		MX2000
TX	------------------------------	RX
RX	------------------------------	TX
V0	------------------------------	V0

Note 1: The 9-conductor cable supplied in the MX system accessory kit (shipped with your unit) should allow easy connection to your PC's serial port. A 25-to-9 pin adaptor is required (user supplied) if the PC port is a 25-pin style

Note 2: Consult your computer manual for the correct pinout of it's serial port. See section 5.3.2.2 of this manual for connections to the MX2000 controller. Also, be sure to read Section 5.1.1 regarding baud rate precautions.

The MX2000 PC program supports four baud rates: 4800, 9600, 19200 and 38400. To set up the serial port, baud rate and Terminal Emulation Mode used for communications, select the **Configure Com Port** item under the System menu (see Figure 6.8). The word length, parity, and number of stop bits are fixed at 8, none, 1 respectively. The baud rate for the MX2000 controller can be set via switches on the front panel (see also Section 5.1). Both the MX2000 Controller and the MX PC program are set to default to a baud rate of 9600 when shipped. The MX2000 PC program will also default to Com 1.

Note: The Terminal Emulation Mode should be set to TTY on the Configure Com Port screen.

The serial communications to the MX2000 Controller can be tested by selecting the **Terminal** item under the **Utility** menu (see Figure 6.7). Simply click on the **Software Revision** command button and the MX2000 Controller will send back the software revision information which will be displayed (see Figure 6.10).

6.3.2 - Axis Configuration & Setup

In order to configure an axis for a new project, a task must first be created and assigned to the project. Here is an outline of the procedure:

1. Select the **New** item under the **Project** menu.
2. Enter the new project name in the **File Name** text box (name.prj) and click on the **OK** command button. This creates a task file (project name.tsk) which is added to the project.

3. If a different name is desired for the task.

   a) Click on the **Remove Task** item under the **Project** menu.
   b) Click on **OK** and **Yes** command buttons. This removes this file name from the project.
   c) Click on the **Save AS** item under the **Task** menu. Type in the desired name in the **File Name** text box (name.tsk) and click on the **OK** command button.
   d) Click on the **Add Task** item under the **Project** menu. Then click on Yes command button to add this task file to the project.

4. If additional tasks need to be added to this project, repeat the following steps until all the new tasks required have been added to the project:
   a) Click on the **New** item under the **Task** menu.
   b) Click on the **Save AS** item under the **Task** menu. Type in the desired name in the **File Name** text box (name.tsk) and click on the **OK** command button.
   c) Click on the **Add Task** item under the **Project** menu. Then click on **Yes** command button to add this new task file to the project.

5. Click on the **Configuration & Setup** Command button. The Configuration Axes Folder screen appears (see Figure 6.19).

Using the Configuration & Setup Folders:

Notes:
Clicking on the **Exit Configuration & setup** command button can be used on any folder to exit the Configuration & setup. If any of the items in the folder have been changed, a query will occur which will allow the user the option of saving the folder data.

Clicking on the **Save changes** command button can be used on any of the folders to save the current folder data.

Clicking on another folder tab, will allow entry into that folder. However, if the data on the current folder has been changed, a query will occur which will allow the user the option of saving the folder data.

Axes Folder:

The **Axes** folder is the first folder opened when entering the Configuration & setup (Figure 6.19). This folder allows the selection of the Number of Axes in this project via an option button.

System folder:

The **System** folder can be opened by clicking on the System tab(Figure 6.20 & 6.21). First assign each axis to a specific task by clicking on the Task assignments list box and selecting a task. The Drive type for each axis must be assigned. The choices are open loop stepper, closed loop stepper or servo. The Motor Direction can be left at the default setting. The Units per motor revolution value must now be entered. This defines the number of units per motor revolution for each axis.

Drive related folders:

The drive folders are **Stepper drive, Closed Loop stepper, Servo drive** and **Encoder**. The drive type of an axis determines which drive folder's should be opened and modified for that axis. The desired folder can be opened by clicking on the tab with its name. The following is a listing of the folders which should be modified for the different drive types:

Drive type	Folder	Figure's
open loop stepper	stepper drive	6.26 & 6.27
closed loop stepper	stepper drive	6.26 & 6.27
	closed loop stepper	6.28 & 6.29
	encoder	6.25
servo	servo drive	6.30 - 6.32
	encoder	6.25

The **Stepper drive** folder **Steps per motor revolution** text box must be modified to reflect the drive resolution of the stepper drive. The **Low Speed, Motor standstill current** and **Motor current delay** settings can be left at the default setting if desired.

The **Closed Loop Stepper** folder **Error Action** drop list should be set to **disabled** for now. The desired **Error action** can be set later, either by changing the axes configuration & setup or in the user program using the ENCMODE command. The **Following error, Position error, Correction attempts** and **Time between attempts** settings can be left at the default settings if desired.

Note: The Closed loop stepper can be tested for proper operation in the Terminal emulation mode after the project has been compiled and downloaded. The following Host commands are used to test a closed loop stepper axis:

ABSPOS(axis)=0 'sets the absolute and encoder position to 0.

MOVE(axis)=100 'move closed loop stepper 100 units.

ENCPOS(axis) 'Do this command after motion has stopped.

1) If the value returned is +100 the closed loop stepper is setup properly.
2) If the direction returned is negative use the other setting for the **Encoder direction** drop list in the **Encoder** folder.
3) If the value returned is the incorrect value, the **Line count** text box in the **Encoder** folder must be changed.
4) Repeat the above Host commands until the closed loop stepper is setup properly.

The **Encoder** folder is used by a servo or closed loop stepper axis. The **Encoder type** drop list should be set for **quadrature**. The **Line count** text box should be set to the line count on the encoder. The **Encoder direction** drop list can be left at the default setting. The **Pulse count** text box requires no entry unless the **Pulse and Direction** option has been selected in the **Encoder Type** drop list. If a value is to be entered it should be equal to the number of input pulses per motor revolution.

The **Servo drive** folder is used to set up the different servo gains and the following error. These settings can be left at the default setting if desired and changed later using the TUNING A SERVO section.

Note: For servo systems, it is recommended to use Auto Tuning (see Section 6.3.5) to determine the proper servo gains for each axis. When the auto tuning is completed, the gain values are automatically transferred to the SERVO DRIVE folder; generally no further modification of these values will be required.

The remaining folders which apply to all axes can now be modified or left at there default settings: **Profile, Analog inputs, Travel Limits** and **Mechanical Home & Mark registration.**

Profile folder:

The **Profile** folder can be opened by clicking on the **Profile** tab (Figures 6.22 & 6.23). This folder allows the **Motion Profile**, Non-coordinated **Speed, Acceleration** rate, **Deceleration** rate, **Maximum Acceleration** rate, **Maximum Speed** and **Delay after motion** to be specified for each axis. These selection can be left at the default settings if desired.

Analog Inputs Folder:

The **Analog Input** folder can be opened by clicking on the **Analog Input** tab (Figure 6.24). The analog input type can be selected, differential or single ended, for each axis. If single ended mode is selected, the analog in+ and in- terminals are two individual inputs for that axis. Refer to the ANALOG command table for input addressing of each axis. The Filter 1 time constant of each axis is used for the analog in+ terminal. The Filter 2 time constant of each axis is used for the analog in- terminal and is only required if the single ended mode is selected.

Travel Limit Folder:

The **Travel Limit** folder can be opened by clicking on the **Travel Limit** tab (Figures 6.33 & 6.34). The **Hardware travel limit** can be disabled or set for a switch closing or opening. The **Hard limit deceleration** rate for a hardware limit activation can be programmed, a 0 specifies an immediate stop. The **Software travel limit** can be enabled or disabled. If enabled the Positive and Negative software limits can be set.

Mechanical Home & Mark Registration Folder:

The **Mechanical Home trigger** and **Mark Registration trigger** can be set for each axis. The travel limit for a Mark Registration cycle can be set. A 0 disables the Mark Registration incremental travel limit. (Figure 6.35)

Parameter Logging:

The MX2000 programmer has the ability of logging 500 sample points for each parameter selected. Up to 8 parameters can be logged on one motion trigger. The **parameters, Trigger Axis, Display Time** and **Trigger Delay** are selected on the **Parameter & Trigger Setup** screen (Figure 6.12 & 6.13). After a motion trigger has occurred the logged data can be transferred (Figure 6.15 & 6.16) from the MX2000 controller in the desired viewing mode (**Zero center, Min-Max** or **Manually** scaled) by selecting the **Data Transfer** screen . The transferred data can now be viewed using the **View Data** screen (Figure 6.17 & 6.18).

Note: the logged data can be transferred or viewed as many times as you desire.

The **Logging** function is accessed by the **Utility** pull-down menu, simply click on the **Utility** menu (Figure 6.7) and then on the **Logging** menu (Figure 6.11). Three selections appear next to the Logging menu. If the **Data Transfer** and **View Data** selections are grayed out then the **Parameter & Trigger Setup** must be accessed first, this indicates that parameters and trigger axis have not been selected or a data log trigger has not occurred.

Parameter & Trigger Setup:

This selection is required when no parameters have been previously selected, a new set of parameters are required or a new motion trigger is required. To access the **Parameter & Trigger Setup** screen simply click on the **Parameter & Trigger Setup** selection (Figure 6.11). The **Parameter & Trigger Setup** screens are shown in Figures 6.12 & 6.13.

There are 23 parameters for each axis. To access a parameter that is not displayed simply click on the (up or down arrow) vertical scroll bar until the parameter appears. To select or deselect a parameter click on the appropriate check box. A maximum of 8 selections are allowed.

The **Trigger Axis** Spin controller selects the motion trigger axis for logging. When this axis starts a motion cycle a parameter logging session is started.

The **Display Time** Spin controller selects the period of the logging session. The time is in seconds and 500 points are logged during this period.

The **Trigger Delay** Spin controller sets the time delay before data logging begins once a motion trigger occurs. This delay time is in milliseconds.

The **Ok** command button sends the parameter listing, trigger, display time and trigger delay to the MX2000. The MX2000 motion trigger is now enabled. The **Terminal Emulation** screen is accessed to allow a motion to be commanded (Figure 6.14). A motion by the Trigger Axis is required to data log the selected parameters. This can be accomplished by executing a program or manual motion.

The **Cancel** command button will exit the Parameter & Trigger Setup screen.

Data Transfer:

The Data Transfer screen is accessed by clicking on the **Utility, Logging,** and **Data Transfer** menus.
This screen allows individual parameter transfer enabling and data display mode to be selected (Figures 6.15 & 6.16). A list of parameter will be shown. If the desired parameter is not displayed simply click on the (up or down arrow) vertical scroll bar until the parameter is listed. To select or deselect a parameter from being transferred simply click on the parameter check box. To select the display format click on the appropriate display mode option button. If the selection is manual two additional text boxes need to be loaded, Full Scale and Offset.

The **Ok** button will request the logged data transfer and the **View Data** screen will be selected (Figures 6.17 & 6.18)..

The **Cancel** command button will exit the Data Transfer screen.

View Data:

The **View Data** screen allows the viewing of the transferred parameters. The Title and scale unit of the data appears in the drop list box (Figure 6.17). To select a different parameter click on the drop list arrow and simply select the desired parameter(Figure 6.18). The vertical cursors that appear on the view port can be moved by clicking on the cursor and dragging it to the desired position. The cursors can be used to measure time or for selecting the Zoom area. The cursors position time values are displayed above the view port. The time between cursors is displayed in the center just below the view port. The start time and end time for the displayed signals are displayed on the left and right hand side just below the view port.

The **Zoom** command button is used to toggle the display Zoom mode. The Zoom area is determined by the position of the cursors.

The **Save** command button is used to save the displayed parameter signal.

The **Graph setup** command button allows selecting the color and style for each logged item.

The **Print** command button allows the screen to be printed.

The **Quit** command button exits the View Data screen.

6.3.3 - Programming

Working with Files & Tasks:

To create a new task, select the **New** menu item under the **TASK** menu. To edit an existing task, select the **Open** menu item under the **TASK** menu. In both cases, a text editing window will appear. Files may also be saved and printed by using the available menu commands.

Editing a task:

Once a new file is created, or an existing file is opened, then it can be edited using the menu commands under the **Edit** menu (see Figure 6.4). These commands allow text to be cut, copied, pasted or deleted. Also provided is the ability to search for or replace text. Most editing commands can be carried out using standard mouse and keyboard functions.

Compiling and Debugging the project:

To compile the project, select the **Compile Project** item under the **Compile** menu (see Figure 6.5). The current project will be compiled.

If the compiler detects any errors during compilation, the line or lines will be displayed in an error window with an error message indicating the type of error(s) detected.

If no errors are detected a successful compilation is indicated. A successful compiled project results in the creation of a file that is suitable for downloading to the MX2000 Controller. These files are saved with the user selected project name; a **.USR** extension is automatically assigned.

To debug the project, load the offending task into the editing window, locate the line or lines containing errors, make the required corrections and recompile the project. It may be necessary to repeat these steps several times until the project compiles without errors.

Downloading the project:

To download the project select the **Download Project** item under the **Download** menu (see Figure 6.6). The project will be downloaded to the MX2000 Controller.

Executing the project:

To execute a project, select the **Terminal** item under the **Utility** menu (see Figure 6.7). A Terminal Emulation screen will appear (see Figure 6.10). Click on the **Run Program** command button to start execution of the project. To stop execution of a project click on the **Stop Program** command button.

Maximum Values:

When writing an MX program, please observe the following maximum values; otherwise, the project will not compile properly.

Item	Maximum Allowed
Line labels	100 total per task
Local variables	100 total per task
Common variables	100 total per project
Quoted literals	100 total per project
Data elements	100 total per task
Nested "FOR" loops	16 total per task
Nested "DO" loops	16 total per task

A discussion regarding the maximum number of program lines can be found in section 6.1, under the subheading "Memory Types and Usage".

6.3.4 - Using the MX2000 PC Program

The following is a summary of the various keyboard commands and their functions. Please note that the use of a mouse is fully supported, and in most cases makes editing quicker and easier.

Using Menus and Commands

Use the menu bars at the top of the screen to select menus and commands.

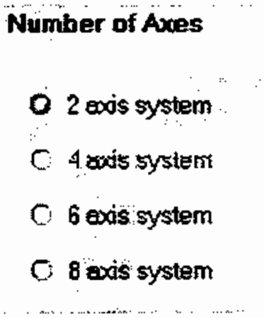
Menu Action	With a Mouse	With the keyboard
Display a menu	Move the mouse pointer to the Menu name, then press and release (Click) the mouse button.	Press the **ALT** key to highlight the menu letters, then press the letter key for the menu you want to display.
Choose a command	Click the command name.	Press the letter key that matches the highlighted letter on the command.
Cancel a command	Click outside the menu.	Press **ESC** key.

Using Dialog Boxes

The MX editor displays dialog boxes when you choose commands that have options. The following figures illustrate different types of dialog boxes and the actions that can be performed.

Dialog Box action

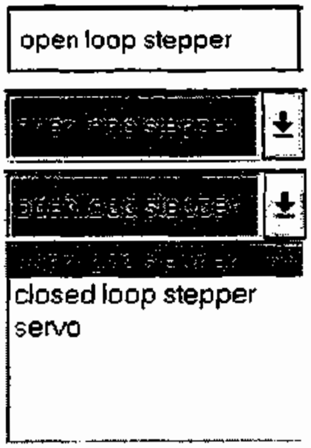
Option Box


Number of Axes

- ● 2 axis system
- ○ 4 axis system
- ○ 6 axis system
- ○ 8 axis system

Click on desired Option.

Drop List


open loop stepper

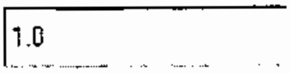
closed loop stepper
servo

Click on Box.

Click on Arrow.

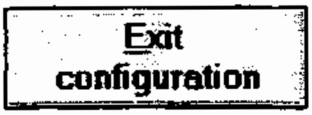
Click on item.

Text Box


1.0

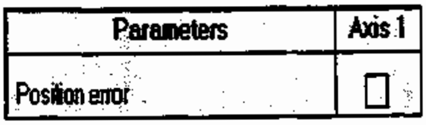
Click on box.
Enter Data.

Command Button

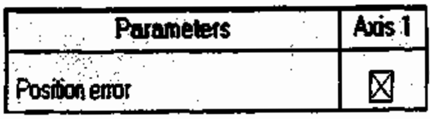


Click on button.

Check Box



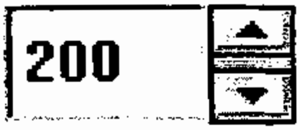
Parameters	Axis 1
Position error	□

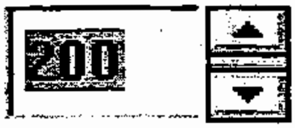


Parameters	Axis 1
Position error	☒

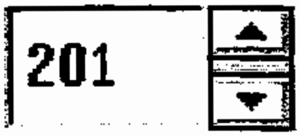
Click on Box.

Spin Controller


200


200

Click on Box.


201

Click on Arrow

Cursor Movement Keys

Character Left	Left Arrow
Character Right	Right Arrow
Word Left	Ctrl Left Arrow
Word Right	Ctrl+Right Arrow
Line up	Up Arrow
Line down	Down Arrow
Beginning of Line	Home

Text Scrolling Keys

Line up	Ctrl+Up Arrow
Line Down	Ctrl+Down Arrow
Page Up	PgUp
Page Down	PgDn
Left one window	Ctrl+PgUp
Right one window	Ctrl+PgDn

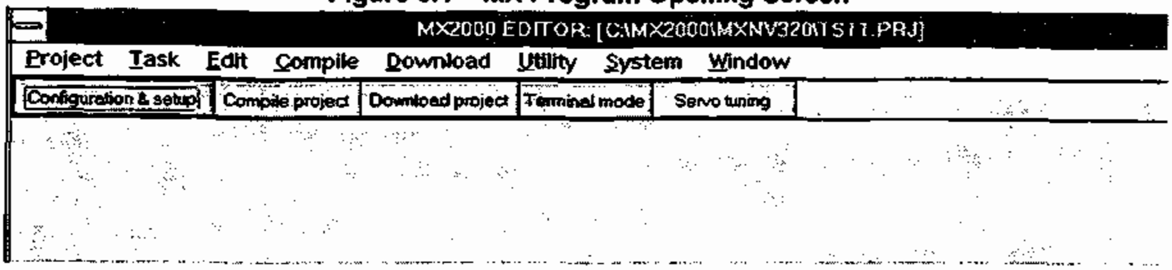
Text Selection Keys

Character left	Shift+Left Arrow
Character right	Shift+Right Arrow
Word Left	Shift+Ctrl+Left Arrow
Word right	Shift+Ctrl+Right Arrow
Current Line	Shift+Down Arrow
Line Above	Shift+Up Arrow
Screen Up	Shift+PgUp
Screen Down	Shift+PgDn
Beginning of file	Shift+Ctrl+Home
End of File	Shift+Ctrl+End

Insert, Delete and Copy Keys

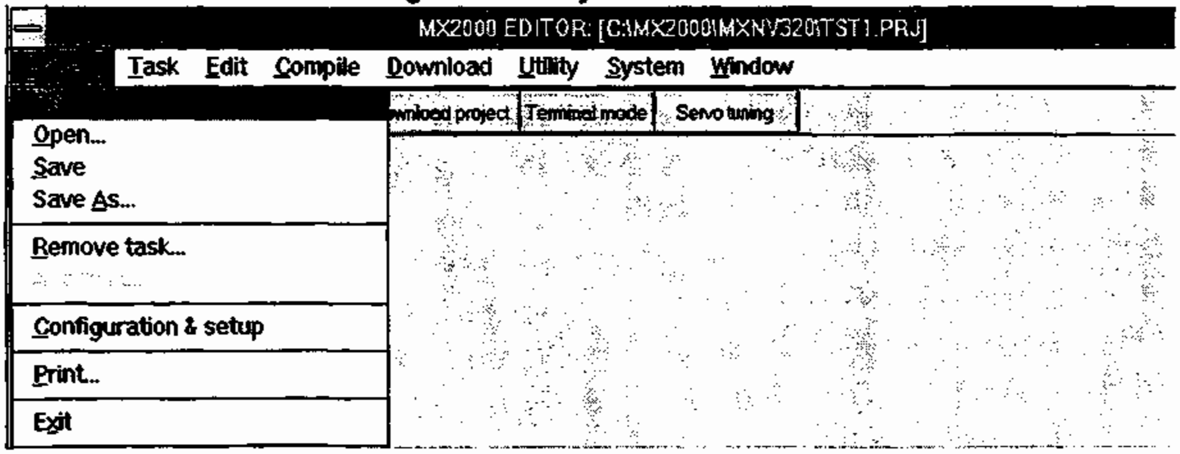
Switch between insert and overstrike modes	Ins	
Copy selected text to the clipboard	Ctrl+Ins	Ctrl+X
Delete selected text and copy it to the clipboard	Shift+Del	Ctrl+C
Paste the contents of the clipboard	Shift+Ins	Ctrl+V
Insert a blank line at the Cursor	Enter	
Delete one character to left of the cursor	Backspace	
Delete one character at the cursor	Del	
Delete selected text	Del	

Figure 6.1 MX Program Opening Screen


MX2000 EDITOR: [C:\MX2000\MXNV320\TST1.PRJ]
Project Task Edit Compile Download Utility System Window
Configuration & setup
Compile project
Download project
Terminal mode
Servo tuning

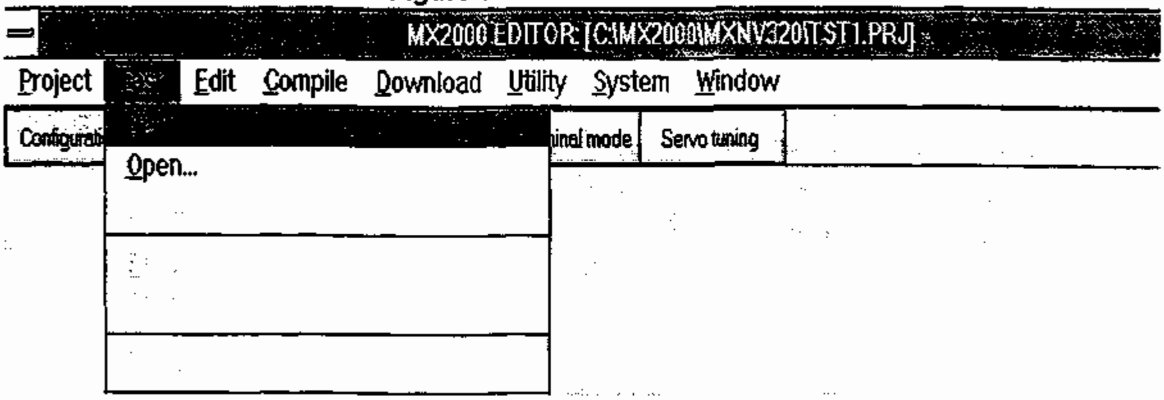
Command	Action
Project	Pull down menu of project related commands.
Task	Pull down menu of task related commands.
Edit	Pull down menu of edit related commands.
Compile	Pull down menu of compile related commands.
Download	Pull down menu of download related commands.
Utility	Pull down menu of utility related commands.
System	Pull down menu of system related commands.
Window	Pull down menu of window related commands.
Configuration & setup	Command button which select the project Configuration.
Compile Project	Command button which compiles the current project.
Download project	Command button which downloads the current project.
Terminal Mode	Command button which starts terminal mode emulation.
Servo tuning	Command button which allows the tuning of a servo motor.

Figure 6.2 Project Menu Screen



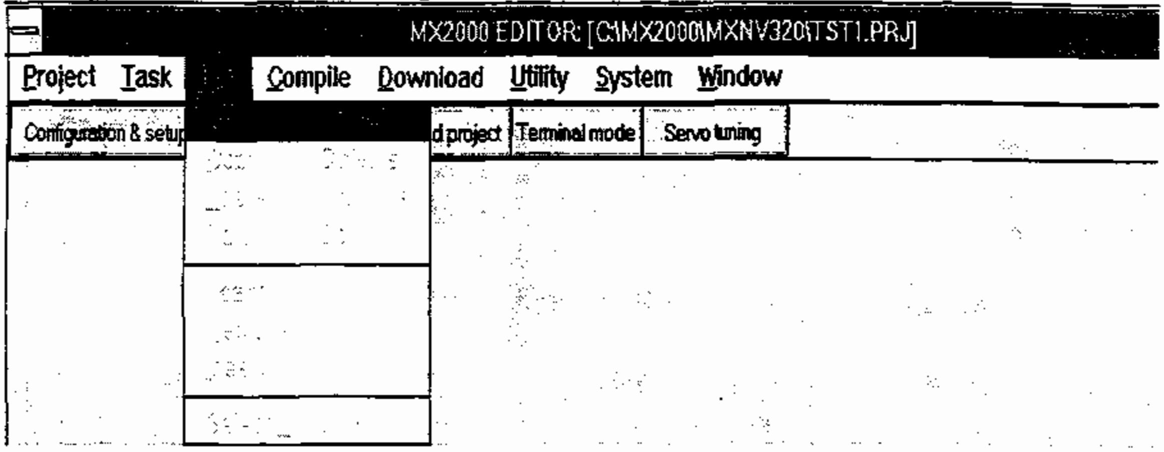
Command	Action
New	Creates a new project.
Open	Opens an existing project.
Save	Saves the current project.
Save As	Saves the current project under a new name.
Remove task	Removes a file from the project.
Add task	Adds a file to the project.
Configuration & setup	Selects the Configuration for editing.
Print	Prints the current project.
Exit	Exits the MX program.

Figure 6.3 Task Menu Screen



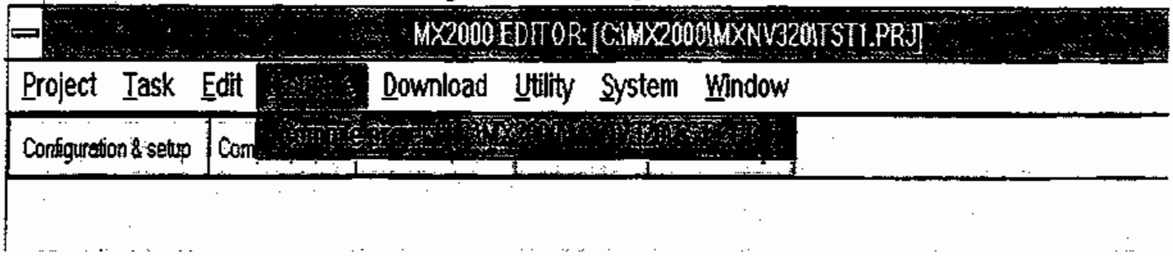
Command	Action
New	Creates a new task file.
Open	Opens an existing task file.
Close	Closes the currently open task file.
Save	Saves the currently open task file.
Save As	Saves the current task file under a new name.

Figure 6.4 Edit Menu Screen



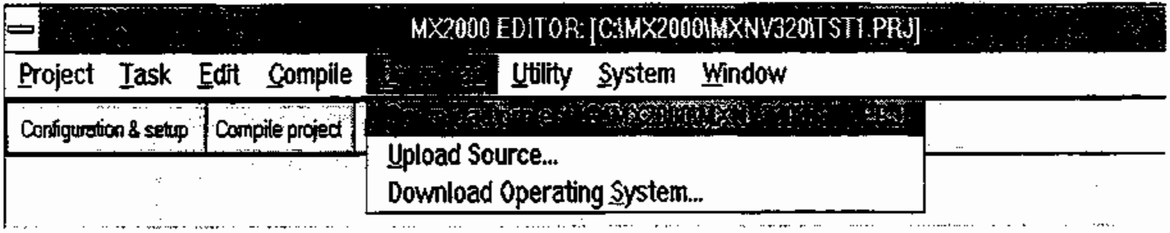
Command		Action
Cut	Shift+Del	Cuts the selected text and places it in the clip board.
Copy	Ctrl+Ins	Copies the selected text and places it in the clip board.
Paste	Shift+Ins	Pastes the contents of the clip board into the file.
Delete	Del	Deletes the selected text.
Search		Searches for the selected string.
Replace		Replaces one string with another string.
Insert		Inserts a selected file at the current position.
Select All	Ctrl+A	Selects all text.

Figure 6.5 Compile Menu



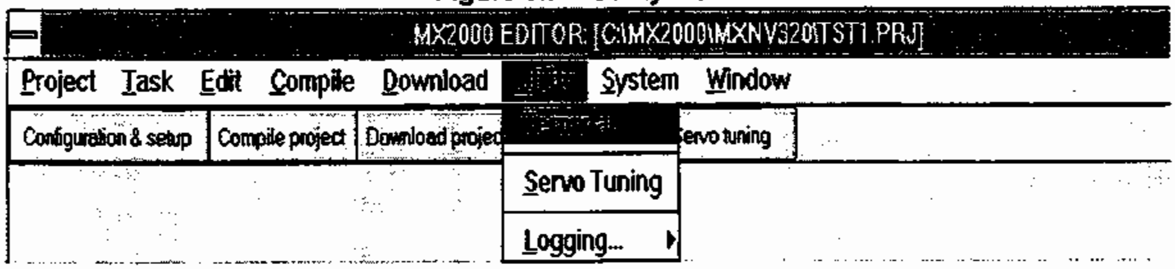
Command	Action
Compile project	Compiles the current project.

Figure 6.6 Download Menu



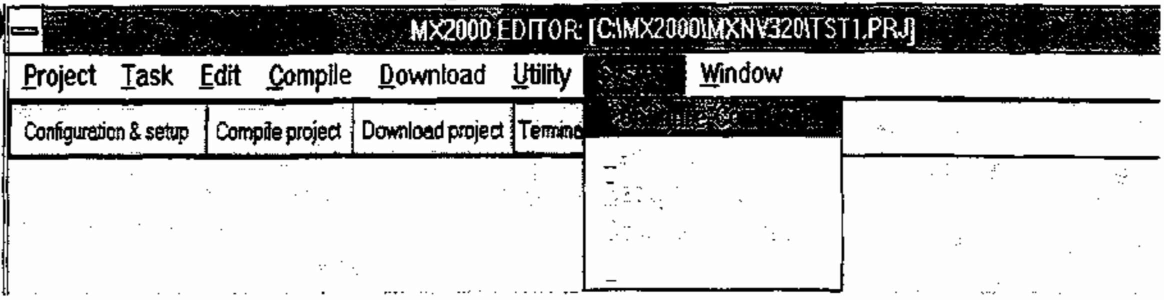
Command	Action
Download project	Downloads the current project to the MX2000 controller through the serial port.
Upload Source	Uploads the user program SEBASIC code from the MX2000 controller to the personal computer through the serial port.
Download Operating System	Downloads a new MX2000 Operating System to the MX2000 through the serial port.

Figure 6.7 Utility Menu



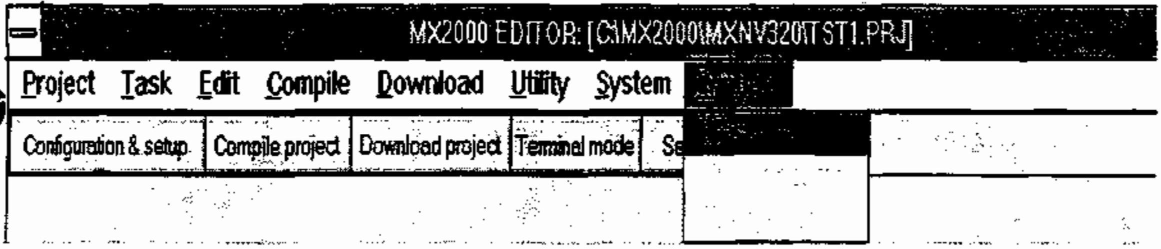
Command	Action
Terminal	Starts terminal mode emulation to communicate with the MX2000 controller over the serial port.
Servo Tuning	Allows the tuning of a servo axis.
Logging	Allows the logging of specific parameters by the MX2000 controller.

Figure 6.8 System Menu



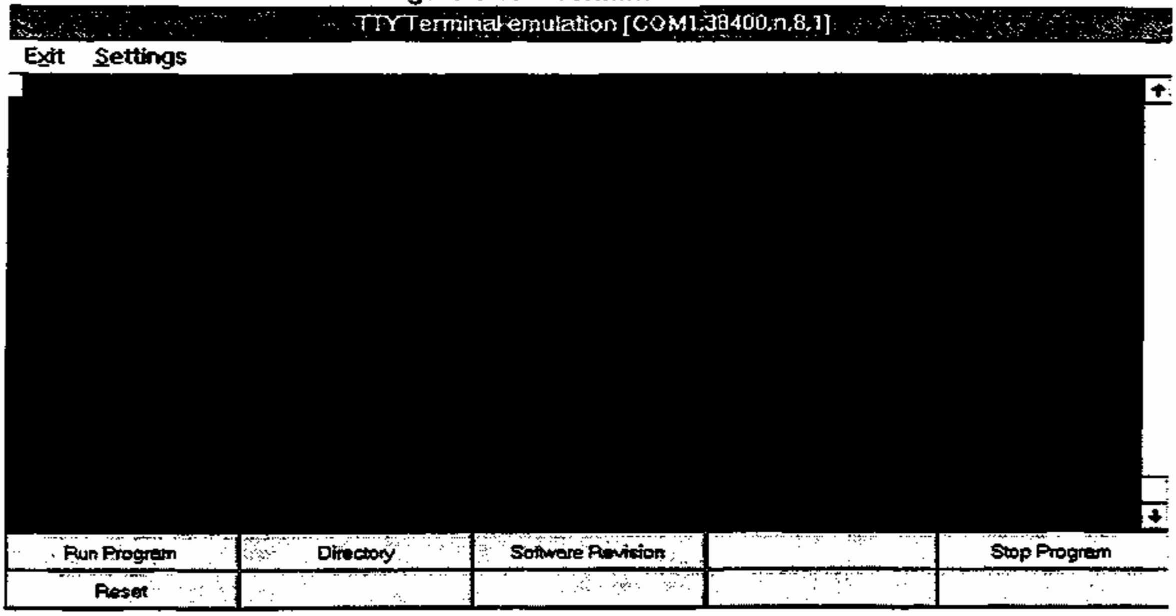
Command	**Action**
Configure Com Port	Selects the com port, baud rate and terminal emulation mode for communicating to the MX2000 controller.
Text Color	Selects the text color for the editor.
Background Color	Selects the background color for the editor.
Document Format	Selects the page size and margins for the document.
Font	Selects the text font for the editor.

Figure 6.9 Window Menu



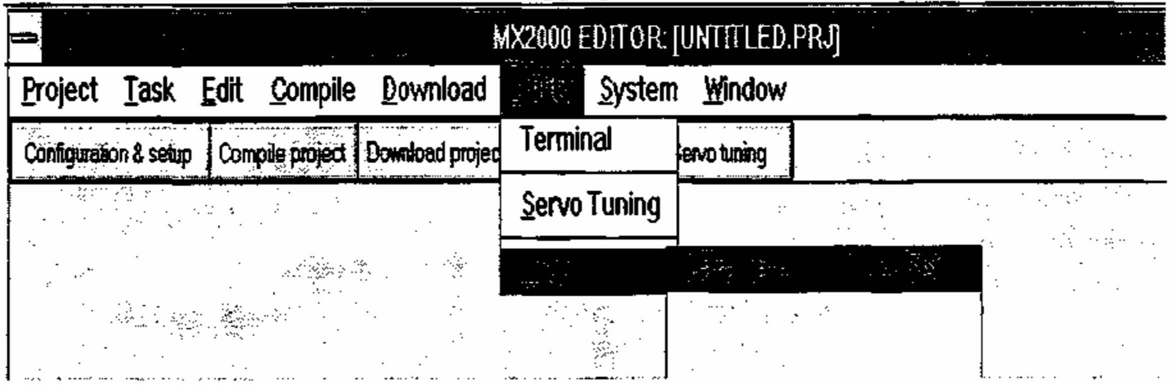
Command	**Action**
Cascade	Cascades the open windows.
Tile Horizontal	Tiles the open windows Horizontally.
Tile Vertical	Tiles the open windows Vertically.

Figure 6.10 Terminal Emulation



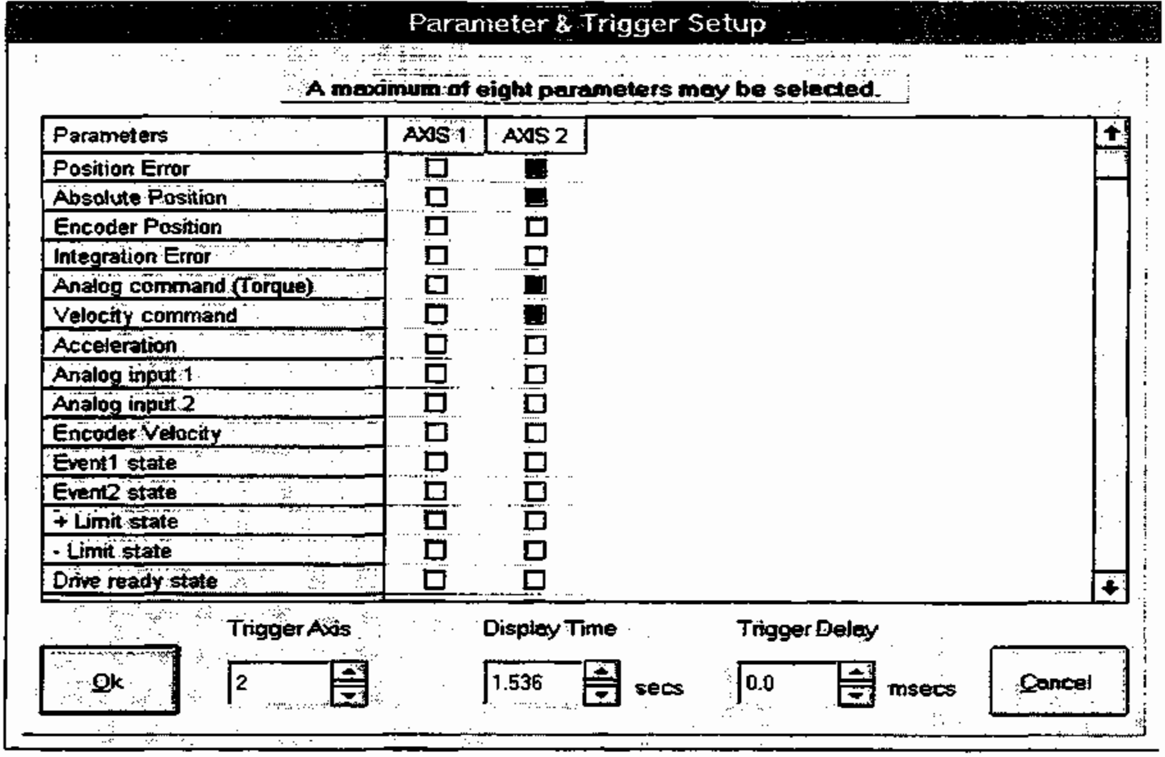
Command	Action
Exit	Pull down menu for exiting terminal emulation.
Settings	Pull down menu which allows the 10 command buttons to be programmed, select the full or half duplex mode and selects the font used in terminal emulation.
Run Program	Command button which runs the currently loaded project.
Directory	Command button which requests the project directory.
Software Revision	Command button which request s the software revision.
Stop Program	Command button which stops a running project.
Reset	Command button which resets the MX2000 controller.

Figure 6.11 Logging Selection Screen



Command	Action
Parameter & Trigger Setup	Selects the parameter to log, trigger, display time, and trigger delay.
Data Transfer	Selects the parameter and viewing mode for the logged data transfer.
View Data	Views the selected parameters.

Figure 6.12 Logging Parameters & Trigger Setup



Parameter & Trigger Setup

A maximum of eight parameters may be selected.

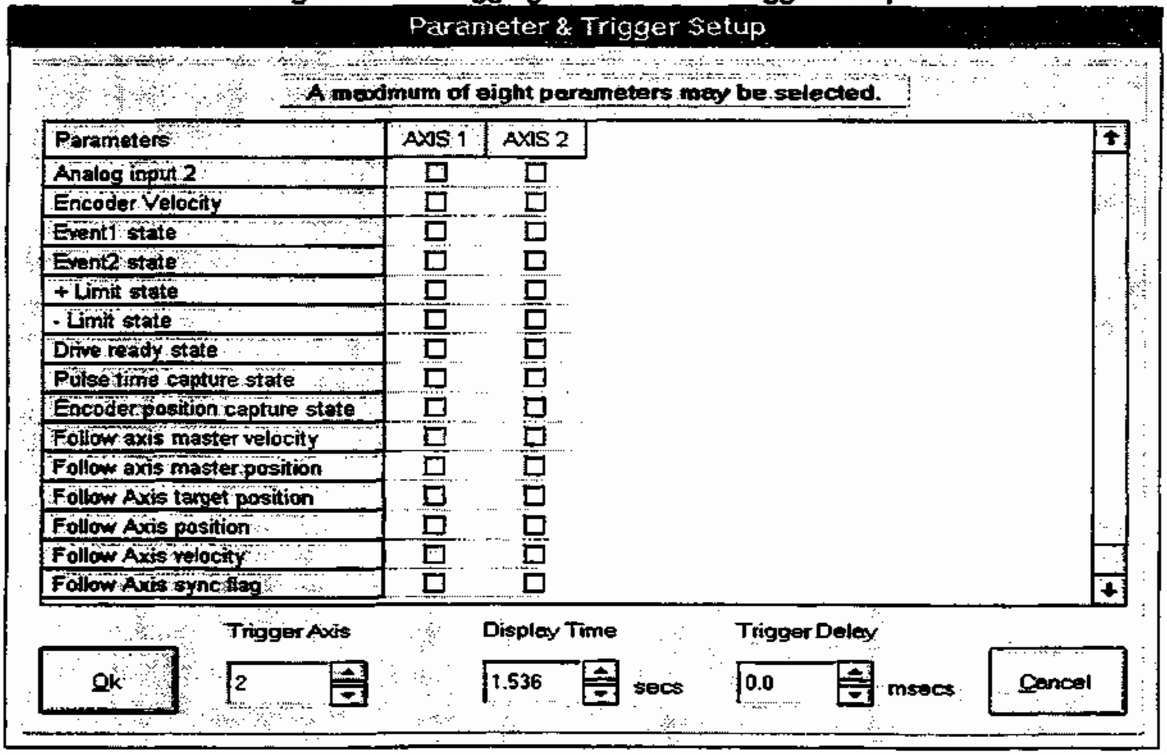
Parameters	AXIS 1	AXIS 2
Position Error	□	■
Absolute Position	□	■
Encoder Position	□	□
Integration Error	□	□
Analog command (Torque)	□	■
Velocity command	□	■
Acceleration	□	□
Analog input 1	□	□
Analog input 2	□	□
Encoder Velocity	□	□
Event1 state	□	□
Event2 state	□	□
+ Limit state	□	□
- Limit state	□	□
Drive ready state	□	□

Trigger Axis: 2

Display Time: 1.536 secs

Trigger Delay: 0.0 msecs

Figure 6.13 Logging Parameters & Trigger Setup



Parameter & Trigger Setup

A maximum of eight parameters may be selected.

Parameters	AXIS 1	AXIS 2
Analog input 2	□	□
Encoder Velocity	□	□
Event1 state	□	□
Event2 state	□	□
+ Limit state	□	□
- Limit state	□	□
Drive ready state	□	□
Pulse time capture state	□	□
Encoder position capture state	□	□
Follow axis master velocity	□	□
Follow axis master position	□	□
Follow Axis target position	□	□
Follow Axis position	□	□
Follow Axis velocity	□	□
Follow Axis sync flag	□	□

Trigger Axis: 2

Display Time: 1.536 secs

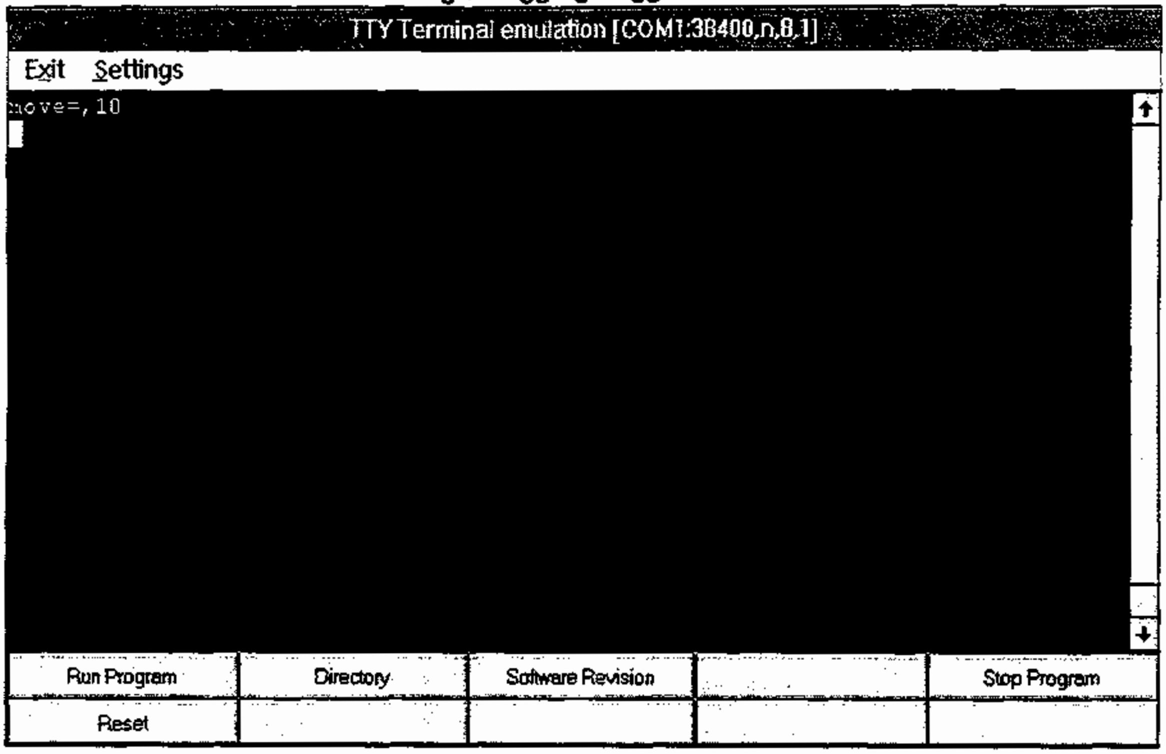
Trigger Delay: 0.0 msecs

Check Box	Action
Position error	Selects the position error of the axis for logging.
Absolute Position	Selects the Absolute position of the axis for logging.
Encoder Position	Selects the Encoder position of the axis for logging.
Integration Error	Selects the Integration error of the axis for logging.
Analog command (Torque)	Select the servo torque command of the axis for logging.
Velocity command	Selects the velocity of the axis for logging.
Acceleration	Selects the acceleration of the axis for logging.
Analog input 1	Selects the Analog input 1 of the axis for logging.
Analog input 2	Selects the Analog input 2 of the axis for logging.
Encoder Velocity	Selects the encoder velocity of the axis for logging.
Event1 state	Selects the Event1 state of the axis for logging.
Event2 state	Selects the Event2 state of the axis for logging.
+ Limit state	Selects the + Limit state of the axis for logging.
- Limit state	Selects the - Limit state of the axis for logging.
Drive ready state	Selects the Drive ready state of the axis for logging.
Pulse time capture state	Selects the Pulse time capture state of the axis for logging.
Encoder position capture state	Selects the Encoder position capture state of the axis for logging.
Follow axis master velocity	Selects the Following axis master velocity for logging.
Follow axis master position	Selects the Following axis master position for logging.
Follow axis target position	Selects the Following axis target position for logging.
Follow axis position	Selects the Following axis position for logging.
Follow axis velocity	Selects the Following axis velocity for logging.
Follow axis sync flag	Selects the Following axis sync flag for logging.

Command button	Action
OK	Sends the parameter listing, Trigger axis, Display Time and Trigger Delay to the MX2000 controller and goes to the Terminal Emulation Screen.
Cancel	Exits the parameter logging.

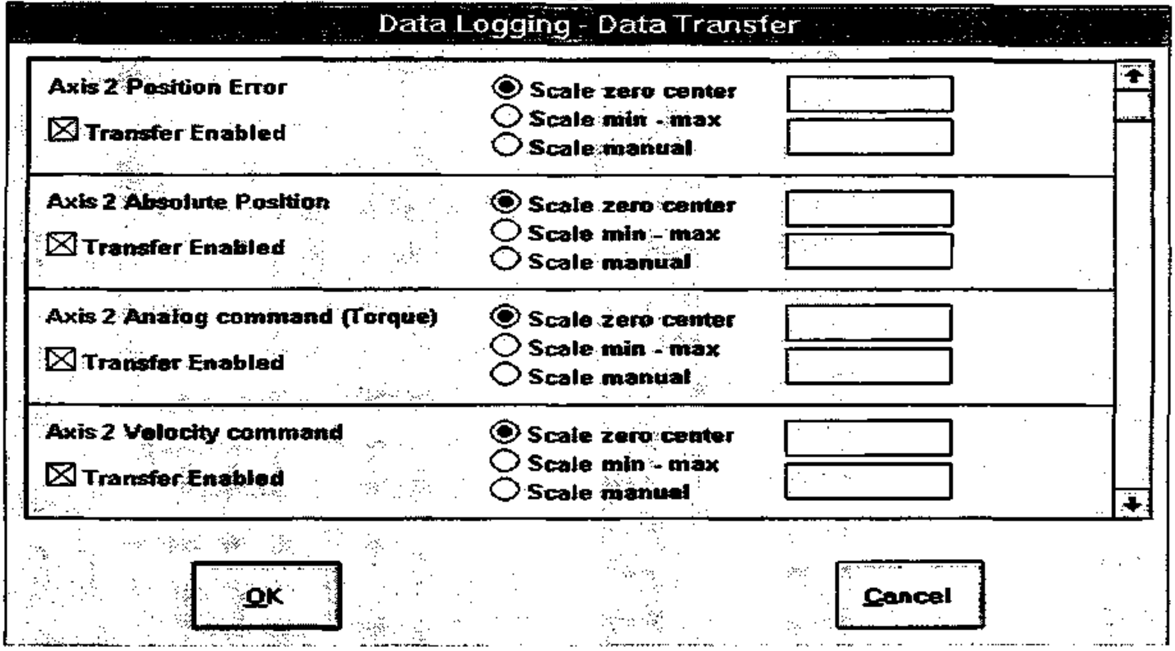
Spin Controller	Action
Trigger Axis	Selects the motion trigger axis for logging.
Display Time	Selects the logging period in seconds.
Trigger Delay	Selects the delay, in milliseconds, after the trigger occurs where parameter logging starts.

Figure 6.14 Terminal Emulation
Creating a Logging Trigger for Axis 2



```
move=,10
```

Figure 6.15 Logging Data Transfer



Data Logging - Data Transfer

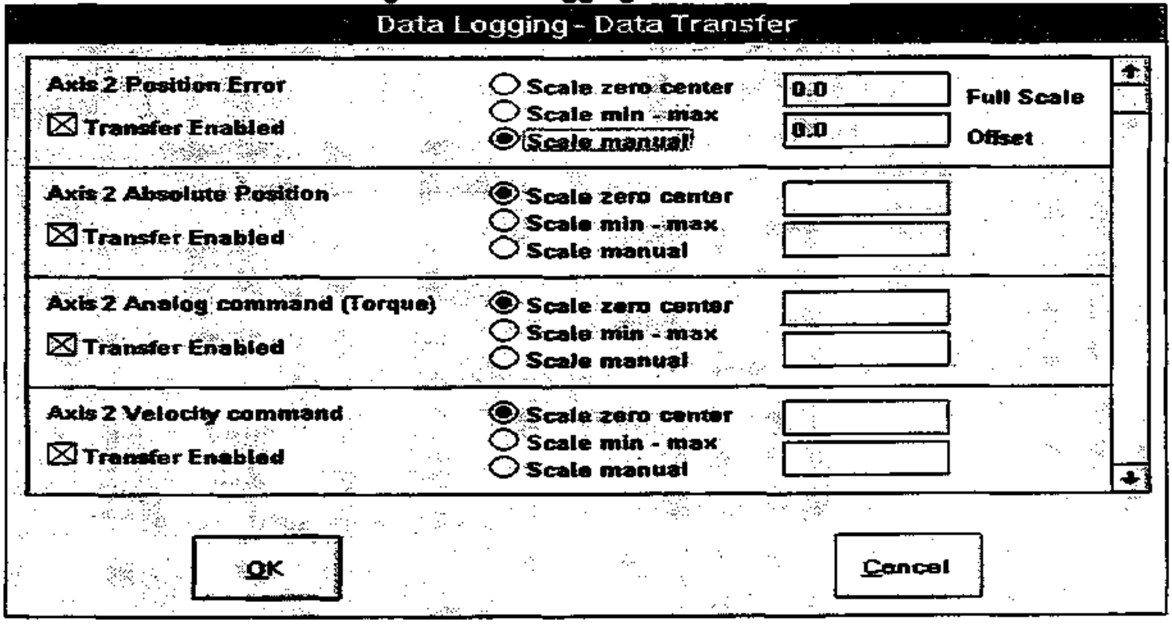
Axis 2 Position Error
☒ Transfer Enabled
● Scale zero center
○ Scale min - max
○ Scale manual

Axis 2 Absolute Position
☒ Transfer Enabled
● Scale zero center
○ Scale min - max
○ Scale manual

Axis 2 Analog command (Torque)
☒ Transfer Enabled
● Scale zero center
○ Scale min - max
○ Scale manual

Axis 2 Velocity command
☒ Transfer Enabled
● Scale zero center
○ Scale min - max
○ Scale manual

Figure 6.16 Logging Data Transfer



Data Logging - Data Transfer

Axis 2 Position Error
☒ Transfer Enabled
○ Scale zero center
○ Scale min - max
● Scale manual
Full Scale: 0.0
Offset: 0.0

Axis 2 Absolute Position
☒ Transfer Enabled
● Scale zero center
○ Scale min - max
○ Scale manual

Axis 2 Analog command (Torque)
☒ Transfer Enabled
● Scale zero center
○ Scale min - max
○ Scale manual

Axis 2 Velocity command
☒ Transfer Enabled
● Scale zero center
○ Scale min - max
○ Scale manual

Command	Action
Transfer Enabled	Check box which enables the parameter for transferred from the MX2000 controller.
Scale	Option button which allows the selection of the display mode: zero centered, min - max or manual.
Ok	Command button which request a logged data transfer from the MX2000 controller.
Cancel	Command button which exits the logging setup.

Figure 6.17 Logging View Data Screen

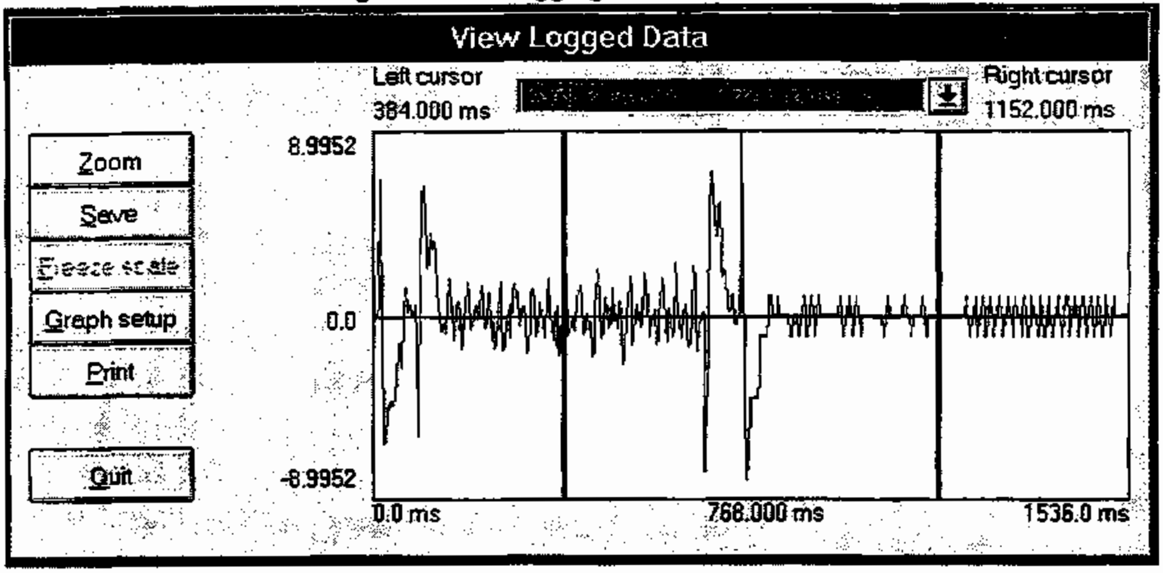
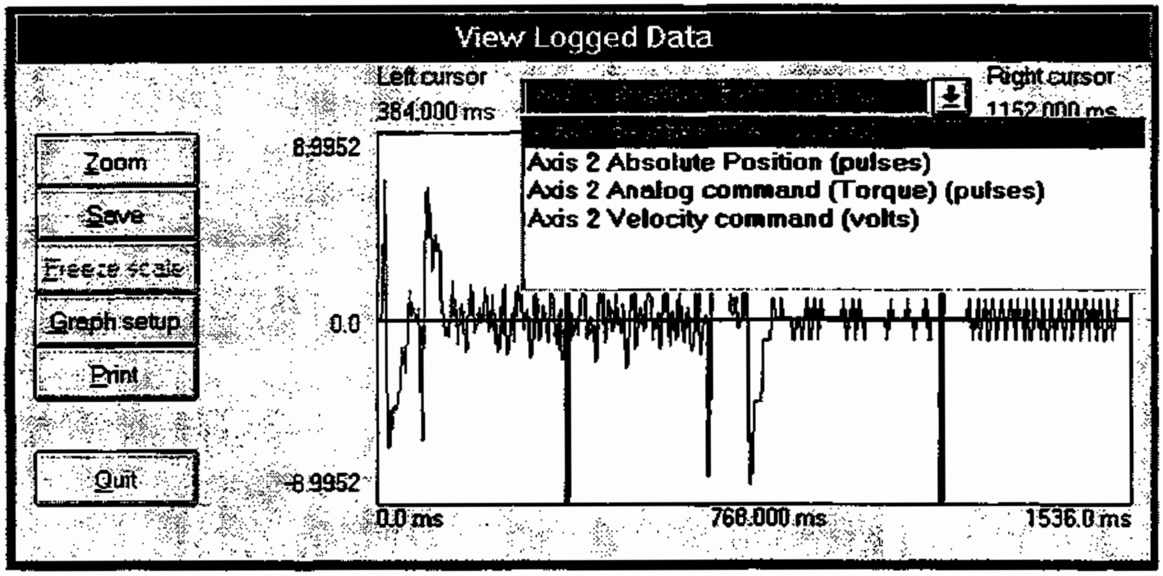


Figure 6.18 Logging View Data
Parameter Select

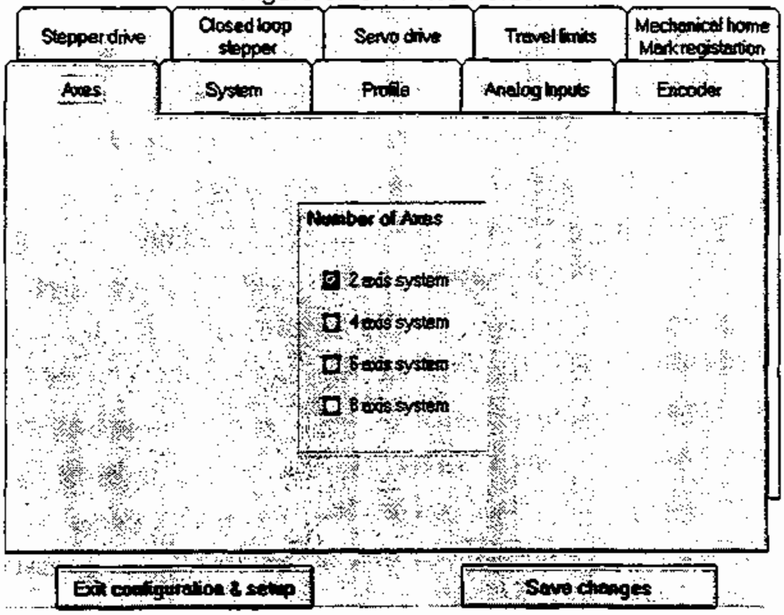


Command Button	Action
Zoom	Zooms the display between cursors.
Save	Saves the current display.
Graph setup	Allows selecting colors and style for each logged item.
Print	Prints the current screen.
Quit	Exits the View Logged Data Screen.

Configuration & Setup Folders:

The following "folder" screens are displayed when **Configuration & Setup** is selected, either via the Project pull-down menu or via the **Configuration & Setup** command button. Each folder allows for setup of specific axis and I/O items. Use the left-right scroll bar at the bottom of some folders to access addition items when there are too many to fit your screen. The folder tabs are displayed at the top of the folder and can be accessed by clicking on the desired tab. Note: not every folder requires scrolling.

Figure 6.19 Axes Folder



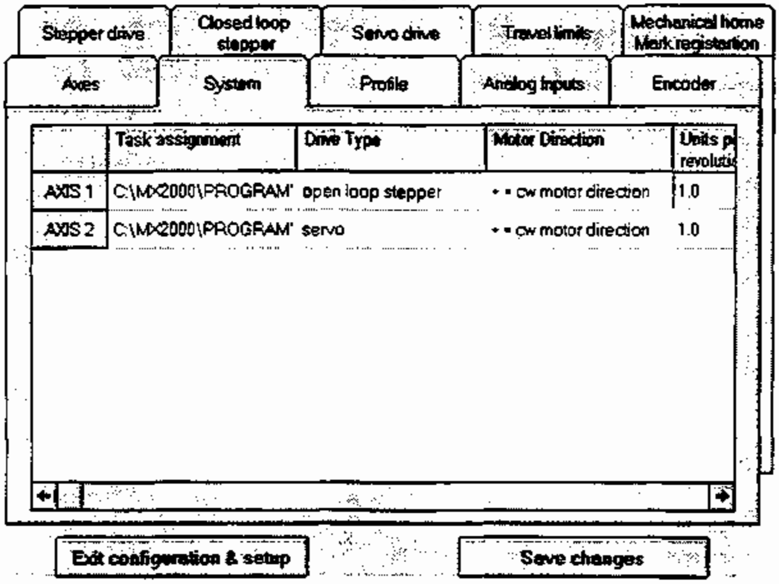
Number of Axes

☑ 2 axis system
□ 4 axis system
□ 6 axis system
□ 8 axis system

Option button	**Action**
Number of Axes	Selects the number of axes in this project.

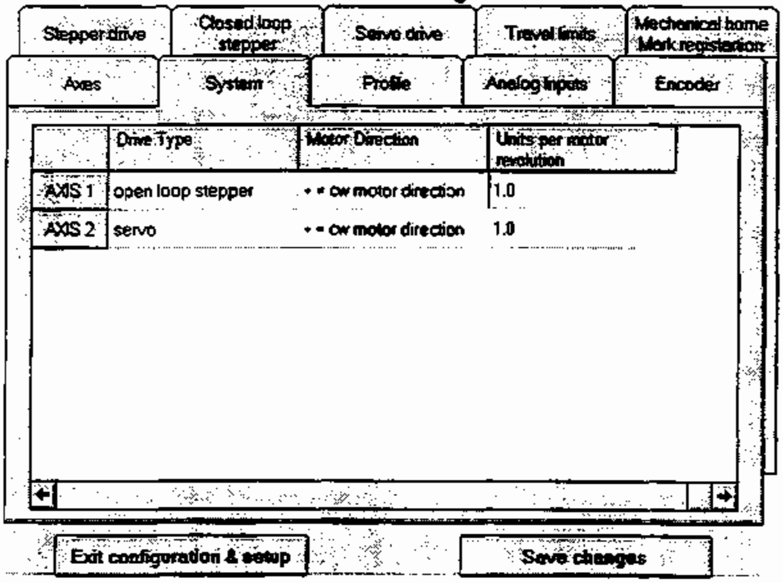
Command button	**Action**
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.20 System Folder
Scrolled to Left



	Task assignment	Drive Type	Motor Direction	Units p revoluti
AXIS 1	C:\MX2000\PROGRAM'	open loop stepper	+ = cw motor direction	1.0
AXIS 2	C:\MX2000\PROGRAM'	servo	+ = cw motor direction	1.0

Figure 6.21 System Folder
Scrolled to Right



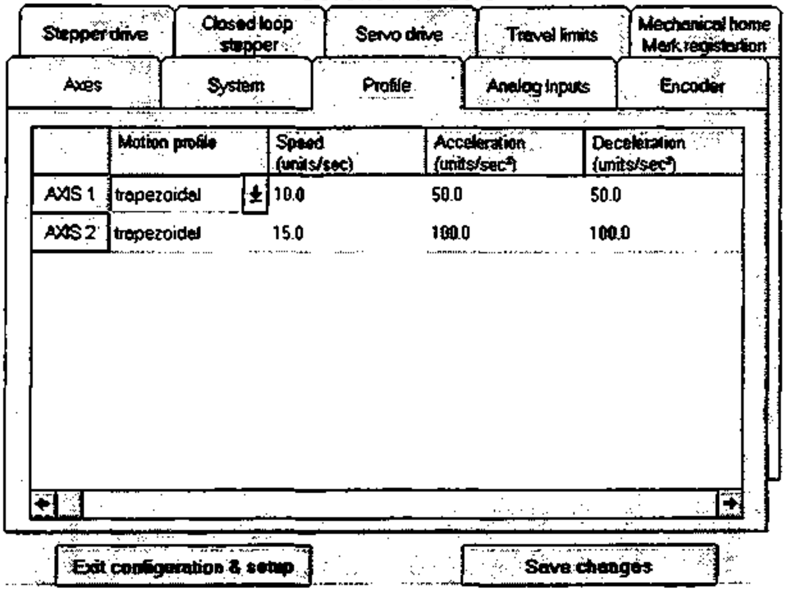
	Drive Type	Motor Direction	Units per motor revolution
AXIS 1	open loop stepper	+ = cw motor direction	1.0
AXIS 2	servo	+ = cw motor direction	1.0

Drop List	**Action**
Task assignment	Assigns an axis to a project task.
Drive Type	Assigns a Drive Type to an axis. The choices are: open loop stepper, closed loop stepper or servo.
Motor Direction	Allows a motor direction to be reversed for a + direction motion. The choices are: += ccwmotor direction or += cw motor direction.

Text box	**Action**
Units per motor revolution	Defines how many revolutions per unit.

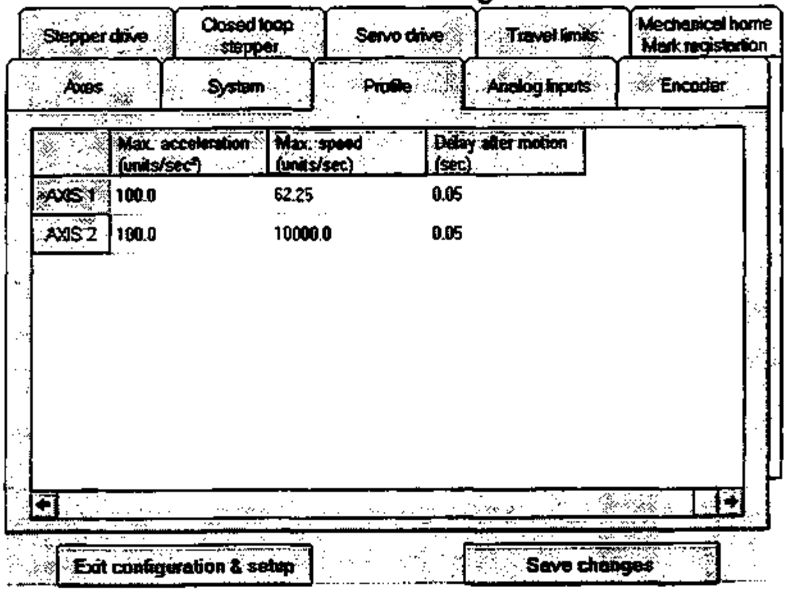
Command button	**Action**
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.22 Profile Folder
Scrolled to Left



	Motion profile	Speed (units/sec)	Acceleration (units/sec²)	Deceleration (units/sec²)
AXIS 1	trapezoidal	10.0	50.0	50.0
AXIS 2	trapezoidal	15.0	100.0	100.0

Figure 6.23 Profile Folder
Scrolled to Right



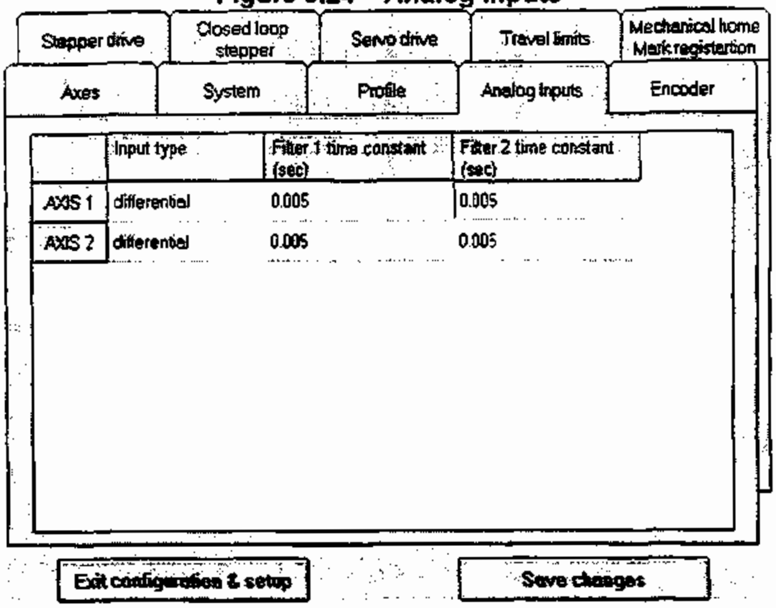
	Max. acceleration (units/sec²)	Max. speed (units/sec)	Delay after motion (sec)
AXIS 1	100.0	62.25	0.05
AXIS 2	100.0	10000.0	0.05

Drop List	Action
Motion profile	Selects the motion profile for the axis. Choices are: trapezoidal or 'S' curve.

Text box	Action
Speed	Sets the non-coordinated speed of an axis.
Acceleration	Sets the acceleration rate of an axis.
Deceleration	Sets the deceleration rate of an axis.
Max. Acceleration	Sets the maximum acceleration or deceleration rate of an axis.
Max. Speed	Sets the maximum speed allowed for an axis.
Delay after motion	Defines the minimum time between motions.

Command button	Action
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.24 Analog Inputs



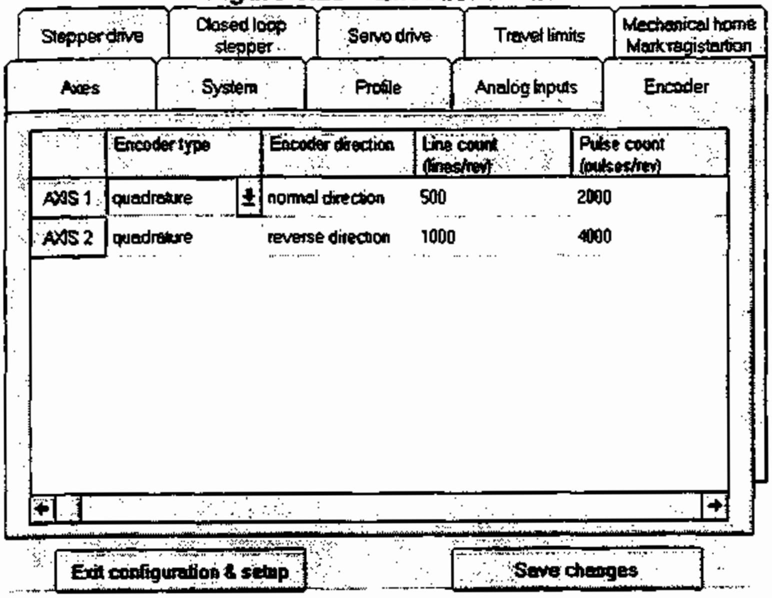
	Input type	Filter 1 time constant (sec)	Filter 2 time constant (sec)
AXIS 1	differential	0.005	0.005
AXIS 2	differential	0.005	0.005

Drop List	**Action**
Input type	Defines the analog input's of an axis. The choices are: differential or single ended.

Text box	**Action**
Filter 1 time constant	Defines the filter time constant for analog input 1 of the axis.
Filter 2 time constant	Defines the filter time constant for analog input 2 of the axis. This only applies if the input type for the axis is single ended.

Command button	**Action**
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.25 Encoder Folder



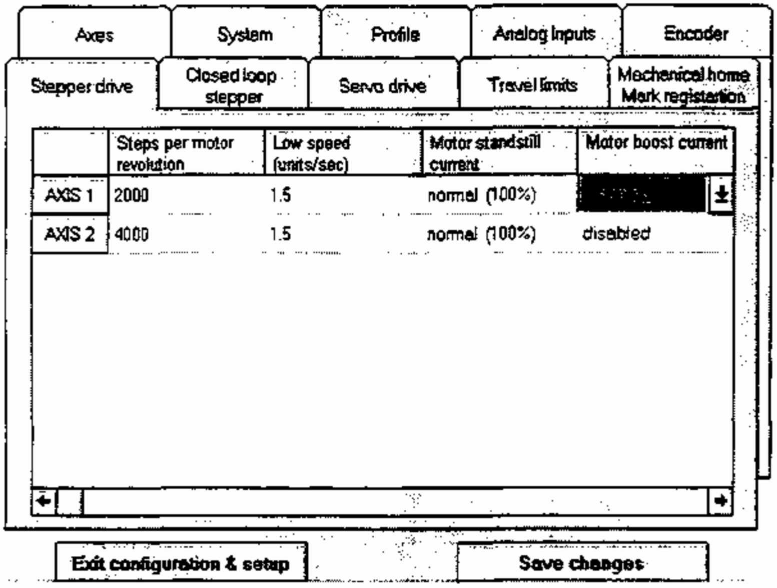
	Encoder type	Encoder direction	Line count (lines/rev)	Pulse count (pulses/rev)
AXIS 1	quadrature	normal direction	500	2000
AXIS 2	quadrature	reverse direction	1000	4000

Drop List	Action
Encoder Type	Selects Quadrature encoder or pulse and direction inputs for the encoder inputs.
Encoder Direction	Allows the encoder direction detected to be reversed. The choices are: reverse direction or normal direction.

Text box	Action
Line Count	Defines the encoder line count for one revolution.
Pulse Count	Defines the pulse count for one revolution. This entry is only required if the encoder type is pulse and direction.

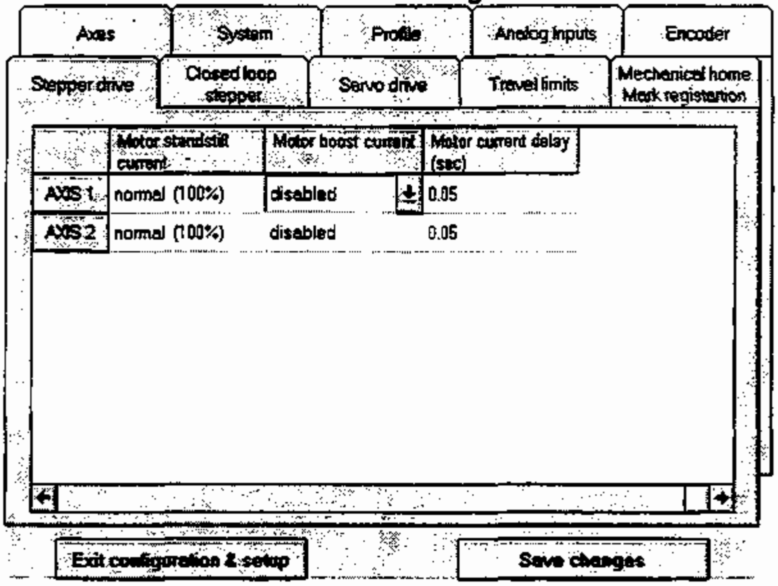
Command button	Action
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.26 Step Drive Folder Scrolled to Left



	Steps per motor revolution	Low speed (units/sec)	Motor standstill current	Motor boost current
AXIS 1	2000	1.5	normal (100%)	[illegible]
AXIS 2	4000	1.5	normal (100%)	disabled

Figure 6.27 Step Drive Folder Scrolled to Right



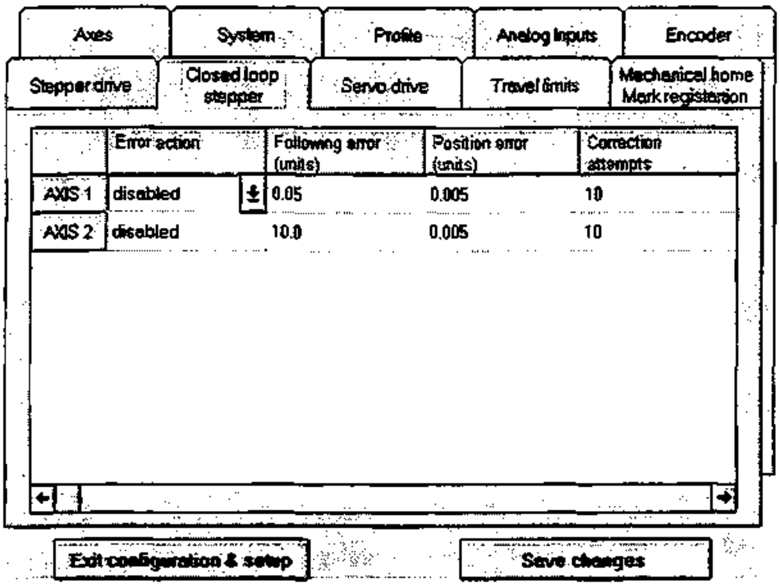
	Motor standstill current	Motor boost current	Motor current delay (sec)
AXIS 1	normal (100%)	disabled	0.05
AXIS 2	normal (100%)	disabled	0.05

Text box	**Action**
Steps per motor revolution	Specifies the stepping motor drive setting for each axis.
Low Speed	Specifies the starting speed for each axis.
Motor current delay	Specifies the time between current mode changes for each axis.

Drop List	**Action**
Motor standstill current	Specifies the state of the motor current at standstill for each axis. The choices are: normal 100%, reduced 50% and off 0%.
Motor boost current	Enables or disables the boost current feature for each stepping motor axis.

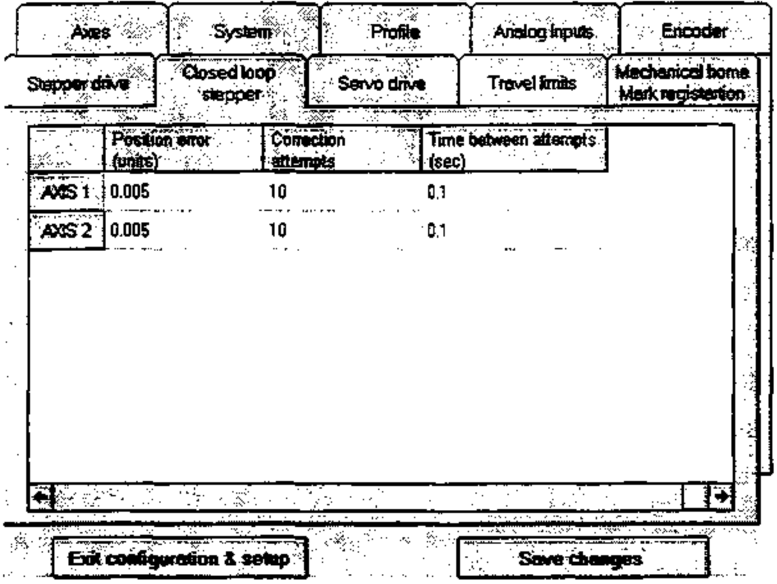
Command button	**Action**
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.28 Closed Loop Stepper Folder Scrolled to Left



	Error action	Following error (units)	Position error (units)	Correction attempts
AXIS 1	disabled	0.05	0.005	10
AXIS 2	disabled	10.0	0.005	10

Figure 6.29 Closed Loop Stepper Folder Scrolled to Right



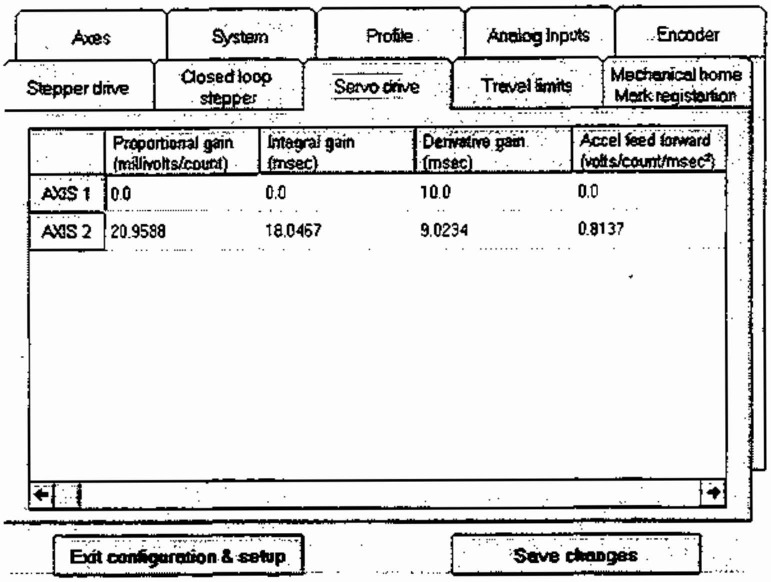
	Position error (units)	Correction attempts	Time between attempts (sec)
AXIS 1	0.005	10	0.1
AXIS 2	0.005	10	0.1

Drop List	**Action**
Error action	Defines the stepper closed loop mode for each axis. The choices are: disabled, stop on error, correct on error and restart on error.

Text box	**Action**
Following error	Specifies the following error for each axis.
Position error	Specifies the deadband error for each axis.
Correction attempts	Specifies the maximum number of consecutive correction attempts allowed for each axis.
Time between attempts	Specifies the time between correction attempts for each axis.

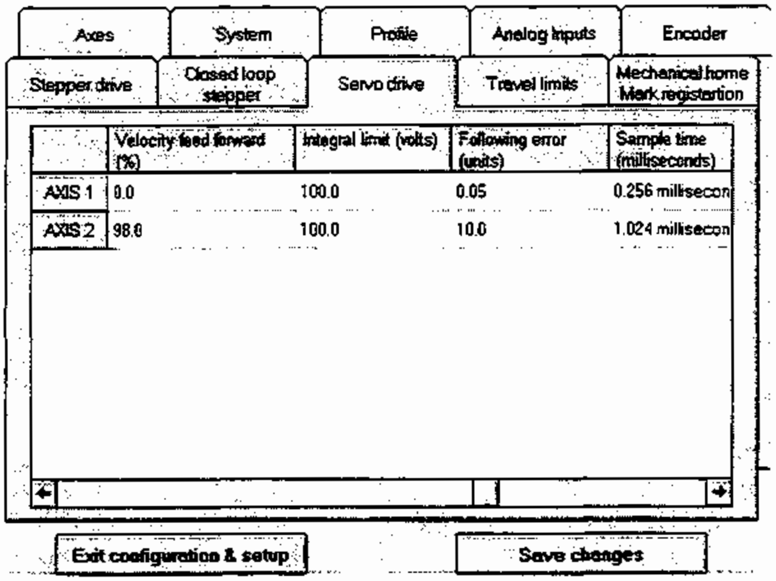
Command button	**Action**
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.30 Servo Drive Folder Scrolled to Left



	Proportional gain (millivolts/count)	Integral gain (msec)	Derivative gain (msec)	Accel feed forward (volts/count/msec^2)
AXIS 1	0.0	0.0	10.0	0.0
AXIS 2	20.9588	18.0467	9.0234	0.8137

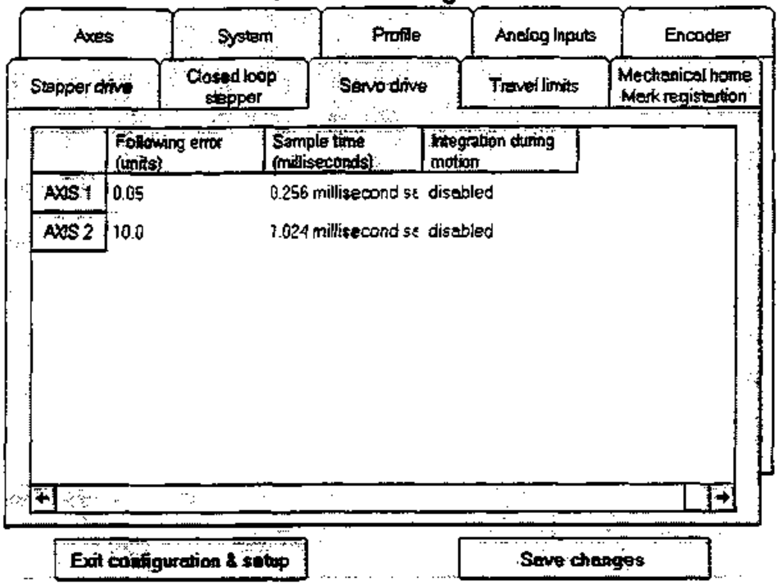
Figure 6.31 Servo Drive Folder Scrolled to Center



	Velocity feed forward (%)	Integral limit (volts)	Following error (units)	Sample time (milliseconds)
AXIS 1	0.0	100.0	0.05	0.256 millisecon
AXIS 2	98.0	100.0	10.0	1.024 millisecon

Note: See Section 6.3.5 for instructions on Autotuning to set the servo gains for each axis.

Figure 6.32 Servo Drive Folder
Scrolled to Right



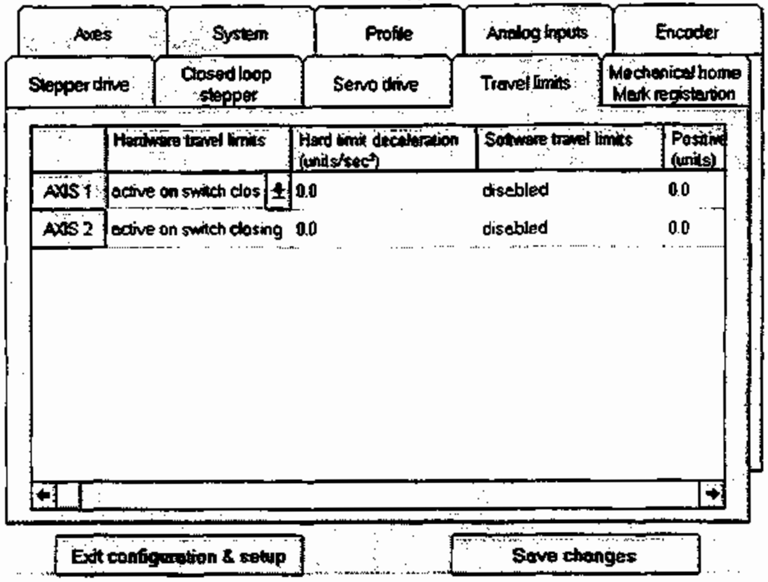
	Following error (units)	Sample time (milliseconds)	Integration during motion
AXIS 1	0.05	0.256 millisecond se	disabled
AXIS 2	10.0	1.024 millisecond se	disabled

Text box	**Action**
Proportional gain (Kp)	Specifies the proportional gain of a servo axis.
Integral gain (Ki)	Specifies the Integral gain of a servo axis.
Derivative gain (Kd)	Specifies the Derivative gain of a servo axis.
Accel feed forward (Kaff)	Specifies the Acceleration feed forward gain of a servo axis.
Velocity feed forward (Kvff)	Specifies the Velocity feed forward gain of a servo axis.
Integral limit	Specifies the Integral limit of a servo axis.
Following error	Specifies the Following error of a servo axis.
Sample time	Specifies the servo sample time of an axis.

Drop List	**Action**
Integration during motion	Enables or disable Integration error of a servo axis during motion.

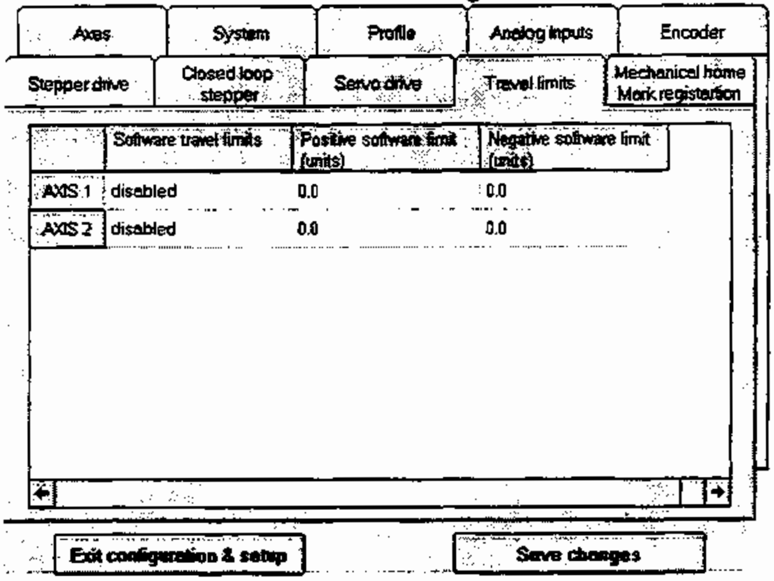
Command button	**Action**
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.33 Travel Limit Folder Scrolled to Left



	Hardware travel limits	Hard limit deceleration (units/sec^2)	Software travel limits	Positive (units)
AXIS 1	active on switch clos ±	0.0	disabled	0.0
AXIS 2	active on switch closing	0.0	disabled	0.0

Figure 6.34 Travel Limit Folder Scrolled to Right



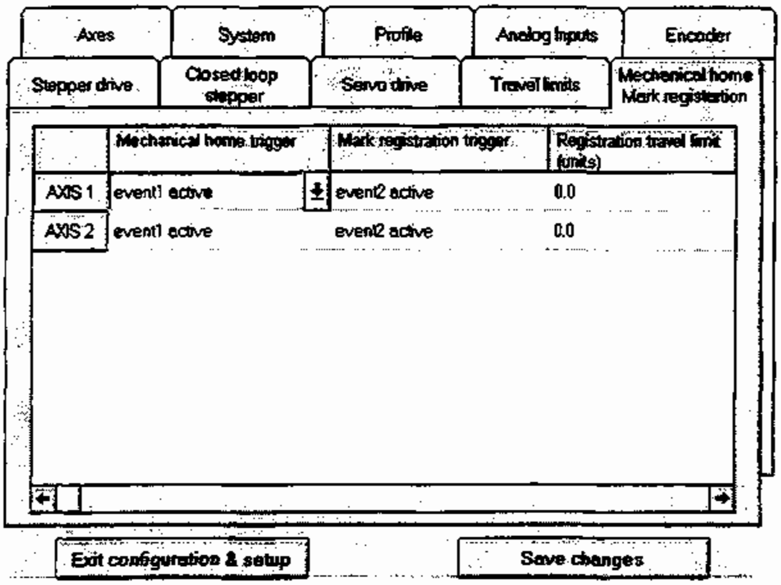
	Software travel limits	Positive software limit (units)	Negative software limit (units)
AXIS 1	disabled	0.0	0.0
AXIS 2	disabled	0.0	0.0

Drop List	**Action**
Hardware travel limit	Specifies the hardware travel limits of an axis. The choices are: disabled, active on switch closing and active on switch opening.
Software travel limit	Enables or disables the software limits of an axis.

Text box	**Action**
Hard limit deceleration	Specifies the axis deceleration rate when a HARDLIMIT is activated. A 0.0 entry specifies that the axis will stop immediately.
Positive software Limit	Specifies the direction and position for a positive motion.
Negative software Limit	Specifies the direction and position for a negative motion.

Command button	**Action**
Save changes	Saves the folder changes.
Exit configuration & setup	Exits configuration & setup.

Figure 6.35 Mechanical Home & Mark Registration Folder



	Mechanical home trigger	Mark registration trigger	Registration travel limit (units)
AXIS 1	event1 active	event2 active	0.0
AXIS 2	event1 active	event2 active	0.0

Drop List — **Action**

Mechanical home trigger — Specifies the mechanical home trigger for each axis. The choices are: event 1 active, event 1 inactive, event 1 active & encoder marker pulse, event 1 inactive & encoder marker pulse, event 2 active, event 2 inactive, encoder marker pulse active and encoder pulse inactive.

Mark registration trigger — Specifies the mark registration trigger for each axis. The choices are: event 1 active, event 1 inactive, event 1 active & encoder marker pulse, event 1 inactive & encoder marker pulse, event 2 active, event 2 inactive, encoder marker pulse active and encoder pulse inactive.

Text box — **Action**

Registration travel limit — Specifies the maximum distance allowed during a Mark registration cycle for an axis. A 0.0 indicates that there is not travel limit.

Command button — **Action**

Save changes — Saves the folder changes.

Exit configuration & setup — Exits configuration & setup.

6.3.5 - Servo Tuning

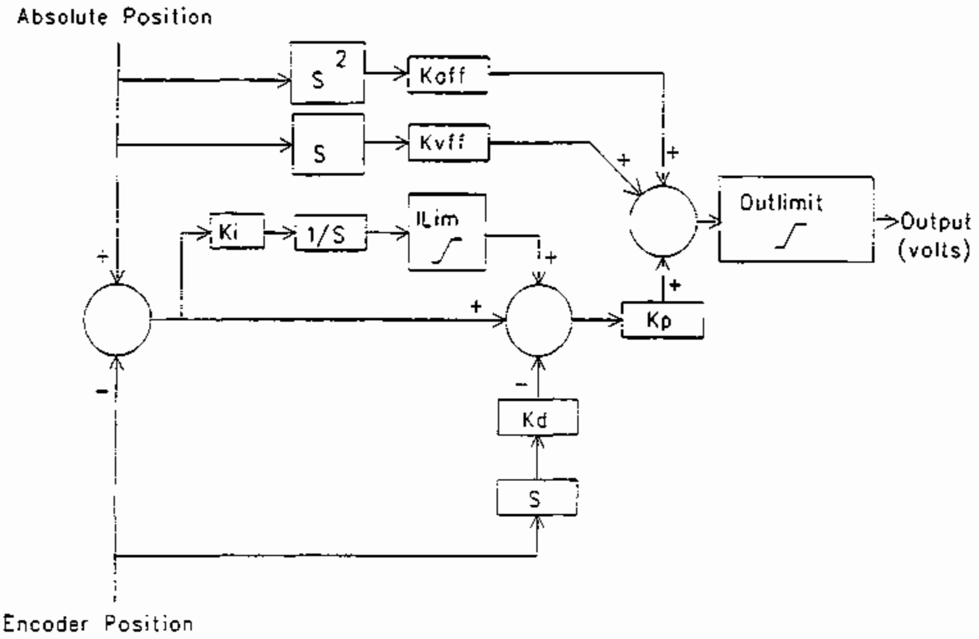
Tuning a Servo Axis:

What is a Servo?

A servo is a closed loop system. The loop is closed by taking a measurement of the actual output (usually a position or velocity), and comparing it to the desired command or reference input. An error signal is generated by subtracting the output signal from the reference. The error signal tells the controller how far away the output is from the desired position. Then, a control law (algorithm) modifies this error signal to provide an output voltage to drive the servo amplifier.

The MX2000 uses a modified form of the classical PID (Proportional, Integral, Derivative) control law with velocity and acceleration feed forward (picture below).

MX2000 Control Law



What is tuning?

Tuning is a process of determining the PID and feed forward gains to get the desired system response. Typical performance indicators like; overshoot, response time, stiffness, settling time, bandwidth and damping can all be used to measure how well the system is tuned. Tuning the gains to improve one performance characteristic may cause another characteristic to get worse.

Tuning a Servo:

Servo Tuning setup:

The initial servo parameters for each axis must be entered in the current project.

System Folder

The desired units per motor revolution value must be entered.

Encoder Folder

The encoder line count value must be entered.

Servo Drive Folder

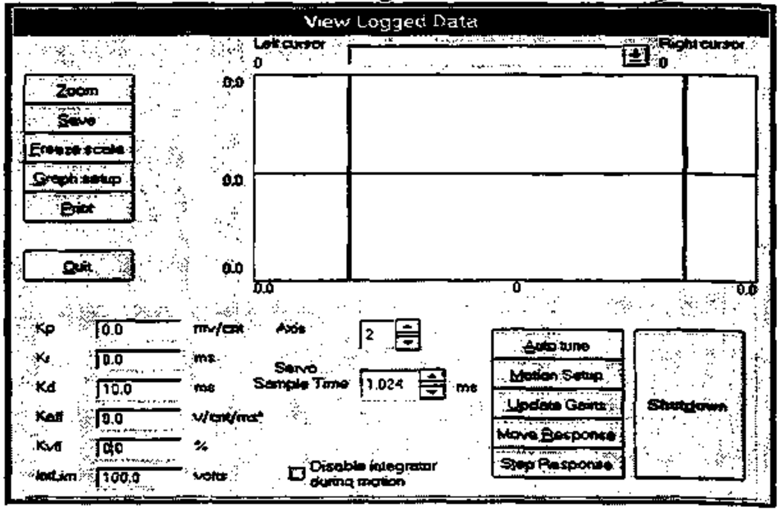
Proportional gain=0
Integral gain=0
Derivative gain=10
Acceleration feed forward=0
Velocity feed forward=0
Following Error set to the desired value.

The Project needs to be compiled and then Downloaded to the MX2000 before proceeding.

Servo Tuning:

The Servo Tuning Main screen is accessed by selecting the **Servo Tuning** item on the **Utility** menu or clicking on the **Servo Tuning** command button. Become familiar with the commands and action described below before proceeding.

Servo Tuning Main Screen



Display Drop List

View Logged Data

Left cursor 0

Right cursor 0

Kp 0.0 mv/cnt
Ki 0.0 ms
Kd 10.0 ms
Kaff 0.0 v/cnt/ms²
Kvff 0.0 %
IntLim 100.0 volts

Axis 2

Servo Sample Time 1.024 ms

□ Disable integrator during motion

Command Button	Action
Zoom	Zooms the display between cursors.
Save	Saves the current display.
Freeze scale	Freezes the current logged scaled values.
unFreeze	Allows the next commanded motion values to be auto scaled.
Graph setup	Allows selecting colors and style for each logged item.
Print	Prints the current screen.
Quit	Exits the Servo Tuning Screen.

Drop List	Action
Display Drop List	Selects the logged item to be displayed.

Text Box	Action
Kp	Displays the present value of proportional gain.
Ki	Displays the present value of integral gain.
Kd	Displays the present value of derivative gain.
Kaff	Displays the present value of acceleration feed forward gain.
Kvff	Displays the present value of velocity feed forward gain.
IntLim	Displays the present value of integral limit.

Spin Controller	Action
Axis	Allows the selection of an axis.
Servo Sample Time	Allows for the selection of the servo sample time.

Check Box	Action
Disable Integrator during motion	Enables or disable the integrator during motion.

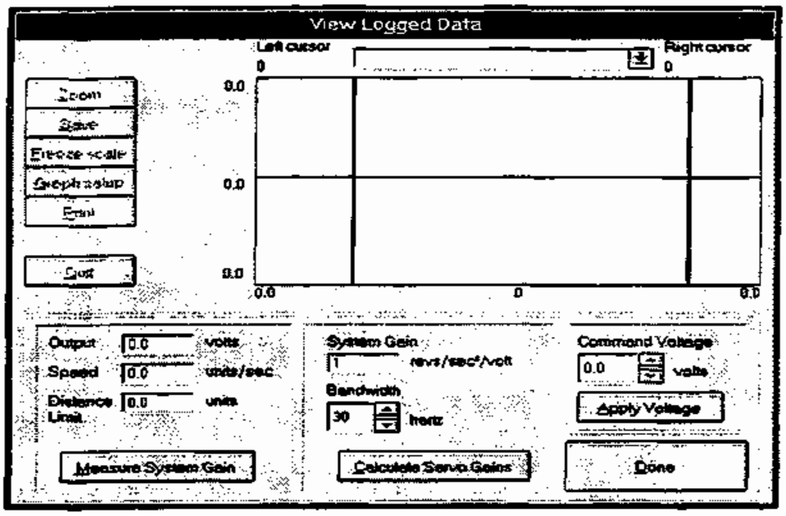
Command Button	Action
Auto tune	Selects auto tuning.
Motion Setup	Allows the setting of motion parameters.
Update Gains	Sends the present servo gains to the MX2000.
Move Response	Creates a logged move cycle.
Step Response	Creates a logged Step Response cycle.
Shutdown	Disables the servo output voltage.

The first step is to select the servo axis by clicking on the Axis spin controller until the desired axis is selected and then perform an Auto Tune for each servo axis.

Auto Tuning:

The Auto Tuning screen is accessed by clicking on the **Auto tune** command button.

Auto Tune Screen



View Logged Data

Left cursor 0

Right cursor 0

Output 0.0 volts

Speed 0.0 units/sec

Distance Limit 0.0 units

System Gain 1 revs/sec²/volt

Bandwidth 30 hertz

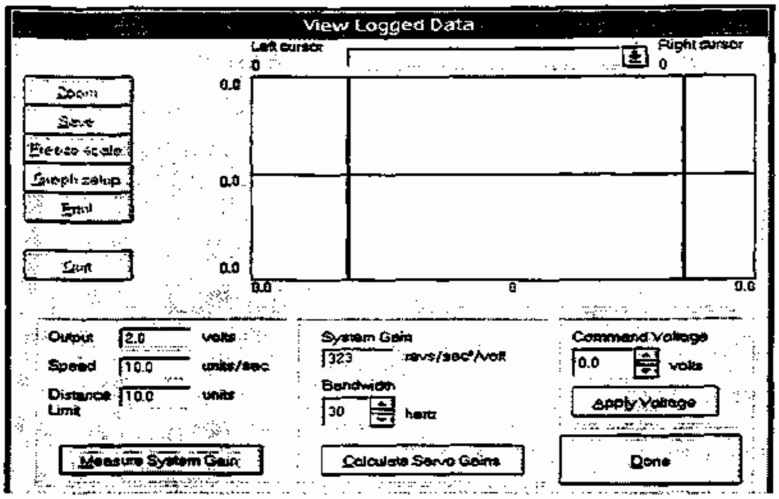
Command Voltage 0.0 volts

The **Output** text box, **Speed** text box, **Distance Limit** text box and **Measure System Gain** command buttons are used to obtain the System Gain of the selected servo axis. The **Output** text box is used to select the commanded servo output voltage for the Gain test. The range for the output voltage is -5 volts to +5 volts with the sign determining the direction of motion. A value must be entered in the Output text box; **2 volts is a good starting point**. The **Speed** text box is used to select the target velocity for the servo Gain test. A positive value must be entered; a good starting value is the maximum speed that will be allowed during motion. The **Distance Limit** text box limits the incremental distance of travel for the servo gain test. A value must be entered in this text box. Use a value that will move your system a safe distance, without exceeding any travel limits.

Clicking on the **Measure System Gain** command button starts the system Gain test. The results are displayed in the **System Gain** text box at the conclusion of the test. The system gain test creates a motion in the specified direction and is terminated if a hardware limit is activated, Distance Limit is exceeded, or the servo is unable to attain the selected speed in 1 second.

For manual motion, the **Command Voltage** can be used to move the servo motor. The range of the command voltage is -5 volts to +5 volts with the sign determining the direction of motion. Click on the **Apply Voltage** command button and slowly adjust the command voltage by clicking on the Command Voltage spin controller, the increment for each click is .05 volts, until motion starts. Click on the **Remove Voltage** command button when the desired position is attained.

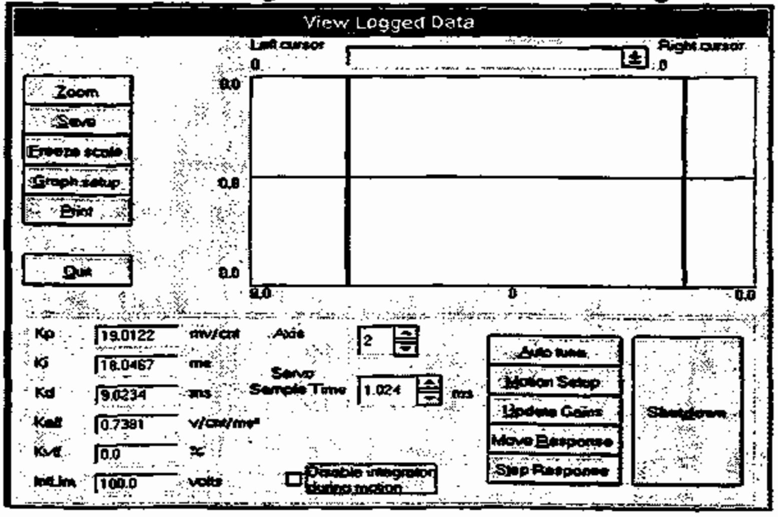
Auto Tune screen after System Gain test



Output 2.0 volts
Speed 10.0 units/sec
Distance Limit 10.0 units
System Gain 323 revs/sec²/volt
Bandwidth 30 hertz
Command Voltage 0.0 volts

The **System Gain** text box, **Bandwidth** spin controller and **Calculate Servo Gains** command button are used to calculate the Kp, Ki , Kd and Kaff gains for the servo axis. The **System Gain** text box can be altered if desired. The **Bandwidth** spin controller selects the servo bandwidth for the gain calculation. The range is 1 hertz to 100 hertz with a default of 30 hertz; **Note: that 30 hertz is a good starting point for tuning most systems.** The **Calculate Servo Gains** command button starts the PID gains calculation and sets the project PID gains for the selected axis to the calculated values. After the calculation the main tuning screen with the new PID gains is viewed. **Note: that the Kvff value is 0% when auto tuning is completed.** Proceed to the **Move Response** section.

Servo Tuning Main Screen after Auto Tuning



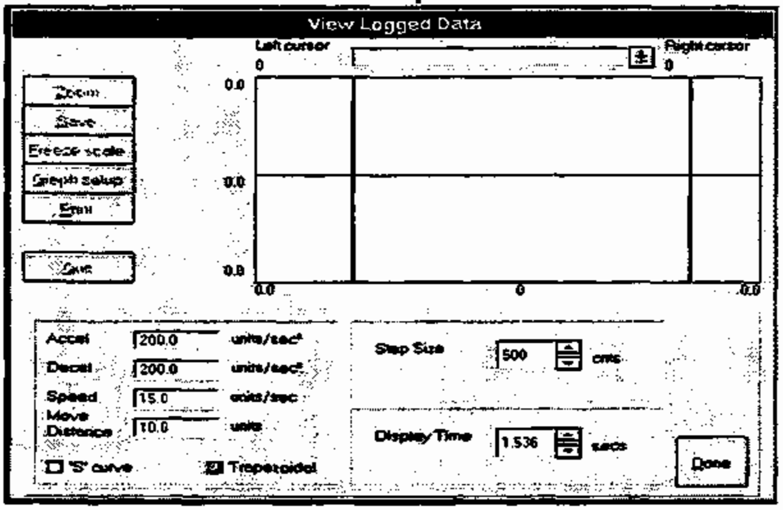
Kp 19.0122 mv/cnt
Ki 18.0467 ms
Kd 9.0234 ms
Kaff 0.7381 v/cnt/ms²
Kvff 0.0 %
IntLim 100.0 volts
Axis 2
Servo Sample Time 1.024 ms
□ Disable integrator during motion

The **Done** command button is used to exit the auto tuning screen without calculating the PID gains.

Move Response:

Move response allows "fine-tuning" of your servo system while making a typical move. This allows The servo axis move parameters are selected by clicking on the **Motion Setup** command button. This allows the **Acceleration, Deceleration, Speed, Profile, Move Distance, Motion Profile** and **Display time** to be modified for the move response. Enter the desired Acceleration, Deceleration, Speed and Move Distance for the move. Set Display Time to 1.536 seconds. When complete click on the **Done** command button to return to the main servo tuning screen.

Motion Setup Screen



Left cursor 0

Right cursor 0

Accel 200.0 units/sec²

Decel 200.0 units/sec²

Speed 15.0 units/sec

Move Distance 10.0 units

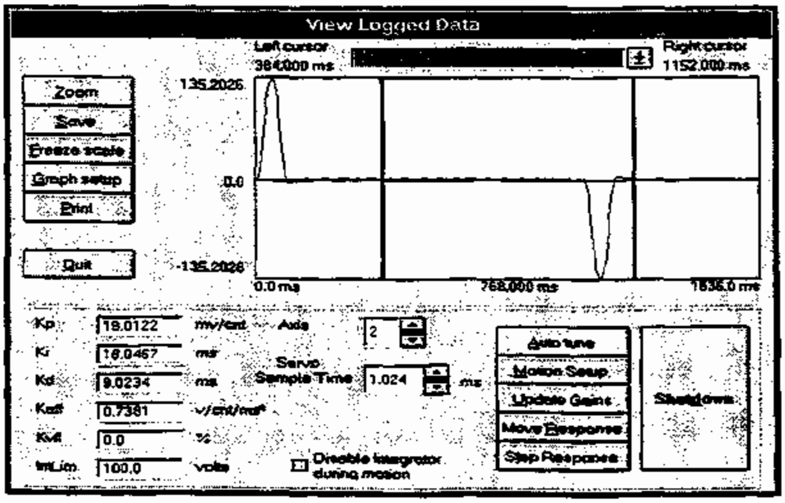
□ 'S' curve ☒ Trapezoidal

Step Size 500 cnts

Display Time 1.536 secs

To perform the move click on the **Move Response** command button and wait for the data logging session to complete. The servo axis Position Error will be displayed. To view other logged items click on the Display Drop List and select the desired item.

Move Response with Kvff=0%
Max Error= 135.2 counts



Left cursor 384.000 ms

Right cursor 1152.000 ms

Kp 19.0122 mv/cnt

Ki 16.0467 ms

Kd 9.0234 ms

Kaff 0.7381 v/cnt/ms²

Kvff 0.0 %

IntLim 100.0 volts

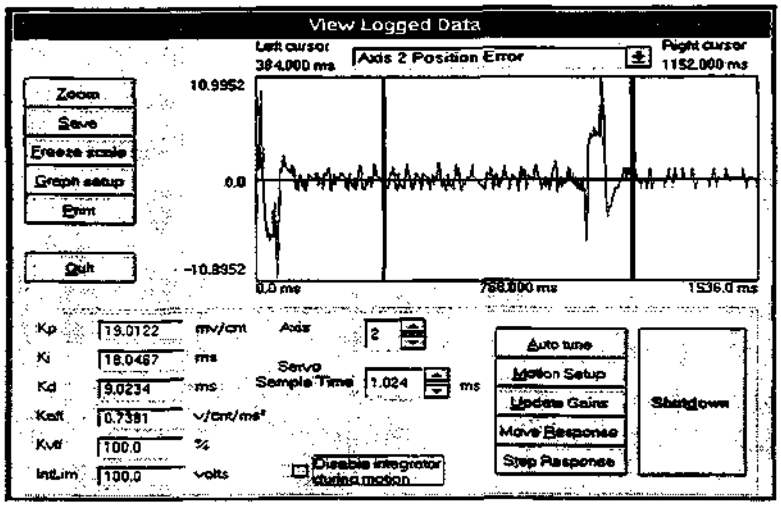
Axis 2

Servo Sample Time 1.024 ms

□ Disable integrator during motion

Set the Kvff value to 100% and perform another Move Response cycle.

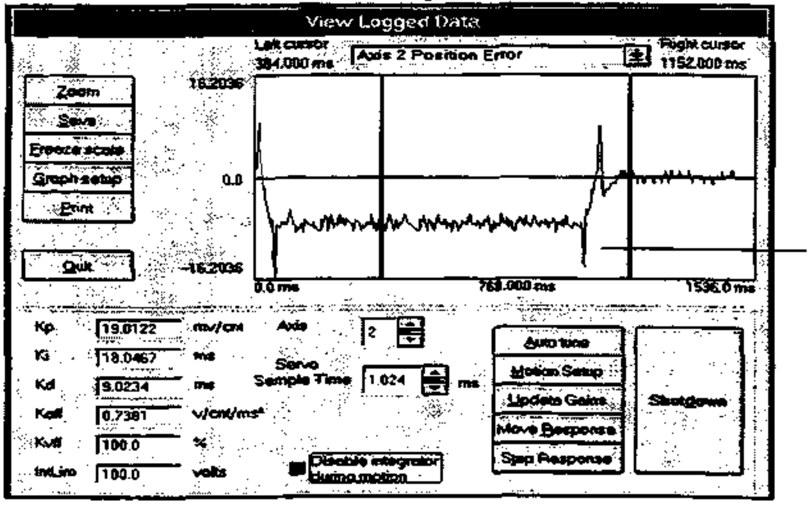
Move Response with Kvff=100%
Max Error=10.99 counts



Kp 19.0122 mv/cnt
Ki 18.0467 ms
Kd 9.0234 ms
Kaff 0.7381 v/cnt/ms²
Kvff 100.0 %
IntLim 100.0 volts
Axis 2
Servo Sample Time 1.024 ms
□ Disable integrator during motion

Click on **Disable integrator during motion** check box (there should be a check in the box), then perform another Move Response cycle.

Move Response Integrator disabled during motion
Kvff is too high

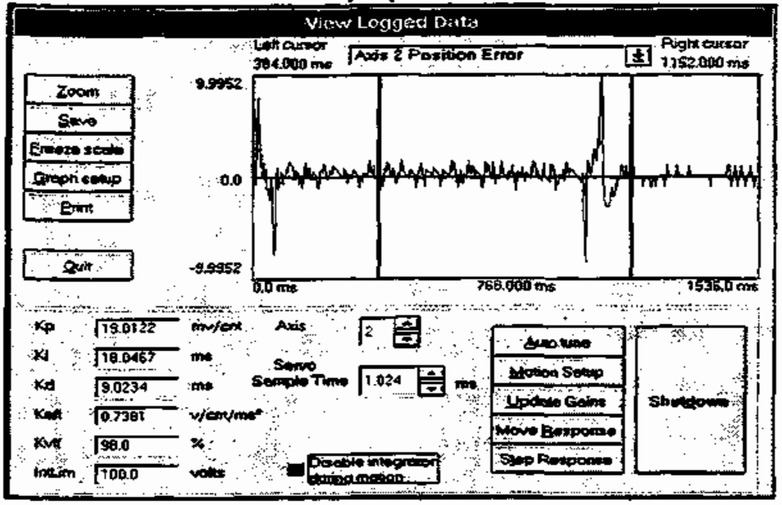


Kp 19.0122 mv/cnt
Ki 18.0467 ms
Kd 9.0234 ms
Kaff 0.7381 v/cnt/ms²
Kvff 100.0 %
IntLim 100.0 volts
Axis 2
Servo Sample Time 1.024 ms
■ Disable integrator during motion

Error below 0.0 line

If the error is above the 0.0 line Kvff must be raised. In this case the error is below the line, so decrease Kvff by 2 % and perform another Move Response cycle. Repeat this procedure until the error is close to the 0.0 line.

Move Response Integrator disabled during motion
Kvff is proper value


View Logged Data

Left cursor 384.000 ms

Axis 2 Position Error

Right cursor 1152.000 ms

Kp 19.0122 mv/cnt

Ki 18.0467 ms

Kd 9.0234 ms

Kaff 0.7381 v/cnt/ms²

Kvff 98.0 %

IntLim 100.0 volts

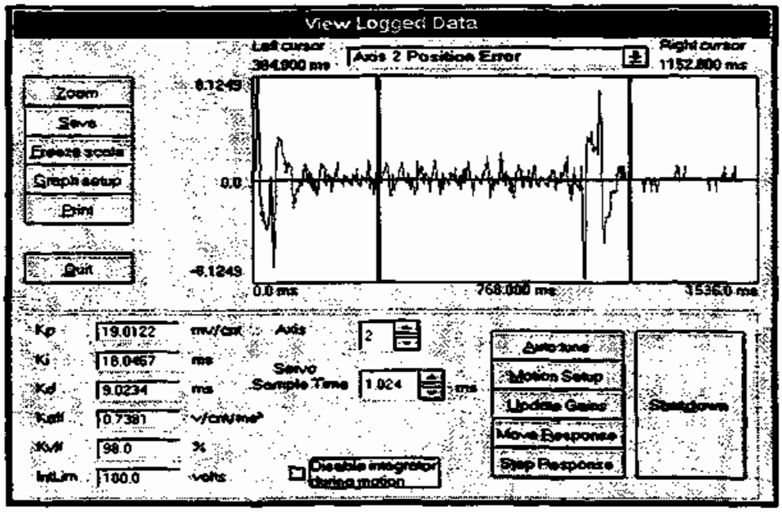
Axis 2

Servo Sample Time 1.024 ms

■ Disable integrator during motion

Click on **Disable integrator during motion** check box (there should be no check in the box), then perform another move response.

Tuned Servo Screen


View Logged Data

Left cursor 384.000 ms

Axis 2 Position Error

Right cursor 1152.800 ms

Kp 19.0122 mv/cnt

Ki 18.0467 ms

Kd 9.0234 ms

Kaff 0.7381 v/cnt/ms²

Kvff 98.0 %

IntLim 100.0 volts

Axis 2

Servo Sample Time 1.024 ms

□ Disable integrator during motion

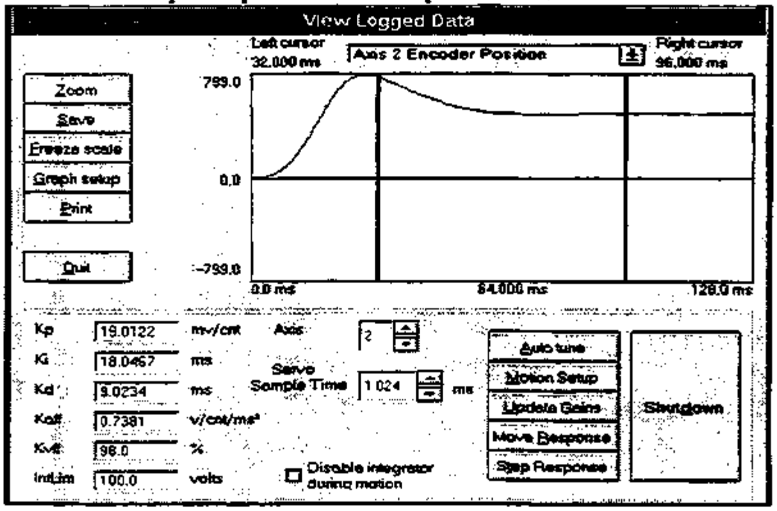
The servo axis is now tuned and you have the option of disabling the integrator during motion.

When all the servo axes have been tuned, compile the current project and download the project to the MX2000. This will save all the new servo values and transfer the project to the MX2000.

Step Response: (optional)

The Step Response, instantaneous error response, of the servo axis can be displayed by clicking on the **Step Response** command button. The instantaneous error can be chosen on the Motion Setup Screen and its value is in encoder counts. 10% overshoot from the commanded value is a good setting for the INTLIM.

Step Response with step size=500 counts



Left cursor 32.000 ms

Axis 2 Encoder Position

Right cursor 96.000 ms

Kp 19.0122 mv/cnt

Ki 18.0467 ms

Kd 9.0234 ms

Kaff 0.7381 v/cnt/ms²

Kvff 98.0 %

IntLim 100.0 volts

Axis 2

Servo Sample Time 1.024 ms

□ Disable integrator during motion

Please Note: **It is nearly impossible, in practical systems, to completely eliminate overshoot using step response in this fashion. Best system performance will be obtained by using the previously-described Move Response method.**

SPECIAL PROGRAMMING NOTES FOR CLOSED-LOOP STEPPER OPERATION

The parameters for closed loop are set during the project configuration of a user's program. These parameters are:

Encoder resolution
: Number of lines the encoder has. The line count times four is the equivalent of encoder pulses/rev. The direction for this parameter controls the quadrature detection direction value.

Encoder position error (units):
: Allowable error at standstill before correction is required.

Encoder following error (units):
: Allowable error during motion before an error is reported. **This value should be a minimum of 1/20 of a motor revolution.**

Number of correction attempts allowed:
: How many consecutive correction cycles are allowed.

Time between correction attempts (seconds):
: Time between correction attempts.

Encoder Mode
: Selects the operation mode for closed loop (see below). This can be controlled during program execution using the ENCMODE command.

The Encoder Modes are described below:

ENCMODE=0 Closed loop operation is disabled.

ENCMODE=1

Position maintenance about a quiescent position is enabled.

If a following error is detected:

1. All motion associated with this task is stopped.
2. The error trap handler routine is called; the task may end or continue, depending upon how the user's error handler routine is written.

ENCMODE=2

Position maintenance about a quiescent position is enabled.

If following error is detected during a MOVE, MOVEHOME or MOVEREG cycle:

1. Motion is stopped for this axis.
2. Motion is restarted after the closed loop timeout has occurred.

If a following error is detected during a JOG or coordinated motion (LINE, ARC, or PATH) cycle:

1. All motion associated with this task is stopped.
2. The error trap handler routine is called; the task may end or continue, depending upon how the user's error handler routine is written.

ENCMODE=3

Position maintenance about a quiescent position is enabled.

If following error is detected during a JOG, MOVEHOME or MOVEREG cycle:

1. Motion is stopped for this axis.
2. Motion is restarted after the closed loop timeout has occurred.

If a following error is detected during a coordinated motion (LINE, ARC, or PATH) cycle:

1. All motion associated with this task is stopped.
2. The error trap handler routine is called; the task may end or continue, depending upon how the user's error handler routine is written.

If following error is detected during MOVE cycle:

1. Motion is stopped for this axis.
2. Motion is restarted in the opposite direction after the closed loop timeout has occurred. The target position is the starting position of the MOVE. When the starting position is reached, a move to the initial target is initiated.

SPECIAL PROGRAMMING NOTES FOR FOLLOWING OPERATION

The MX2000 axes can be programmed to follow an **encoder, pulse and direction inputs**, or **Analog voltage**. The encoder or pulse and direction source for the master axis requires that the axis be set up as a closed loop stepper with Position Verification disabled. To select these options, use the Program Configuration menu. The two basic modes of operation are **Velocity following** (FOLMODE=0 or 1) or **Velocity and Position following** (FOLMODE= 2 or 3).

Theory of Operation
The Master Velocity is attained for a time slice and is added to the Master Position. If a motion request is pending, MOVE or JOGSTART command, the FOLTRIG value is tested for the trigger requirement. If the trigger condition exists, the specified motion for an axis begins and sets the following variables: Following Target=0, Following Current=0, Following Offset=0, motion request=0, Accelerate to Master flag=1. The Master Position is set to 0 if the FOLTRIG value is 0-2. If the FOLTRIG value is 3, the new Master Position is set to the difference between the old Master Position and the FOLDIST value.

When a JOGSTART cycle is taking place and a JOGSTOP request exists, the axis will decelerate to a stop and clear the JOGSTOP request.

When the ACCEL to Master flag=1, the following axis will accelerate until the Following Target Velocity is achieved. Once the target velocity is attained, the ACCEL to Master flag is cleared. If the Velocity Following mode is selected, the Master Position and the Following Target are set to 0.

To obtain a following sync for an axis, the Following Current value must be equal to the Sum of the Following Offset and Following Target values. An axis is commanded to accelerate or decelerate if not in following sync.

Definitions:

Master Velocity Defined in units/time. If Encoder or Pulse and Direction following is selected, the master velocity is the difference between the previous encoder value and the new encoder value. If analog following is selected, the master velocity is calculated as follows:
(((analog input / 10) x FOLDEVIATION) + VELOCITY) x time slice

Master Position Current master position in units. The Master Velocity is added to the Master Position in each time slice.

Following Target This is the current Target-position of a following axis. The target position of an axis for an offset of 0. The Current Target is equal to the Master Position times the FOLRATIO of each axis + following offset.

Following Offset The current offset value in units from the Following target Position to attain the true target position of an axis. This value is set to 0 when motion starts and is changed by the FOLOFFSET command.

Following Current This is the following axis' current position. This value is set to 0 when motion starts and is changed when the axis pulses are outputted.

Velocity Following:
The Following axis is only concerned with the master axis velocity. In this mode the following axis ramps at the programmed acceleration, ACCEL value for this axis, until the following axis velocity matches the master axis velocity times the FOLRATIO. This position is then established as a sync position of 0, which is used as a positional reference for the FOLOFFSET command. Thus a positional advance or recede can be commanded using the FOLOFFSET command. The two types of following motion that can be done are MOVE or JOGSTART.

Velocity and Position following:
The Following axis must maintain a specific positional relationship with the Master axis and match the master velocity. In this mode, the following axis ramps at the programmed acceleration, ACCEL value of the axis, up to the master axis speed. The target offset position is interrogated and either an advance cycle or a recede-cycle is initiated to get into the specific positional relationship with the Master axis. This position will be maintained until another offset command is issued or it is time to stop. Thus a positional advance or recede can be commanded using the FOLOFFSET command. The default for FOLOFFSET is 0. Two types of following motion can be done: MOVE or JOGSTART. The FOLRATIO should be set at 100%.

Selecting the master source:
The master axis following source of a task can be an **incremental encoder, pulse and direction input**, or an **analog voltage**. Use the Program Configuration to select either the encoder or pulse and direction input source and enable the function with the FOLENCODER command. The analog voltage source is enabled using the FOLANALOG command.

The master encoder source should not be a servo axis if the master axis is to travel in the same direction for an extended period. The Travel limit is 2,147,483,648 encoder pulses in one direction. The limit for a 1000 line encoder is approximately 536,870 revolutions. Motion must be stopped before the travel limit is reached and the ABSPOS of this axis should be set to zero before proceeding in the same direction.

Encoder or Pulse and Direction:
In the **Configuration** section of the MX2000 PC Program select the desired master axis and set it up as a **Closed loop stepping motor**. In the **Encoder Setup menu** select the Encoder type, Quadrature for encoder or Pulse and Direction. Enter the Encoder line count value. If using Pulse and Direction, this value should be expressed in pulses/revolution divided by 4. This value is used to generate an encoder scale factor that converts encoder pulses to units. In the **Closed Loop Setup menu,** select the Disabled function under the error action menu. The remaining Configuration parameters for this axis should be programmed also.

The scale factors for the master axis are a combination of the following parameters: motor steps per revolution, units per motor revolution, and encoder line count. The unit scale factor uses the motor steps per revolution and units per motor revolution. The encoder scale factor is calculated from the encoder line count and the unit scale factor.

Analog Voltage Following:
This mode uses an analog voltage for the master velocity deviation. The Nominal velocity is set by the VELOCITY command in units/sec and the velocity deviation for a 10-volt analog input is controlled by the FOLDEVIATION command. The velocity is linearly reduced by a negative analog voltage and linearly increased by a positive analog voltage.

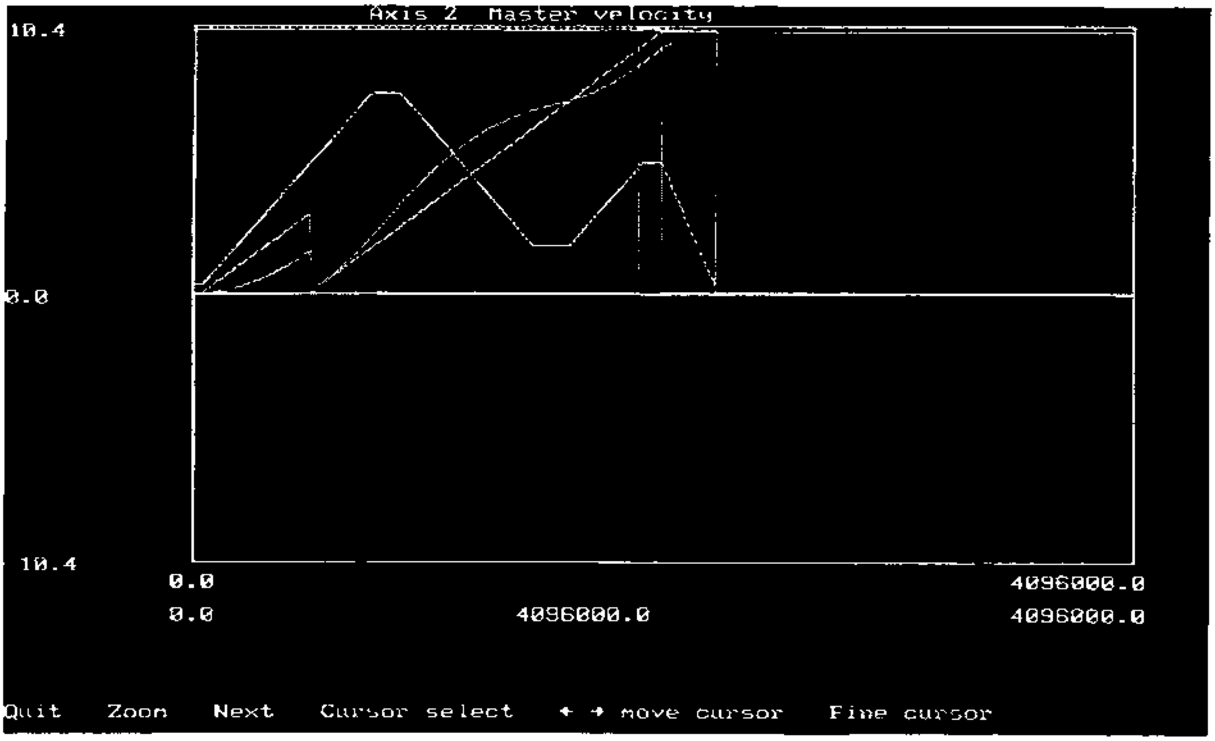
The unit scale factor uses the motor steps per revolution and units per motor revolution values that are set in the MX2000 PC Program Configuration section.

Examples of Following programs are provided below. The first is an example of an Encoder Following program the second is an example of an Analog Following program. Additional examples of encoder and analog following programs are given in Section 7 - Application Examples.

Encoder Following Programming Example

The encoder for axis 2 is the master axis and axis 1 is the following axis. The following axis will run at 50% of the master axis. Velocity following motion will be triggered by event 1.

```
ACCEL=10000                      ' axis 1 ACCEL 10000 units/sec/sec
DECEL=20000                      ' axis 1 DECEL 20000 units/sec/sec
LOWSPD=3000                      ‘ following starting speed
FOLRATIO=.5                      ' axis 1 50% following ratio
FOLMODE=0                        ' Velocity following in programmed direction
FOLENCODER=102,1                 ' enable encoder following
FOLTRIG=1                        ' axis 1 motion trigger is event 1 on axis 1
FOLMAXRATIO=1                    ' axis 1 advance cycle max velocity  is 100%
FOLMINRATIO=.1                   ' axis 1 recede-cycle  min velocity is 10%
JOGSTART(1)= 1                   ' axis 1 JOGSTART in +direction on event 1 trigger
DO: LOOP UNTIL FOLSYNC(1)=1      ' wait for velocity sync
FOLOFFSET=1000                   ' positional advance axis 1 1000 units
DO: LOOP UNTIL FOLSYNC(1)=1      ' wait for positional sync
FOLOFFSET= -500                  ' positional recede axis 1 1500 units
DO: LOOP UNTIL EXIN(101)=1       ' wait for expansion input
JOGSTOP(1)=0                     ' stop velocity following mode
FOLEND                           ' disables the Encoder following cycle
FOLENCODER=0,0                   ' end Encoder following mode
FOLEND
```

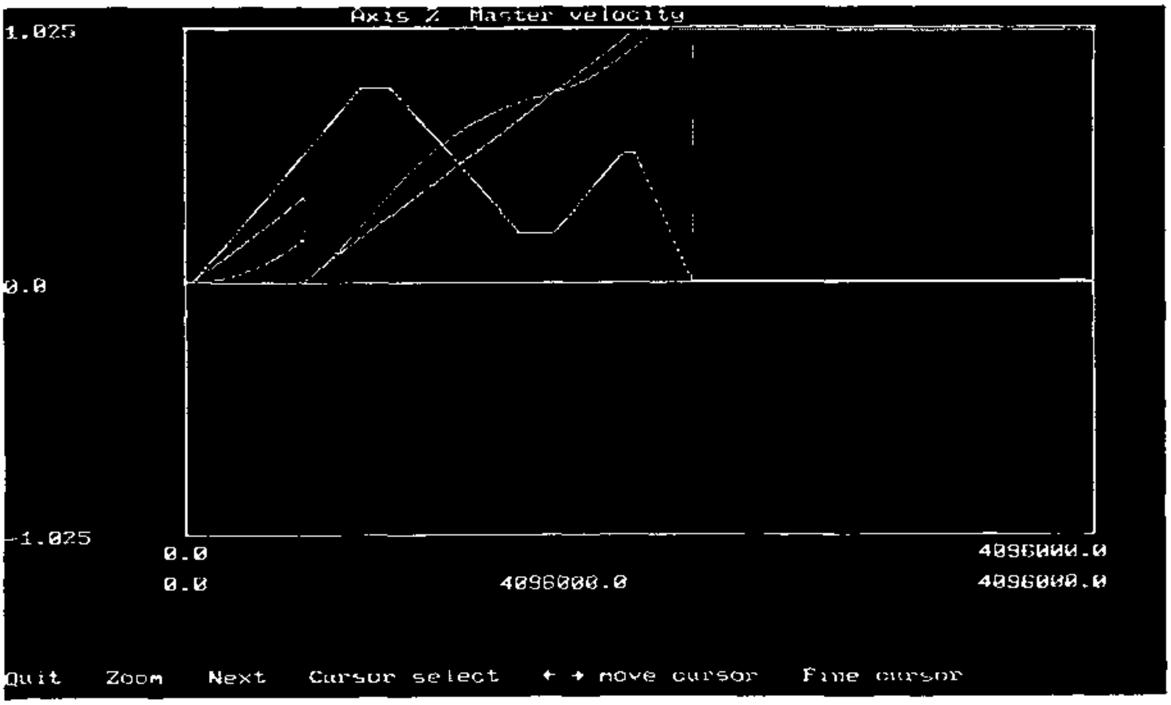


Signals:
Master Velocity
Following Velocity
Following Target Position
Following Position
Following Sync

Analog Voltage Following Example

The analog input for master axis velocity deviation is analog port 102 and axis 1 is the following axis. The nominal master axis velocity for a 0-volt analog input voltage is 1000 units/sec. The velocity deviation for the 10-volt analog input is 500 units/sec. The secondary axis will run at 50% of the master axis. Velocity following motion will be triggered by event 1.

```
LOWSPD=0                          ' following starting speed in units/sec
VELOCITY=1000                     ' velocity for the master axis is 1000 units/sec
ACCEL=1000                        ' axis 1 ACCEL 1000 units/sec/sec
DECEL=2000                        ' axis 1 DECEL 2000 units/sec/sec
FOLRATIO=.5                       ' axis 1 50% following ratio
FOLMODE=0                         ' Velocity following in programmed direction
FOLDEVIATION=500                  ' deviation for a 10-volt signal is 500 units/sec
FOLANALOG=102,1                   ' enable analog following
FOLTRIG=1                         ' axis 1 motion trigger is event 1 on axis 1
FOLMAXRATIO=1                     ' axis 1 advance cycle max velocity  is 100%
FOLMINRATIO=.1                    ' axis 1 recede-cycle  min velocity is 10%
JOGSTART(1)= -1                   ' axis 1 JOGSTART in - direction on trigger
DO: LOOP UNTIL FOLSYNC(1)=1       ' wait for velocity sync
FOLOFFSET=100                     ' positional advance axis 1 100 units
DO: LOOP UNTIL FOLSYNC(1)=1       ' wait for positional sync
FOLOFFSET= -50                    ' positional recede-axis 1 150 units
DO: LOOP UNTIL FOLSYNC(1)=1       ' wait for positional sync
DO: LOOP UNTIL EXIN(101)=1        ' wait for expansion input
FOLEND                            ' disables ANALOG following
FOLANALOG=0,0                     ' end analog following
FOLEND
```



Signals: Master Velocity
Following Velocity
Following Target Position
Following Position

SECTION 6.4

SOFTWARE REFERENCE GUIDE

6.4.1 Programming Commands

Programming Commands Grouped By Function

Motion

ABSPOS	Sets or returns the absolute position.
ACCEL	Sets or returns the acceleration rate in units/sec/sec.
ARC	Initiate a coordinated motion to move in an arc.
BUSY	Returns the motion status of the axis.
DECEL	Sets or returns the deceleration rate in units/sec/sec.
DIST	Returns the incremental distance moved.
DONE	Returns the motion status of the axis.
ENCERR	Returns the encoder error count.
ENCMODE	Sets or returns the encoder mode of operation.
ENCPOS	Returns the encoder absolute position.
EVENT1	Returns the state of an axis input.
EVENT2	Returns the state of an axis input.
FEEDRATE	Sets the feedrate override value for a path motion.
FOLANALOG	Selects the analog input and following axes for the task, enables Analog following.
FOLDEVIATION	Master deviation velocity of a task in units/sec for a 10-volt analog input voltage.
FOLDIST	Sets the axis triggering position, in units, for master position triggering.
FOLENCODER	Selects the encoder axis and the following axis for the task and enables Encoder or Pulse and Direction following.
FOLEND	This command ends the encoder following mode.
FOLMAXRATIO	Sets the axis velocity clamp for an advance cycle.
FOLMINRATIO	Sets the axis velocity clamp for a recede-cycle.
FOLMODE	Selects the task's following mode and direction.
FOLOFFSET	Sets the axis offset position in units from the initial sync position in velocity following or from the initial starting position in velocity and position following.
FOLRATIO	Sets the ratio of the master axis to the following axis.
FOLSYNC	Returns the following sync status of an axis.
FOLTRIG	Sets the trigger mode for starting a following axis in motion.
HARDLIMIT	Enables/disables or returns the axes limit switch enabled status.
HARDLIMNEG	Returns the status of the axis input LIM-.
HARDLIMPOS	Returns the status of the axis input LIM+.
JOGSTART	Run continuously in the specified direction.
...JOGSTOP	Stop continuous run.
JOYSTICK	
..JOYSTICK END	Enables/disables Joystick motion.
LINE	Initiates a coordinated motion to move in a straight line.
LOWSPD	Sets or returns the low speed (starting speed).
MAXSPD	Sets or returns the maximum allowed speed.
MOVE	Initiates an indexed move.
MOVEHOME	Run until the home input is activated.
MOVEREG	Run until the registration input is activated, then move the specified distance.

PATH..PATH END	Begin a continuous motion path.
POINT	Specify coordinates, which the motor will move through in a path.
POSMODE	Sets or returns the positioning mode.
PROFILE	Sets or returns the acceleration/deceleration profile.
RADIUS	Sets the arc radius for Path blending
SOFTLIMIT	Enable/disable or returns SOFTLIMIT enabled state of an axis.
SOFTLIMNEG	Sets or returns the absolute negative travel limit position.
SOFTLIMPOS	Sets or returns the absolute positive travel limit position.
SPEED	Sets or returns the an individual axes target speed.
VELOCITY	Sets or returns the target velocity for coordinated motion.
WAITDONE	Waits for motion to be done for the specified axes.

Motor Control

BOOST	Enables or disables the motor boost current feature, or returns status thereof.
REDUCE	Enables or disables the reduce current feature, or returns status thereof.
WNDGS	Enables or disables the motor winding current, or returns status thereof.

Input-Output Control

ANALOG	Sets or returns a numeric value representing the voltage on the analog port.
BCD	Returns the value on the BCD port.
DRVREADY	Enables or disables checking of the drive ready (READY) signal on the axis board.
EXIN	Returns the state of the inputs on the expansion board.
EXOUT	Sets, resets, or returns the state of the expansion port outputs.
IN	Returns the state of the inputs on the I/O board.
OUT	Sets, resets, or returns the state of the outputs on the I/O board.
OUTLIMIT	Sets or returns the maximum analog output voltage allowed on a servo axis.

Mathematics

ABS	Returns the absolute value of an expression.
LOG	Returns the natural logarithm of x.
MOD	Returns the divide remainder
SIGN	Returns the sign of the expression.
SQRT	Returns the square root.

Trigonometry

ATN	Returns the arctangent (2 quadrant) of a value.
ATN2	Returns the arctangent (4 quadrant) of the relative 2 values.
COS	Returns the cosine of the angle.
SIN	Returns the sine of the angle.
TAN	Returns the tangent of the angle.

Boolean Logic

AND	Logical conjunction operator.
NOT	Logical complement operator.
OR	Logical inclusive or operator.
XOR	Logical exclusive or operator.
&	Bitwise AND operator
|	Bitwise inclusive or operator
^	Bitwise exclusive or operator
~	Bitwise complement operator
>>	Bitwise shift bits right
<<	Bitwise shift bits left

Timing Functions

TIMER	Sets or reads Task Timer
WAIT	Wait for the period of time to expire.

String Manipulation

ASC	Returns the ASCII code of character.
CHR$	Returns a one character string for the given ASCII code.
FORMAT	Enables or disables the formatting of the STR$ returned string.
GETCHR	Wait for a character to be received via the serial port.
HEX$	Returns the HEX character equivalent of the argument.
HVAL	Returns the hex value of a string.
INCHAR	Returns a character from the serial port.
INPUT	Reads data from the selected serial port.
INSTR	Returns the first occurrence of a character in a string.
LCASE$	Converts a string to lower case letters.
LEFT$	Returns the leftmost characters of a string
LEN	Returns the number of characters in the string.
MID$	Returns the designated middle number of characters of a string.
PRINT	Transmit data to the selected serial port.
RIGHT$	Returns the right most characters from a string.
STR$	Returns a string representation of a numeric expression
STRING$	Returns a string of characters
UCASE$	Converts a string to upper case letters.
VAL	Returns the value of a string.

Program Flow Control

DO...LOOP	Begin a repeatable a block of statements.
END	End of program
FOR..NEXT...STEP ...EXIT FOR	Begin a repeatable block of statements.
GOSUB...Returns	Branch to a subroutine and returns.
GOTO	Branch unconditionally to the specified label.
IF..THEN ..ELSE..END IF	Begin a conditional block of statements.

Miscellaneous

COMMON	Defines common variables to be shared between tasks.
DATA	Define numeric values.
#DEFINE	Defines a symbolic name to be a particular string of characters.
DIM	Defines an array.
#INCLUDE	Includes a file name in a user's task.
ERR	Returns the error status number for the task.
NVR	Non volatile storage of a variable (1-2048).
READ	Reads values from data statements into variables or arrays.
REM, '	Remark
RESET	Initializes System to power on condition.
RESTORE	Restores data list pointer to beginning.
SETCOM	Sets the baud rate and data format of the AUX serial port.
SHIFT	Shifts an array
WARNING	Sets or returns task warning count

Servo Gains and Limits

INTLIM	Sets or returns the servo axis integral limit.
KAFF	Sets or returns the servo axis acceleration feed forward gain.
KD	Sets or returns the servo axis derivative gain.
KI	Sets or returns the servo axis integral gain.
KP	Sets or returns the servo axis proportional gain.
KVFF	Sets or returns the velocity feed forward gain value for a servo axis.

Programming Command Summary (alphabetical list)

&	Bitwise AND operator
|	Bitwise inclusive or operator
^	Bitwise exclusive or operator
~	Bitwise complement operator
>>	Bitwise shift right
<<	Bitwise shift left
A	
ABS	Returns the absolute value of an expression.
ABSPOS	Sets or returns the absolute position of an axis.
ACCEL	Sets or returns the acceleration rate of an axis in units/sec/sec.
ANALOG	Sets or returns a numeric value representing the voltage on the analog port.
AND	Logical "conjunction" operator.
ARC	Initiate a coordinated motion to move in an arc.
ASC	Returns the ASCII code of a character.
ATN	Returns the arctangent of a value.
ATN2	Returns the arctangent of the value y/x.
B	
BCD	Returns the value on the BCD switches connected to an expansion port.
BOOST	Enables or disables the motor boost current feature.
BUSY	Returns the motion status of an axis.
C	
CHR$	Returns a one-character string for the given ASCII code.
COMMON	Defines common variables to be shared between tasks.
COS	Returns the cosine of the angle.
D	
DATA	Defines numeric values (used with READ).
DECEL	Sets or returns the deceleration rate of an axis in units/sec/sec.
#DEFINE	Defines a symbolic name to be a particular string of characters.
DIM	Defines an array.
DIST	Returns the incremental distance moved.
DO...LOOP	Begins a repeatable a block of statements.
DONE	Returns the motion status of an axis.
DRVREADY	Enables or disables checking of the drive ready (READY) signal on the axis board.
E	
ENCERR	Returns the encoder error count.
ENCMODE	Sets or returns the encoder mode of operation.
ENCPOS	Returns the encoder absolute position.
END	End of program
ERR	Returns the MX controller error status number for this task.
EVENT1	Returns the EVENT1 hardware state of an axis.
EVENT2	Returns the EVENT2 hardware state of an axis.
EXIN	Returns the state of the inputs on an expansion board.
EXOUT	Sets or returns the state of the outputs on an expansion board.

F	
FEEDRATE	Sets the feedrate override value for a path motion.
FOLANALOG	Selects the analog input and following axes for the task, enables Analog following.
FOLDEVIATION	Master deviation velocity of a task in units/sec for a 10-volt analog input voltage.
FOLDIST	Sets the axis triggering position, in units, for master position triggering.
FOLENCODER	Selects the encoder and following axis for the task. Enables encoder or pulse & direction following.
FOLEND	This command ends the encoder following mode.
FOLMAXRATIO	Sets the axis velocity clamp for an advance cycle.
FOLMINRATIO	Sets the axis velocity clamp for a recede-cycle.
FOLMODE	Selects the task's following mode and direction.
FOLOFFSET	Sets the axis offset position from the initial sync position in velocity following or from the initial starting position in velocity and position following.
FOLRATIO	Sets the ratio of the master axis to the following axis.
FOLSYNC	Returns the following sync status of an axis.
FOLTRIG	Sets the trigger mode for starting a following axis in motion.
FORMAT	Enables or disables the formatting of the STR$ returned string.
FOR..NEXT...STEP ...EXIT FOR	Begins a repeatable block of statements.
G	
GETCHAR	Waits for a character to be received via the serial port.
GOSUB...RETURN	Branches to a subroutine and returns.
GOTO	Branches unconditionally to the specified label.
H	
HARDLIMIT	Enables/disables or returns the HARDLIMIT enabled state of an axis.
HARDLIMNEG	Returns the status of the -LIM hardware state of an axis input.
HARDLIMPOS	Returns the status of the +LIM hardware state of an axis input.
HEX$	Returns the Hex character equivalent of the argument.
HVAL	Returns the Hex value of a string.
I	
IF..THEN..ELSE ..END IF	Begins a conditional block of statements.
IN	Returns the state of the inputs on a digital I/O board.
INCHAR	Returns a character from the selected serial port.
#INCLUDE	Includes a file name in a user's task.
INPUT	Reads data from the selected serial port.
INSTR	Return the first occurrence of a character in a string.
INTLIM	Sets or returns the servo axis integral limit.
J	
JOGSTART	Runs an axis continuously in the specified direction.
JOGSTOP	Stop continuous run.
JOYSTICK ..JOYSTICK END	Enable/Disables Joystick motion.
K	
KAFF	Sets or returns the servo axis acceleration feed forward gain.
KD	Sets or returns the servo axis derivative gain.
KI	Sets or returns the servo axis integral gain.
KP	Sets or returns the servo axis proportional gain.
KVFF	Sets or returns the velocity feed forward gain value for a servo axis.

L	
LCASE$	Converts a string to lower-case letters.
LEFT$	Returns the left-most characters of a string
LEN	Returns the number of characters in the string.
LINE	Initiates a coordinated straight-line move.
LOG	Returns the natural logarithm of x.
LOWSPD	Sets or returns the low speed (starting speed) of an axis.
M	
MAXSPD	Sets or returns the maximum allowed speed of an axis.
MID$	Returns the designated middle number of characters of a string.
MOD	Returns the divide remainder.
MOVE	Initiates a non-coordinated move.
MOVEHOME	Runs an axis until the home input is activated.
MOVEREG	Runs an axis until the registration input is activated.
N	
NOT	Logical "complement" operator.
NVR	Non-volatile storage of a variable (1-2048).
O	
OR	Logical "inclusive or" operator.
OUT	Sets or resets the specified outputs on a digital I/O board.
OUTLIMIT	Sets or returns the maximum analog output voltage allowed on a servo axis.
P	
PATH .. PATH END	Begin a continuous motion path, end a continuous motion path.
POINT	Specify coordinates, of a point on a path.
POSMODE	Sets or returns the positioning mode.
PRINT	Transmits data to the selected serial port.
PROFILE	Sets or returns the acceleration/deceleration profile of an axis.
R	
RADIUS	Sets the arc radius for blending from one line to another in a path.
READ	Reads values from data statements into variables or arrays.
REDUCE	Enables or disables the reduce current feature.
REM	Remark (also can use ')
RESET	Initializes System to power on condition.
RESTORE	Restores data list pointer to beginning.
RIGHT$	Returns the right-most characters from a string.
S	
SETCOM	Sets the baud rate and data format of the AUX serial port.
SHIFT	Shifts an array
SIGN	Returns the sign of the expression.
SIN	Returns the sine of the angle.
SOFTLIMIT	Enables/disables or returns SOFTLIMIT enabled state of an axis.
SOFTLIMNEG	Sets or returns the software absolute negative travel limit position.
SOFTLIMPOS	Sets or returns the software absolute positive travel limit position.
SPEED	Sets or returns tan individual axes target speed.
SQRT	Returns the square root of a number.
STR$	Returns a string representation of a numeric expression
STRING$	Returns a string of characters

T	
TAN	Returns the tangent of the angle.
TIMER	Sets or reads Task Timer
U	
UCASE$	Converts a string to upper-case letters.
V	
VAL	Returns the value of a string.
VELOCITY	Sets or returns the target velocity used for coordinated motion.
W	
WAIT	Waits for the period of time to expire.
WAITDONE	Waits for motion to be done for the specified axes. "Done" means motion is complete.
WARNING	Sets or returns task warning count
WNDGS	Enables or disables the motor winding current.
X	
XOR	Logical "exclusive or" operator.

SEBASIC CONVENTIONS

Superior Electric's BASIC-like language ("SEBASIC") conforms to most of the rules and conventions of modern implementations of the BASIC programming language, such as "QuickBasic", etc. Following is a summary of the considerations to be used in writing your programs.

AXIS CONVENTIONS and SETTING-READING VALUES

Many SEBASIC commands can address a single axis at a time, or they can address multiple axes. The number of axes in a program is assigned during the program configuration. The axis assignment to a task is also assigned during program configuration. **Note: When a program is loaded all axes are assigned to the host commands.**

The single axis format can be used to write data, read data, or command motion of a specific axis. The axis is the absolute axis value of a system (1-8). The following restrictions apply to the single axis format:

1) The read axis format can be used in any task or via a host command.
2) The write data or motion format can only be used in the task assigned to it or via a host command.

The write data or command motion format for a **single axis** is: COMMAND(axis #) = value or expression

```
        Example:        ACCEL(1) = 2000 sets the accel of axis 1 to 2000 units/sec².
```

The read single axis value format is: Expression = Command(axis#)

```
        Example:        X=ABSPOS(1)   'set X to the absolute position of axis 1
```

The multiple axis format can only be used for writing data or commanding motion of a designated axis. Commas are used as axis separators. The selected axis for these commands can only be used in the task assigned to them in the program configuration.
Note: When a program is loaded all axes are assigned as host commands.

The write data or command motion format for **multiple axes** is:

```
        COMMAND = value for axis 1, value for axis 2, , , ,value for axis n    (note: values may be expressions, too)
                ACCEL=,2000             'Sets the accel of axis 2 to 2000 units/sec²
                MOVE = ,1000,,2000      'Commands a move of 1000 units in axis 2 and a move of 2000 units in axis 4
```

In addition, many of the SEBASIC commands can be used either to set a controller parameter, or to read its present value. The format used to **set** values is shown in the above two examples. To **read** the present value of a parameter, the following syntax is used:

```
        COMMAND(axis #)
                (e.g.,  a1 =  ACCEL(1)   sets the program variable a1 equal to the present value of accel on axis 1)
```

When using the HOST mode of communicating with the controller, typing a command in the "set" format assigns the value typed to that parameter. Using "read" format returns the present setting.

ARITHMETIC OPERATORS

The SEBASIC arithmetic operators are listed in order of precedence:

Operator	Function
-	Negation
*, /	Multiplication and division
+, -	Addition and subtraction

Parentheses change the order in which arithmetic operations are performed. Operations within parentheses are performed first. Inside parentheses, the usual order of operation is maintained.

NOTE: Squaring and exponentiation are not supported; use multiplication to perform these operations.
Example: to calculate X^3, do X*X*X.

LOGICAL OPERATORS

The logical operators in SEBASIC, listed in order of precedence, are as follows:

Operator	Function
NOT	Bit-wise complement
AND	Conjunction
OR	Disjunction (inclusive "or")
XOR	Exclusive "or"

Logical operators perform tests on multiple relations, bit manipulations, or Boolean operations, and return a true (nonzero) or false (zero) value to be used in making a decision.

RELATIONAL OPERATORS

Relational operators are used to compare two values. The result of the comparison is either "true" (non-zero) or "false" (zero). This result can then be used to make a decision regarding program flow. Although SEBASIC treats any nonzero value as true, true is usually represented by the value 1.

Operator	Relation	Expression
=	Equality *	X = Y
<>	Inequality	X <> Y
<	Less than	X < Y
>	Greater than	X > Y
<=	Less than or equal to	X <= Y
>=	Greater than or equal to	X >= Y

* The equal sign (=) is also used to assign a value to a variable.

BASIC DATA TYPES

Two basic data types exist: floating point values and string values. All values are assumed to be floating point unless a $ suffix is used.

x	x is a floating point value.
x$	x$ is a string value.

NUMERIC FORMAT AND RANGE

Numeric data* may be represented in Standard format:

1234567,
-1.234567,
0.1234567,
1234.567,
etc.

or in Scientific notation format using the "e" or "E" to designate an exponent of base 10:

2e6	(2,000,000),
2.0456E4	(20456),
-3.14159e0	(-3.14159),
6.78E-2	(0.0678),
etc.	

The largest number that can be used is 2×2^{127} ($\cong 3.4 \times 10^{38}$)

The smallest number that can be used is 1×2^{-128} ($\cong 2.9 \times 10^{-39}$)

* The numeric resolution is 1 part in 8,388,608 ($\cong 1.2 \times 10^{-7}$), there are seven significant digits.

CASE SENSITIVITY IN STATEMENTS & COMMANDS

Some programming statements and commands are case-sensitive; others are not. The following table defines case sensitivity in SEBASIC:

BASIC LANGUAGE ELEMENT	**CASE SENSITIVE?**	**MAX. LENGTH (characters)**
Label	No	80
Variable name (symbolic constant)	No	80
String constant	Yes	80
BASIC keyword	No	N/A

Example of case sensitivity: String constant "Hello" is not the same as "HELLO"

Example of case insensitivity: Variable names XPOS, Xpos and xpos all reference the same variable.

The Host commands are not case sensitive; that is, upper and lower case letters can be used interchangeably.

Programming Example

Programs for the MX2000 controller are written in a BASIC-like language as shown by the following example.
NOTE: Additional programming examples are given in Section 7 - Application Examples.

```
' **************** PROJECT ************************** (consists of 3 Tasks)****************

        ' ******************** TASK 1 *****************************************************
        common x_done, y_done                  ' define variables used in other tasks
        common progdone

        progdone = 0                           ' initialize variables:  progdone, x_done, y_done, dis
        x_done = 0
        y_done = 0
        dis = 12                               ' set X axis move distance

        for x = 1 to 3                         ' repeat the following code 3 times:
                move = dis                     ' move the X axis  (+12) units, then
                move = -dis                    ' move it back (-12)

                do
                loop until done(1)             ' looping, waiting  for X axis move to finish

                x_done=1                       ' signals task 2, that the move is done

                do : loop until y_done=1       ' waits for "done" signal from task 2,

                y_done=0                       ' clear "done" signal from task 2
        next x
        progdone=1                             ' signals task 3 that the program is done
        end                                    ' end Task 1

        ' ******************** TASK 2 *****************************************************
        common x_done, y_done                  ' define variables used in task 1

        loop1:                                 ' note label called "loop1"
                do :  loop until x_done=1      ' wait until x_done =1, set by task 1
                                               ' note multiple stmts. on one line, separated by colon
                x_done = 0                     ' clear "done" signal from task 1
                move = ,6                      ' move Y axis 6 units

                do :  loop until done(2)       ' wait for Y axis move to finish

                y_done=1                       ' signal task 1, that the Y move is done
                goto loop1
        end                                    ' end Task 2

        ' ******************** TASK 3 *****************************************************
        common progdone                        ' define variable used (set)  in task 1

        do : loop until progdone=1        ' wait until progdone = 1
        print#1,"program finished"        ' print message on serial port 1

        end                                    ' end Task 3
```

Alphabetical List of Programming Commands with Syntax and Examples

& &

ACTION: Returns the bitwise AND of the expressions.

PROGRAM SYNTAX: result= expression1 & expression2

REMARKS: A 24 bit binary AND is performed on the two arguments.

EXAMPLES:

```
X = 10 & 2
0000 0000 0000 0000 0000 1010    (10)
0000 0000 0000 0000 0000 0010    (2)
0000 0000 0000 0000 0000 0010    (result)
Returns a 2
```

| |

ACTION: Returns the bitwise inclusive OR of the expressions.

PROGRAM SYNTAX: result= expression1 | expression2

REMARKS: A 24 bit binary OR is performed on the two arguments.

EXAMPLES:

```
X = 10 | 4
0000 0000 0000 0000 0000 1010    (10)
0000 0000 0000 0000 0000 0100    (4)
0000 0000 0000 0000 0000 1110    (result)
Returns a 14
```

^ ^

ACTION: Returns the bitwise eXclusive OR of the expressions.

PROGRAM SYNTAX: result= expression1 ^ expression2

REMARKS: A 24 bit binary eXclusive OR performed on the two arguments.
If a binary bit in expression2 exist the resulting bit will be inverse of the expression1 bit..

EXAMPLES:

```
X = 10 ^ 6
0000 0000 0000 0000 0000 1010    (10)
0000 0000 0000 0000 0000 0110    (6)
0000 0000 0000 0000 0000 1100    (result)
Returns a 12.
```

~ ~

ACTION: Returns the Ones complement of the argument.

PROGRAM SYNTAX: result= ~expression1

REMARKS: A 24 bit binary Ones complement is performed on the argument.

EXAMPLES:

```
X = X~1
0000 0000 0000 0000 0000 0001    (1)
1111 1111 1111 1111 1111 1110    (result)
Returns a -2.
```

>> >>

ACTION: Returns the bitwise shift right of the argument.

PROGRAM SYNTAX: expression= >>shift cnt

REMARKS: A 24 bit binary shift right is performed on the argument. The shift cnt determined the number of shift performed. 0's are shifted in starting at the MSB.

EXAMPLES:

```
x=10
x = x >>1
0000 0000 0000 0000 0000 1010    (10)
0000 0000 0000 0000 0000 0101    (result)
Returns a 5.
```

<< <<

ACTION: Returns the bitwise shift left one place of the argument.

PROGRAM SYNTAX: result= <<expression1

REMARKS: A 24 bit binary shift left is performed on the argument. The shift cnt determined the number of shift performed. 0's are shifted in starting at the LSB.

EXAMPLES:

```
x=10
x = x<<1
0000 0000 0000 0000 0000 1010    (10)
0000 0000 0000 0000 0001 0100    (result)
Returns a 20.
```

ABS

ABS

ACTION: Returns the absolute value of an expression.

PROGRAM SYNTAX: result=ABS(expression1)

REMARKS: The absolute value is the unsigned magnitude of the expression.

EXAMPLES:

```
X = -57.5
A = ABS(X)          'A = 57.5
```

ABSPOS

ABSPOS

ACTION: Sets or returns the absolute position of an axis.

PROGRAM SYNTAX:

```
ABSPOS(axis)=expression
ABSPOS=expression1,expression2, ... ,expression8
ABSPOS(axis) -  used in an expression
```

REMARKS: The "axis" specifies the number of the axis (1-8).

ABSPOS(axis)=expression
Sets the absolute position for the specified axis.

ABSPOS=expression1,expression2, ... ,expression8
Sets the absolute position for all defined axes. If an axis is unchanged a comma must be inserted in its place.

ABSPOS(axis) - used in an expression
Evaluates and returns the current absolute position for the specified axis.

The absolute position, ABSPOS, can not be changed during motion. If this is attempted during motion , this command will trap until motion has been completed and will then set the absolute position.

EXAMPLES:

ABSPOS(3)=2
sets axis 3 absolute position to 2 units.

ABSPOS=3,3,,6,6,5,5,3
sets the absolute position for axis 1, 2 and 8 to 3 units. Axis 4 and 5 to 6 units. Axis 6 and 7 to 5 units. Axis 3 remains unchanged.

a=ABSPOS(3)
returns the ABSPOS value for axis 3 to variable "a".

ACCEL

ACTION:

Sets or returns the acceleration value of an axis.

PROGRAM SYNTAX:

```
ACCEL(axis)=expression
ACCEL=expression1,expression2, ... ,expression8
ACCEL(axis)  -  used in an expression
```

REMARKS:

The "axis" specifies the number of the axis (1-8).

ACCEL(axis)=expression
Sets the acceleration rate for the specified axis.

ACCEL=expression1,expression2, ... ,expression8
Sets the acceleration rate for all defined axis.

ACCEL(axis) - used in an expression
Evaluates and returns the present acceleration for the specified axis.

EXAMPLES:

ACCEL(3)=2
Sets axis 3 acceleration rate to 2 units/sec^2.

ACCEL=3,3,,6,6,5,5,3
Sets the acceleration rate for axis 1,2 and 8 to 3 units/sec^2. Axis 4 and 5 to 6 units/sec^2 and axis 6 and 7 to 5 units/sec^2. Axis 3 remains unchanged.

ANALOG ANALOG

ACTION: Sets or returns a numeric value representing the voltage on the analog port.

PROGRAM SYNTAX: ANALOG(b0n)
ANALOG)b0n)=expression

REMARKS: The b specifies the board number (1-4).

The n specifies the analog input (1-4) or output(1-2).

ANALOG(b0n)
Returns the present value of the specified analog input. The range is +10.0 volts to -10.0 volts.

ANALOG(b0n)=expression
Sets the analog output equal to the expression. The range is +10.0 volts to -10.0 volts.

Board Analog Configuration				
b0n value	A inputs differential B inputs differential	A inputs single ended B inputs differential	A inputs differential B inputs single ended	A inputs single ended B inputs single ended
101	board 1 A+ & A-	board 1 A+	board 1 A+ & A-	board 1 A+
102	board 1 B+ & B-	board 1 B+ & B-	board 1 B+	board 1 B+
103		board 1 A-		board 1 A-
104			board 1 B-	board 1 B-
201	board 2 A+ & A-	board 2 A+	board 1 A+ & A-	board 2 A+
202	board 2 B+ & B-	board 2 B+ & B-	board 1 B+	board 2 B+
203		board 2 A-		board 2 A-
204			board 1 B-	board 2 B-
301	board 3 A+ & A-	board 3 A+	board 1 A+ & A-	board 3 A+
302	board 3 B+ & B-	board 3 B+ & B-	board 1 B+	board 3 B+
303		board 3 A-		board 3 A-
304			board 1 B-	board 3 B-
401	board 4 A+ & A-	board 4 A+	board 1 A+ & A-	board 4 A+
402	board 4 B+ & B-	board 4 B+ & B-	board 1 B+	board 4 B+
403		board 4 A-		board 4 A-
404			board 1 B-	board 4 B-

EXAMPLES: X=ANALOG(102)
Sets x equal to the present voltage on board 1 input 2.

ANALOG(102)=2.5
Sets the voltage on board 1 output 2 equal to +2.5 volts

AND AND

ACTION: The logical AND operator is used in boolean expressions.

PROGRAM SYNTAX: result = expression1 AND expression2

REMARKS: The AND operator uses this "truth table":

expression1	expression2	Condition result
True	True	True
True	False	False
False	True	False
False	False	False

The result is true if both expressions are true.

EXAMPLES:

```
if (x >2 AND y < 3)  then
```

ARC ARC

ACTION: Initiates a coordinated motion to move in an arc.

PROGRAM SYNTAX:

```
ARC=x,y,xcenter,ycenter,±angle  (normal)
ARC=xcenter,ycenter,±angle (in a path)
```

REMARKS: The x specifies the axis number for one of the coordinated axes, and the y specifies the axis number for the other axis.
Note x and y are not required in a path, since the PATH command defines the axes used.

The xcenter specifies the x axis coordinate of the arc center, and the ycenter specifies the y axis coordinate of the arc center.

The angle specifies the direction of rotation as well as the arc angle.
This angle is in degrees. Clockwise rotation is +.

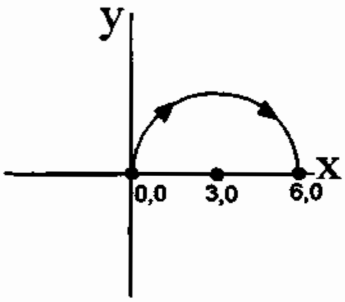
EXAMPLES:

```
ARC=1,2,3,0,+180
```

Initiates a 180° arc motion, using axis 1 and 2, with a 3 unit radius in the clockwise direction.

or

```
PATH=1,2
ARC=3,0,+180
LINE=2,.1
PATH END
```



In both examples the radius is 3 if:
1) In incremental position mode or in,
2) Absolute position mode and present position is 0,0

ASC ASC

ACTION: Returns the ASCII code for the first character in the string expression.

PROGRAM SYNTAX: ASC(n$)

REMARKS: The ASCII code returned is for the first character in the string expression (n$). If the string is a null string then 0 will be returned.

EXAMPLES:

```
x=ASC(a$)

IF ASC(INCHAR(1)) = 65 THEN
        Statements ...
ENDIF
```

If the character "A" is pressed on the keyboard, then do the statements

ATN ATN

ACTION: Returns the angle (in radians) whose tangent is x̄.

PROGRAM SYNTAX: ATN(x) - used in an expression

REMARKS: The arctangent returns an angle in the range -π/2 to π/2 radians. π/2 radians equals 90 degrees.

To convert values from degrees to radians, multiply the angle (in degrees) times π /180 (or .017453).

To convert a radian value to degrees, multiply it by 180/π (or 57.295779).

EXAMPLES:

```
ATN(1)    ' returns .785398 radians, which is 45 degrees
```

ATN2

ACTION: Returns the angle (in radians) whose tangent is y/x.

PROGRAM SYNTAX: ATN(y,x) - used in an expression

REMARKS: The arctangent returns an angle in the range -π to +π radians.
π radians equals 180 degrees.

To convert values from degrees to radians, multiply the angle (in degrees) times π/180 (or .017453).

To convert a radian value to degrees, multiply it by 180/π (or 57.295779).

EXAMPLES: ATN(2.5,3) 'returns .694727 radians, which is 39.8 degrees

BCD

ACTION: Returns the value on the BCD switches connected to an Expansion I/O port.

PROGRAM SYNTAX: BCD(b0n) - used in an expression

REMARKS: The b specifies the Expansion I/O board number (1-4).

The n specifies the BCD switch bank number (1 to 8).

BCD (b0n) - used in an expression
Evaluates and returns the number set on BCD board switch bank "n", connected to Expansion I/O board "b".

EXAMPLES: x=BCD (101)
Sets x equal to the value read on board 1, BCD switch bank 1.

x=BCD(405)
Sets x equal to the value read on board 4, BCD switch bank 5.

BOOST

BOOST

ACTION:

Enables or disables the Boost Current feature or returns the boost enable status for the specified axis. Note: requires compatible drive to function. When enabled the stepper drive BOOST output turns on during motion. This causes the drive to boost the motor current by 50%.

PROGRAM SYNTAX:

BOOST(axis)=expression
BOOST=expression1,expression2, ... ,expression8
BOOST(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).

If the expression is true (non-zero) then the BOOST feature is enabled for the specified axis. If the expression is false (zero) then the BOOST feature is disabled for the specified axis.

BOOST(axis)=expression
Enables or disables the motor boost current feature for the specified axis.

BOOST=expression1,expression2, ... ,expression8
Enables or disables the motor boost current feature for all defined axis.

BOOST(axis) - used in an expression
Evaluates to 1 if boost is enabled for the specified axis, otherwise evaluates to zero.

EXAMPLES:

BOOST(7)=1
Enables the BOOST feature for axis 7.
BOOST=1,1,,0,0,0,1,0
Enables the BOOST feature for axis 1,2,7. Disables the feature for axis 4,5,6,8. Axis 3 is unchanged.

BUSY

BUSY

ACTION:

Returns the motion status of the specified axis.
An axis is "busy" if motion is taking place.

PROGRAM SYNTAX:

BUSY(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).

If the commanded motion is incomplete BUSY(axis) returns a true (+1) otherwise BUSY(axis) returns a false (0). Busy is the complement of DONE.

EXAMPLES:

```
BUSY(1)          ' Evaluates to 1 if axis 1 is busy, or zero if it is not busy.

DO WHILE BUSY(1)   ' does the loop only if axis 1 is in motion

LOOP
```

CHR$

CHR$

ACTION: Returns a one-character string whose ASCII code is the argument.

PROGRAM SYNTAX: CHR$(code)

REMARKS: CHR$ is commonly used to send a special character to the serial port.

EXAMPLES:

```
PRINT#1,"Input Accel";CHR$(27)
```

transmits "Input Accel"plus the "escape" code (ASCII character 27) to the Host serial port.

COMMON

COMMON

ACTION: Allows variables to be shared by tasks.

PROGRAM SYNTAX: COMMON variable[,variable][,variable]

REMARKS: If a variable defined in one task is to be used in another task, the variable name must be declared by common statements in both tasks. Common statements should be placed at the start of the program.

EXAMPLES:

```
------------------- TASK 1 -----------------------------------

COMMON  X                    'X declared common in Task 1

X = 1

------------------- TASK 2 -----------------------------------

COMMON X                     'X declared common in Task 2

DO : LOOP UNTIL  X=1         'Wait until X is set to 1 by Task 1
```

COS

COS

ACTION: Returns the cosine of the angle x, where x is in radians.

PROGRAM SYNTAX: COS(x) - used in an expression

REMARKS: To convert values from degrees to radians, multiply the angle (in degrees) times π/180 (or .017453) where π = 3.141593. To convert a radian value to degrees, multiply it by 180/π (or 57.295779).

EXAMPLES:

```
PI = 3.141593
A = COS(PI/3)          ' sets A=0.5, which is the cosine of π/3 (60 degrees)
```

DATA

DATA

ACTION: Stores the numeric constants used by the READ statement.

PROGRAM SYNTAX: DATA constant, constant, etc.

REMARKS: The constant is a numeric constant.

EXAMPLES: DATA 1,2,3,4,5,6
Also see the example for the "READ" command

DECEL

DECEL

ACTION: Sets or returns the deceleration value of the axis.

PROGRAM SYNTAX:

```
DECEL(axis)=expression
DECEL=expression1,expression2, ... ,expression8
DECEL(axis) -  used in an expression
```

REMARKS: The "axis" specifies the number of the axis (1-8).

DECEL(axis)=expression
Sets the deceleration rate for the specified axis.

DECEL=expression1,expression2, ... ,expression8
Sets the deceleration rate for all defined axis.

DECEL(axis) - used in an expression
Evaluates and returns the present deceleration for the specified axis.

EXAMPLES: DECEL(2)=3.1
sets the deceleration value of axis 2 to 3.1 units/sec^2.

DECEL=3,3, ,6,6,5,5,3
Sets the deceleration rate for axis 1,2 and 8 to 3 units/sec^2, axis 4 and 5 to 6 units/sec^2 and axis 6 and 7 to 5 units/sec^2. Axis 3 remains unchanged.

X = DECEL(1)
Sets variable X equal to the value of deceleration for axis 1.

#DEFINE #DEFINE

ACTION:

Defines a symbolic name to be a particular string of characters.

COMMAND SYNTAX:

```
#DEFINE  name @1, ... ,@10  replacement text
#DEFINE replacement text
```

REMARKS:

The **name** has the same form as a variable name: a sequence of letters and digits that begins with a letter. The name is case sensitive. Typically upper case is used for the name.

The **@1, ..., @10** are the program command substitution arguments for the replacement text.

The **replacement text** can be any sequence of letters of characters.

Any occurance of the **name** in the program, not in quotes and not part of another name, will be replaced by the corresponding **replacement text** when the program is compiled.

EXAMPLES:

```
#DEFINE TRUE 1
```

substitutes a 1 when the name TRUE is encountered.

```
#DEFINE FALSE 0
```

substitutes a 0 when the name FALSE is encountered.

```
#DEFINE SENDPOS @1,@2  print#@1,abspos(@2)
```

Sends absolute position of axis @2 via port @1.

```
SENDPOS 1,2
```

Sends the absolute position of axis 2 via port #1. The 1 is substituted for the @1 argument and 2 is substituted for @2 argument.

```
#DEFINE CLR    print#2,Chr$(12);
#DEFINE LOCATE @1,@2 print#2,chr$(27);"[@1;@2H";

CLR            'Clear display
LOCATE 1,2     'Locate cursor at row 1 column 2
```

DIM DIM

ACTION: Declares an array variable and allocates storage space.

PROGRAM SYNTAX:

```
DIM variable(dimension,dimension,etc)
DIM variable_string(dimension,dimension,etc)
```

REMARKS: SEBASIC uses "option base zero" for array notation, in which the first element of each array dimension is annotated as element "0". Therefore, the total number of elements in the array is: (dimension 1 + 1)*(dimension 2 + 1)* ... *(dimension n + 1)

Example notation for a two-dimensional array:

	Y_0	Y_1	Y_2	→	Y_n
X_0				→	
X_1				→	
X_2				→	
↓	↓	↓	↓	↘	↓
X_n				→	

EXAMPLES:

DIM x(10,10,10)
The variable x is a three-dimensional array with 11*11*11, or 1331 elements.

DIM a$(3,3,3)
The variable string a$ is a three-dimensional array with 64 elements.

DIST DIST

ACTION: Returns the incremental distance moved for the last commanded motion.

PROGRAM SYNTAX: DIST(axis) - used in an expression

REMARKS: The "axis" specifies the number of the axis (1 - 8).

EXAMPLES:

x=DIST(1)
sets x to the last incremental distance moved in axis 1.

DO...LOOP DO...LOOP

ACTION: Repeats a block of statements while a condition is true or until a condition becomes true.

PROGRAM SYNTAX 1:

```
DO {UNTIL | WHILE} [condition]
        [statementblock]
[EXITDO]
        [statementblock]
LOOP
```

PROGRAM SYNTAX 2:

```
DO
        [statementblock]
[EXITDO]
        [statementblock]
LOOP {UNTIL | WHILE} [condition]
```

REMARKS: Syntax 1 allows the condition to be tested at the top of the loop. Syntax 2 allows the condition to be tested at the bottom of the loop therefore the loop will always execute at least once.

EXITDO is an alternative exit from a DO...LOOP.

EXITDO transfers control to the statement following the LOOP statement. When used within nested DO...LOOP statements, EXITDO transfers out of the immediately enclosing loop. EXITDO can be used only in a DO...LOOP statement.

EXAMPLES:

```
DO WHILE EVENT1(1) <> 1

LOOP
```

Note: Several additional examples are shown in Section 7.5.

DONE DONE

ACTION: Returns the motion status of the designated axis. "Done" means motion is complete.

SYNTAX: DONE(axis) - used in an expression

REMARKS: The "axis" specifies the number of the axis (1-8).

If the commanded motion is complete Done(axis) returns a true (1) otherwise, DONE(axis) returns a false (0). DONE is the complement of BUSY.

EXAMPLES:

DONE(1)
Returns the motion done status of axis 1.

```
DO


UNTIL DONE(1)
```

ACTION:

Enables or disables the checking of the drive ready (READY) signal on the axis card.

PROGRAM SYNTAX:

```
exp= DRVREADY
DRVREADY=exp
```

REMARKS:

DRVREADY
Returns the current state of the drive ready inhibit ored with the drive ready of each axis. The value returned is 0-255.

DRVREADY=exp
Sets the state of the drive ready inhibit. A 1 in the decimal weighted bits bypasses the drive ready signal check for that axis. This command should only be used if the drive connected to the MX2000 does not have a drive ready signal. The decimal weights for each axis are shown below.

axis 8	axis 7	axis 6	axis 5	axis 4	axis 3	axis 2	axis 1
128	64	32	16	8	4	2	1

The drive ready of an axis is only checked if motion is to occur in this axis. If the drive ready signal is checked and the drive is not ready an error 33 will occur and the cycle will be aborted.

EXAMPLES:

DRVREADY=15
Bypasses the drive ready signal check for axis 1 thru axis 4 and enables the drive ready check for axis 5 thru axis 8.

ENCERR

ACTION:

Returns the position error of the designated axis. Position error is the difference between the absolute position and the encoder position (ABSPOS minus ENCPOS).

PROGRAM SYNTAX:

ENCERR(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).

ENCERR(axis) - used in an expression
Evaluates and returns the axis' position error.

EXAMPLES:

```
x = ENCERR(1)
IF x > 10  THEN
        PRINT#1, "Large Error"
ENDIF
```

ENCMODE

ACTION:

Sets or returns the encoder closed loop operating mode of the selected axis.

COMMAND SYNTAX:

ENCMODE(axis)=expression
ENCMODE=expression1,expression2, ...,expression8
ENCMODE(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).

ENCMODE(axis)=expression
Sets the encoder closed loop operating mode for the specified axis.

ENCMODE=expression1,expression2, ... ,expression8
Sets the encoder closed loop operating mode for all defined axes.

ENCMODE(axis) - used in an expression
Evaluates and returns the encoder closed loop operating mode of a specified axis.

The operating modes are as follows:

0	close loop disabled - operate open loop
1	halt execution on excessive following error
2	correct position on excessive following error
3	restart move on excessive following error

EXAMPLES:

ENCMODE(1)=0
Sets axis 1 to open loop operation.

ENCMODE=0,0,1,1,0,1,1,1
Sets axis 1,2 and 5 to open loop operation. Axis 3,4,6,7 and 8 to halt execution on excessive following error.

```
x = ENCMODE(1)          ' returns the closed loop operating mode of axis 1.
PRINT#1,"Axis 1 encoder mode is", x          ' prints the mode number on port1
```

ENCPOS

ENCPOS

ACTION: Returns the encoder position of the specified axis.

PROGRAM SYNTAX: ENCPOS(axis) - used in an expression

REMARKS: The "axis" specifies the number of the axis (1-8).

ENCPOS(axis)
Evaluates and returns the present encoder position of the specified axis.

EXAMPLES:

```
y = ENCPOS(2)              ' returns the encoder position of axis 2.

PRINT#2, "Axis 4 position =", ENCPOS(4)    ' prints to port to the axis 4 position.
```

END

END

ACTION: Signifies the end of a program.

COMMAND SYNTAX: END

REMARKS: This command signifies the end of a program.

EXAMPLES:

```
statement
statement
.
.
END
```

ERR ERR

ACTION: Returns the MX controller error status number for this task.

PROGRAM SYNTAX:

ERR
ERR=expression1,expression2

REMARKS:

ERR
Returns the last task error number set if any.
The predefined error codes are:

1 Axis +Limit input activated while moving in the + direction.
2 Axis -Limit input activated while moving in the - direction.
3 Axis Soft Limit exceeded while moving in the + direction.
4 Axis Soft Limit exceeded while moving in the - direction.
5 Closed Loop Correction attempts exceeded.
6 Closed Loop Following Error exceeded in Encoder Mode 1.
7 Closed Loop Following Error exceeded during a POINT, LINE or ARC move.
8 Closed loop following Error exceeded during a Jog cycle and Encoder mode 2.
9 Warning. Command axis is not in task group.
10 Warning. Analog I/O selected out of range.
11 Warning. BCD selected out of range (b01 - b08).
12 Warning. Encmode selected out of range (0 - 3).
13 Warning. EXIN selected out of range (b00 - b47).
14 Warning. EXOUT selected out of range (b00 - b47).
15 Warning. Digital Input selected out of range (b01 - b24).
16 Warning. Digital Output selected out of range(b01 - b24).
17 Warning. Log command argument 0 or negative.
18 Warning. SQRT command argument is negative.
19 Warning. NVR element out of range.
20 Warning. Read command out of data arguments.
21 Warning. Accel value out of range (1 - max decel).
22 Warning. Decel value out of range (1 - max decel).
23 Warning. Low Speed value out of range (0 - max spd).
24 Warning. Axis Speed value out of range (1 - max spd).
25 Warning. Max Spd out of range.
26 Warning. Profile value out of range (0 or 1).
27 Warning. Encoder Mode value out of range (0-3)
28 Warning. Coordinated Speed value out of range (1 - max spd).
29 Warning. Radius value out of range.
30 Motion occurring at program end.
31 RS232 Configuration Error.
32 Warning. Servo Parameter out of range.
33 Drive not Ready Fault.
34 Servo following error exceeded.
35 Warning. OUTLIMIT out of range (1-10 volts).
36 Program Area out of memory.

Note: if a warning is issued the value or the command is ignored.

If an error occurs during program execution the fault LED will blink the error code. If the error code is >= 10 the fault LED blinks on .25 sec and off .5 sec for each ten's digit. The LED goes off for 1.25 sec. If the LSB digit is 0 the LED stays on for 1 sec and then goes off for 2.5 sec. Otherwise, the fault LED blinks on .25 sec and off .5 sec for each unit digit then goes off for 2.5 sec.

(ERR, cont'd.)

ERR=expression1,expression2

Used as a user definable error to set an error number and severity of the error. Expression1 set the error number and expression2 sets the severity of the error. If the error number is a 0 the fault LED will be turned off and the severity argument will be ignored. Multiple severity levels can be set by adding the desired severity level numbers. The severity levels are:

1	Stop all motors in Task immediately.
2	Stop all motors in task using deceleration value of each axis.
4	Stop all motors in Task using Max decel value of each axis.
8	Increment Warning Count for this task.
16	Create an ERROR Trap.

EXAMPLES:

x=ERR
Sets x equal to the present controller error number for this task.

ERR=0,0
Sets error number to 0 for this task.

ERR=nnn, 1
Sets error number = nnn. Stops all motors in Task Immediately.

ERR=nnn,17
Sets error number=nnn. Stops all motors in Task immediately, creates an error trap condition.

ERROR HANDLING:

The MX2000 Controller will handle errors in one of two ways. The first method is to stop execution of the task in which the error occurred; all remaining tasks continue executing. This is the default method of handling errors. The second method of handling errors is for the user to write an error trapping routine. The routine must use the label **ERROR_HANDLER**. This routine will be called whenever an error occurs. The user may then evaluate the condition that invoked the error handling routine.

EXAMPLE:

ERR=99,17
When this line of code is executed, an error condition is created. If an error handling routine has been written execution will jump to the **ERROR_HANDLER** routine. If no error no handling routine was used, execution would end. The error number is set to 99 and all motors in this task will stop.

ERROR_HANDLER:

This label defines the start of the code that is to be executed if an error occurs. The last statement in the ERROR_HANDLER code should be an "END" or "GOTO label".

EVENT1 EVENT1

ACTION:

Returns the state of the trigger input labeled EVNT1 for the selected axis or sets the trigger polarity and trigger enable, which are used in movehome and movereg cycles.

NOTE: EVENT1 is the software command used to access the hardware input labeled "EVNT1" on the Dual Axis Interface card.

PROGRAM SYNTAX:

EVENT1(axis) - used in an expression
EVENT1(axis)=expression
EVENT1=expression1, ... , expression8

REMARKS:

EVENT1 is a software interface to the EVNT1 input on the axis card that is primarily used as a home trigger in a movehome cycle or a trigger to start the index portion of a movereg cycle. (Note: if not used for one of these purposes, then it can be used as a general-purpose, program-testable input.)

The EVENT1 triggering for a movehome or movereg cycle is assigned in the user program configuration.

EVENT1(axis) - used in an expression
Evaluates and returns the state of the EVNT1 input. The "axis" value (1-8) selects the axis. A returned value of (0) indicates an inactive state and a (1) an active state.

EVENT1(axis)=expression
Sets the trigger edge or trigger enable for the EVNT1 input of the specified axis. A value of 0 disables the event1 trigger if assigned as a movereg trigger. A value of 1 or greater sets the event1 trigger to positive edge triggering and enables the trigger if assigned as a movereg or movehome trigger. A value which is negative set the event1 trigger to negative edge triggering if assigned as a movereg or movehome trigger.

EVENT1=expression1, ... , expression8
Sets the trigger edge or trigger enable for the EVNT1 input of the specified axis. A value of 0 disables the event1 trigger if assigned as a movereg trigger. A value of 1 or greater sets the event1 trigger to positive edge triggering and enable the trigger if assigned as a movereg or movehome trigger. A value which is negative set the event1 trigger to negative edge triggering if assigned as a movereg or movehome trigger.

EXAMPLES:

```
if EVENT1(1) then
        statement
end if
```

EVENT1(1)= 0
disables axis 1 event1 trigger if assigned as a movereg trigger.

EVENT1(1)= 1
sets axis 1 event1 trigger to positive edge and enables the event1 trigger if assigned as a movehome or movereg trigger.

EVENT1(1)= -1
sets axis1 event1 trigger to negative edge and enables the event1 trigger if assigned as a movehome or movereg trigger.

EVENT1=,,1
sets axis 3 event1 trigger to positive edge and enables the event1 trigger if assigned as a movehome or movereg trigger.

EVENT2

EVENT2

ACTION:

Returns the state of the trigger input labeled EVNT2 for the selected axis or sets the trigger polarity and trigger enable, which are used in movehome and movereg cycles.

NOTE: EVENT2 is the software command used to access the hardware input labeled "EVNT2" on the Dual Axis Interface card.

PROGRAM SYNTAX:

EVENT2(axis) - used in an expression
EVENT2(axis)=expression
EVENT2=expression1, ... , expression8

REMARKS:

EVENT2 is a software interface to the EVNT2 input on the axis card that is primarily used as a home trigger in a movehome cycle or a trigger to start the index portion of a movereg cycle. (Note: if not used for one of these purposes, then it can be used as a general-purpose, program-testable input.)

The EVENT2 triggering for a movehome or movereg cycle is assigned in the user program configuration.

EVENT2(axis) - used in an expression
Evaluates and returns the state of the EVNT2 input. The "axis" value (1-8) selects the axis. A returned value of (0) indicates an inactive state and a (1) an active state.

EVENT2(axis)=expression
Sets the trigger edge or trigger enable for the EVNT2 input of the specified axis. A value of 0 disables the event2 trigger if assigned as a movereg trigger. A value of 1 or greater sets the event2 trigger to positive edge triggering and enable the trigger if assigned as a movereg or movehome trigger. A value which is negative set the event2 trigger to negative edge triggering if assigned as a movereg or movehome trigger.

EVENT2=expression1, ... , expression8
Sets the trigger edge or trigger enable for the EVNT2 input of the specified axis. A value of 0 disables the event2 trigger if assigned as a movereg trigger. A value of 1 or greater sets the event2 trigger to positive edge triggering and enable the trigger if assigned as a movereg or movehome trigger. A value which is negative set the event2 trigger to negative edge triggering if assigned as a movereg or movehome trigger.

EXAMPLES:

```
if EVENT2(1) then
  .     statement
end if
```

EVENT2(1)= 0
disables axis 1 event2 trigger if assigned as a movereg trigger.

EVENT2(1)= 1
sets axis 1 event2 trigger to positive edge and enables the event2 trigger if assigned as a movehome or movereg trigger.

EVENT2(1)= -1
sets axis1 event2 trigger to negative edge and enables the event2 trigger if assigned as a movehome or movereg trigger.

EVENT2=,,1
sets axis 3 event2 trigger to positive edge and enables the event2 trigger if assigned as a movehome or movereg trigger.

EXIN EXIN

ACTION: Returns the state of the specified expansion I/O inputs.

PROGRAM SYNTAX: EXIN(nnn) (used in an expression)
EXIN(nnn,len) (used in an expression)

nnn is the I/O point terminal number.

nnn = (100-147) or (200-247) or (300-347) or (400-447)
board 1 board 2 board 3 board 4

len is the number of I/O points.

len = 1 - 24

SINGLE INPUT PROGRAM SYNTAX: EXIN(nnn) returns the state (1 or 0) of input nnn.
nnn = (100-147) or (200-247) or (300-347) or (400-447)
board 1 board 2 board 3 board 4

EXAMPLE:

EXIN(207) = 1	input 207 is on.
EXIN(207) = 0	input 207 is off.

MULTIPLE INPUT PROGRAM SYNTAX: EXIN(nnn,len) returns a number corresponding to the states of multiple inputs (calculated from the binary weighting of inputs nnn to (nnn+len-1))

Where: nnn is the first input and len is the number of inputs.

nnn = (100-147) or (200-247) or (300-347) or (400-447)
board 1 board 2 board 3 board 4
len = (1-24)

EXIN(nnn,len) is equivalent to:

EXIN(nnn) +2 EXIN(nnn+1) +4 EXIN(nnn+2) + ... + 2^{len-1} EXIN(nnn+len-1)

EXAMPLE: EXIN(207,3) is equivalent to:

EXIN(207) +2 EXIN(208) +4 EXIN(209) depending on the states of

inputs 207 - 209, EXIN(207,3) will return a number between 0 and 7.

So, if input states are: 207 = off, 208 = on, 209 = on

```
the value of EXIN(207,3) = 1 1 0 (binary)
                           | | └──output 207 = off
                           | └────output 208 = on
                           └──────output 209 = on
```

EXOUT

EXOUT

ACTION: Sets or returns the state of the specified expansion I/O outputs.

PROGRAM SYNTAX:

EXOUT(nnn) = expression
EXOUT(nnn) (used in an expression)
EXOUT(nnn,len) = expression
EXOUT(nnn,len) (used in an expression)

nnn is the I/O point terminal number

```
nnn =  (100-147) or  (200-247) or  (300-347) or  (400-447)
          board 1       board 2       board 3       board 4
```

len is the number of I/O points

len = 1 - 24

SET SINGLE OUTPUT PROGRAM SYNTAX: EXOUT(nnn) = expression

"expression" turns output nnn on (expression is non-zero) or off (expression =0).

EXAMPLE:

EXOUT(207) = -3	turns output 207 on.
EXOUT(207) = 0	turns output 207 off.

READ SINGLE OUTPUT PROGRAM SYNTAX: EXOUT(nnn) (used in an expression)

evaluates to the last output commanded (1 or 0) for this I/O pin. Note this is different from the state of the I/O pin.

EXAMPLE:

EXOUT(207) = 1	turns output 207 on.
A = EXOUT(207) *2	A = 1 (last commanded output for 207) *2 A = 2

SET MULTIPLE OUTPUTS PROGRAM SYNTAX: EXOUT(nnn,len) = expression

The expression is evaluated and converted to an integer value. The least significant "len" bits of the binary representation are then used to set outputs "nnn" to "nnn+len-1" respectively.

EXAMPLE: EXOUT(207,3) = 6.2 sets output 207 - 209.

```
6.2 converted to integer 6
binary representation of 6 is  1 1 0
                               | | └──        output 207 = off
                               | └────        output 208 = on
                               └──────        output 209 = on
```

(continued on next page)

(EXOUT, cont'd.)

READ MULTIPLE OUTPUTS
PROGRAM SYNTAX: EXOUT(nnn,len) (used in an expression)

evaluates to a number corresponding to the last outputs commanded (1 or 0) for these I/O pins. The number is the binary weighted sum of last commanded outputs nnn to (nnn+len-1). Note this is different from the state of the I/O pins.

EXAMPLE: EXOUT(207,3) = 4 binary rep. of 4 1 0 0

output 207 = off
output 208 = off
output 209 = on

A = EXOUT(208,2) A=2
(last output for 208 = 0)
+ 2 (last output for 209 = 1)

FEEDRATE FEEDRATE

ACTION: Sets a feedrate override during Path execution.

PROGRAM SYNTAX: FEEDRATE=expression

REMARKS: The expression range is .01 to 2.0 (1% to 200%). This value scales the commanded velocity to obtain a target velocity.

This command is only honored during path execution.

EXAMPLES:

```
PATH=1,2
        FEEDRATE=.5
        LINE=expression1,expression2
PATH END
```

FOLANALOG

ACTION:

Command enables or disables analog following, defines the master analog input and the following axes.

PROGRAM SYNTAX:

FOLANALOG=b0n, axis1, . . . , axis8
FOLANALOG=0,0

REMARKS:

FOLANALOG=b0n, axis1, . . . , axis8
This syntax enables analog following and defines the analog input used for deviating the master velocity. The "b" defines the axis board. The "n" defines the Analog input on the selected board (1-4). Master velocity monitoring is enabled by this command.

Axis I/O Analog Configuration				
b0n value	A- inputs: differential B- inputs: differential	A- inputs: single ended B- inputs: differential	A- inputs: differential B- inputs: single ended	A- inputs: single ended B- inputs: single ended
101	board 1 Ain+ & Ain-	board 1 Ain+	board 1 Ain+ & Ain-	board 1 Ain+
102	board 1 Bin+ & Bin-	board 1 Bin+ & Bin-	board 1 Bin+	board 1 Bin+
103		board 1 Ain-		board 1 Ain-
104			board 1 Bin-	board 1 Bin-
201	board 2 Ain+ & Ain-	board 2 Ain+	board 1 Ain+ & Ain-	board 2 Ain+
202	board 2 Bin+ & Bin-	board 2 Bin+ & Bin-	board 1 Bin+	board 2 Bin+
203		board 2 Ain-		board 2 Ain-
204			board 1 Bin-	board 2 Bin-
301	board 3 Ain+ & Ain-	board 3 Ain+	board 1 Ain+ & Ain-	board 3 Ain+
302	board 3 Bin+ & Bin-	board 3 Bin+ & Bin-	board 1 Bin+	board 3 Bin+
303		board 3 Ain-		board 3 Ain-
304			board 1 Bin-	board 3 Bin-
401	board 4 Ain+ & Ain-	board 4 Ain+	board 1 Ain+ & Ain-	board 4 Ain+
402	board 4 Bin+ & Bin-	board 4 Bin+ & Bin-	board 1 Bin+	board 4 Bin+
403		board 4 Ain-		board 4 Ain-
404			board 1 Bin-	board 4 Bin-

The arguments, axis1 through axis8, define which following axes are involved in the following-mode. The value entered, in the axis field, is irrelevant.

The nominal velocity for the master axis is defined by the VELOCITY command. This Velocity value must be greater than zero.

The FOLDEVIATION command defines the velocity deviation from the nominal velocity for a 10-volt analog input signal on the designated analog port.

FOLANALOG=0,0
This command ends analog following and disables master velocity monitoring.

(FOLANALOG, cont'd.)

A MOVE command between the FOLANALOG and FOLEND commands will generate a following move cycle. The move axis must be a defined following axis.

A JOGSTART command between the FOLANALOG and FOLEND commands will generate a JOGSTART following cycle. The JOGSTART axis must be a defined following axis. This cycle can be stopped using the JOGSTOP command.

The following commands are not allowed between the FOLANALOG and FOLEND statements:
ARC
DIM
FOLANALOG
FOLENCODER
FOLMODE
GOSUB
GOTO
JOYSTICK
LINE
MOVEHOME
MOVEREG
PATH
PATHEND
POINT

EXAMPLES:

VELOCITY=1000	Sets the nominal master velocity at 1000 units/second.
FOLDEVIATION=100	Sets the maximum deviation at 100 units/second.
FOLANALOG=101,1,1,,1	Selects board 1 A-analog inputs and enables analog following for axis 1, axis 2 and axis 4
Program Statements	
FOLEND	
FOLANALOG=0,0	ends analog following
FOLEND	

FOLDEVIATION

FOLDEVIATION

ACTION: Sets the maximum velocity deviation for a 10-volt analog input.

PROGRAM SYNTAX:

```
FOLDEVIATION=expression
FOLDEVIATION
```

REMARKS: The expression's define the deviation for a 10-volt analog input. This value is in units/second.

Analog Input Voltage	Commanded Master Velocity
+10-volts	Velocity(task)+FOLDEVIATION
0-volts	Velocity
-10-volt	Velocity(task) - FOLDEVIATION

FOLDEVIATION=expression
Sets the maximum velocity deviation allowed for a 10-volt analog input. The value is specified in units/second.

FOLDEVIATION
Returns the current value for FOLDEVIATION.

This command only applies to Analog following.

EXAMPLES:

```
VELOCITY= 1000          Sets the nominal velocity at 1000 units/second.
FOLDEVIATION=100        Sets the maximum velocity deviation at 100
                        units/second
FOLANALOG=101,1,2,4     enable analog following
Program Statements
FOLEND
FOLANALOG=0,0           ended analog following
FOLEND
```

FOLDIST FOLDIST

ACTION:

Defines the distance between master position triggering.

PROGRAM SYNTAX:

FOLDIST(axis)=expression
FOLDIST=expression1, . . . , expression8
FOLDIST(axis)

REMARKS:

The value of "axis" specifies the axis wanted(1-8).

The expression's define the distance in units between master position triggering.

FOLDIST(axis)=expression
Defines the trigger distance for the specified axis.

FOLDIST=expression1, . . . , expression8
Defines the trigger distance for all defined axes. If an axis is unchanged, a comma must be inserted in its place.

FOLDIST(axis)
Returns the trigger distance for the specified axis.

This command is only used when FOLTRIG for an axis is set to 3.

EXAMPLES:

```
ABSPOS(1)=0                     sets following starting position
ACCEL(1)=100000                 axis 1 ACCEL rate
DECEL(1)=200000                 axis 1 DECEL rate
POSMODE(1)=1                    absolute positioning mode
FOLMODE(1)=2                    velocity and position following using the programmed
                                direction
FOLTRIG(1)=1                    starts the cycle on the event 1 trigger
FOLRATIO(1)=1.0                 following ratio 100%
FOLMAXRATIO(1)=1.5              maximum secondary velocity 150%
FOLMINRATIO(1)=.5               minimum secondary velocity 50%
FOLDIST(1)=10000                10000 units between cuts
CUTLOOP:                        start of the loop
FOLENCODER=101,1                enables encoder following
FOLOFFSET(1)=0                  sync to master starting position
JOGSTART(1)=1                   starts Jog velocity and position following on the event 1
                                trigger
DO: LOOP UNTIL FOLSYNC(1)=1  wait for positional sync
EXOUT(100)=1                    starts the cutting cycle
DO: LOOP UNTIL EXIN(101)=1      wait until cutting complete
EXOUT(100)=0                    stops the cutting cycle
JOGSTOP(1)=0                    stops  the position-following cycle
FOLEND                          wait for jogging to stop
MOVE(1)=0                       moves back to starting position
WAITDONE(1)=1
FOLTRIG(1)=3                    trigger on master position
IF EXIN(102)=1 THEN GOTO CUTLOOP  if input true, continue
FOLENCODER=0,0                  end encoder following
FOLEND
END
```

FOLENCODER FOLENCODER

ACTION:

This enables or ends Encoder following and defines the master axis and the following axes.

PROGRAM SYNTAX:

FOLENCODER=b0n, axis1, . . . , axis8
FOLENCODER=0,0

REMARKS:

FOLENCODER=b0n, axis1, . . . , axis8
This enables encoder following and defines the encoder axis used for the master velocity. The "b" defines the board(1-4) where the encoder is found. The "n" defines the encoder axis on the board. The master velocity monitoring is enabled by this command.

b0n value	axis selected
101	1
102	2
201	3
202	4
301	5
302	6
401	7
402	8

The arguments, axis1 through axis8, define the following axes involved in following. The value of each axis field is irrelevant.

The source for the encoder input can be pulse and direction or the quadrature encoder input. This is selected in the Program Configuration section.

FOLENCODER=0,0
This ends encoder following. The master velocity monitoring is disabled.

A MOVE command between the FOLENCODER and FOLEND commands will generate a following move cycle. The move axis must be a defined following axis.

A JOGSTART command between the FOLENCODER and FOLEND commands will generate a JOGSTART following cycle. The JOGSTART axis must be a defined following axis. This cycle can be stopped using the JOGSTOP command.

The following commands are not allowed between a FOLENCODER and FOLEND:

ARC DIM FOLANALOG FOLENCODER
FOLMODE GOSUB GOTO JOYSTICK
LINE MOVEHOME MOVEREG PATH
PATHEND POINT

EXAMPLES:

FOLENCODER=101,1,1,,1 Sets encoder axis 1 as the master axis and axis 1, axis 2 and axis 4 as the following axes.

Program Statements
FOLEND
FOLENCODER=0,0 ends encoder following
FOLEND

FOLEND FOLEND

ACTION: This command disables the following modes for all axes involved and waits for all following motion to come to a stop.

PROGRAM SYNTAX: FOLEND

REMARKS: If an axis is JOGSTART following, a JOGSTOP command will be issued for this axis. This command disables analog or encoder following. However, the master velocity will continue to be monitored until an FOLANALOG=0,0 or FOLENCODER=0,0 is issued.

EXAMPLES:

```
FOLMODE(2)=2                    Velocity and position following
FOLRATIO(2)=1                   following-ratio is 100%
FOLMAX(2)=1.5                   maximum ratio 150%
FOLMIN(2)=.5                    minimum ratio 50%
FOLOFFSET(2)=100                sync to master position 100 units
FOLTRIG(2)=1                    start cycle trigger input 1
FOLENCODER=101,,1               enables encoder following for axis 2.
JOGSTART(2)= +1                 starts Jog on trigger input 1
DO: LOOP UNTIL FOLSYNC(2)=1     wait for sync

   program statements
DO: LOOP UNTIL EXIN(100)=1      wait for input
FOLOFFSET(2)= -20               Recede secondary -20 units
DO: LOOP UNTIL FOLSYNC(2)=1     wait for sync

   program statements
DO: LOOP UNTIL EXIN(101)=1      wait for input
FOLOFFSET(2)= +20               advance secondary 40 units
DO: LOOP UNTIL FOLSYNC(2)=1     wait for sync

   program statements
DO: LOOP UNTIL EXIN(102)=1      wait for input
JOGSTOP(2)=0                    stop the Jogging cycle
FOLEND                          disables encoder following
FOLENCODER=0,0                  cancel encoder following
FOLEND
END
```

FOLMAXRATIO

FOLMAXRATIO

ACTION:

Sets the maximum velocity allowed by the following axis during an advance cycle.

PROGRAM SYNTAX:

```
FOLMAXRATIO(axis)=expression
FOLMAXRATIO=expression1, . . . , expression8
FOLMAXRATIO(axis)
```

REMARKS:

The "axis" specifies the number for the axis(1-8).

The expression value specifies the maximum velocity attainable as a ratio of the master. **This value must be greater than the FOLRATIO for this axis but less than or equal to 2.0 (200%).**

FOLMAXRATIO(axis)=expression
Sets the maximum velocity for the specified axis.

FOLMAXRATIO=expression1, . . . , expression8
Sets the maximum velocity for all defined axes. If an axis is unchanged, a comma must be inserted in its place.

FOLMAXRATIO(axis)
Returns the current maximum velocity for the specified axis.

EXAMPLES:

FOLMAXRATIO(3)=1.5
Maximum velocity for axis 3 is set to 150%.

FOLMAXRATIO=1.25,1.5,,1.75
Sets the maximum velocity for axis 1 to 125%, axis 2 to 150%, axis 3 unchanged and axis 4 to 175%.

FOLMINRATIO

FOLMINRATIO

ACTION:

Sets the minimum velocity allowed by the following axis during a receding cycle.

PROGRAM SYNTAX:

FOLMINRATIO(axis)=expression
FOLMINRATIO=expression1, . . . , expression8
FOLMINRATIO(axis)

REMARKS:

The "axis" specifies the number for the axis(1-8).

The expression value specifies the minimum velocity attainable as a ratio of the master. **This value must be less than the FOLRATIO for this axis but greater less than 0.**

FOLMINRATIO(axis)=expression
Sets the minimum velocity for the specified axis.

FOLMINRATIO=expression1, . . . , expression8
Sets the minimum velocity for all defined axes. If an axis is unchanged, a comma must be inserted in its place.

FOLMINRATIO(axis)
Returns the current minimum velocity for the specified axis.

EXAMPLES:

FOLMINRATIO(3)=.5
Sets the minimum velocity for axis 3 at 50%.

FOLMINRATIO=.25, .5,,.75
Sets the maximum velocity for axis 1 to 25%, axis 2 to 50%, axis3 unchanged and axis 4 to 75%.

FOLMODE FOLMODE

ACTION: Sets the following and direction mode of operation of an axis.

PROGRAM SYNTAX:

```
FOLMODE(axis)=expression
FOLMODE=expression1, . . . , expression 8
FOLMODE(axis)
```

REMARKS: The "axis" specifies the number for the axis(1-8).

The expression's value specifies the mode of operation.

0 velocity following using the programmed direction.
1 velocity following using the master axis direction.
2 velocity and position following using the programmed direction.
3 velocity and position following using the master axis direction.

This command must precede an FOLENCODER or FOLANALOG command.

FOLMODE(axis)=expression
Sets the following and direction mode for the specified axis.

FOLMODE=expression1, . . . , expression8
Sets the following and direction mode for all defined axes. If an axis is unchanged, a comma must be inserted in its place.

FOLMODE(axis)
Returns the current mode for the specified axis.

EXAMPLES:

FOLMODE(3)=0
Axis 3 is set to the velocity following mode using the programmed direction.

FOLMODE=1,2,,3
Axis 1 is set to the velocity following mode using the master axis direction. Axis 2 is set for velocity and position following using the programmed direction. Axis 3 is unchanged, and axis 4 is set to velocity and position following using master axis direction.

FOLOFFSET

ACTION: Defines an offset relative with the initial sync position of a following cycle.

PROGRAM SYNTAX:

```
FOLOFFSET(axis)=expression
FOLOFFSET=expression1, . . . , expression8
FOLOFFSET(axis)
```

REMARKS: The "axis" specifies the number for the axis(1-8).

The expression's value specifies the relative offset from the initial sync position. This value is in following axis units.

FOLOFFSET(axis)=expression
Sets the offset for the specified axis.

FOLOFFSET=expression1, . . . , expression8
Sets the offset for all defined axes. If an axis is unchanged, a comma must be inserted in its place.

FOLOFFSET(axis)
Returns the current offset for the specified axis.

EXAMPLES:

```
FOLMODE(2)=2                     velocity and position following.
FOLRATIO(2)=1                    ratio 100% of master
FOLMAXRATIO(2)=1.5               maximum 150%
FOLMINRATIO(2)=.5                minimum 50%
FOLENCODER=101,,2                enable encoder following
FOLOFFSET(2)=100                 sync to master position 100 units
FOLTRIG(2)=1                     starts the cycle on trigger input 1
JOGSTART(2)= +1                  starts Jog on trigger input 1
DO: LOOP UNTIL FOLSYNC(2)=1      wait for positional sync
  program statements
DO: LOOP UNTIL EXIN(100)=1       wait for input
FOLOFFSET(2)= -20                Offset = -20 units, a receding cycle
DO: LOOP UNTIL FOLSYNC(2)=1      wait for position sync
  program statements
DO: LOOP UNTIL EXIN(101)=1       wait for input
FOLOFFSET(2)= +20                advance secondary 40 units
DO: LOOP UNTIL FOLSYNC(2)=1      wait for position sync
  program statements
DO: LOOP UNTIL EXIN(102)=1       wait for input
JOGSTOP(2)=0                     jogging stops
FOLEND                           disables encoder following
FOLENCODER=0,0                   end encoder following
FOLEND
```

FOLRATIO FOLRATIO

ACTION: Sets the ratio of the following axis to the master axis.

PROGRAM SYNTAX:

```
FOLRATIO(axis)=expression
FOLRATIO=expression1, . . . , expression8
FOLRATIO(axis)
```

REMARKS: The "axis" specifies the number for the axis(1-8).

The "expression" value specifies the ratio (.001 to 2.0) .1% to 200% of the master axis.

FOLRATIO(axis)=expression
Sets the ratio for the specified axis.

FOLRATIO=expression1, . . . , expression8
Sets the ratio for all defined axes. If an axis is unchanged, a comma must be inserted in its place.

FOLRATIO(axis)
This syntax returns the current ratio for the specified axis.

When using the velocity and position following mode, this value should be set to 100%.

When this value is changed, make sure that the FOLMAXRATIO for this axis is greater than the new value and that the FOLMINRATIO is less than the new value.

EXAMPLES:

FOLRATIO(3)=.5
Sets axis 3 to 50% of the master axis.

FOLRATIO=1,1.5,,1.75
Sets axis 1 to 100%, axis 2 to 150%, and axis 4 to 175% of the master axis. Axis 3 is unchanged.

FOLSYNC FOLSYNC

ACTION: Returns the sync status of the specified axis.

PROGRAM SYNTAX: FOLSYNC(axis)

REMARKS: The "axis" specifies the number for the axis(1-8).

The value returned is either a 0(out of sync) or 1(in sync).

EXAMPLES:

```
FOLANALOG=101,1
  Program statements
DO: LOOP UNTIL FOLSYNC(2)=1 Waits for axis 2 to be in sync
  Program statements
FOLEND
```

FOLTRIG

ACTION: Defines the starting trigger for a following cycle.

PROGRAM SYNTAX:

```
FOLTRIG(axis)=expression
FOLTRIG=expression1, . . . , expression8
FOLTRIG(axis)
```

REMARKS: The "axis" specifies the number for the axis(1-8).

The "expression" value specifies the trigger mode for the axis.

0	no starting trigger command required.
1	start the cycle on event 1 trigger.
2	start the cycle on event 2 trigger.
3	start the cycle on a master position trigger

FOLTRIG(axis)=expression
Sets the trigger mode for the specified axis.

FOLTRIG=expression1, . . . , expression8
Sets the trigger mode for all defined axes. If an axis is unchanged, a comma must be inserted in its place.

FOLTRIG(axis)
Returns the trigger mode for the specified axis.

EXAMPLES:

FOLTRIG(2)=0
Sets axis 2 trigger mode to no trigger required to start on command.

FOLTRIG=1,,2
Sets axis 1 to start cycle on event 1 trigger, axis 2 unchanged and axis 3 to start cycle on event 2 trigger.

FORMAT

FORMAT

ACTION: Enables or disables the formatting of the STR$ returned string.

PROGRAM SYNTAX: FORMAT=m,n,d

REMARKS: This command is used in conjunction with the STR$ command to set the format of the returned string.

The m specifies the format mode.

0	disable format
1	leading and trailing zero's will be returned in the string.
2	sign followed by leading spaces and trailing 0's will be returned in the string.

The n specifies the number of whole digits to be returned in the string. This number does not include the sign of the returned string. If the sign is positive a space will be inserted in place of the sign. If this value is 0 the whole number value will be ignored.

The d specifies the number of decimal digits to be returned in the string. If this value is 0 no decimal point will be returned and the fractional portion of the variable will be ignored.

FORMAT=0,n,d
Disables the format mode. No leading or trailing characters are inserted in the string.

If the number converted is outside the whole number digit the returned string will have * substitutions for the numbers.

EXAMPLES:

```
FORMAT=1,4,2             ‘ sign leading & trailing 0'S
ABSPOS(1)= -200.254
A$=STR$(ABSPOS(1))       ‘ A$ =“-0200.25"

FORMAT=2,4,4             ‘ sign leading space trailing 0's
ABSPOS(1)= -200.254
A$=STR$(ABSPOS(1))       ‘ A$ =“- 200.2540"

FORMAT=1,2,4             ‘Sign leading & trailing 0's
ABSPOS(1)= -200.254
A$=STR$(ABSPOS(1))       ‘ A$ =“-**.****"

FORMAT=0,1,1             ‘Disable formatting
```

FOR...NEXT...STEP

ACTION:

Repeats a block of statements a specified number of times.

PROGRAM SYNTAX:

```
FOR counter = start TO end [STEP increment]
        [statementblock]
[EXITFOR]
        [statementblock]
NEXT [counter]
```

REMARKS:

Counter is a variable used as the loop counter.
Start is the initial value of the counter.
End is the ending value of the counter.
Increment is the amount the counter is changed each time through the loop. If STEP is not specified, *increment* defaults to one.

If *end* is greater than *start* then *increment* must be positive. If *start* is greater than *end* then *increment* must be negative. If these conditions are not met the loop will not execute, control is transferred to the statement following the NEXT statement. If *start* equals *end* then the loop will execute once regardless of the *increment* value. If *increment* equals zero the loop will execute indefinitely.

EXITFOR is an alternative exit from a FOR...NEXT loop.

EXITFOR transfers control to the statement following the NEXT statement. When used within nested FOR...NEXT statements, EXITFOR transfers out of the immediately enclosing loop. EXITFOR can be used only in a FOR...NEXT statement.

EXAMPLES:

```
for x=1 to 8 step 1
        statements
next x
```

GETCHAR

GETCHAR

ACTION: Waits for and returns the ASCII code of the first character in the selected serial port's receiver buffer.

PROGRAM SYNTAX: GETCHAR(n) - used in an expression

REMARKS: The n specifies the number of serial port (1 or 2).
Port 1 is the Host port and Port 2 is the Auxiliary port.

Program execution is suspended while GETCHAR waits for a character to be received in the serial port receive buffer. If a character is already in the receive buffer GETCHAR returns that character's ASCII code immediately.

EXAMPLES: The serial port 1 receive buffer contains a single character "X".

```
A = GETCHAR(1)        ' sets A = 88, which is the ASCII code for  X
B = GETCHAR(1)        ' waits for next character, then sets B = its ASCII code
```

GOSUB...RETURN

GOSUB...RETURN

ACTION: Branches to, and returns from, a subroutine.

PROGRAM SYNTAX: GOSUB [linelabel]

REMARKS: You can call a subroutine any number of times in a program. You can call a subroutine from within another subroutine (nesting).

How deeply you can nest subroutines is limited only by the available stack space. Subroutines that call themselves (recursive subroutines) can easily run out of stack space.

The execution of the RETURN statement causes program execution to continue with the line immediately following the line that called the subroutine.

Subroutines can appear anywhere in the program, but it is good programming practice to make them readily distinguishable from the main program.

EXAMPLES:

```
GOSUB GET_CHAR        'go to subroutine at label "GET_CHAR"

GET_CHAR:             'label for subroutine
              :
      statement block 'statements to perform action of the subroutine
              :
RETURN                'return to program line following GOSUB GET_CHAR
```

See Section 7.3 for further examples of how to use this command

GOTO GOTO

ACTION: Branches unconditionally to the specified label.

PROGRAM SYNTAX: GOTO [label]

REMARKS: The GOTO statement provides a means for branching unconditionally to another label.

It is good programming practice to use subroutines or structured control statements (DO... UNTIL, FOR...NEXT, IF..THEN...ELSE) instead of GOTO statements, because a program with many GOTO statements can be difficult to read and debug. Avoid using "GOTO" !

EXAMPLES:

```
if x=1 then GOTO coolant_off
coolant_off:
        (statements)
```

HARDLIMIT HARDLIMIT

ACTION: Limit switch inputs can be enabled/disabled by this command and enable/disable status can be detected.

PROGRAM SYNTAX:

HARDLIMIT(axis) - used in an expression
HARDLIMIT=exp1, ... , exp8
HARDLIMIT(axis)=expression

REMARKS: The "axis" (1-8) selects the designated axis.

HARDLIMIT(axis) - used in an expression
Returns the HARDLIMIT enabled status for the designated axis.
A value of 1 is returned if enabled, and a value of zero is returned if disabled.

HARDLIMIT=exp1, ... , exp8
Sets the HARDLIMIT state for all designated axes. A "0" disables the +LIMIT and -LIMIT inputs of the designated axis. Any other value will enable the +LIMIT and -LIMIT of the designated axis.

HARDLIMIT(axis)=expression
Sets the HARDLIMIT state of the designated axis. A "0" disables the +LIMIT and -LIMIT of the designated axis. Any other value will enable the +LIMIT and -LIMIT of the designated axis.

EXAMPLES:

HARDLIMIT(1)=0
Disables the +LIMIT and -LIMIT of axis 1.

HARDLIMIT(1)=1
Enables the +LIMIT and -LIMIT of axis 1.

HARDLIMIT=0,1,,1,,0,1,0
Enables the +LIMIT and -LIMIT for axis 2,4 and 7. Disables the +LIMIT and -LIMIT for axis 1,6 and 8. Axis 3 and 5 unchanged.

a = HARDLIMIT(1)   ' Returns the present HARDLIMIT enabled status of axis 1.

HARDLIMNEG HARDLIMNEG

ACTION: Returns the -LIMIT hardware state for the selected axis.

PROGRAM SYNTAX: HARDLIMNEG(axis) - used in an expression

REMARKS: The "axis" (1-8) selects the axis to be returned. A false (0) or true (1) is returned. Limit switches can be configured as normally open or closed. The normal switch state (limit inactive) returns (0), .

EXAMPLES:

```
if HARDLIMNEG(1) then
        Statements ...
end if
```

HARDLIMPOS HARDLIMPOS

ACTION: Returns the +LIMIT hardware state for the selected axis.

PROGRAM SYNTAX: HARDLIMPOS(axis) - used in an expression

REMARKS: The "axis" (1-8) selects the axis to be returned. A false (0) or true (1) is returned. Limit switches can be configured as normally open or closed. The normal switch state (limit inactive) returns (0), .

EXAMPLES:

```
if HARDLIMPOS(1) then
end if
```

HEX$ HEX$

ACTION: Return the hex character's equivalent of the argument.

PROGRAM SYNTAX: A$=HEX$(expression)

REMARKS: The expression must be an integer value.

EXAMPLES:

```
a$=HEX$(255)
a$="FF"
```

HVAL

HVAL

ACTION: Returns the hex value of a string.

PROGRAM SYNTAX:

```
x=HVAL(A$)
```

REMARKS: A$ is the designated string variable or string literal.

The string variable format for the conversion is:
"0xHH" or "HH"
where: H is a hex character 0-F.

a decimal value is returned.

EXAMPLES:

```
x=HVAL("0xFF")
x is set to 255.

a$="1F"
x=HVAL(a$)
x is set to 31.
```

IF...THEN...ELSE...END IF

IF...THEN...ELSE...END IF

ACTION: Allows conditional execution based on the evaluation of a Boolean condition.

PROGRAM SYNTAX 1:

```
IF condition THEN thenpart [ELSE elsepart]
```

PROGRAM SYNTAX 2:

```
IF condition1 THEN
        [statementblock-1]
[ELSE
        [statementblock-2]]
END IF
```

REMARKS: The argument condition is an expression that SEBASIC evaluates as true (nonzero) or false (zero).

The argument statementblock includes any number of statements on one or more lines.

The argument thenpart includes the statements or branches performed when condition is true.

The argument elsepart includes the statements or branches performed when condition is false. The syntax is the same as thenpart. If the ELSE clause is not present, control passes to the next statement in the program.

EXAMPLES:

```
if x=0 then
        statement block
else
        statement block
end if
```

IN IN

ACTION: Returns the state of the specified digital I/O inputs.

PROGRAM SYNTAX: IN(nnn) - used in an expression
IN(nnn,len) - used in an expression

nnn is the I/O point terminal

```
nnn = (101-124) or (201-224) or (301-324) or (401-424)
        board 1       board 2       board 3       board 4
```

len is the number of I/O points.
len = 1 - 24

SINGLE INPUT PROGRAM SYNTAX: IN(nnn) returns the state (1 or 0) of input nnn.

EXAMPLE: IN(207) = 1, if input 207 is on.
IN(207) = 0, if input 207 is off.

MULTIPLE INPUT PROGRAM SYNTAX: IN(nnn,len) returns a number corresponding to the states of multiple inputs.
(binary weighting of inputs nnn to (nnn+len-1))

IN(nnn,len) is equivalent to:

IN(nnn) +2 IN(nnn+1) +4 IN(nnn+2) + ... + 2^{len-1} IN(nnn+len-1)

EXAMPLE: IN(207,3) is equivalent to:

IN(207) +2 IN(208) +4 IN(209) depending on the states of

inputs 207 - 209, IN(207,3) will return a number between 0 and 7.

INCHAR INCHAR

ACTION: Returns the ASCII code of the first character in the selected serial port's receive buffer.

PROGRAM SYNTAX: INCHAR(n) - used in an expression

REMARKS: The n specifies the number of the serial port (1 or 2).
Port 1 is the Host port and Port 2 is the Auxiliary port.

INCHAR checks the serial port receive buffer. If a character is already in the receive buffer INCHAR returns with that characters ASCII code. If no character is in the serial port receive buffer INCHAR returns with a zero.

EXAMPLES: The serial port 1 receive buffer contains a single character "A".

```
B = INCHAR(1)  'B = 65, which is the ASCII code for A.
C = INCHAR(1)  'C = 0, since no more characters are in the buffer.
```

#INCLUDE

#INCLUDE

ACTION: Includes a file name with define statement's in a user task.

COMMAND SYNTAX: #INCLUDE drive:\subdir\ ...\subdir\filename.inc

REMARKS: "Drive" is the root directory.

"Subdir\" is the path required to find the file.

"Filename" is the include filename with extension ".inc"

The include file must be a series of #DEFINE statements only. The **#INCLUDE** command with this file name can be used in any project task file.

The IWS.inc file is included in the MX2000 software for Windows. This file can be used to control a Superior Electric IWS-127-SE or IWS-30-SE interface panel.

EXAMPLES:

```
#INCLUDE c:\mx2000\iws.inc

#DEFINE GET_CHAR   @1       do: @1=inchar(COM) : until @1> 0

CLRSCRN             'Clear IWS interface screen
LINE_CURSOR         'IWS cursor is a flashing underline
DISP 1,5,"MX2000"   'display MX2000 starting at row 1 col 5 on IWS interface
                    screen.
LOCATE 1,12         'locate cursor at row 1 col 12
KYBRD_ON            'Enable Keyboard on IWS interface panel
ECHO_ON             'Display key pressed and send key code to MX2000
FLUSH_RX_BUF        'removes characters in reciever buffer
GET_CHAR a          'The ASCII code of the key pressed is returned in variable "a"
```

(#INCLUDE, cont'd.)

IWS.INC listing:

```
         SYMBOL           arguments   substitution code
#DEFINE  ESC                          CHR$(27)
#DEFINE  FALSE                        0
#DEFINE  TRUE                         1
#DEFINE  COM                  2
#DEFINE  FLUSH_RX_BUF                 do:KEYPRESS=INCHAR(COM):loop until KEYPRESS=0
#DEFINE  CURSOR_ON                    PRINT#COM,ESC+"[>5l";
#DEFINE  CURSOR_OFF                   PRINT#COM,ESC+"[>5h";
#DEFINE  SOLID_CURSOR                 PRINT#COM,ESC+"[>4h";
#DEFINE  LINE_CURSOR                  PRINT#COM,ESC+"[>4l";
#DEFINE  SAVE_CURSOR                  PRINT#COM,ESC+"[s";
#DEFINE  RESTORE_CURSOR               PRINT#COM,ESC"[u";
#DEFINE  BELL                         PRINT#COM,CHR$(7);
#DEFINE  TX_LINE                      PRINT#COM,ESC;"[lp";
#DEFINE  CLRSCRN                      PRINT#COM,CHR$(12);
#DEFINE  CLRLINE                      PRINT#COM,ESC+"[2K";
#DEFINE  LOCATE             @1,@2     PRINT#COM,ESC+"["+STR$(@1)+";"+STR$(@2)+"H";
#DEFINE  DISP              @1,@2,@3   LOCATE @1,@2,@3;
#DEFINE  ECHO_ON                      PRINT#COM,ESC+"[>14h";
#DEFINE  ECHO_OFF                     PRINT#COM,ESC+"[>14l";
#DEFINE  FLASHING_ON                  PRINT#COM,ESC+"[5m";
#DEFINE  FLASHING_OFF                 PRINT#COM,ESC+"[0m";
#DEFINE  KYBRD_ON                     PRINT#COM,ESC+"[2l";
#DEFINE  KYBRD_OFF                    PRINT#COM,ESC+"[2h";
#DEFINE  SET_TIME          @1,@2,@3   PRINT#COM,ESC+"["+STR$(@1)+";"+STR$(@2)+";" +STR$(@3) +"?s";
#DEFINE  SET_DATE          @1,@2,@3   PRINT#COM,ESC+"["+STR$(@1)+";"+STR$(@2)+";" +STR$(@3) +"?t";
#DEFINE  TIME_ON            @1,@2     LOCATE @1,@2: PRINT#COM,ESC,"[?15h";
#DEFINE  DATE_ON            @1,@2     LOCATE @1,@2: PRINT#COM,ESC;"[?14h";
#DEFINE  TIME_OFF                     PRINT#COM,ESC;"{15l";
#DEFINE  DATE_OFF                     PRINT#COM,ESC;"{14l";
```

(#INCLUDE, cont'd.)

IWS.INC Macros:

FLUSH_RX_BUF	Reads and dumps any character in the receive buffer.
CURSOR_ON	Makes cursor visible.
CURSOR_OFF	Makes cursor invisible.
SOLID_CURSOR	Makes cursor a solid block.
LINE_CURSOR	Makes cursor a flashing underline.
SAVE_CURSOR	Saves the current cursor position.
RESTORE_CURSOR	Moves cursor to last saved position.
TX_LINE	Transmit the current line from IWS to MX2000.
CLRSCRN	Home the cursor and clear the screen.
CLRLINE	Clear the current line.
LOCATE row,column	Locate cursor at the designated row and column.
DISP row,column,text	Move the cursor to the designated row and column and display the designated text.
ECHO_ON	Enables the echo mode, displays a key pressed and sends the ASCII code to the MX2000.
ECHO_OFF	Disable the echo mode, sends the ASCII code for the key pressed to the MX2000 only.
FLASHING_ON	All subsequent characters will be displayed as flashing characters.
FLASHING_OFF	All subsequent characters will be displayed normally.
KYBRD_ON	Enable keyboard, key presses will be processed.
KYBRD_OFF	Disable keyboard, key presses will not be processed.
SET_TIME hour,minute,second	Sets the clock (military time)in the IWS interface panel.
SET_DATE month,day,year	Sets the date in the IWS interface panel. The year entry is two digit only.
TIME_ON row,column	Enable display of time at row and column. Require a delay before disabeling.
DATE_ON row,column	Enable display of date at row and column. Require a delay before disabeling.
TIME_OFF	Disable display of time.
DATE_OFF	Disable display of date.

INPUT

INPUT

ACTION: Reads data from the selected serial port into the string variable list.

PROGRAM SYNTAX:

```
INPUT#1,n$[,][x$]        INPUT#1, var1$[,var2$] . . . [,var n$]
INPUT#2,n$[,][x$]        INPUT#2, var1$[,var2$] . . . [,var n$]
```

REMARKS: The string variables (e.g. n$, x$) are separated by commas.

This command accepts input characters until a carriage return or a linefeed is received.

Port 1 is the Host port and Port 2 is the Auxiliary port.

This command reads data into "string" variables, to use an input numeric value in a mathematical expression or in a function that requires numerical data. The string must be converted to a number with the "VAL" command, e.g., x = VAL (acc$).

EXAMPLES: The following command inputs string data into variables acc$, dcc$ and vel$ from the host port:

```
Input#1 acc$,dcc$,vel$
        Serial Port 1 input characters "100.25, 200.5, 10.1" CrLf
        acc$ = "100.25";     dcc$ = "200.5";     vel$ = "10.1"
```

INSTR INSTR

ACTION: Returns the character position of the first occurrence of a specified string in another string.

PROGRAM SYNTAX: INSTR(string1$,string2$) - used in an expression

REMARKS: The comparison is case sensitive. Returns a 0 if no match is found.

EXAMPLES:

a$ = "SE part # 215629"
a=INSTR(a$,"part #")
returns the starting position of "part #" in a$; in this case, the value 4 is returned.

INTLIM INTLIM

ACTION: Sets the Integral limit of a servo axis.

PROGRAM SYNTAX:

INTLIM(axis)=expression
INTLIM=expression1, . . . , expression8
INTLIM(axis)

REMARKS: The axis specifies the number for the axis (1-8).

The expression is the Integral Limit of the servo axis in volts.

The range for the Integral Limit is 0 to 10 volts.

INTLIM(axis)=expression
Sets the Integral limit of the specified axis to the expression value.

INTLIM=expression1, . . . , expression8
Sets the integral limit of the designated axes to the expression value.

INTLIM(axis)
Returns the integral limit value of an axis.

EXAMPLES:

INTLIM(2)=5
Sets the integral limit of axis 2 to 5 volts.

INTLIM=5, , , ,3
Sets the integral limit for axis 1 to 5 volts. Axes 2, 3 and 4 are unchanged. Axis 5 is set to 3 volts.

INTLIM(4)
Returns the integral limit for axis 4.

JOGSTART

JOGSTART

ACTION:

Runs an axis continuously in the specified direction.

PROGRAM SYNTAX:

```
JOGSTART(axis) = expression
JOGSTART = expression1, ..., expression8
```

REMARKS:

The "axis" specifies the number of the axis (1 - 8).

If expression is positive or 0, jogging will take place in the positive direction. If the expression is negative, jogging will take place in the negative direction. Otherwise the value of the expression does not matter. Speed of the jog move is determined by the last SPEED command for that axis. If no SPEED command was used, then jogging occurs at the speed set in the axis configuration menu during setup.

JOGSTART(axis)=expression
Starts jogging the motor for the specified axis.

JOGSTART=expression1, ..., expression8
Starts jogging the motor for all defined axis.

Use the JOGSTOP command for stopping the motor(s).

EXAMPLES:

JOGSTART(1)=+1
start axis 1 jogging in the positive direction.

JOGSTART=,-1,,,,,+1
starts axis 2 jogging in the negative direction and axis 7 jogging in the positive direction.

JOGSTOP

JOGSTOP

ACTION:

Stops jog motion for the specified axis.

PROGRAM SYNTAX:

```
JOGSTOP(axis) = expression
JOGSTOP=expression1,... , expression8
```

REMARKS:

The "axis" specifies the number of the axis (1 - 8).
The value of the expression does not matter.

EXAMPLES:

JOGSTOP=0,,0
generates a jog stop command for axis 1 and 3.

JOYSTICK..JOYSTICK END JOYSTICK..JOYSTICK END

ACTION:

Enables/Disables Joystick motion.

PROGRAM SYNTAX:

```
JOYSTICK = ax1, ax2
JOYSTICK END
```

REMARKS:

The joystick command sets up two axes, ax1 and ax2, to move in response to the voltage applied to their respective analog inputs (IN+ and IN-). Each axis will run at a speed proportional to the input voltage and in the direction determined by the polarity of the input voltage. There is a ±0.25 V dead band, therefore an input -0.25 and +0.25 volts will not cause motion.

Input Voltage	**Speed**	**Direction**
-10.0 to -0.25	$(\frac{Vin+0.25}{10})vel$	Minus
-0.25 to +0.25	0	—
+0.25 to +10.0	$(\frac{Vin-0.25}{10})vel$	Plus

vel is the velocity set by the velocity command. The joystick mode is in effect until a "joystick end" command is executed.

EXAMPLES:

The following example allows the user to manually position the axes to their home position with a joystick supplying voltage to the axes analog inputs. A switch connected to an input, signals when the home position is reached.

```
velocity = 10                       'velocity set to 10
joystick=1,2                        ' set joystick active for selected axes
do : loop until exin(100) = 1       ' allows joystick motion until button pressed
joystick end
abspos=0,0                          ' set absolute position to 0
```

Note: See Section 7.5 for a more detailed example of using a joystick.

KAFF KAFF

ACTION: Sets the acceleration feed forward gain for a servo axis.

PROGRAM SYNTAX:

KAFF(axis)=expression
KAFF=expression1, . . . , expression8
KAFF(axis)

REMARKS: The axis specifies the number for the axis (1-8).

The expression is the acceleration feed forward gain of the servo axis.
The **Units** are volts/encoder count/msec^2.

The expression value must be positive.

KAFF(axis)=expression
Sets the acceleration feed forward gain of the specified axis to the expression value.

KAFF=expression1, . . . , expression8
Sets the acceleration feed forward gain of the designated axes to the expression value.

KAFF(axis)
Returns the acceleration feed forward gain value of the specified axis.

EXAMPLES:

KAFF(2)=.5
Sets the acceleration feed forward gain value for axis 2 to .5 volts/encoder count/msec^2.

KAFF=.2, ,0
Sets the acceleration feed forward gain value for axis 1 to .2 volts/encoder count/msec^2.
Axis 2 is unchanged. Axis 3 is set to 0 volts/encoder count/msec^2.

KAFF(2)
Return the acceleration feed forward gain for axis 2.

KD

ACTION:

Sets the derivative gain for a servo axis.

PROGRAM SYNTAX:

KD(axis)=expression
KD=expression1, . . . , expression8
KD(axis)

REMARKS:

The axis specifies the number for the axis (1-8)

The expression is the derivative gain value of the servo axis.
The **Units** are milliseconds.

The expression value must be positive.

KD(axis)=expression
Sets the derivative gain of the axis to the expression value.

KD=expressio1, . . . , expression8
Sets the derivative gain of the designated axes to the expression value.

KD(axis)
Returns the derivative gain value of the specified axis.

EXAMPLES:

KD(2)=4
Sets the derivative gain of axis 2 to 4 milliseconds.

KD=4, , 4
Sets the derivative gain of axis 1 and axis 3 to 4 milliseconds. Axis 2 is unchanged.

KD(2)
Returns the derivative gain of axis 2.

KI

ACTION:

Sets the Integral gain of a servo axis.

PROGRAM SYNTAX:

KI(axis)=expression
KI=expression1, . . . , expression8
KI(axis)

REMARKS:

The axis specifies the number for the axis (1-8)

The expression is the integral gain value of the servo axis.
The **Units** are milliseconds.

The expression value must be positive.

KI(axis)=expression
Sets the integral gain of the specified axis to the expression value.

KI=expression1, . . . , expression8
Sets the integral gain of the designated axes to the expression value.

KI(axis)
Returns the integral gain value of the specified axis.

EXAMPLES:

KI(2)=1
Sets the integral gain for axis 2 to 1 millisecond.

KI=1, ,0
Sets the integral gain for axis 1 to 1 millisecond. Axis 2 is unchanged.
Axis 3 is set to 0 milliseconds.

KI(2)
Returns the integral gain for axis 2.

KP

ACTION:

Sets the proportional gain of a servo axis.

PROGRAM SYNTAX:

KP(axis)=expression
KP=expression1, . . . , expression8
KP(axis)

REMARKS:

The axis specifies the number for the axis (1-8).

The expression is the proportional gain value of the servo axis.
The **Units** are millivolts/encoder count.

The expression value must be positive.

KP(axis)=expression
Sets the proportional gain of the specified axis to the expression value.

KP=expression1, . . . , expression8
Sets the proportional gain of the designated axes to the expression value.

KP(axis)
Returns the proportional gain value of the specified axis.

EXAMPLES:

KP(2)=50
Sets the proportional gain for axis 2 to 50 millivolts/encoder count (.05 volts/encoder count).

KP=30, ,50
Sets the proportional gain for axis 1 to 30 millivolts/encoder count. Axis 2 is unchanged. Axis 3 is set to 50 millivolts/encoder count.

KP(2)
Return the proportional gain for axis 2.

KVFF

ACTION:

Sets the velocity feed forward gain value for a servo axis.

PROGRAM SYNTAX:

KVFF(axis)=expression
KVFF=expression1, ... , expression8
KVFF(axis)

REMARKS:

The axis specifies the number of the axis (1-8).

The expression is the velocity feed forward gain value of the servo axis. The **Units** are in percent. A value of 0 disable velocity feed forward gain. A Value of 100 is a good starting point for velocity feed forward.

The expression value must be positive.

KVFF(axis)=expression
Sets the velocity feed forward gain of axis to the expression value.

KVFF=expression1, ... , expression8
Sets the velocity feed forward gain of the designated axes to the expression value.

KVFF(axis)
Returns the velocity feed forward gain value of axis.

EXAMPLES:

KVFF(2)=100
Sets the velocity feed forward gain for axis 2 to 100%.

KVFF=98, ,99
Sets the velocity feed forward gain for axis 1 to 98% and axis 3 to 99%.

KVFF(2)
Returns the velocity feed forward gain for axis 2.

LCASE$ LCASE$

ACTION: Returns a string with all letters converted to lower case.

PROGRAM SYNTAX: string1$=lcase$(string2$)

REMARKS: String2$ is copied and all upper case letters are converted to lower case.

This command is useful for making the INSTR command case insensitive.

EXAMPLES: a$=lcase$(b$)
Converts all upper case letters to lower case and returns the string to a$.

```
a$ = "HELLO"
b$ = LCASE$(a$)          ' sets b$ = "hello"
```

LEFT$ LEFT$

ACTION: Returns a string consisting of the specified number of left-most characters in the indicated string.

PROGRAM SYNTAX: string2$=left$(string1$,n)

REMARKS: String2$ is the result, where n specifies the number of characters in string1$ to return. If n is greater than the length of the string then the entire string is returned.

EXAMPLES: a$=left$(b$,4)
Sets a$ equal to the first 4 characters of b$.

LEN LEN

ACTION: Returns the length of a string.

PROGRAM SYNTAX: LEN(string$) - used in an expression

REMARKS: LEN returns the length of the string. If the string is a null string LEN returns zero.

EXAMPLES: A = LEN("ABCD")   ' Returns A=4

LINE

LINE

ACTION: Initiates a coordinated linear move.

PROGRAM SYNTAX:

```
LINE=expression1,expression2, ... ,expression8 (normal)
LINE=expression1,expression2 (in a path)
```

REMARKS: The expression represents the move distance. All defined axes will start and stop at the same time.

In a path, expression1 is the move distance for first designated path axis and expression2 for the second.

EXAMPLES:

```
LINE=+1.0,,-2.0,3.0
```

Linearly interpolates axes 1,3 and 4. Axis 1 move +1.0 units, axis 3 moves -2.0 units, and axis 4 moves +3.0 units.

```
PATH=1,2
LINE=1,3.5
     statements...
PATH END
```

Linearly interpolates axes 1 and 2. Axis 1 move +1.0 units and axis 2 moves +3.5 units.

LOG

LOG

ACTION: Returns the natural logarithm of a numeric expression.

PROGRAM SYNTAX: LOG(expression) - used in an expression

REMARKS: The argument expression must be greater than zero. The natural logarithm is the logarithm to the base **e** . The constant **e** is approximately equal to 2.718282.

You can calculate base-10 logarithms as follows., Log10(x)=LOG(x)*.4342945

EXAMPLES:

```
result=LOG(2.718282)
```

Returns a 1 to result.

LOWSPD

ACTION: Sets or returns the low speed (starting speed) value of the specified axis.

PROGRAM SYNTAX:

```
LOWSPD(axis)=expression
LOWSPD=expression1,expression2, ... ,expression8
LOWSPD(axis) -  used in an expression
```

REMARKS: The "axis" specifies the number of the axis (1-8).

LOWSPD(axis)=expression
Sets the low speed value for the specified axis

LOWSPD=expression1,expression2, ... ,expression8
Sets the low speed value for all defined axes.

LOWSPD(axis) - used in an expression
Evaluates and returns the low speed value of the specified axis.

EXAMPLES:

LOWSPD(1)= 4
Sets the low speed value of axis 1 to 4 units.

LOWSPD=4,4,,,10,10,5,5
Sets the low speed value for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

MAXSPD MAXSPD

ACTION:

Sets or returns the maximum allowed speed of the specified axis.
Motion will not be performed at speeds higher than this value,
even if an axis is programmed or commanded to do so.

PROGRAM SYNTAX:

MAXSPD(axis)=expression
MAXSPD=expression1,expression2, ... ,expression8
MAXSPD(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).

MAXSPD(axis)=expression
Sets the maximum speed value for the specified axis

MAXSPD=expression1,expression2, ... ,expression8
Sets the maximum speed value for all defined axes.

MAXSPD(axis) - used in an expression
Evaluates and returns the maximum speed value of the specified axis.

EXAMPLES:

MAXSPD(1)= 4
Sets the maximum speed value of axis 1 to 4 units.

MAXSPD=4,4,,,10,10,5,5
Sets the maximum speed value for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

MAXSPD(2)
returns the maximum speed value of axis 2.

MID$ MID$

ACTION:

Returns the designated middle number of characters of a string.

PROGRAM SYNTAX:

string1$=mid$(string2$,start,number)

REMARKS:

The **start** specifies the starting position in the string. If the **start** is greater than the length of the string or the **start** value is negative a "" string is returned.

The **number** specifies the number of characters to return. If the **number** is greater than the length of the string, the string returned starts at the start value through the end of the string. If the **number** is negative a "" string is returned.

EXAMPLES:

```
a$ = "P/N_123AC"
b$ = mid$(a$,5,3)          ' Sets b$ equal to "123"
```

MOD MOD

ACTION: Return the remainder of a number divided by the base.

PROGRAM SYNTAX: y=x MOD base

REMARKS: The y is the remainder.

The x is number that is divided by the base.

The base is the divisor.

EXAMPLES: y= 31 MOD 16
y is set to 15 which is the remainder.

MOVE MOVE

ACTION: Initiate a non-coordinated move.

PROGRAM SYNTAX: MOVE(axis)=expression
MOVE=expression1, expression2, ..., expression8

REMARKS: The "axis" value specifies the number of the axis (1 - 8).

The expression represents the distance to move. The sign of the expression determines the direction (positive or negative) of motion for the move.

EXAMPLES: MOVE(1)= -1.0
moves axis 1 -1.0 units.

MOVE=,2.0,3.0
moves axis 2 +2.0 units and axis 3 +3.0 units.

MOVEHOME

ACTION:

Runs an axis motor until the home input is activated, captures and registers this position as home (electrical zero), then stops.

PROGRAM SYNTAX:

MOVEHOME(axis)=expression
MOVEHOME=expression1, expression2, ..., expression8

REMARKS:

The "axis" specifies the number of the axis (1 - 8).

The sign of the expression determines the direction (positive or negative) of motion for the home cycle. The non-zero value of the expression is not significant.

EXAMPLES:

MOVEHOME(1)= -1.0
initiates a mechanical home cycle for axis 1 in the negative direction.

MOVEHOME=,+1.0,,1.0
initiates a mechanical home cycle for axis 2 & 4 in the positive direction.

MOVEREG

ACTION:

Runs an axis motor until the registration input is activated; then without stopping, moves that axis the desired registration distance.

PROGRAM SYNTAX:

MOVEREG(axis)=expression
MOVEREG=expression1, expression2, ..., expression8

REMARKS:

The "axis" specifies the number of the axis (1 - 8).

The expression represents the distance to move after the input has been activated. The sign of the expression determines the direction (positive or negative) of motion for the mark registration cycle.

The distance must be an incremental distance.

EXAMPLES:

MOVEREG(1)=+1.0
initiates a registration cycle for axis 1 in the positive direction.

MOVEREG=,-1.0,,2.0
initiates a registration cycle for axis 2 in the negative direction and axis 4 in the positive direction.

NOT NOT

ACTION: The logical NOT operator is used in boolean expressions that appear as conditions IF, DO, and LOOP statements.

PROGRAM SYNTAX: result = NOT expression

REMARKS: The NOT operator uses the "truth table":
The result is TRUE if the expression is FALSE

expression	condition result
True	False
False	True

EXAMPLES: Loop while (NOT (x=1 AND y=0))

NVR NVR

ACTION: The NVR array is used for non-volatile variable storage. Up to 2048 variables can be saved.

PROGRAM SYNTAX:

```
NVR(number)
NVR(number)=expression
```

REMARKS: The NVR array has 2048 elements (1-2048) and is accessible by all program tasks.

number is the NVR element number (1-2048).

expression must be a numeric value.

To start the NVR array to zero, run the following program.

```
   FOR X=1 TO 2048
      NVR(X)=0
NEXT X
```

EXAMPLES:

A=NVR(2)
Return the NVR element 2 value to variable A.

NVR(2048)=10.5
Sets the NVR element 2048 to a value of 10.5

NVR(3)=A
Sets the NVR element 3 to the value of variable A.

ACTION: The logical OR operator is used in boolean expressions that appear as conditions in IF, DO, and LOOP statements.

PROGRAM SYNTAX: expression1 OR expression2

REMARKS: The OR operator uses this "truth Table":
The result is TRUE, if either expression is TRUE.

Expression1	Expression2	Condition Result
True	True	True
True	False	True
False	True	True
False	False	False

EXAMPLES: Do until (A > 5 OR X = 0)

OUT OUT

ACTION: Sets or returns the state of the specified digital I/O outputs.

PROGRAM SYNTAX:

OUT(nnn) = expression
OUT(nnn) (used in an expression)
OUT(nnn,len) = expression
OUT(nnn,len) (used in an expression)

nnn is the I/O point number.

```
nnn =  (101-124) or  (201-224) or  (301-324) or  (401-424)
         board 1         board 2       board 3       board 4
```

Outputs x01-x16 are physical outputs, x17-x24 are internal flags which can be set and read just as the physical outputs.

len is the number of I/O points

len = 1 - 24

SET SINGLE OUTPUT PROGRAM SYNTAX: OUT(nnn) = expression

"expression" turns output nnn on (expression is non-zero) or off (expression =0).

EXAMPLE:

```
OUT(207) = 1          turns output 207 on.
OUT(207) = 0          turns output 207 off.
```

READ SINGLE OUTPUT PROGRAM SYNTAX: OUT(nnn) (used in an expression)
evaluates to the last output commanded (1 or 0) for this I/O pin. Note this is different from the state of the I/O pin.

EXAMPLE:

```
OUT(207) = 3          turns output 207 on.
A = OUT(207)          A = 1 (last commanded output for 207)
```

SET MULTIPLE OUTPUTS PROGRAM SYNTAX: OUT(nnn,len) = expression
The expression is evaluated and converted to an integer value. The least significant "len" bits of the binary representation are then used to set outputs "nnn" to "nnn+len-1" respectively.

EXAMPLE:

```
OUT(216,3) = 6.2        sets output 216 and flags 217, 218

    6.2 converted to integer 6
    binary representation of 6 is    1 1 0
                                     | | └──       output 216 = off
                                     | └───        flag 217   = on
                                     └────         flag 218   = on
```

(continued on next page)

(OUT, cont'd)

READ MULTIPLE OUTPUTS
PROGRAM SYNTAX: OUT(nnn,len) (used in an expression)

evaluates to a number corresponding to the last outputs commanded (1 or 0) for these I/O pins. The number is the binary weighted sum of last commanded outputs nnn to (nnn+len-1). Note this is different from the state of the I/O pins.

EXAMPLE: OUT(207,3) = 4 binary rep. of 4 = 1 0 0

output 207 = off
output 208 = off
output 209 = on

If the last output for 208 = 0 and 209 = 1, then the expression :
A=OUT(208,2) evaluates to A = 2 since binary 2 = 1 0

last output for 208 = 0
last output for 209 = 1

OUTLIMIT OUTLIMIT

ACTION: Sets or returns the maximum analog ouput voltage allowed on a servo axis.

PROGRAM SYNTAX:

OUTLIMIT(axis)=expression
OUTLIMIT=expression1, ... ,expression8
expression=OUTLIMIT(axis)

REMARKS: This command only applies to a servo axis and clamps the peak level of the commanded analog output voltage to the servo.

Axis is the axis number (1 to 8).

exp is a value (1.0 to 10.0). This value is in volts.

OUTLIMIT(axis)=expression
Sets the OUTLIMIT for the selected axis equal to the expression.

OUTLIMIT=expression1, ... , expression8
Sets the OUTLIMIT for the designated axes equal to the corresponding expression.

expression=OUTLIMIT(axis)
Returns the OUTLIMIT value for the selected axis to the expression.

EXAMPLES:

OUTLIMIT(1)=5
Sets the analog output limit for a servo drive (axis 1) to 5 volts.

OUTLIMIT=,10
Sets the analog output limit for a servo drive (axis 2) to 10 volts. Axis 1 is unchanged.

PATH .. PATH END

PATH .. PATH END

ACTION:

Specifies a continuous motion path.

PROGRAM SYNTAX:

```
PATH=axis1,axis2
    EXOUT(nnn)=expression
     LINE=axis1 distance,axis2 distance
    EXOUT(nnn,len)=expression
    FEEDRATE=expression
    ARC=axis1center,axis2center,±angle
    OUT(nnn)=expression
    POINT=axis1 distance,axis2 distance
    OUT(nnn,len)=expression
PATH END
```

REMARKS:

Axis1 and axis2 are the axes used in the path. The commands LINE , ARC, POINT, EXOUT, OUT, FEEDRATE and RADIUS are the only commands allowed in a PATH..PATHEND block of statements. Path motion proceeds from one segment (LINE, ARC or POINT) to another without stopping. The path speed can be changed from segment to segment with the FEEDRATE command. Outputs can be set in various segments with standard output commands. When two consecutive segments are lines, then a radius is inserted if the last RADIUS command specified a non-zero radius.

When PATH statements are used in each task, a maximum of 100 points are allowed per PATH...PATHEND block. Multiple, consecutive PATH's are allowed within a task; however, motion stops between PATH's. NOTE: Up to 700 points may be used to specify a single PATH under the following conditions:

1. The PATH...PATHEND block must be in the first task and,
2. No other tasks may contain PATH or ARC statements.

EXAMPLES:

```
PATH=1,2
  LINE=1.5,3
  EXOUT(101) = 1
  ARC=3,0,+360
  EXOUT(101) =0
PATH END
```

The above example will move from the present position to position (1.5,3) using LINE motion. EXOUT(101) is set, a 360 degree ARC is executed and then EXOUT(101) is cleared.

POINT POINT

ACTION: Specify coordinates, which the motors will move through in a path.

PROGRAM SYNTAX: POINT=expression1,expression2

REMARKS: This command is only valid between PATH and PATHEND statements, where expression1 is the first axis coordinate, expression2 the second axis coordinate. The path connecting points is smooth.

EXAMPLES:

```
PATH=1,2
  POINT=1.5,3
  POINT=4,5
  POINT=6,7
PATH END
```

The above example will move the axes from the present position, through points (1.5,3) and (4,5) to position (6,7) smoothly. The point data can be incremental or absolute as set by the POSMODE command.

POSMODE POSMODE

ACTION: Sets or returns the positioning mode for the specified axis.

PROGRAM SYNTAX:

```
POSMODE(axis) = expression
POSMODE=expression1, expression2, ..., expression8
POSMODE(axis)  -  used in an expression
```

REMARKS: The "axis" specifies the number of the axis (1 - 8).

If expression is true(1) then the absolute positioning mode is enabled. If expression is false(0) then the incremental positioning mode is enabled. Incremental positioning is the default mode.

POSMODE(axis) = expression
Sets the positioning mode for the specified axis.

POSMODE = expression1, expression2, ..., expression8
Sets the positioning mode for all defined axis.

POSMODE(axis) - used in an expression
Evaluates and returns the positioning mode for the specified axis.

EXAMPLES: POSMODE(1)=1 or POSMODE(1) = TRUE
Sets axis 1 to absolute positioning mode.

POSMODE=0,0,,,1,1,0,1
Sets axes 1, 2 and 7 to incremental positioning mode. Axes 5, 6 and 8 to absolute positioning mode. Axes 3 and 4 are unchanged.

PRINT

ACTION:

Outputs data to the designated serial port.

PROGRAM SYNTAX:

```
PRINT#1,[expression][ , or ; ][expression][ , or ; ]
PRINT#2,[expression][ , or ; ][expression][ , or ; ]
```

REMARKS:

Port 1 is the Host port and Port 2 is the Auxiliary port.

expression can be a variable or String variable or Literal string. Literal strings must be enclosed in quotation marks.

If a comma "," is used between expressions four spaces will separate expressions.

If a semicolon ";" is used between expressions there will be no spaces between expressions.

Up to 20 expressions can be used with one PRINT command.

If a ";" is used at the end of the Print command, no carriage-return / line-feed sequence will be generated.

The PRINT#1 or PRINT#2 command should be used in one task, but not necessarily the same task. If a message from another task must be printed pass the message using a COMMON string variable and flag variable.

EXAMPLES:

```
acc=10.5 dcc=2.1
Print#1,"accel= ";acc,"decel= ";dcc
     Outputs on Host serial port :
     accel= 10.5    decel= 2.1

acc=10.5 dcc=2.1
Print#2,"accel= ";acc,"decel= ";dcc
     Outputs on Auxiliary serial port :
     accel= 10.5    decel= 2.1

a$="accel= "  b$="decel= "  acc=10.5  dcc=2.1
Print#1,a$;acc,b$;dcc
     Outputs on Host serial port #1:
     accel= 10.5    decel= 2.1
```

ACTION:

Sets or returns the acceleration/deceleration profile for the specified axis.

PROGRAM SYNTAX:

PROFILE(axis)=expression
PROFILE=expression1,expression2, ... ,expression8
PROFILE(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).

PROFILE(axis)=expression
Sets the acceleration/deceleration profile for the specified axis

PROFILE=expression1,expression2, ... ,expression8
Sets the acceleration/deceleration profile for all defined axes.

PROFILE(axis) - used in an expression
Evaluates and returns the acceleration/deceleration profile of the specified axis.

The acceleration/deceleration profiles are:

value	description
0	trapezoidal profile
1	S Curve profile

EXAMPLES:

PROFILE(1)= 0
Sets the acceleration/deceleration profile of axis 1 to trapezoidal.

PROFILE=0,0,,,1,1,1,0
Sets the acceleration/deceleration profile for axis 1,2 and 8 to trapezoidal. Axis 5,6 and 7 to S curve. Axis 3 and 4 are unchanged.

RADIUS

RADIUS

ACTION: Sets or returns the ARC radius for Path blending.

PROGRAM SYNTAX:

```
RADIUS=expression
expression=RADIUS
```

REMARKS: Blending only occurs between lines in a path.

The first syntax type sets the ARC radius for Path blending equal to the expression. The second syntax type (expression=RADIUS) returns the current value of Radius.

EXAMPLES:

```
X=RADIUS
```

sets X equal to the current RADIUS value.

```
RADIUS=.25
```

Sets the RADIUS for Path blending to .25 units.

```
Path=1,2
    radius=.125
    Line=1,1
    Line=2,-.5     ' blending occurs
    Line=3,-.5     ' blending occurs
Path end
```

READ

READ

ACTION: Reads numbers from data statements and assigns them to the variables in the list. The data statements are contained in the BASIC program. Refer to the DATA statement description for more detail.

PROGRAM SYNTAX:

```
READ variable,variable,etc
```

REMARKS: All numbers in the data statements are floating point numbers. The DATA statement must always appear ahead of the READ statement.

EXAMPLES:

```
DATA 1,2,3,4
READ a,b,c,d
```

Reads next four values from the data statement into variables a,b,c and d.

REDUCE REDUCE

ACTION:

Enables, disables, sets or returns the Reduce Current Feature or returns the reduce enable status for the specified axis. When enabled, the stepper drive "Reduce" output turns on when there is no motion. This causes the drive to reduce the motor current 50%.
Note: Requires compatible drive.

PROGRAM SYNTAX:

REDUCE(axis)=expression
REDUCE=expression1,expression2, ... ,expression8
REDUCE(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).

If the expression is true (non-zero) then the REDUCE feature is enabled for the specified axis. If the expression is false (zero) then the REDUCE feature is disabled for the specified axis.

REDUCE(axis)=expression
Enables or disables the motor reduce current feature for the specified axis.

REDUCE=expression1,expression2, ... ,expression8
Enables or disables the motor reduce current feature for all defined axis.

EXAMPLES:

REDUCE(7)=1
Enables the REDUCE feature for axis 7.

REDUCE=1,1,,0,0,0,1,0
Enables the REDUCE feature for axis 1,2,7. Disables the feature for axis 4,5,6,8. Axis 3 is unchanged.

REM or ' REM or '

ACTION:

Allows source code comments to be inserted in the program.

PROGRAM SYNTAX:

REM or '

REMARKS:

all text to the right of REM or ' to the end of the line is not considered part of the command during execution.

EXAMPLES:

```
accel(1)=10.2     REM axis 1 acceleration=10.2 units   <- these are REMARKS
decel(1)=5.4      'axis 1 deceleration=5.4 units       <- these are REMARKS, too
```

RESET

RESET

ACTION: Resets the system.

COMMAND SYNTAX: RESET

REMARKS: This command causes the system to halt, and then restart as though power had been recycled. This command can be used to start a different project , as selected by the SEL1 SEL2 and SEL4 inputs on the DSP Controller card.

RESTORE

RESTORE

ACTION: Allows DATA statements to be read again.

PROGRAM SYNTAX:

```
RESTORE
RESTORE(number)
```

REMARKS: Sets the pointer for DATA statements to the start (0) or to a designated position (number).

EXAMPLES:

```
Restore(10)
```

Sets the pointer for DATA statements to position 10.
The first variable in the next READ statement will be loaded with element 10.

RIGHT$

RIGHT$

ACTION: Returns the specified number of rightmost characters in a string.

PROGRAM SYNTAX:

```
string2$=RIGHT$(string1$,n)
```

REMARKS: String2$ is the output. The n specifies the number of characters in string 1$ to return. If n is greater than the length of the string then the entire string is returned.

EXAMPLES:

```
a$=RIGHT$(b$,4)
```

Sets a$ equal to the last 4 characters of b$.

SETCOM SETCOM

ACTION: Sets the baud rate and data format for Auxiliary serial port.

PROGRAM SYNTAX: SETCOM#n,baud,parity,data,stop

REMARKS: The variable "n" signifies the port number. Presently, only the second serial port (the Auxiliary Port) is supported, therefore only a value of 2 is valid for "n".

The baud rate can be any value up to 38,400.

parity setting:

0	no parity
1	odd parity
2	even parity

data

7	7 bit data length
8	8 bit data length

stop

1	1 stop bit
2	2 stop bits

If the inputs are outside the above setting the command will be ignored and an error warning will be issued.

EXAMPLES: SETCOM#2,9600,0,8,1

Sets Auxiliary port to 9600 baud, no parity, 8 bit data and 1 stop bit.

SHIFT SHIFT

ACTION: Shifts the elements of a single-dimension numeric array up or down.

PROGRAM SYNTAX: SHIFT (array,n)

REMARKS: n is the number of shifts to perform on the array. If n is a positive number, the array is shifted up and the top elements are discarded. If n is a negative number the array is shifted down and the bottom elements are discarded. Zeroes are shifted into the array.

EXAMPLES: This example illustrates the effect of shift commands on a 4-element array "x".

x(0)	x(1)	x(2)	x(3)	
1	2	3	4	x before shift command
0	1	2	3	x after SHIFT(x,1)
2	3	4	0	x after SHIFT(x,-1)

SIGN SIGN

ACTION: Returns the sign of an expression.

PROGRAM SYNTAX: SIGN(expression) - used in an expression

REMARKS: If expression is positive, the SIGN function returns 1.

If expression is zero, the SIGN function returns 0.

If expression is negative, the SIGN function returns -1.

EXAMPLES:

SIGN(-10.0)
Evaluates to -1.

SIGN(10)
Evaluates to 1.

SIGN(0)
Evaluates to 0.

SIN SIN

ACTION: Returns the sine of the angle x, where x is in radians.

PROGRAM SYNTAX: SIN(x) - used in an expression

REMARKS: To convert values from degrees to radians, multiply the angle (in degrees) times π/180 (or .017453) where π = 3.141593.

To convert a radian value to degrees, multiply it by 180/π (or 57.295779).

EXAMPLES:

```
conv = 3.141593 / 180     ' converts degrees to radians
A = SIN (conv * 45)       ' A = sin (45 degrees ) or .7071
```

SOFTLIMIT

SOFTLIMIT

ACTION:

Enables/disables or returns the SOFTLIMIT enable state for the selected axis.

PROGRAM SYNTAX:

SOFTLIMIT(axis) - used in an expression
SOFTLIMIT=expression1, ... , expression8
SOFTLIMIT(axis)=expression

REMARKS:

The "axis" (1-8) selects the designated axis.

SOFTLIMIT(axis) - used in an expression
Evaluates to and returns an enabled (1) or disabled (0) SOFTLIMIT state for the designated axis.

SOFTLIMIT(axis)=expression
Sets the SOFTLIMIT state of the designated axis. A "0" disables the SOFTLIMPOS and SOFTLIMNEG softlimits of the designated axis. Any other value will enable the SOFTLIMPOS and SOFTLIMNEG softlimits of the designated axis.

SOFTLIMIT=expression1, ... , expression8
Sets the SOFTLIMIT state of the designated axes. A "0" disables the SOFTLIMPOS and SOFTLIMNEG softlimits of the designated axis. Any other value will enable the SOFTLIMPOS and SOFTLIMNEG softlimits of the designated axis.

EXAMPLES:

SOFTLIMIT(1)=0
Disables the SOFTLIMPOS and SOFTLIMNEG softlimits of axis 1.

SOFTLIMIT(1)=1
Enables the SOFTLIMPOS and SOFTLIMNEG softlimits of axis 1.

SOFTLIMIT=,1,0
Enables the SOFTLIMPOS and SOFTLIMNEG softlimits of axis 2 and disables the SOFTLIMPOS and SOFTLIMNEG softlimits of axis 3.

a=SOFTLIMIT(1)   Returns the current SOFTLIMIT state of axis 1.

SOFTLIMNEG

SOFTLIMNEG

ACTION:

Programmable "software limit switch" for motion in the negative direction.
Sets or returns the absolute negative travel position value for the specified axis.

PROGRAM SYNTAX:

SOFTLIMNEG(axis)=expression
SOFTLIMNEG=expression1, ... ,expression8
SOFTLIMNEG(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).
If during motion the absolute position becomes less
than its limit, the move is aborted.

SOFTLIMNEG(axis)=expression
Sets the absolute travel distance for the specified axis.

SOFTLIMNEG=expression1, ... ,expression8
Sets the absolute travel distance for all defined axes.

SOFTLIMNEG(axis) - used in an expression
Evaluates and returns the absolute software travel distance of the specified axis.

EXAMPLES:

SOFTLIMNEG(1) = -4
Sets the absolute software travel distance of axis 1 to -4 units.

SOFTLIMNEG=4,4,,,10,10,5,5
Sets the absolute software travel distance for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

SOFTLIMPOS SOFTLIMPOS

ACTION:

Programmable "software limit switch" for motion in the positive direction.
Sets or returns the absolute positive travel position value for the specified axis.

PROGRAM SYNTAX:

SOFTLIMPOS(axis)=expression
SOFTLIMPOS=expression1, ... ,expression8
SOFTLIMPOS(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1-8).
If during motion the absolute position becomes greater than its limit, the move is aborted.

SOFTLIMPOS(axis)=expression
Sets the absolute travel distance for the specified axis.

SOFTLIMPOS=expression1,expression2, ... ,expression8
Sets the absolute travel distance for all defined axes.

SOFTLIMPOS(axis) - used in an expression
Evaluates and returns the absolute travel distance of the specified axis.

EXAMPLES:

SOFTLIMPOS(1) = -4
Sets the absolute travel distance of axis 1 to -4 units.

SOFTLIMPOS=4,4, , ,10,10,5,5
Sets the absolute travel distance for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

SPEED SPEED

ACTION:

Sets or returns the target velocity of an axis.

PROGRAM SYNTAX:

SPEED(axis) = expression
SPEED=expression1,expression2,..., expression8
speed(axis) - used in an expression (note lower case OK for command name)

REMARKS:

The "axis" specifies the number of the axis (1 - 8).

SPEED(axis) = expression
Sets the velocity for the specified axis.

SPEED=expression1, expression2, ..., expression8
Sets the velocity for all defined axes.

SPEED(axis) - used in an expression
Evaluates and returns the velocity of the specified axis.

EXAMPLES:

SPEED=3.0
Sets the velocity for axis 1 to 3.0 units.

SPEED=4,4,,,10,10,5,5
Sets the velocity for axes 1 and 2 to 4 units. Axes 5 and 6 to 10 units. Axes 7 and 8 to 5 units. Axes 3 and 4 are unchanged.

SQRT SQRT

ACTION: Returns the square root of the expression.

PROGRAM SYNTAX: SQRT(expression) - used in an expression

REMARKS: The expression must be greater than or equal to zero, or an error occurs.

EXAMPLES:

```
x = SQRT(16)
```

Sets x equal to the value 4, which is the square root of the number 16.

STR$ STR$

ACTION: Returns a string representation of a numeric expression.

PROGRAM SYNTAX: String1$=STR$(numeric_expression)

REMARKS: The STR$ command is the complement of a VAL command.

EXAMPLES:

```
a$ = STR$(accel(1))
```

Returns the ASCII string representing the accel value of axis 1.

```
b = 1000
a$ = STR$ (b)
```

Returns ASCII string "1000" to variable a$.

STRINGS$ STRINGS$

ACTION: Returns a string of characters.

PROGRAM SYNTAX: String1$=STRING$(num,code)

REMARKS: The num indicates the length of the string to return.

The code is the ASCII code of the character to use to build the string.

EXAMPLES:

```
a$=STR$(10,63)
```

returns 10 question marks to a$.

TAN TAN

ACTION: Returns the tangent of the angle x, where x is in radians.

PROGRAM SYNTAX: TAN(x) - used in an expression

REMARKS: To convert values from degrees to radians, multiply the angle (in degrees) times π/180 (or .017453) where π = 3.141593.

To convert a radian value to degrees, multiply it by 180/π (or 57.295779).

EXAMPLES:

```
PI = 3.141593          ' assign the constant "pi"
x = TAN (PI/4)         ' calculate tangent of 45 degrees
```

Sets x equal to the value 1.0, which is the tangent of 45 degrees.

TIMER

TIMER

ACTION: Sets or reads the timer.

PROGRAM SYNTAX:

```
TIMER  - used in an expression
TIMER=expression
```

REMARKS: Each task has a unique timer.

When used in an expression, TIMER returns the timer value (in seconds).

The TIMER=expression command sets the timer to a specific time.

The timer should be set to a value at the beginning of the task in which it is used.

The timers are incremented every millisecond.

EXAMPLES:

```
TIMER=0                 ' set the timer to zero

Do

while timer < 1.0       ' do the loop for 1 second
```

UCASE$

UCASE$

ACTION: Returns a string with all letters converted to upper-case.

PROGRAM SYNTAX:

```
string1$=ucase$(string2$)
```

REMARKS: String2$ is copied and all lower-case letters are converted to upper-case.

This command is useful for making the INSTR command case insensitive.

EXAMPLES:

```
b$ = "Hello"
a$=ucase$(b$)
```

Converts all lower-case letters to upper-case and returns the string to a$, so a$ is set to "HELLO".

VAL VAL

ACTION: Returns the floating point value of the designated string variable.

PROGRAM SYNTAX: VAL(n$) - used in an expression

REMARKS: n$ is the designated string variable.

The string variable format for conversion is:
[sign]digits[.digits[e or E[sign]integer]

The sign and scientific notions are optional.

Only numeric values are returned. The first character that cannot be part of the number terminates the string. If no digits have been processed, a value of zero is returned. (see the second example).

EXAMPLES:

```
a$ = "134 Main St."
x = VAL(a$)            'returns x = 134

b$ = "p/n 221"
y = VAL(b$)            'returns y = 0
```

VELOCITY VELOCITY

ACTION: Sets or returns the path speed to be used for coordinated motion.

PROGRAM SYNTAX: VELOCITY = expression
VELOCITY - used in an expression

REMARKS: This velocity is used in the LINE, ARC, and PATH commands.

EXAMPLES: VELOCITY=10.1
Sets the coordinated velocity for a LINE or ARC command to 10.1 units/sec.

k1 = VELOCITY
Sets the variable k1 equal to the value of VELOCITY used in the program.

WAIT WAIT

ACTION: Waits for the period of time (expressed in seconds) to expire before continuing.

PROGRAM SYNTAX: WAIT = expression

REMARKS: Program execution is suspended until the desired time has elapsed.

EXAMPLES: WAIT = 1.1
Wait 1.1 seconds and then continue.

WAITDONE

WAITDONE

ACTION: Waits for motion to be done for the specified axes. "Done" means motion is complete.

PROGRAM SYNTAX:

WAITDONE(axis)
WAITDONE=expression1, expression2,....., expression8

REMARKS: The "axis" specifies the number of the axis (1-8).

WAITDONE(axis)
Waits for motion to complete on the specified axis before program execution continues.

WAITDONE=expression1, expression2,....., expression8
Waits for motion to complete on all specified axes before program execution continues.

An alternate way to accomplish the WAITDONE function is as follows:

```
DO: LOOP WHILE BUSY(1)              Waits until axis 1 motion is completed.
```

EXAMPLES:

WAITDONE = 1
Waits for axis 1 motion to be complete before continuing program execution .

WAITDONE = 3,3,,6,6,5,5,3
Waits for axis 1, 2, 4, 5, 6, 7, and 8 motion to be complete before continuing program execution. The motion status of axis 3 is ignored.

WARNING

WARNING

ACTION: Sets or returns the task Warning count.

PROGRAM SYNTAX:

expression=WARNING
WARNING=expression

REMARKS: result=WARNING returns the current warning count.

WARNING=expression sets the Warning count equal to the expression. The expression must be a whole number.

EXAMPLES:

X=WARNING
set X equal to the current warning count.

WARNING=10
sets the warning count equal to 10.

Note: Also see the ERR command.

WNDGS WNDGS

ACTION:

Enables or disables the windings off function, or returns the WNDGS OFF status for the specified axis. When enabled, the stepper drive "WNDGS" output turns on when there is no motion. This causes the drive to turn off the motor current.
Note: requires a compatible drive.

PROGRAM SYNTAX:

WNDGS(axis)=expression
WNDGS=expression1,expression2, ..., expression8
WNDGS(axis) - used in an expression

REMARKS:

The "axis" specifies the number of the axis (1 - 8).
If the expression is true then the WNDGS feature is enabled for the specified axis. If the expression is false then the WNDGS feature is disabled for the specified axis.

WNDGS(axis) = expression
Enables or disables the motor windings for the specified axis.

WNDGS = expression1, expression2, ..., expression8
Enables or disables the motor windings feature for all defined axis.

EXAMPLES:

WNDGS(1)=1
axis 1 windings off mode active

WNDGS=0,,1
axis 1 windings on mode active, axis 2 unchanged and axis 3 windings off mode active.

a=WNDGS(1)
set variable a to the windings mode of axis 1.

XOR XOR

ACTION:

The logical XOR operator is used in boolean expressions that appear as conditions in IF, DO, and LOOP statements.

PROGRAM SYNTAX:

expression1 XOR expression2

REMARKS:

The XOR operator uses this "truth table": The result is TRUE if both expressions are TRUE or both expressions are FALSE.

expression1	expression2	Condition result
True	True	False
True	False	True
False	True	True
False	False	False

EXAMPLES:

Loop While (A = 1 XOR B = 1)

SECTION 6.4.2

HOST COMMANDS

REFERENCE GUIDE

6.4.2 Host Commands

One method of operating the SLO-SYN MX2000 controller is to program it via a PC using the commands detailed in the previous section, then set it up as a stand-alone system. In that case, after it is programmed, the MX does not need to communicate with any outside computer system. However, another method of system operation involves connecting the MX to some type of "host" computer, via the "HOST RS-232" or "HOST RS-485" port. Then, that computer may directs it's operation and query its status from time to time, if desired. To do this, you use the "Host Commands" detailed below. **Note: items placed in quotes (e.g., "?") are keypresses or ASCII characters.**

Host Commands Grouped by Function

Motion

ABSPOS	Sets or returns the absolute position of an axis.
ACCEL	Sets or returns the acceleration value of an axis.
ARC	Initiate a coordinated motion to move in an arc.
BUSY	Returns the motion status of the axis.
DECEL	Sets or returns the deceleration value of the axis.
ENCBAND	Sets or returns the position error band of the axis.
ENCERR	Returns the position error of the axis.
ENCFOL	Sets or returns the following error limit of the axis.
ENCMODE	Sets or returns the encoder closed loop operating mode of the axis.
ENCPOS	Returns the encoder position of the axis.
ENCRES	Sets or returns the encoder line count of the axis.
EVENT1	Returns the EVENT1 state for the selected axes.
EVENT2	Returns the EVENT1 state for the selected axes
HARDLIMNEG	Returns the HARDLIMNEG state of an axis.
HARDLIMPOS	Returns the HARDLIMPOS state of an axis.
JOGSTART	Runs an axis continuously in the specified direction.
JOGSTOP	Stops continuous Run.
LINE	Initiates a coordinated linear move.
LOWSPD	Sets or returns the low speed value of the axis.
MAXSPD	Sets or returns the maximum allowed speed of the axis.
MOVE	Initiates a non coordinated move .
MOVEHOME	Runs motor until the home input is activated.
MOVEREG	Runs motor until the registration input is activated .
POSMODE	Sets or returns the positioning mode of an axis.
PROFILE	Sets or returns the acceleration/deceleration profile of an axis.
SOFTLIMNEG	Sets or returns the software absolute negative travel distance for the axis.
SOFTLIMPOS	Sets or returns the software absolute positive travel distance for the axis.
SPEED	Sets or returns the axis speed to be used for non-coordinated motion.
STOP	Stops all motion and all tasks.
VELOCITY	Sets or returns the target velocity (path speed) for coordinated motion.
WNDGS	Enables or disables the windings off function, or returns the windings off status for the specified axis.

Input-Output Control

ANALOG	Sets or returns a numeric value representing the voltage on the analog port.
BCD	Returns the value on the BCD switches on the expansion I/O board.
DRVREADY	Allows checking of the drive ready (READY) on the axis card to be bypassed.
EXIN	Returns the state of the inputs on the expansion I/O board.
EXOUT	Sets or returns the state of the outputs on the expansion I/O board.
FILTER	Sets the filter value for the defined analog input.
IN	Returns the state of the inputs on the digital I/O board.
OUT	Sets or returns the state of the outputs on the digital I/O board.
OUTLIMIT	Sets or return the maximum analog ouput voltage allowed on a servo axis.

Program Control

"<n"	Activates/deactivates a controller from accepting commands from a host.
"?"	Requests the space remaining in the Host Receiver buffer.
"BACKSPACE"	The Backspace key (or ASCII code 08) can be used to delete one character from the host serial port receiver buffer.
DIR	Lists the names of projects and Tasks stored in non-volatile memory.
ERASE	Erases a specific project or all projects stored in non-volatile memory.
ERR	Returns the error status number for each task (0-7).
ERRM	Returns an error message.
"ESC"	The Escape key (ASCII 27) is used during program execution to designate that a command in the host serial port buffer is to be executed.
FREE	Transfers the free space in sectors, available in non-volatile memory.
FREEMEM	Returns the amount of free memory for program execution allocation.
LOAD	Loads the designated project from non-volatile to operating memory.
RESET	Resets the system after a watchdog timeout, as though power was cycled.
REVISION	Returns the current revision of the operating system software.
RUN	Runs the loaded project or specified task number.
STOP	Stops all tasks and all motion.

Miscellaneous

AXISBRD	Sets or returns the number of axis boards in the system.
NVR	Array used for non-volatile variable storage. (1-2048)
UNIT	Sets or returns the pulses/ unit value of an axis.
WARNING	Returns the program execution warning count of each task.
"XON","XOFF"	The XON/XOFF serial communication protocol command characters; "XON" is ASCII 17 and "XOFF" is ASCII 19).
"Ctrl C"	Stops all motion (same as STOP) - press "Control" and "C" keys simultaneously (this is ASCII 03).

Servo Gains and Limits

INTLIM	Sets or returns the servo axis integral limit.
KAFF	Sets the velocity feed forward gain value for a servo axis.
KD	Sets or returns the servo axis derivative gain.
KI	Sets or returns the servo axis integral gain.
KP	Sets or returns the servo axis proportional gain.
KVFF	Sets or returns the servo axis velocity feed forward gain.

Host Command Summary (alphabetical list)

Note: items placed in quotes (e.g., "?") are keypresses or ASCII characters.

"<n" activates/deactivates a controller from accepting commands from a host.
"?" Requests the space remaining in the Host Receiver buffer.
"CTRL C" Stops all motion and all tasks - same as STOP command.

A

ABSPOS Sets or returns the absolute position of an axis.
ACCEL Sets or returns the acceleration value of an axis.
ANALOG Sets or returns the analog voltage on an axis card.
ARC Initiate a coordinated motion to move in an arc.
AXISBRD Sets or returns the number of axis cards in the system.

B

"BACKSPACE" The Backspace key (or ASCII code 08) can be used to delete one character from the host serial port receiver buffer.
BCD Returns the value on the BCD switches on the expansion I/O board.
BUSY Returns the motion status of the axis.

C

"CTRL C" Stops all motion and all tasks - same as STOP command.

D

DECEL Sets or returns the deceleration value of the axis.
DIR Lists the names of projects and Tasks stored in non-volatile memory.
DRVREADY Allows checking of the drive ready (READY) on the axis card to be bypassed.

E

ENCBAND Sets or returns the position error band of the axis.
ENCERR Returns the position error of the axis.
ENCFOL Sets or returns the following error limit of the axis.
ENCMODE Sets or returns the encoder closed loop operating mode of the axis.
ENCPOS Returns the encoder position of the axis.
ENCRES Sets or returns the encoder line count of the axis.
ERASE Erases a specific project or all projects stored in non-volatile memory.
ERR Returns the error number.
ERRM Returns an error message.
"ESC" The Escape key or character (ASCII 27) is used during program execution to designate that a command in the host serial port buffer is to be executed.
EVENT1 Returns the EVENT1 state for the selected axes.
EVENT2 Returns the EVENT1 state for the selected axes
EXIN Returns the state of the inputs on the expansion I/O board.
EXOUT Sets or returns the state of the outputs on the expansion I/O board.

F	
FREE	Transfers the free space in sectors, available in non-volatile memory.
FREEMEM	Returns the amount of free memory for program execution allocation.
FILTER	Sets the filter value for the defined analog input.
H	
HARDLIMNEG	Returns the -LIMIT hardware state of an axis.
HARDLIMPOS	Returns the +LIMIT hardware state of an axis.
I	
IN	Returns the state of the inputs on the digital I/O board.
INTLIM	Sets or returns the servo axis integral limit.
J	
JOGSTART	Runs an axis continuously in the specified direction.
JOGSTOP	Stops continuous Run.
K	
KAFF	Sets the velocity feed forward gain value for a servo axis.
KD	Sets or returns the servo axis derivative gain.
KI	Sets or returns the servo axis integral gain.
KP	Sets or returns the servo axis proportional gain.
KVFF	Sets or returns the servo axis velocity feed forward gain.
L	
LINE	Initiates a coordinated linear move.
LOAD	Loads the designated project from non-volatile memory into operating memory.
LOWSPD	Sets or returns the low speed value of the axis.
M	
MAXSPD	Sets or returns the maximum allowed speed of the axis.
MOVE	Initiates a non coordinated move.
MOVEHOME	Runs motor until the home input is activated.
MOVEREG	Runs motor until the registration input is activated .
N	
NVR	Array used for non-volatile variable storage. (1-2048)
O	
OUT	Sets or returns the state of the outputs on the digital I/O board.
OUTLIMIT	Sets or return the maximum analog ouput voltage allowed on a servo axis.
P	
POSMODE	Sets or returns the positioning mode of an axis.
PROFILE	Sets or returns the acceleration/deceleration profile of an axis.

R	
RESET	Resets the system after a watchdog timeout.
REVISION	Returns the current revision of the operating system software.
RUN	Runs the loaded project or specified task number.
S	
SOFTLIMNEG	Sets or returns the software absolute negative travel distance value for the axis.
SOFTLIMPOS	Sets or returns the software absolute positive travel distance value for the axis.
SPEED	Sets or returns the axis target velocity for non-coordinated motion.
STOP	Stops all tasks and all motion immediately.
U	
UNIT	Sets or returns the pulses/ unit value of an axis.
V	
VELOCITY	Sets or returns the path speed to be used for coordinated motion.
W	
WARNING	Returns the program execution warning count of each task.
WNDGS	Enables or disables the windings off function, or returns the windings off status for the specified axis.
X	
"XON/XOFF"	The XON/XOFF serial communication protocol command characters.

Alphabetical list of all HOST commands with syntax and examples

Note: items placed between quotes (e.g., "?") are keypresses or ASCII characters to be sent via a serial port and the term "cr" means the carriage return key.

"<n" "<n"

ACTION: This command activates/deactivates a controller from accepting commands from a host computer.

COMMAND SYNTAX:

```
<n  or <n cr
<n?
<0  or <0 cr
```

REMARKS: In order to daisy chain multiple controllers to communicate with a single host, each controller must be given a unique identification number. The Unit ID # selector switch defines the identification number of the control. This switch is interrogated on power turn on only. The factory setting is device 1.

Each Controller must be given a unique identification (1-9) before the system is wired. See Section 5.1.1.

In order to accept commands from a host device, a Control must be set to the active mode. To do this, the host must send the device attention command (<) followed by the device identification followed by a carriage return, line feed or non-numeric character. If n matches the controller id number, that unit becomes the active controller.

If the host requires an acknowledgement of the active controller the <n? command is transmitted by the host and if the device exists it will respond with its id number.

If all controllers are to be placed in the listen mode the host issues a <0cr command. No data can be transferred from the Control to the host in this mode. However, all other commands will be honored by the controllers.

"?" "?"

ACTION: The ? key (or ? character code, ASCII 63, sent via a serial port) requests the space remaining in the Host Receiver buffer.

COMMAND SYNTAX: ?

REMARKS: The controller receiver buffer is 255 character long.

ABSPOS ABSPOS

ACTION: Sets or returns the absolute position of an axis.

COMMAND SYNTAX:

ABSPOS(n)=number cr
ABSPOS=number1, number2, . . . , number8 cr
ABSPOS cr
ABSPOS(n) cr

REMARKS: The n specifies the axis (1-8). "Number" is the value of ABSPOS to be set.

ABSPOS(n)=number
Sets the absolute position for the specified axis.

ABSPOS=number1, number2, . . . , number8
Sets the absolute position for all defined axis.

ABSPOS
returns the absolute position for all axes.

ABSPOS(n)
returns the absolute position for the specified axis.

EXAMPLES:

ABSPOS(3)=2
sets axis 3 absolute position to 2 units.

ABSPOS=3,3, ,6,6,5,5,3
sets the absolute position for axis 1, 2 and 8 to 3 units. Axis 4 and 5 to 6 units. Axis 6 and 7 to 5 units. Axis 3 remains unchanged.

ABSPOS(3)
returns the absolute position for axis 3.

ACCEL ACCEL

ACTION:

Sets or returns the acceleration value of an axis.

COMMAND SYNTAX:

```
ACCEL(n)=number  cr
ACCEL=number1, number2, . . . , number8  cr
ACCEL(n) cr
ACCEL cr
```

REMARKS:

The n specifies the axis (1-8).

ACCEL(n)=number
Sets the acceleration rate for the specified axis.

ACCEL=number1, number2, . . . , number8
Sets the acceleration rate for all defined axis.

ACCEL
returns the acceleration rate for all axes.

ACCEL(n)
returns the acceleration rate for the specified axis.

EXAMPLES:

ACCEL(3)=2
Sets axis 3 acceleration rate to 2 units/sec^2.

ACCEL=3,3, ,6,6,5,5,3
Sets the acceleration rate for axis 1,2 and 8 to 3 units/sec^2. Axis 4 and 5 to 6 units/sec^2 and axis 6 and 7 to 5 units/sec^2. Axis 3 remains unchanged.

ACCEL(1)
returns the acceleration value of axis 1.

ANALOG ANALOG

ACTION: A numeric value representing the voltage on the analog port is returned

PROGRAM SYNTAX:

ANALOG(b0n) cr
ANALOG(b0n)=number cr

REMARKS: The b specifies the board number (1-4).

The n specifies the analog input (1-4) or output(1-2).

ANALOG(b0n)
Returns the present value(+10.0 volts to -10.0 volts) of the specified analog input.

ANALOG(b0n)=number
Sets the analog output equal to the number. The values allowed are +10.0 volts to -10.0 volts.

Board Analog Configuration

b0n value	A inputs differential B inputs differential	A inputs single ended B inputs differential	A inputs differential B inputs single ended	A inputs single ended B inputs single ended
101	board 1 Ain+ & Ain-	board 1 Ain+	board 1 Ain+ & Ain-	board 1 Ain+
102	board 1 Bin+ & Bin-	board 1 Bin+ & Bin-	board 1 Bin+	board 1 Bin+
103		board 1 Ain-		board 1 Ain-
104			board 1 Bin-	board 1 Bin-
201	board 2 Ain+ & Ain-	board 2 Ain+	board 1 Ain+ & Ain-	board 2 Ain+
202	board 2 Bin+ & Bin-	board 2 Bin+ & Bin-	board 1 Bin+	board 2 Bin+
203		board 2 Ain-		board 2 Ain-
204			board 1 Bin-	board 2 Bin-
301	board 3 Ain+ & Ain-	board 3 Ain+	board 1 Ain+ & Ain-	board 3 Ain+
302	board 3 Bin+ & Bin-	board 3 Bin+ & Bin-	board 1 Bin+	board 3 Bin+
303		board 3 Ain-		board 3 Ain-
304			board 1 Bin-	board 3 Bin-
401	board 4 Ain+ & Ain-	board 4 Ain+	board 1 Ain+ & Ain-	board 4 Ain+
402	board 4 Bin+ & Bin-	board 4 Bin+ & Bin-	board 1 Bin+	board 4 Bin+
403		board 4 Ain-		board 4 Ain-
404			board 1 Bin-	board 4 Bin-

EXAMPLES: X=ANALOG(102)
Sets x equal to the present voltage on board 1 input 2.

ANALOG(102)=2.5
Sets the voltage on board 1 output 2 equal to +2.5 volts

ARC

ARC

ACTION: Initiates a coordinated motion to move in an arc.

PROGRAM SYNTAX: ARC=x,y,xcenter,ycenter,±angle

REMARKS: The x specifies the axis number for one of the axis.

The y specifies the axis number for the other axis.

Xcenter specifies the x axis coordinate of the arc center.

Ycenter specifies the y axis coordinate of the arc center.

The angle specifies the direction of rotation as well as the arc angle.
This angle is in degrees. Clockwise rotation is +.

EXAMPLES: ARC=1,2,3,0,+180
Initiates a 180° arc motion, using axis 1 and 2, with a 3 unit radius in the + direction.

AXISBRD

AXISBRD

ACTION: Sets or returns the number of axis cards in the system.

COMMAND SYNTAX: AXISBRD cr
AXISBRD=number cr

REMARKS: The AXISBRD command returns the current value for the number of axis cards.

The AXISBRD=number command is only honored if the project directory is empty, DIR command return no project names. The number (1-4) sets the number of axis cards in the system.

This value is altered when a project is loaded into active ram and reflects the number of axis defined in a project.

The Power-on default with no projects is 1.

The Current value determines the maximum number of axes to be returned during a Host command.

EXAMPLES:

AXISBRD cr	returns the current value of axis cards.
AXISBRD=4 cr	sets the current value of axis cards to 4
ABSPOS cr	returns the Absolute Position for 8 axes

"BACKSPACE"

"BACKSPACE"

ACTION: The Backspace key or ASCII code 08 can be used to delete one character from the host receiver buffer.

COMMAND SYNTAX: BACKSPACE (ASCII 08)

BCD

BCD

ACTION: Returns the number set on a BCD switch bank connected to an expansion I/O board.

COMMAND SYNTAX: BCD(b0n) cr

REMARKS: The b specifies the expansion board number (1-4).

The n specifies the BCD switch bank number, 1-8 on a full expansion board and 1-4 on the power supply board. See Section 5.3 for further details regarding connection of BCD switch banks.

EXAMPLES: BCD (101) returns the setting of BCD switch bank 1 connected to expansion board 1.

BUSY

BUSY

ACTION: Returns the motion status of the selected axis. An axis is "busy" if motion is occurring.

COMMAND SYNTAX:

```
BUSY(n) cr
BUSY cr
```

REMARKS: The n specifies the axis number (1-8).

If the commanded motion is incomplete BUSY(n) returns a true (1) otherwise BUSY(n) returns a false (0).

The BUSYcr returns the motion busy status of all axes.

EXAMPLES:

```
BUSY(1) cr
```

returns the motion status of axis 1.

"CTRL C" "CTRL C"

ACTION: Stops all motion and all tasks.

COMMAND SYNTAX: Simultaneously press the keyboard keys marked "C" and the control key "CTRL".

REMARKS: "CTRL C" will stop execution of all tasks presently running on the controller; all motion ceases immediately.

If the axis is a servo axis "CTRL C" turns off the servo output voltage. To turn the servo output back on use the "WNDGS(axis)=1 " command.

DECEL DECEL

ACTION: Sets or returns the deceleration value of the selected axis.

COMMAND SYNTAX:

```
DECEL(n)=number cr
DECEL=number1, number2, . . . , number8 cr
DECEL(n) cr
DECEL cr
```

REMARKS: The n specifies the axis (1-8). "Number" is the value of DECEL to be set.

DECEL(n)=number
Sets the deceleration for the specified axis.

DECEL=number1, number2, . . . , number8
Sets the deceleration for all defined axis.

DECEL
returns the deceleration for all axes.

DECEL(n)
returns the deceleration for the specified axis.

(continued on next page)

EXAMPLES:

DECEL(2)=3.1
sets the deceleration value of axis 2 to 3.1 units/sec^2.

DECEL=3,3, ,6,6,5,5,3
Sets the deceleration rate for axis 1,2 and 8 to 3 units/sec^2. Axis 4 and 5 to 6 units/sec^2 and axis 6 and 7 to 5 units/sec^2. Axis 3 remains unchanged.

DECEL(2)
returns the deceleration value of axis 2.

ACTION: List the names of projects and tasks stored in non-volatile memory.

COMMAND SYNTAX: DIR cr

REMARKS: The transfer format is:

```
n*s Project name  checksum
     Task name
     Task name
    etc
n*s Project name  checksum
    Task name
    Task name
    etc
free space = nnnn
```

where the n is the project number (0 to 6); the * indicates that this project is loaded into DSP card memory; the s indicates that the project source code is loaded; and the checksum number is a unique value for that project.

Note: If the CLR input is open circuited at power-on no projects will be loaded into the DSP memory.

EXAMPLES: DIR cr
transfers the names of the user projects and tasks stored in memory.

DRVREADY DRVREADY

ACTION: Allows the checking of the drive ready (READY) signal on the axis card to be bypassed.

PROGRAM SYNTAX:

```
DRVREADY cr
DRVREADY=0xhh cr
DRVREADY=number cr
```

REMARKS:

DRVREADY
Returns the current state of the drive ready inhibit. Two hex values are returned. The first hex value represent axis 8 thru axis 5. The second hex value represents axis 4 thru axis 1.

The hex value weight for each axis are shown below.

1st hex value				2nd hex value			
axis 8	axis 7	axis 6	axis 5	axis 4	axis 3	axis 2	axis 1
8	4	2	1	8	4	2	1

DRVREADY=0xhh
Sets the state of the drive ready inhibit. A 1 in the hex weighted bits bypasses the drive ready signal check for that axis. The hh represents the two hex values.

DRVREADY=number
Sets the state of the drive ready inhibit. A 1 in the decimal weighted bits bypasses the drive ready signal check for that axis. The decimal weights for each axis are shown below.

axis 8	axis 7	axis 6	axis 5	axis 4	axis 3	axis 2	axis 1
128	64	32	16	8	4	2	1

The drive ready of an axis is only checked if motion is to occur in this axis. If the drive ready signal is checked and the drive is not ready an error 33 will occur and the cycle will be aborted.

EXAMPLES:

DRVREADY=0x0f or DRVREADY=0xf
Bypasses the drive ready signal check for axis 1 thru axis 3 and enables the drive ready check for axis 4 thru axis 8.

DRVREADY=15
Bypasses the drive ready signal check for axis 1 thru axis 3 and enables the drive ready check for axis 4 thru axis 8.

ENCBAND

ENCBAND

ACTION:

Sets or returns the maximum position error allowed when motion is stopped, referred to herein as "position error band."

COMMAND SYNTAX:

ENCBAND(n)=number cr
ENCBAND=number1, number2, . . . , number8 cr
ENCBAND(n) cr
ENCBAND cr

REMARKS:

The n specifies the axis (1-8). "Number" is the value of ENCBAND to be set.
Position error = absolute position - encoder position.

ENCBAND(n)=number
Sets the position error band for axis n.

ENCBAND=number1, number2, . . . , number8
Sets the position error band for all defined axes.

ENCBAND(n)
Returns the current position error band of axis n.

ENCBAND
Returns the current position error band of all axes.

EXAMPLES:

ENCBAND(1)=.4 Sets the position error band of axis 1 to .4 units.

ENCBAND=.4,.4, , ,1,1,.5,.5 Sets the position error band for axis 1 and 2 to .4 units. Axis 5 and 6 to 1 units. Axis 7 and 8 to .5 Units. Axis 3 and 4 are unchanged.

ENCBAND(2) returns the position error band of axis 2.

ENCERR

ENCERR

ACTION:

Returns the position error (absolute position - encoder position) of the selected axis.

COMMAND SYNTAX:

ENCERR(n) cr
ENCERR cr

REMARKS:

The n specifies the number of the axis (1-8).

ENCERR(n) Returns the present position error of axis n.

ENCERR Returns the present position error for all axes.

EXAMPLES:

ENCERR(1)
returns the present position error of axis 1.

ENCFOL

ACTION:

Sets or returns the maximum position error allowed during motion, herein referred to as "following error."

COMMAND SYNTAX:

ENCFOL(n)=number cr
ENCFOL=number1, number2, . . . , number8 cr
ENCFOL(n) cr
ENCFOL cr

REMARKS:

The n specifies the axis (1-8). "Number" is the value of ENCFOL to be set.
Position error = absolute position - encoder position.

ENCFOL(n)=number
Sets the following error for axis n.

ENCFOL=number1, number2, . . . , number8
Sets the following error for all defined axes.

ENCFOL(n)
Returns the current following error of axis n.

ENCFOL
Returns the current following error of all axes.

EXAMPLES:

ENCFOL(1)=.4
Sets the following error of axis 1 to .4 units.

ENCFOL=.4,.4, , ,1,1,.5,.5
Sets the following error for axis 1 and 2 to .4 units. Axis 5 and 6 to 1 units. Axis 7 and 8 to .5 Units. Axis 3 and 4 are unchanged.

ENCFOL(2) Returns the following error for axis 2.

ENCMODE ENCMODE

ACTION:

Sets or returns the encoder closed loop operating mode of the selected axis.

COMMAND SYNTAX:

ENCMODE(n)=number cr
ENCMODE=number1, number2, . . . , number8 cr
ENCMODE(n) cr
ENCMODE cr

REMARKS:

"Number" corresponds to the closed-loop operating modes, as follows:

0	closed loop disabled - operates open loop
1	halt execution on excessive following error
2	correct position on excessive following error
3	restart move on excessive following error

The n specifies the number of the axis (1-8).

ENCMODE(n)=number
Sets the encoder closed loop operating mode for the specified axis.

ENCMODE=number1, number2, . . . , number8
Sets the encoder closed loop operating mode for all defined axes.

ENCMODE(n)
Returns the current encoder closed loop operating mode of axis n.

ENCMODE
Returns the current encoder closed loop operating mode of all axes.

EXAMPLES:

ENCMODE(1)=0
Sets axis 1 to open loop operation.

ENCMODE=0,0,1,1,0,1,1,1
Sets axis 1,2 and 5 to open loop operation. Axis 3,4,6,7 and 8 to halt execution on excessive following error.

ENCMODE(1)
returns the closed loop operating mode number of axis 1.

ENCPOS ENCPOS

ACTION: Returns the encoder position of the selected axis.

COMMAND SYNTAX: ENCPOS(n) cr
ENCPOS cr

REMARKS: The n specifies the number of the axis (1-8).

ENCPOS(n)
Returns the present encoder position of axis n.

ENCPOS
Returns the present encoder position of all axes.

EXAMPLES: ENCPOS(2)
returns the encoder position of axis 2.

ENCRES ENCRES

ACTION: Returns the encoder line count of the selected axis.

COMMAND SYNTAX: ENCRES(n)=number cr

REMARKS: The n specifies the axis (1-8).

ENCRES(n) Returns the current line count of axis n.

ENCRES Returns the current line count of all axes.

EXAMPLES: ENCRES(2) returns the encoder line count of axis 2.

ERASE ERASE

ACTION: Erases a specific project or all projects stored in non-volatile memory.

COMMAND SYNTAX:

```
ERASE DIR cr
ERASE project name cr
```

REMARKS: The project erased is not recoverable.

ERASE DIR Erases all projects stored in non-volatile memory.

ERASE project name
Erases the defined project name in non_volatile memory.

EXAMPLES:

ERASE DIR Erases all projects stored in non-volatile memory.

ERASE CONVEYOR Erases project CONVEYOR if it exists.

ERR ERR

ACTION: Returns the MX controller error status number for each task (0-7).

COMMAND SYNTAX:

```
ERR cr
```

REMARKS: This command returns the error status for all tasks and clears the errors. See the ERR program command description, Section 6.4.1 for further details.

ERRM

ACTION: Returns an error message.

COMMAND SYNTAX: ERRM cr

REMARKS: The error message are:

Error	Text
0	no Errors
1	+Limit activated
2	-Limit activated
3	Soft Limit in +dir
4	Soft Limit in -dir
5	CL attempts
6	CL Fol Err Mode 1
7	CL Fol Err Coord Motion
8	CL Fol Err JOG Mode 2
9	Warn Axis not in task
10	Warn ANALOG I/O range
11	Warn BCD range
12	Warn ENCMODE range
13	Warn EXIN range
14	Warn EXOUT range
15	Warn IN range
16	Warn OUT range
17	Warn LOG value<=0
18	Warn SQRT arg negative
19	Warn NVR range
20	Warn Read out of arg
21	Warn ACCEL range
22	Warn DECEL range
23	Warn LOWSPD range
24	Warn SPEED range
25	Warn MAXSPD range
26	Warn PROFILE range
27	Warn ENCMODE range
28	Warn VELOCITY range
29	Warn RADIUS range
30	motion at program end
31	SETCOM error
32	Warn Servo Gain range
33	DRVREADY fault
34	Servo Following Error
35	Warn OUTLIMIT range
36	Program out of memory
>36	User define Error nnn

EXAMPLE:

```
ERRM cr     'with error 34 set in task 1
            Task1 Servo Following Error
```

"ESC"

"ESC"

ACTION: The ESC key (or ESC character code sent via a serial port) is used during program execution to force execution of a command in the host buffer.

COMMAND SYNTAX: ESC (ASCII 27) command

REMARKS: When the controller is executing a BASIC program, any host commands received are qued for execution after the BASIC program finishes. The execution of a host command can be forced to happen immediately by preceding it with the ESC character (ASCII 27). The command will consist of all characters from the ESC to the cr (carriage return). Multiple commands can be placed on one line, but they must be separated by colons (:).

EXAMPLE: ESC ABSPOS(1) cr
Returns the absolute position of axis 1 during program execution.

EVENT1

EVENT1

ACTION: Returns the EVENT1 state for the selected axes.

PROGRAM SYNTAX: EVENT1 cr
EVENT1(n) cr

REMARKS: The n specifies the axis to return

EVENT1
Returns the EVENT1 state of all axes. A return of a 1 indicates the EVENT1 input is active.

EVENT1(n)
Returns the EVENT1 state of axis n. A return of a 1 indicates the EVENT1 input is active.

EXAMPLES: EVENT1(2)
Returns the EVENT1 state of axis 2.

EVENT2 EVENT2

ACTION: Returns the EVENT2 state for the selected axes.

PROGRAM SYNTAX:

EVENT2 cr
EVENT2(n) cr

REMARKS: The n specifies the axis to return

EVENT2
Returns the EVENT2 state of all axes. A return of a 1 indicates the EVENT2 input is active.

EVENT2(n)
Returns the EVENT2 state of axis n. A return of a 1 indicates the EVENT2 input is active.

EXAMPLES:

EVENT2(2)
Returns the EVENT2 state of axis 2.

EXIN EXIN

ACTION: Returns the state of the specified expansion I/O inputs.

COMMAND SYNTAX:

EXIN(nnn) cr
EXIN(nnn,len) cr

nnn is the I/O point terminal

nnn =	(100-147) or	(200-247) or	(300-347) or	(400-447)
	board 1	board 2	board 3	board 4

len is the number of I/O points.

len = 1 - 24

SINGLE INPUT COMMAND SYNTAX: EXIN(nnn) cr returns the state (1 or 0) of INPUT nnn.

EXAMPLE:

EXIN(207) cr returns 1, if input 207 is on.
EXIN(207) cr returns 0, if input 207 is off.

MULTIPLE INPUT COMMAND SYNTAX: EXIN(nnn,len) cr returns a number corresponding to the states of multiple inputs: (binary weighting of inputs nnn to (nnn+len-1))

$$INPUT_{nnn} +2\ INPUT_{nnn+1} +4\ INPUT_{nnn+2}+ \ldots + 2^{len-1}\ INPUT_{nnn+len-1}$$

EXAMPLE: EXIN(207,3) cr will return: $INPUT_{207} +2\ INPUT_{208} +4\ INPUT_{209}$ which is a number between 0 and 7, depending on the states of INPUTS 207-209.

EXOUT

ACTION: Sets or returns the state of the specified expansion I/O outputs.

COMMAND SYNTAX:

```
EXOUT(nnn) = number cr
EXOUT(nnn) cr
EXOUT(nnn,len) = number cr
EXOUT(nnn,len) cr
```

nnn is the I/O point terminal

```
nnn =  (100-147) or  (200-247) or  (300-347) or  (400-447)
         board 1       board 2       board 3       board 4
```

len is the number of I/O points; len = 1 - 24

"Number" turns an output on (number is non-zero) or off (number =0).

SET SINGLE OUTPUT COMMAND SYNTAX: EXOUT(nnn) = number cr

turns output nnn on (number is non-zero) or off (number =0).

EXAMPLE:

```
EXOUT(207) = -3 cr      turns output 207 on.
EXOUT(207) =  0 cr      turns output 207 off.
```

READ SINGLE OUTPUT COMMAND SYNTAX: EXOUT(nnn) cr

Returns the last commanded output (1 or 0) for this I/O pin.
Note this is different from the state of the I/O pin.

EXAMPLE:

```
EXOUT(207) = 1 cr       turns output 207 on.
EXOUT(207) cr           returns 1, the last commanded output for 207
```

SET MULTIPLE OUTPUTS COMMAND SYNTAX: EXOUT(nnn,len) = number cr

The number is converted to an integer value. The least significant "len" bits of the binary representation are then used to set outputs "nnn" to "nnn+len-1" respectively.

EXAMPLE:

```
EXOUT(207,3) = 6.2 cr    sets output 207 - 209
```

6.2 converted to integer 6 binary representation of 6 is:

```
1 1 0
| | |___      output 207 = off
| |_____      output 208 = on
|_______      output 209 = on
```

EXOUT, (cont'd)

READ MULTIPLE OUTPUTS
COMMAND SYNTAX: EXOUT(nnn,len) cr

Evaluates to a number corresponding to the last commanded outputs (1 or 0) for these I/O pins. The number is the binary weighted sum of last commanded outputs nnn to (nnn+len-1). Note this is different from the state of the I/O pins.

EXAMPLE:

```
EXOUT(207,3) = 4  cr     binary rep. of 4 = 1 0 0
                                            | | └── output 207 = off
                                            | └──── output 208 = off
                                            └────── output 209 = on

EXOUT(208,2)  cr         returns 2
                         (last output for 208 = 0 )
                         + 2 (last output for 209 = 1)
```

FILTER FILTER

ACTION: Sets the filter value for the for the defined analog input

PROGRAM SYNTAX:

FILTER(b0n)=number cr
FILTER(b0n) cr

REMARKS: The "b" specifies the board (1-4).

The "n" specifies the analog input (1-4)

The number sets the filter value (.01 - 1). Where 1.0 is no filtering

FILTER(b0n)=number
Sets the filter value for the designated board and input.

FILTER(b0n)
Returns the filter value for the designated board and input.

EXAMPLES:

FILTER(101)=.1
Sets the filter value for board 1 input 1 to a value of .1 .

FILTER(302)=.1
Sets the filter value for board 3 input 2 to a value of .1 .

FREE FREE

ACTION: Transfers the free space, in sectors, available in non-volatile memory.

COMMAND SYNTAX: FREE cr

REMARKS: A sector consists of 128 bytes of non-volatile memory. With no program(s) loaded, the free space value is 2044 sectors.

The transfer format is:
free space = nnnncr

FREEMEM FREEMEM

ACTION: Returns the amount of free memory for program execution allocation.

COMMAND SYNTAX: FREEMEM cr

REMARKS: The value returned is the number of 32 bit word free for allocation.

The DIM command uses free memory for allocating an area for arrays.

A new variable string uses free memory for storing the string characters.

The maximum free memory size is 32768 words.

If an "Out of Memory" error occurs during program execution the FREEMEM command can be used to determine whether the error was created by a string command or that the memory allocated for program storage was exceeded. If the FREEMEM command returns a negative value the memory allocated for program storage was exceeded.

EXAMPLE: FREEMEM cr

HARDLIMNEG HARDLIMNEG

ACTION: Returns the -LIMIT state for the selected axis.

PROGRAM SYNTAX: HARDLIMNEG cr
HARDLIMNEG(n) cr

REMARKS: The n specifies the axis to return.

HARDLIMNEG
Returns the -LIMIT state of all axes. A return of a 1 indicates the limit is active.

HARDLIMNEG(n)
Returns the -LIMIT state of axis n. A return of a 1 indicates the limit is active.

EXAMPLES: HARDLIMNEG(2)
Returns the -LIMIT state of axis 2.

HARDLIMPOS HARDLIMPOS

ACTION: Returns the +LIMIT state of the selected axis.

PROGRAM SYNTAX:
HARDLIMPOS cr
HARDLIMPOS(n) cr

REMARKS: The n specifies the axis to return.

HARDLIMPOS
Returns the +LIMIT state of all axes. A return of a 1 indicates the limit is active.

HARDLIMPOS(n)
Returns the +LIMIT state of axis n. A return of a 1 indicates the limit is active.

EXAMPLES:
HARDLIMPOS(2)
Returns the +LIMIT state of axis 2.

IN IN

ACTION: Returns the state of the specified digital I/O inputs.

COMMAND SYNTAX:
IN(nnn) cr
IN(nnn,len) cr

nnn is the I/O point terminal number.

nnn =	(101-124) or	(201-224) or	(301-324) or	(401-424)
	board 1	board 2	board 3	board 4

len is the number of I/O points.

len = 1 - 24

SINGLE INPUT COMMAND SYNTAX:

IN(nnn) cr returns the state (1 or 0) of input nnn.

EXAMPLE:

IN(207) cr returns 1, if input 207 is on.
IN(207) cr returns 0, if input 207 is off.

MULTIPLE INPUT COMMAND SYNTAX:

N(nnn,len) cr returns a number corresponding to the states of multiple inputs: (binary weighting of inputs nnn to (nnn+len-1)

$$INPUT_{nnn} + 2\ INPUT_{nnn+1} + 4\ INPUT_{nnn+2} + \ldots + 2^{len-1}\ INPUT_{nnn+len-1}$$

EXAMPLE:

IN(207,3) cr will return: $INPUT_{207} + 2\ INPUT_{208} + 4\ INPUT_{209}$
which is a number between 0 and 7, depending on the states of INPUTS 207-209.

INTLIM INTLIM

ACTION:

Sets the Integral limit of a servo axis.

PROGRAM SYNTAX:

INTLIM(axis)=number cr
INTLIM=number1, ... , number8 cr
INTLIM(axis) cr
INTLIM cr

REMARKS:

number is the Integral Limit of the servo axis in volts.

The range for the Integral Limit is 0 to 10 volts.

INTLIM(axis)=number
Sets the Integral limit of the specified axis to the number value.

INTLIM=number1, ... , number8
Sets the integral limit of the designated axes to the number value.

INTLIM(axis)
Returns the integral limit value of axis.

INTLIM
Returns all the axes integral limit values.

EXAMPLES:

INTLIM(2)=5
Sets the integral limit of axis 2 to 5 volts.

INTLIM=5, , , ,3
Sets the integral limit for axis 1 to 5 volts. Axes 2, 3 and 4 are unchanged. Axis 5 is set to 3 volts.

INTLIM(4)
Returns the integral limit for axis 4.

JOGSTART JOGSTART

ACTION:

Runs a specified axis or axes continuously in the designated direction.

COMMAND SYNTAX:

JOGSTART(n) = number cr
JOGSTART = number1, number2, . . . , number8 cr

REMARKS:

The n specifies the axis (1 - 8). The sign of "number" is used to specify the jogging direction; the magnitude of the number is not significant. If the number is positive, jogging will take place in the positive direction. If the number is negative, jogging will take place in the negative direction. Jog speed is determined by the previously executed velocity command. Typically, "number" = +1 or -1.

JOGSTART(n) = number
Starts jogging the motor for the specified axis.

JOGSTART = number1, number2, . . . , number8
Starts jogging the motor for all defined axis.

Use the JOGSTOP command for stopping the motor(s).

EXAMPLES:

JOGSTART(1) = +1
starts jogging axis 1 in positive direction

JOGSTART= +1,-1
starts jogging axis 1 in positive direction and axis 2 in the negative direction.

JOGSTOP JOGSTOP

ACTION:

Stops JOG motion for the specified axis.

PROGRAM SYNTAX:

JOGSTOP(n) cr
JOGSTOP = number1, number2, . . . , number8 cr

REMARKS:

The n specifies the axis (1 - 8).
The value of "number" in the command syntax is not significant; any number will work.

EXAMPLES:

JOGSTOP(1)
stop axis 1 from jogging

JOGSTOP=0,0
stops axis 1 and axis 2 from jogging.

KAFF

ACTION:

Sets the acceleration feed forward gain for a servo axis.

PROGRAM SYNTAX:

```
KAFF(axis)=number cr
KAFF=number1, ... , number8 cr
KAFF(axis) cr
KAFF cr
```

REMARKS:

number is the acceleration feed forward gain of the servo axis.
The **Units** are volts/encoder count/msec^2.

The number value must be positive.

KAFF(axis)=number
Sets the acceleration feed forward gain of the specified axis to the number value.

KAFF=number1, ... , number8
Sets the acceleration feed forward gain of the designated axes to the number value.

KAFF(axis)
Returns the acceleration feed forward gain value of the specified axis.

KAFF
Returns all the axes acceleration feed forward gain values.

EXAMPLES:

KAFF(2)=.5
Sets the acceleration feed forward gain value for axis 2 to .5 volts/encoder count/msec^2.

KAFF=.2, ,0
Sets the acceleration feed forward gain value for axis 1 to .2 volts/encoder count/msec^2.
Axis 2 is unchanged. Axis 3 is set to 0 volts/encoder count/msec^2.

KAFF(2)
Return the acceleration feed forward gain for axis 2.

KD

ACTION:

Sets the derivative gain for a servo axis.

PROGRAM SYNTAX:

KD(axis)=number cr
KD=number1, ... , number8 cr
KD(axis) cr
KD cr

REMARKS:

number is the derivative gain value of the servo axis.
The **Units** are milliseconds.

The number value must be positive.

KD(axis)=number
Sets the derivative gain of axis to the number value.

KD=number1, ... , number8
Sets the derivative gain of the designated axes to the number value.

KD(axis)
Returns the derivative gain value of axis.

KD
Returns all the axes derivative gain values.

EXAMPLES:

KD(2)=4
Sets the derivitive gain of axis 2 to 4 milliseconds.

KD=4, , 4
Sets the derivitive gain of axis 1 and axis 3 to 4 milliseconds. Axis 2 is unchanged.

KD(2)
Returns the derivitive gain of axis 2.

KI

ACTION: Sets the Integral gain of a servo axis.

PROGRAM SYNTAX:

KI(axis)=number cr
KI=number1, ... , number8 cr
KI(axis) cr
KI cr

REMARKS:

number is the integral gain value of the servo axis.
The **Units** are milliseconds.

The number value must be positive.

KI(axis)=number
Sets the integral gain of the specified axis to the number value.

KI=number1, ... , number8
Sets the integral gain of the designated axes to the number value.

KI(axis)
Returns the integral gain value of the specified axis.

KI
Returns all the axes integral gain values.

EXAMPLES:

KI(2)=1
Sets the integral gain for axis 2 to 1 millisecond.

KI=1, ,0
Sets the integral gain for axis 1 to 1 millisecond. Axis 2 is unchanged.
Axis 3 is set to 0 milliseconds.

KI(2)
Returns the integral gain for axis 2.

KP

ACTION:

Sets the proportional gain of a servo axis.

PROGRAM SYNTAX:

KP(axis)=number cr
KP=number1, ... , number8 cr
KP(axis) cr
KP cr

REMARKS:

number is the proportional gain value of the servo axis.
The **Units** are millivolts/encoder count.

The number value must be positive.

KP(axis)=number
Sets the proportional gain of the specified axis to the number value.

KP=number1, ... , number8
Sets the proportional gain of the designated axes to the number value.

KP(axis)
Returns the proportional gain value of the specified axis.

KP
Returns all the axes proportional gain values.

EXAMPLES:

KP(2)=50
Sets the proportional gain for axis 2 to 50 millivolts/encoder count (.05 volts/encoder count).

KP=30, ,50
Sets the proportional gain for axis 1 to 30 millivolts/encoder count. Axis 2 is unchanged. Axis 3 is set to 50 millivolts/encoder count.

KP(2)
Return the proportional gain for axis 2.

KVFF KVFF

ACTION:

Sets the velocity feed forward gain value for a servo axis.

PROGRAM SYNTAX:

KVFF(axis)=number cr
KVFF=number1, ... , number8 cr
KVFF(axis) cr

REMARKS:

The axis specifies the number of the axis (1-8).

The number is the velocity feed forward gain value of the servo axis. The **Units** are in percent. A value of 0 disable velocity feed forward gain. A Value of 100 is a good starting point for velocity feed forward.

The number value must be positive.

KVFF(axis)=number
Sets the velocity feed forward gain of axis to the number value.

KVFF=number1, ... , number8
Sets the velocity feed forward gain of the designated axes to the number value.

KVFF(axis)
Returns the velocity feed forward gain value of axis.

EXAMPLES:

KVFF(2)=100 cr
Sets the velocity feed forward gain for axis 2 to 100%.

KVFF=98, ,99 cr
Sets the velocity feed forward gain for axis 1 to 98% and axis 3 to 99%.

KVFF(2) cr
Returns the velocity feed forward gain for axis 2.

LINE LINE

ACTION: Initiates a coordinated linear move for specified axes.

COMMAND SYNTAX: Line = number1, number2, . . . , number8 cr

REMARKS: The "number" represents the distance to move. The sign of the number determines the direction of motion for the move. All defined axes will start at the same time.

The VELOCITY command value is used for the coordinated speed.

EXAMPLES: LINE=2,4,,,,,,1.5
creates a linear motion of axis 1 2.0 units, axis 2 motion of 4 units and axis 8 of 1.5 units.

LOAD LOAD

ACTION: Loads the designated project from non-volatile memory into operating memory.

COMMAND SYNTAX: LOAD project name cr

REMARKS: The name is limited to eight characters.

EXAMPLES: LOAD CONVEYOR
Loads project CONVEYOR into operating memory.

LOWSPD LOWSPD

ACTION:

Sets or returns the low speed value (starting speed) of the specified axis.

COMMAND SYNTAX:

```
LOWSPD(n)=number  cr
LOWSPD=number1, number2, . . . , number8  cr
LOWSPD(n)  cr
LOWSPD  cr
```

REMARKS:

The n specifies the axis (1-8).
The "number" specifies the value of the low speed to be set.

LOWSPD(n)=number
Sets the low speed value for the specified axis n.

LOWSPD=number1, number2, . . . , number8
Sets the low speed value for all defined axes.

LOWSPD(n)
Returns the low speed value of the specified axis n.

LOWSPD
Returns the low speed value of all axes.

EXAMPLES:

LOWSPD(1)= 4
Sets the low speed value of axis 1 to 4 units.

LOWSPD=4,4, , ,10,10,5,5
Sets the low speed value for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

LOWSPD(2)
returns the low speed value of axis 2.

MAXSPD MAXSPD

ACTION: Sets or returns the maximum allowed speed of the specified axis.

COMMAND SYNTAX:

```
MAXSPD(n)=number cr
MAXSPD=number1, number2, . . . , number8  cr
MAXSPD(n) cr
MAXSPD cr
```

REMARKS: The n specifies the axis (1-8).
The "number" specifies the value of the maximum speed to be set.

MAXSPD(n)=number
Sets the maximum speed value for the specified axis n.

MAXSPD=number1, number2, . . . , number8
Sets the maximum speed value for all defined axes.

MAXSPD(n)
Returns the maximum speed value of the specified axis n.

MAXSPD
Returns the maximum speed value of all axes.

EXAMPLES: MAXSPD(1)= 4
Sets the maximum speed value of axis 1 to 4 units.

MAXSPD=4,4, , ,10,10,5,5
Sets the maximum speed value for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

MAXSPD(2)
returns the maximum speed value of axis 2.

MOVE MOVE

ACTION: Initiates a non-coordinated move for one or more specified axes.

COMMAND SYNTAX:

MOVE(n) = number cr
MOVE = number1, number2, . . . , number8 cr

REMARKS: The n specifies the axis (1 - 8). The "number" represents the distance to move. The sign of the number determines the direction (positive or negative) of motion for the move.

EXAMPLES:

MOVE(1)=2.5
axis 1 moves 2.5 units.

MOVE=2.5,4,,,,,,8.2
axis 1 moves 2.5 units, axis 2 moves 4 units and axis 8 moves 8.2 units.

MOVEHOME MOVEHOME

ACTION: Runs the designated axis motor until the home input is activated; captures and registers this position as HOME (electrical zero); then stops.

COMMAND SYNTAX:

MOVEHOME(n) = number cr
MOVEHOME = number1, number2, . . . , number8 cr

REMARKS: The n specifies the axis (1 - 8). The sign of the "number" determines the direction (positive or negative) of motion for the home cycle; its value is unimportant. Typically, values or +1 or -1 are used. The speed will be determined by the last velocity command that was executed.

The trigger is determined by the program loaded configuration.

EXAMPLES:

MOVEHOME(1)=+1
axis 1 returns to mechanical home in the Positive direction

MOVEHOME=,,,,,,,-1
axis 8 returns to mechanical home in the Negative direction.

MOVEREG MOVEREG

ACTION:

Runs the designated axis motor until the registration input is activated, then moves the specified registration distance.

COMMAND SYNTAX:

```
MOVEREG(n) =number cr
MOVEREG = number1, number2, . . . , number8  cr
```

REMARKS:

The n specifies the axis (1 - 8). The "number" represents the distance to move after the input has been activated. The sign of the number determines the direction (positive or negative) of motion for the mark registration cycle.

The trigger is determined by the program loaded configuration.

EXAMPLES:

MOVEREG(1)=+2.5
starts an axis 1 move registration cycle in the positive direction.

MOVEREG=,,,-3.0
starts an axis 4 move registration cycle in the negative direction.

NVR NVR

ACTION:

The NVR array is used for non-volatile variable storage. Up to 2048 variables can be saved.

PROGRAM SYNTAX:

```
NVR(element) cr
NVR(element)=number cr
```

REMARKS:

The NVR array has 2048 element (1-2048) and is accesible by all program tasks.

Element is the NVR element number (1-2048).

Number is the value assigned to the element.

EXAMPLES:

NVR(2)
Returns the NVR element 2 value.

NVR(2048)=10.5
Sets the NVR element 2048 to a value of 10.5

OUT OUT

ACTION: Sets or returns the state of the specified digital I/O outputs.

COMMAND SYNTAX:

```
OUT(nnn) = number cr
OUT(nnn) cr
OUT(nnn,len) = number cr
OUT(nnn,len) cr
```

nnn is the I/O point terminal number

```
nnn = (101-124) or (201-224) or (301-324) or (401-424)
        board 1      board 2      board 3      board 4
```

Outputs x01-x16 are physical outputs, x17-x24 are internal flags which can be set and read just as the physical outputs.

len is the number of I/O points; len = 1 - 24

"number" is zero (0) to turn an output off or one (1) to turn an output on

SET SINGLE OUTPUT COMMAND SYNTAX:

```
OUT(nnn) = number cr
```

turn output nnn on (number = 1) or off (number = 0).

EXAMPLE:

```
OUT(207) = 1 cr  turns output 207 on.
OUT(207) = 0 cr  turns output 207 off.
```

READ SINGLE OUTPUT COMMAND SYNTAX:

```
OUT(nnn) cr
```

evaluates to the last output commanded (1 or 0) for this I/O pin. Note this is different from the state of the I/O pin.

EXAMPLE:

```
OUT(207) = 3 cr        turns output 207 on.
OUT(207) cr            returns 1 (last commanded output for 207)
```

SET MULTIPLE OUTPUTS COMMAND SYNTAX:

```
OUT(nnn,len) = number cr
```

The expression is evaluated and converted to an integer value. The least significant "len" bits of the binary representation are then used to set outputs "nnn" to "nnn+len-1" respectively.

EXAMPLE:

OUT(216,3) = 6.2 cr sets output 216 and flags 217, 218
6.2 is converted to integer 6 and the binary representation of 6

```
is  1 1 0
    | | └───   output 216 = off
    | └─────   flag 217   = on
    └───────   flag 218   = on
```

OUT, (cont'd)

READ MULTIPLE OUTPUTS
COMMAND SYNTAX: OUT(nnn,len) cr

Evaluates to a number corresponding to the last outputs commanded (1 or 0) for these I/O pins. The number is the binary weighted sum of the last commanded outputs nnn to (nnn+len-1). Note this is different from the state of the I/O pins.

EXAMPLE:

```
OUT(207,3) = 4 cr  binary rep. of 4  1 0 0
                                     | | └── output 207 = off
                                     | └──── output 208 = off
                                     └────── output 209 = on
OUT(208,2) cr              returns 2
     (last output for 208 = 0 ) + 2 (last output for 209 = 1)
```

OUTLIMIT OUTLIMIT

ACTION: Sets or return the maximum analog ouput voltage allowed on a servo axis.

PROGRAM SYNTAX:

OUTLIMIT(axis)=number cr
OUTLIMIT=number1, ... ,number8 cr
OUTLIMIT(axis) cr

REMARKS: This command only applies to a servo axis and clamps the peak level of the commanded analog output voltage to the servo.

Axis is the axis number (1 to 8).

Number is a value (1.0 to 10.0). This value is in volts.

OUTLIMIT(axis)=number
Sets the OUTLIMIT for the selected axis equal to the number.

OUTLIMIT=number1, ... , number8
Sets the OUTLIMIT for the designated axes equal to the corresponding number.

OUTLIMIT(axis)
Returns the OUTLIMIT value for the selected axis.

EXAMPLES:

OUTLIMIT(1)=5
Sets the analog output limit for a servo drive (axis 1) to 5 volts.

OUTLIMIT=,10
Sets the analog output limit for a servo drive (axis 2) to 10 volts. Axis 1 is unchanged.

POSMODE

ACTION:

Sets or returns the positioning mode for the specified axis.

COMMAND SYNTAX:

```
POSMODE(n) = number  cr
POSMODE = number1, number2, . . . , number8  cr
POSMODE(n)  cr
POSMODE  cr
```

REMARKS:

The n specifies the axis (1 - 8). "Number" specifies the positioning mode, as follows:

Number	Positioning Mode
non-zero	absolute
zero	incremental (default mode)

POSMODE(n) = number
Sets the positioning mode for the specified axis n.

POSMODE = number1, number2, . . . , number8
Sets the positioning mode for all defined axis.

POSMODE(n)
returns the positioning mode for the specified axis n.

POSMODE
returns the positioning mode for all axes.

EXAMPLES:

POSMODE(1)=0
sets axis 1 to incremental position mode.

POSMODE=0,1
sets axis 1 to incremental position mode and axis 2 to absolute position mode.

POSMODE(1)
returns axis 1 positioning mode

PROFILE PROFILE

ACTION: Sets or returns the acceleration/deceleration profile for the specified axis.

COMMAND SYNTAX:

```
PROFILE(n)=number  cr
PROFILE=number1, number2, . . . , number8  cr
PROFILE(n)  cr
PROFILE  cr
```

REMARKS: The n specifies the number of the axis (1-8).

PROFILE(n)=number
Sets the acceleration/deceleration profile for the specified axis n.

PROFILE=number1, number2, . . . , number8
Sets the acceleration/deceleration profile for all defined axes.

PROFILE(n)
Returns the acceleration/deceleration profile of the specified axis n.

PROFILE
Returns the acceleration/deceleration profile of all axes.

The acceleration/deceleration profiles are:

value	description
0	trapezoidal profile
1	S Curve profile

EXAMPLES:

PROFILE(1)= 0
Sets the acceleration/deceleration profile of axis 1 to trapezoidal.

PROFILE=0,0, , ,1,1,1,0
Sets the acceleration/deceleration profile for axis 1,2 and 8 to trapezoidal. Axis 5,6 and 7 to S curve. Axis 3 and 4 are unchanged.

PROFILE(2)
returns the acceleration/deceleration profile of axis 2.

RESET RESET

ACTION: Resets the system.

COMMAND SYNTAX: RESET cr

REMARKS: This command causes the system to halt, and then restart as though power had been recycled.

REVISION REVISION

ACTION: Returns the current revision level of the controller's operating system software.

COMMAND SYNTAX: REVISION cr

REMARKS: The return format for this command is:

MX2000 REV n , date
where "n" is the current revision number and "date" is the release date.

RUN RUN

ACTION: Runs the loaded project or specified task number.

COMMAND SYNTAX:

```
RUN cr
RUN(n) cr
```

REMARKS: RUN starts execution of all loaded tasks from their respective beginnings.

RUN(n) (n is 1-7) starts execution of the specified task from its beginning, if it is loaded.

EXAMPLE:

```
RUN(1)
```

Starts task 1 from the beginning if loaded.

SOFTLIMNEG SOFTLIMNEG

ACTION: Sets or returns the software absolute negative travel distance value for the specified axis.

COMMAND SYNTAX:

SOFTLIMNEG(n)=number cr
SOFTLIMNEG=number1, number2, . . . , number8 cr
SOFTLIMNEG(n) cr
SOFTLIMNEG cr

REMARKS: The n specifies the axis (1-8). "Number" is the value of SOFTLIMNEG to be set.

SOFTLIMNEG(n)=number
Sets the absolute negative travel distance for the specified axis n.

SOFTLIMNEG=number1, number2, . . . , number8
Sets the absolute negative travel distance for all defined axes.

SOFTLIMNEG(n)
Returns the absolute negative travel distance of the specified axis n.

SOFTLIMNEG
Returns the absolute negative travel distance of all axes.

EXAMPLES:

SOFTLIMNEG(1)= -4
Sets the absolute negative travel distance of axis 1 to -4 units.

SOFTLIMNEG=4,4, , ,10,10,5,5
Sets the absolute negative travel distance for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

SOFTLIMNEG(2)
returns the absolute negative travel distance value of axis 2.

SOFTLIMPOS SOFTLIMPOS

ACTION: Sets or returns the software absolute positive travel distance value for the specified axis.

COMMAND SYNTAX:

```
SOFTLIMPOS(n)=number cr
SOFTLIMPOS=number1, number2, . . . , number8  cr
SOFTLIMPOS(n) cr
SOFTLIMPOS cr
```

REMARKS: The n specifies the axis (1-8). "Number" is the value of SOFTLIMPOS to be set.

SOFTLIMPOS(n)=number
Sets the absolute positive travel distance for the specified axis n.

SOFTLIMPOS=number1, number2, . . . , number8
Sets the absolute positive travel distance for all defined axes.

SOFTLIMPOS(n)
Returns the absolute positive travel distance of the specified axis n.

SOFTLIMPOS
Returns the absolute positive travel distance of all axes.

EXAMPLES:

SOFTLIMPOS(1)=-4
Sets the absolute positive travel distance of axis 1 to 4 units.

SOFTLIMPOS=4,4, , ,10,10,5,5
Sets the absolute positive travel distance for axis 1 and 2 to 4 units. Axis 5 and 6 to 10 units. Axis 7 and 8 to 5 units. Axis 3 and 4 are unchanged.

SOFTLIMPOS(2)
returns the absolute positive travel distance value of axis 2.

SPEED

SPEED

ACTION:

Sets or returns target speed of a specified axis for non-coordinated motion.

COMMAND SYNTAX:

```
SPEED(n) = number  cr
SPEED = number1, number2, . . . , number8   cr
SPEED(n) cr
SPEED cr
```

REMARKS:

The n specifies the number of the axis (1 - 8).

SPEED(n) = number
Sets the velocity for the specified axis n.

SPEED = number1, number2, . . . , number8
Sets the velocity for all defined axis.

SPEED(n)
returns the velocity of the specified axis n.

SPEED
Returns the velocities of all axes.

EXAMPLES:

SPEED(1)=2.0
sets axis 2 velocity for 2 units/sec.

SPEED=1.2,3.5
sets axis 1 velocity for 1.2 units/sec and axis 2 velocity for 3.5 units/sec.

SPEED(1)
returns the velocity for axis 1.

STOP

STOP

ACTION:

Stops all motion and stops execution of all tasks.

COMMAND SYNTAX:

```
STOP  cr
```

REMARKS:

STOP cr will stop execution of all tasks presently running on the controller; all motion ceases immediately.

EXAMPLE:

STOP
stops all tasks that are running.

UNIT

ACTION:

Sets or returns the pulses/ unit value. Used for programming in "user units," such as inches or revolutions or meters, etc.

COMMAND SYNTAX:

UNIT(n)=number cr
UNIT(n)=number1, number2, . . . , number8 cr
UNIT(n) cr
UNIT cr

REMARKS:

The n specifies the axis (1-8). "Number" is a value derived by taking the number of pulses/rev (drive specified) and dividing it by the number of programming units/rev.

Unit is a signed value, calculated as follows: UNIT=(pulses / motor rev) / (units / motor rev) where CW is defined as positive direction.

UNIT(n)=number
Sets the unit value for the specified axis n.

UNIT=number1, number2, . . . , number8
Sets the unit value for all defined axes.

UNIT(n)
Returns the unit value of the specified axis n.

UNIT
Returns the unit value of all axes.

EXAMPLES:

Calculate the UNIT value so that a 1/10 microstepper can be programmed in degrees:

UNIT = (2000 pulses / motor rev) / (360 degrees / motor rev) = 5.555556
UNIT(1)=5.555556

UNIT(2)
returns the unit value of axis 2.

VELOCITY VELOCITY

ACTION:

Sets or returns the path speed to be used for coordinated motion.

COMMAND SYNTAX:

VELOCITY=number cr
VELOCITY cr

REMARKS:

This velocity is only used as the path speed for the Host commands LINE and ARC.

EXAMPLES:

VELOCITY= 1.0
sets the coordinated speed to 1.0 units/second.

VELOCITY
returns the current velocity for coordinated motion

WARNING WARNING

ACTION:

Returns the program execution warning count of each task.

COMMAND SYNTAX:

WARNING cr
WARNING(task number) cr

REMARKS:

The WARNING command returns the warning counts for all task.

The WARNING(task number) command returns the warning count for the designated task (1-7).

EXAMPLES:

WARNING cr	returns the warning counts for all tasks
WARNING(1) cr	returns the warning counts for task 1

WNDGS WNDGS

ACTION:

Enables or disables the windings off function, or return the WNDGS OFF status for the specified axis. When enabled, the stepper drive "WNDGS" output turns on when there is no motion. This causes the drive to turn off the motor current.
Note: requires a compatible drive.

PROGRAM SYNTAX:

```
WNDGS(axis)=number cr
WNDGS=number1, ... ,number8 cr
WNDGS(axis) cr
WNDGS cr
```

REMARKS:

The "axis" specifies the axis number (1-8).

The number specifies the winding condition (0 or 1). A number value which is not a 0 (true) enables the WNDGS feature. If the number value is 0 (false) disables the WNDGS feature.

WNDGS(n)=number
Enables or disables the motor winding for the specified axis (n).

WNDGS=number1, ... , number8
Enables or disables the motor winding for all designated axes.

WNDGS(n)
Returns the WNDGS condition of the designated axis (n).

WNDGS
Returns the WNDGS condition of all axes.

EXAMPLES:

WNDGS(1)=1
Enables the WNDGS condition for axis 1.

WNDGS=1,1,,,0,0,1,1
Enables the WNDGS condition for axes 1,2,7,8. Disables the WNDGS condition for axes 5 and 6. Axes 3 and 4 are unchanged.

"XON" ... "XOFF"

"XON" ... "XOFF"

ACTION:

The XON/XOFF command is a serial communication protocol, executed in software, that allows communications between two devices without the need for additional hardware control. The protocol is used for controlling the flow of data between the Control and another device.

COMMAND SYNTAX:

Xon (ASCII 17)
Xoff (ASCII 19)

REMARKS:

The Xoff character is used to stop the transmission of RS232 or RS485 characters. When one device sends an Xoff to the other device, it is telling the other to stop transmitting characters. The transmitting device should comply with the request.

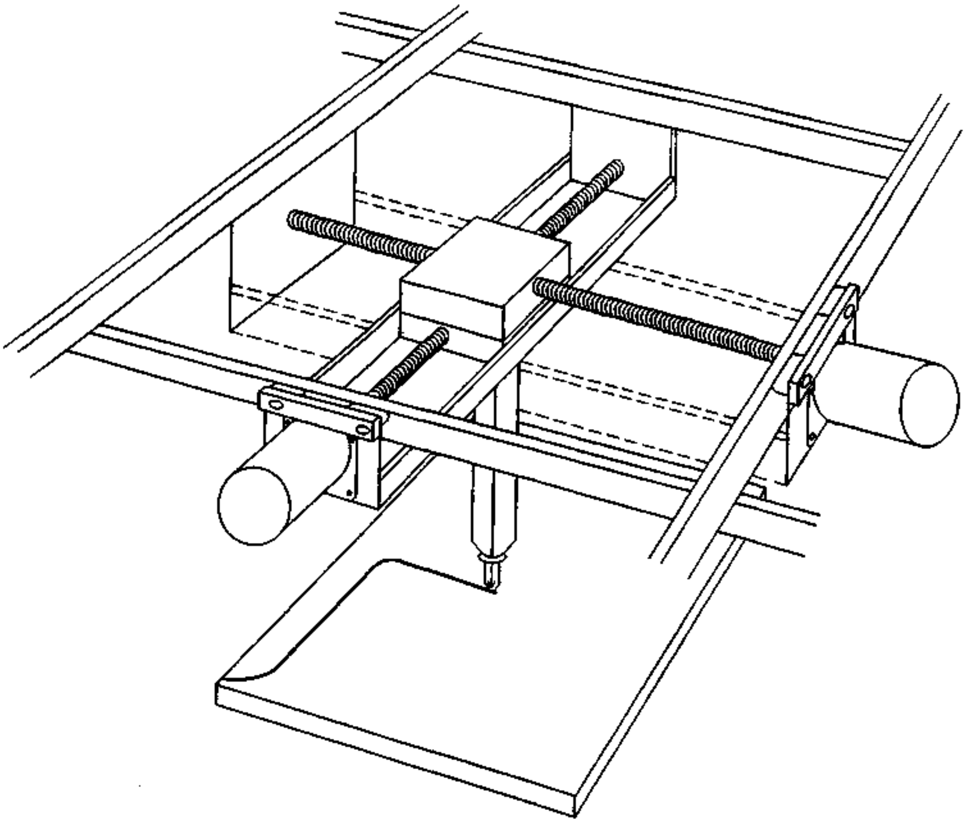
The Xon character is used to resume transmission of the RS232 or RS485 characters. When one device sends an Xon character to the other, it is signifying that it is ready to receive more characters.

SECTION 7 - APPLICATION EXAMPLES

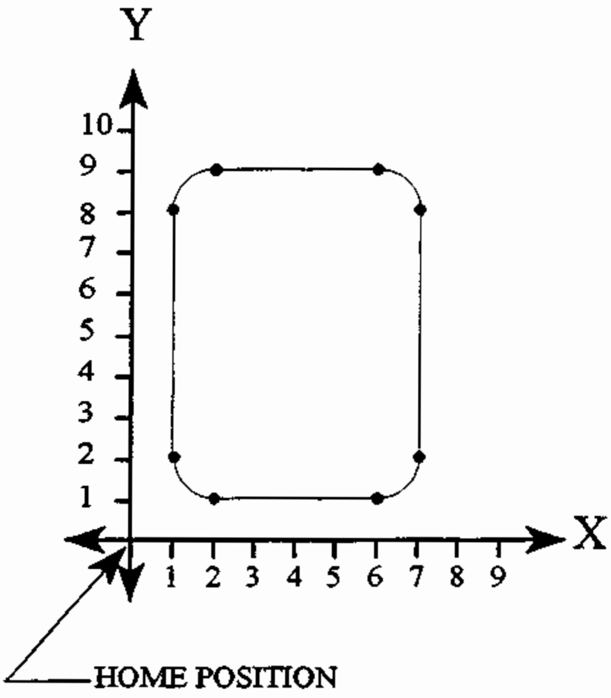
7.1 - APPLICATION EXAMPLE NUMBER 1 (ARBITRARY CONTINUOUS MOTION)

This program illustrates the simplicity of using an arbitrary continuous motion path in any application.

In this application, the operator places a sheet of sponge material on a sponge cutting machine and then activates a cycle start switch (IN101). The cutting blade moves to a starting position, lowers, then cuts a predetermined shape sponge. After a sponge is cut, the blade is raised and returned to a home position.



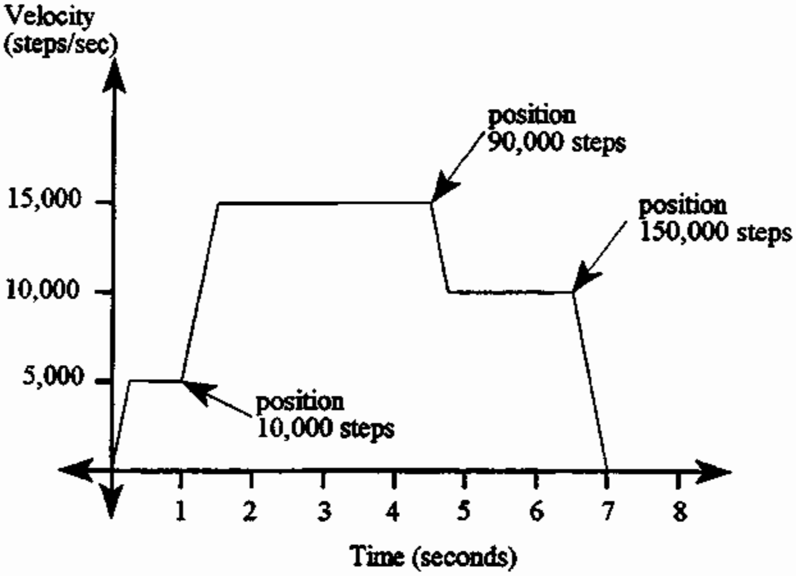
```
POSMODE=1,1                          'enable absolute mode
DO : LOOP UNTIL IN(101)=1            'loop until input 101 is high
MOVE=2,1                             'move to starting position
OUT(111)=1                           'turn output 111 on (high)
PATH=1,2                             'begin continuous motion path
LINE=6,1                             'first coordinate of the path
POINT=7,2                            'second coordinate of the path
LINE=7,8
POINT=6,9
LINE=2,9
POINT=1,8
LINE=1,2
POINT=2,1
PATH END                             'end of continuous motion path
OUT(111)=0                           'turn output 111 off (low)
MOVEHOME=1,1                         'return to home position
```



7.2 - APPLICATION EXAMPLE NUMBER 2 (CHANGING VELOCITY "ON-THE-FLY")

This program illustrates changing velocity "on the fly" based on a position.

```
VELOCITY=10000                'set velocity to 10000 steps/sec
ACCEL=20000,20000             'set acceleration to 20000 steps/sec^2
MOVEHOME=1,1                  'go to home position
WAITDONE =1,1                 'wait until axes 1 and 2 are at home position
PATH=1,2
FEEDRATE=0.5                  'set velocity to 50% its value (0.5 x 10000 steps/s = 10000 steps/s)
LINE=10000,0                  'move to 10,000 steps
FEEDRATE=1.5                  'change velocity to 1.5 x 10000 steps/s = 15000 steps/s
LINE=90000,0                  'move to 90,000 steps
FEEDRATE=1                    'set velocity back to 100% (10000 steps/sec)
LINE=150000,0                 'move to 150,000 steps
PATH END                      'stop motion
```

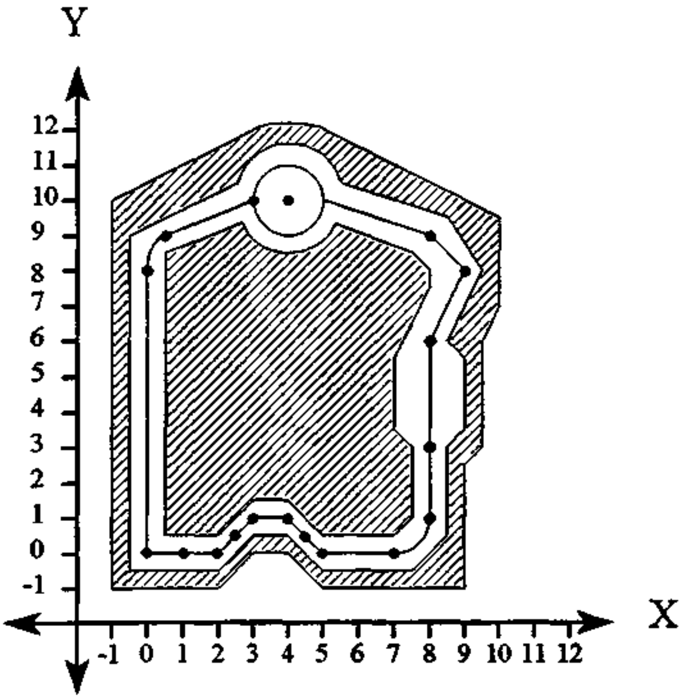


7.3 - APPLICATION EXAMPLE NUMBER 3 (GLUE APPLICATION ON A GASKET)

This program generates a complex continuous motion path for applying glue on a gasket.

An operator activates a cycle start switch (EXIN111), the glue head returns to a home position, and an absolute position is set to zero. The glue head is moved 5" from home to a starting position, and absolute position is set to zero again (all path coordinates are relative to this point). Next the glue head is lowered (EXOUT102) and waits for .25 seconds. Then glue is applied along the pattern, which is described by the x-y coordinates of the lines, arcs, and paths in the Gluing Subroutine section of the program. Finally, the glue is turned off and the glue head is raised (EXOUT101 and 102).

Gasket Pattern:



```
'***************************** PARAMETER SETUP **********************************************************

RADIUS=0                                'radius for path blending
VELOCITY=5                              'path speed = 5in/sec
ACCEL=10,10                             'acceleration rate = 10in/sec^2
DECEL=10,10                             'deceleration rate = 10in/sec^2
SOFTLIMIT=0,0                           'disable software limits
HARDLIMIT=1,1                           'enable hard limits
POSMODE=1,1                             'enable absolute mode
```

```
'************************ START OF MAIN PROGRAM *******************************************************

BEGIN:
DO : LOOP WHILE EXIN(111)=0                 'wait for cycle start input
GOSUB HOME                                  'go to home position
LINE = 5,5                                  'offset each axis to starting position (5" from home)
WAITDONE=1,1                                'wait until axes 1 and 2 are in position
ABSPOS = 0,0                                'reset absolute position to zero
GOSUB GLUE_PATH                             'go to subroutine GLUE_PATH
GOTO BEGIN                                  'go to begin
END                                         'end of main program

'*********************** START OF GLUING SUBROUTINE ************************************************

GLUE_PATH:
EXOUT(101)=1                                'head down
EXOUT(102)=1                                'glue on
WAIT=.25                                    'wait for glue to start flowing
PATH = 1,2                                  'beginning of path
LINE = 0,8
POINT = 0.5,9
LINE = 2,10
ARC = 4,10,+540
LINE = 8,9
POINT = 8.5,8.5
POINT = 9,8
LINE = 8.5,6.5
FEEDRATE = 0.5                              'decrease velocity to 50% = 2.5in/sec
LINE = 8.5,3
FEEDRATE = 1                                'increase velocity back to 100% = 5in/sec
LINE =8.5,1
POINT = 8,0
LINE =7,0
POINT = 6.5,.5
POINT = 5.5,1
LINE = 4.5,1
POINT = 3.5,.5
POINT = 3,0
LINE = 0.5,0
POINT = 0,0.5
PATH END                                    'end of path
EXOUT(101,2)=0                              'turn glue off and raise glue head
RETURN                                      'end of subroutine

'**************************** Home routine **************************************************************

HOME:
EXOUT(123,2)=0                              'glue off and head up
SPEED =2,2                                  'home speed = 2in/sec
MOVEHOME =-1,-1                             'move to home switch x & y "-" dir
DO: LOOP WHILE BUSY(1) OR BUSY(2)           'wait until x & y are home  -  NOTE: alternate form of  WAITDONE=1,1
ABSPOS = 0,0                                'set absolute position to 0
RETURN                                      'end of subroutine
```

7.4 - APPLICATION EXAMPLE NUMBER 4 (SPRING WINDING MACHINE)

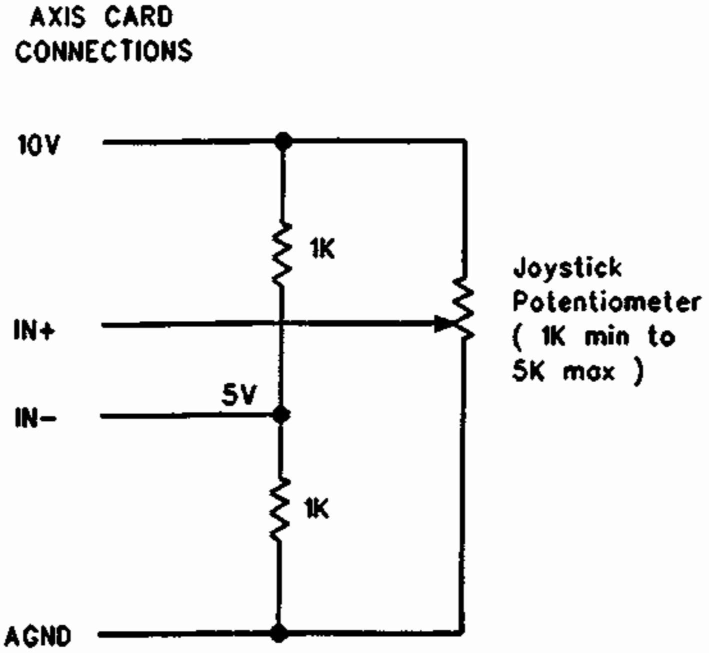
In this application two motors must be moved simultaneously to wind a spring. An expansion I/O board is used to provide the required inputs to the controller.

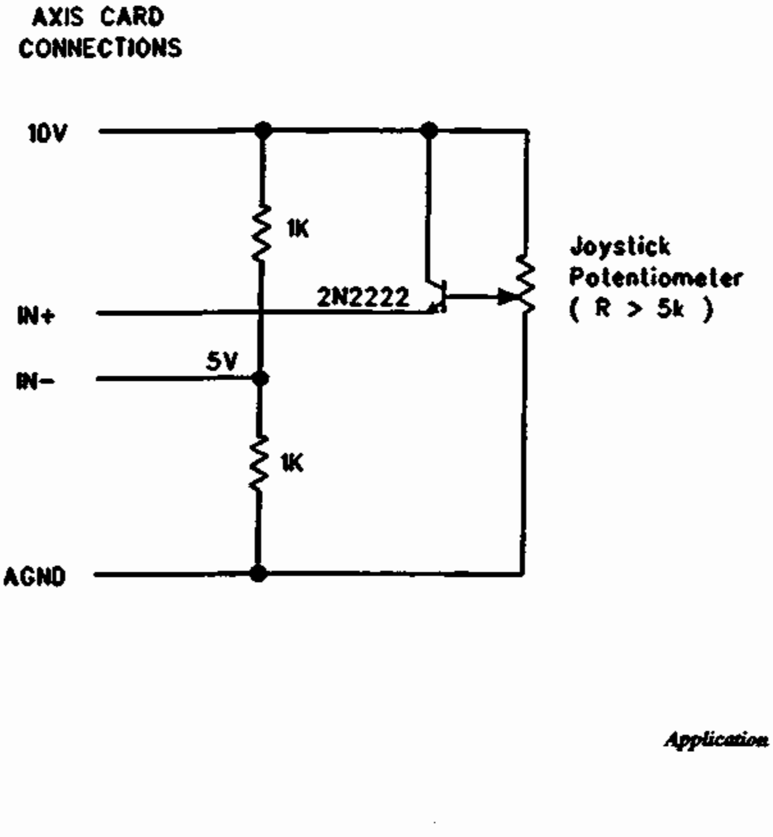
The sequence of events for this application is as follows: a cam will actuate a switch (EXIN101) to start the machine cycle, the wire will be fed (EXOUT112), and a delay of 0.1 seconds will be used to feed enough wire out before a clamp (EXOUT111), used to hold the wire in place, is turned on. Next a centerform clamp (EXIN102), activated by a cam, is moved into position, the winding pin (EXOUT113) slides in and the wire is cut (EXOUT114). The wire is stopped from being fed (EXOUT112) then the wire clamp and the cutter is lifted up (EXOUT111 and 114). Then the cam actuated U-bender (EXIN103) bends the wire into a U shape and the spring is wound. Once the spring has been wound, wire sensing probes move in (EXOUT115 & 116) and check if it has been wound enough (EXIN105 and EXIN106). If not, the spring is wound one step and checked again. This procedure is continued for a predefined number of steps. Recoil to release the spring from the arbors, retract the wire sensing probes (EXOUT115 &116), and slide the winding pin out (EXOUT113) to drop the spring in a bin. Move back to absolute zero. Check whether the auxiliary feed has been depleted, if so end the cycle, otherwise go back to the beginning of program an make another spring.

```
'**************************** PARAMETER SETUP ******************************************************

WIND=145                                'number of steps to wind wire
AUX=20                                  '# of steps for auxiliary wind
RECOIL=50                               '# of steps to recoil
BOOST=1,1                               'enable boost current function
ABSPOS=0,0                              'set absolute position to zero
'****************************** START OF MAIN PROGRAM **********************************************

BEGIN:
DO: LOOP UNTIL EXIN(101) =1             'wait for switch to be activated by cam
EXOUT(112)=1                            'feed wire (expansion output 112 on)
WAIT=.1                                 'wait .1 sec
EXOUT(111)=1                            'turn clamp on (expansion output 111) to hold wire
WAIT=.1                                 'wait .1 sec
DO: LOOP UNTIL EXIN(102)=1              'wait until centerform clamp is in position
EXOUT(113,2)=3                          'winding pin in & cutter down
EXOUT(111,2)=0                          'turn output 112 (feed) and output 111 (clamp) off
EXOUT(114)=0                            'turn output 114(cutter) off
DO: LOOP UNTIL EXIN(103)=1              'wait until U-bender bends spring
MOVE=WIND,WIND                          'wind spring "wind" # of steps
EXOUT(115,2)=3                          'turn on probe x (out 115) & probe y (out 116)
FOR X=1 TO AUX                          'go through loop A number of times
        IF EXIN(105)=0 THEN             'if input 105 is off
            MOVE=1                      'move X-axis 1 step
            A=X                         'A=number of auxiliary feed steps
        END IF                          'end if statement
        IF EXIN(106)=0 THEN             'if input 106 is off
            MOVE=,1                     'move y-axis 1 step
            A=X                         'A=number of auxiliary feed steps
        END IF
        WAIT=.1
NEXT X
MOVE=-RECOIL,-RECOIL                    'move "recoil" # of steps
EXOUT(115,2)=0                          'turn output 115 (probe x) & output 116 (probe y) off
EXOUT(113)=0                            'turn out 113( winding pin) off
POSMODE=1,1                             'enable absolute mode
MOVE=0,0                                'move to absolute zero
WAITDONE=1,1                            'wait until motion stops on axes 1 and 2
POSMODE=0,0                             'switch back to incremental mode
IF A=AUX THEN                           'if auxiliary feed equals aux then part is bad
END                                     'end program
ELSE GOTO BEGIN                         'if go back to beginning and wind another spring
END IF                                  'end of if statement
END                                     'end of program
```

7.5 - EXAMPLE NUMBER 5 (USING A JOYSTICK TO CONTROL MOTION)

A joystick is easily interfaced to the MX2000's analog inputs to control (jog) two motors on an X-Y table, for example. This can allow positioning of a router for setup, or for capturing positions on an arbitrary shape prior to machining, etc. The following diagrams show the methods of connecting a relatively low-impedance joystick (1k to 5k ohms), and a relatively high impedance joystick (>5 k ohms). The center position is at 0 volts; the extremes of the potentiometer movement are at + and - 5 volts. Also given is a programming example used to operate a typical X-Y application.





Sample Code: This teach-mode routine for an X-Y plotter table allows the user to "trace" an arbitrary shape by positioning the pen under joystick control over points on the shape's periphery. By pressing the "fire" button, the coordinates of each point are automatically captured into the MX2000 memory. Then, when finished "tracing" the shape, the user can have the plotter redraw the original shape.

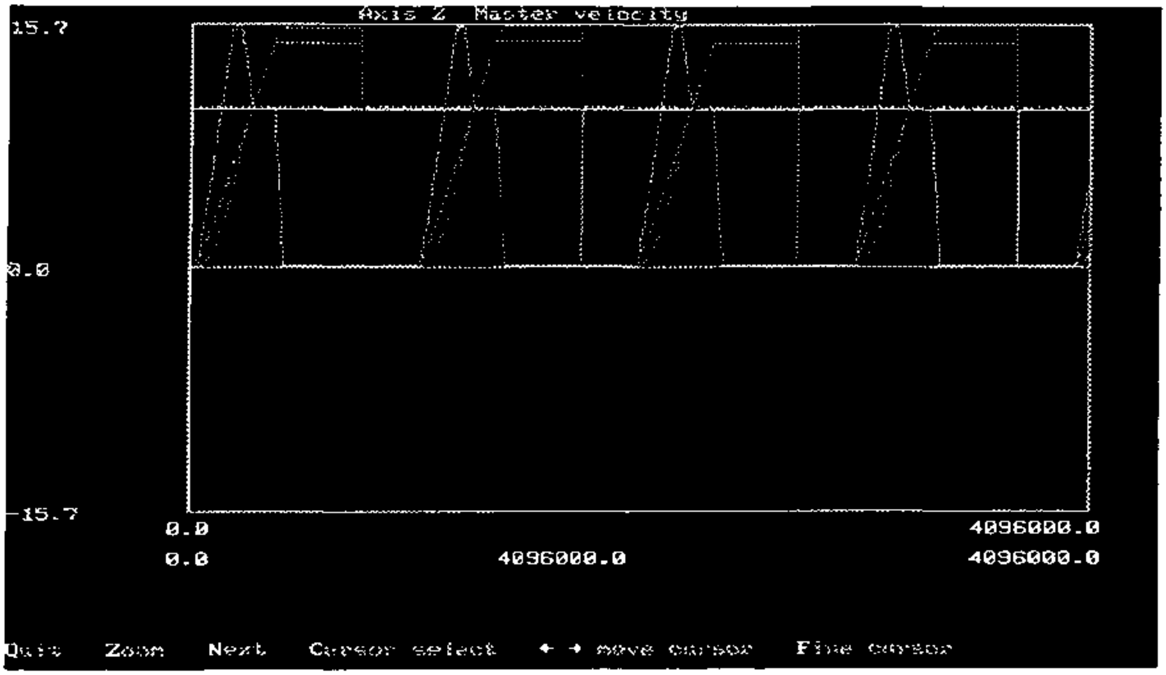
```
*********************** SETUP PARAMETERS ***************************************************
DIM XYPOS(100,2)                                          ' dimension array to 100 by 2
POSMODE=1,1                                               ' set absolute position mode
VELOCITY=1                                                ' set coordinated speed to 1 unit/sec
PT=0                                                      ' initialize array element select

**********************MOVE TO STARTING POSITION USING JOYSTICK*******************************
PRINT #1,"Joy stick Active"
PRINT #1,"Press Button to set a home position"
JOYSTICK=1,2                                              ' set joystick active for selected axes
DO : LOOP UNTIL EXIN(100) = 1                             ' allows joystick motion until button pressed
JOYSTICK END
DO : LOOP UNTIL EXIN(100) = 0                             ' wait for button release
ABSPOS=0,0                                                ' set absolute position to 0

*********************** RECORD DATA POINTS ALONG DESIRED PATH *****************************

PRINT#1, "Move to a position and press the button to record the position"
PRINT#1, "Press D when done"
JOYSTICK=1,2                                              ' set joystick active for selected axes
DO
        A = INCHAR(1)                                     ' load character if received
        IF A > 0 THEN PRINT #1
        IF A >= 0X61 AND A <=0X7A THEN A = A - 0X20       'make lower case
        IF A <> 0X44 AND A <> 0 THEN
                PRINT#1, "Please press D when done"
        END IF
        IF EXIN(100) = 1 THEN                             ' button pressed ?
                EXOUT(123)=1                              ' pen down (mark point)
                DO : LOOP UNTIL EXIN(100) = 0             ' wait for release of button
                EXOUT(123)=0                              ' pen up
                PT=PT+1                                   ' advance to next element in array
                XYPOS(PT,1)=ABSPOS(1)                     ' load axis 1 position
                XYPOS(PT,2)=ABSPOS(2)                     ' load axis 2 position
                PRINT #1,PT,XYPOS(PT,1),XYPOS(PT,2)
        END IF
LOOP UNTIL A = 0X44                                       ' loop until "D" received
JOYSTICK END                                              ' disable joystick

********************** MOVE TO HOME AND EXECUTE PATH ***************************************

PRINT#1, "Moving to home position"
MOVE=0,0                                                  ' move to home position
WAITDONE=1,1                                              ' wait until axes 1 & 2 are at home
EXOUT(123)=1                                              ' pen down
print#1, "Executing path"
path = 1,2                                                ' path start
        FOR Y = 1 TO PT
                POINT = XYPOS(Y,1),XYPOS(Y,2)             ' load point data
         NEXT Y
PATH END                                                  ' execute path command
EXOUT(123)=0                                              ' pen up
PRINT#1, "Program execution complete"
END
```

7.6 - EXAMPLE NUMBER 6 (MATERIAL CUTTING, ENCODER FOLLOWING)

The following is an example of a material cutting program using the Encoder Following feature of the MX2000 software.

The encoder for axis 1 is the master axis and is also the following axis. The following axis will run at 100% of the master axis. Velocity and position following motion will be triggered by event 1 at the start. When in position sync, start cutting the material. When cutting is complete, the axis will ramp to a stop then return to the starting position and wait for a trigger on master position. This cycle will be repeated until EXIN 102 is a 1. The distance between cuts is defined by the FOLDIST value. The only constraint is the time it takes a following axis to complete the cycle and return to the starting position. This time must be less than the time the master takes to move the cutting distance.

```
SPEED(1)=20000                      ' cycle home speed
ABSPOS(1)=0                         ' set secondary starting position
POSMODE(1)=1                        ' absolute positioning mode for axis 1
LOWSPD(1)=3000                      ' following axis starting speed
ACCEL=100000                        ' axis 1 ACCEL 100000 units/sec/sec
DECEL=200000                        ' axis 1 DECEL 200000 units/sec/sec
FOLMODE(1)=2                        ' velocity & position following using a program dir.
FOLTRIG(1)=1                        ' starts the initial cycle on an event 1 trigger on axis 1
FOLRATIO(1)=1                       ' ratio 100%
FOLMAXRATIO(1)=1.5                  ' axis 1 advance cycle max velocity 150%
FOLMINRATIO(1)=.5                   ' axis 1 recede-cycle min velocity 50%
FOLOFFSET(1)=0                      ' sync to master starting position
FOLDIST(1)=10000                    ' 10000 units between cuts
CUTLOOP:                            ' Loop Start
FOLENCODER=101,1                    ' enable encoder following
JOGSTART(1)=1                       ' axis 1: Start Jog on the trigger
DO: LOOP UNTIL FOLSYNC(1)=1         ' wait for positional sync
EXOUT(100)=1                        ' starts cutting
DO: LOOP UNTIL EXIN(101)=1          ' wait until cutting complete
EXOUT(100)=0                        ' stop cutting
FOLEND                              ' wait for JOGSTOP and disable following
WAIT=.05                            ' wait for motor to settle
MOVE(1)=0                           ' move to starting position
WAITDONE(1)                         ' waits for the move to complete
FOLTRIG(1)=3                        ' set trigger on master position
IF EXIN(102)=0 THEN GOTO CUTLOOP    ' do loop again if input=0
FOLENCODER=0,0                      ' end encoder following
FOLEND
END
```



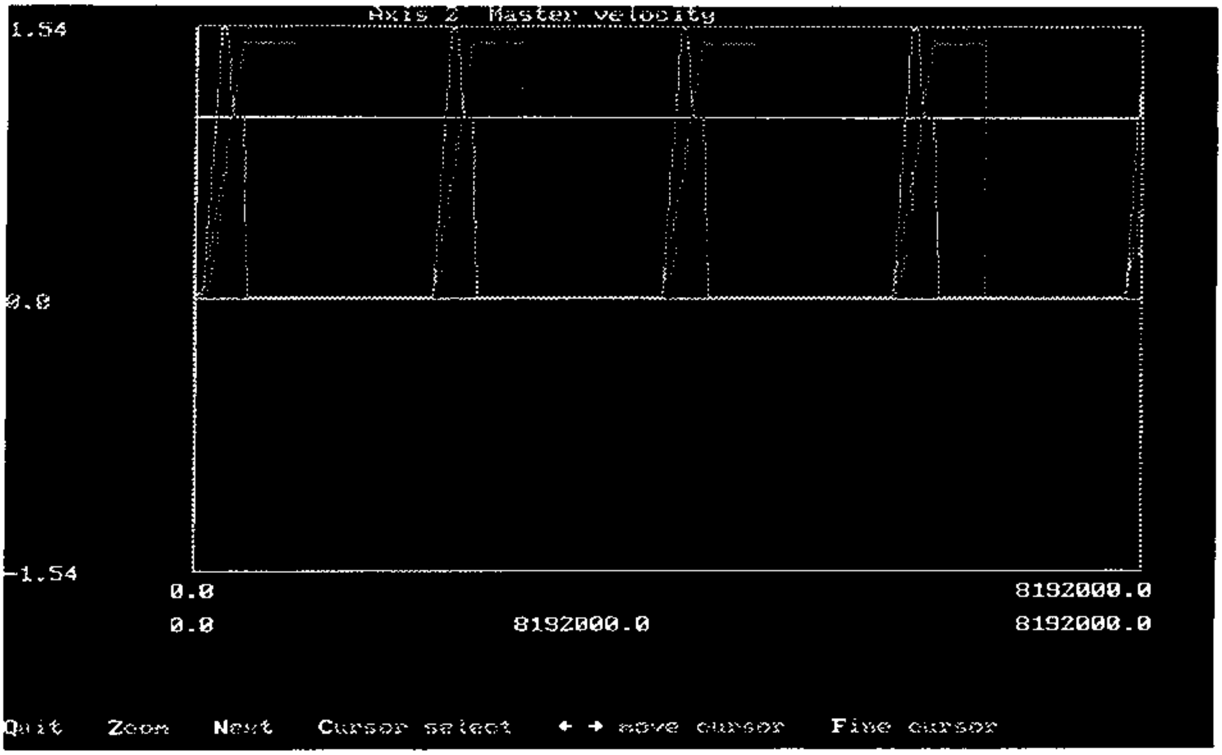
Signals: Master Velocity
Following Velocity
Following Target Position
Following Position

7.7 - EXAMPLE NUMBER 7 (MATERIAL CUTTING, ANALOG FOLLOWING)

The following is an example of a material cutting program using the Analog Following feature of the MX2000 software.

The analog input for master axis velocity deviation is analog port 102 and axis 1 is the following axis. The nominal master axis velocity for a 0-volt analog input voltage is 1000 units/sec. The velocity deviation for 10-volt analog input is 500 units/sec. The following axis will run at 100% of the master axis. Velocity and position following motion will be triggered by event 1 at the start. When in position sync, the cutting cycle starts. When cutting is complete, the axis will ramp to a stop then return to the starting position and wait for a trigger on master position. This cycle will be repeated until EXIN 102 is a 1. The distance between cuts is defined by the FOLDIST value. The only constraint is the time it takes a following axis to complete the cycle and return to the starting position. This time must be less than the time the master takes to move the cutting distance.

```
SPEED=2000                          ' following home speed in units/sec
LOWSPD=0                            ' following starting speed
ABSPOS(1)=0                         ' set secondary starting position
POSMODE(1)=1                        ' absolute positioning mode for axis 1
VELOCITY=1000                       ' nominal velocity of the master axis is 1000 units/sec
ACCEL=10000                         ' axis 1 ACCEL 10000 units/sec/sec
DECEL=20000                         ' axis 1 DECEL 20000 units/sec/sec
FOLMODE(1)=2                        ' velocity & position following using a program dir.
FOLTRIG(1)=1                        ' the initial cycle starts on event 1 trigger of axis 1
FOLRATIO(1)=1                       ' following ratio 100%
FOLMAXRATIO(1)=1.5                  ' axis 1 advance cycle max velocity 150%
FOLMINRATIO(1)=.5                   ' axis 1 recede-cycle min velocity 50%
FOLDIST(1)=2000                     ' 2000 units between cuts
FOLDEVIATION=500                    ' deviation for a 10-volt signal is 500 units/sec
CUTLOOP:
FOLANALOG=102,1                     ' enable analog following
FOLOFFSET(1)=0                      ' sync to master starting position
JOGSTART(1)=1                       ' axis 1 Jog-start on trigger
DO: LOOP UNTIL FOLSYNC(1)=1         ' wait for positional sync
EXOUT(100)=1                        ' starts cutting
DO: LOOP UNTIL EXIN(101)=1          ' wait until cutting complete
EXOUT(100)=0                        ' stop cutting
FOLEND                              ' wait for JOGSTOP and disable following
WAIT=.05                            ' allow the motor to settle
MOVE(1)=0                           ' move to starting position
WAITDONE(1)                         ' wait for motion to stop
FOLTRIG(1)=3                        ' set trigger on master position
IF EXIN(102)=0 then GOTO CUTLOOP    ' loops if not complete
FOLANALOG=0,0                       ' end analog following
FOLEND
```



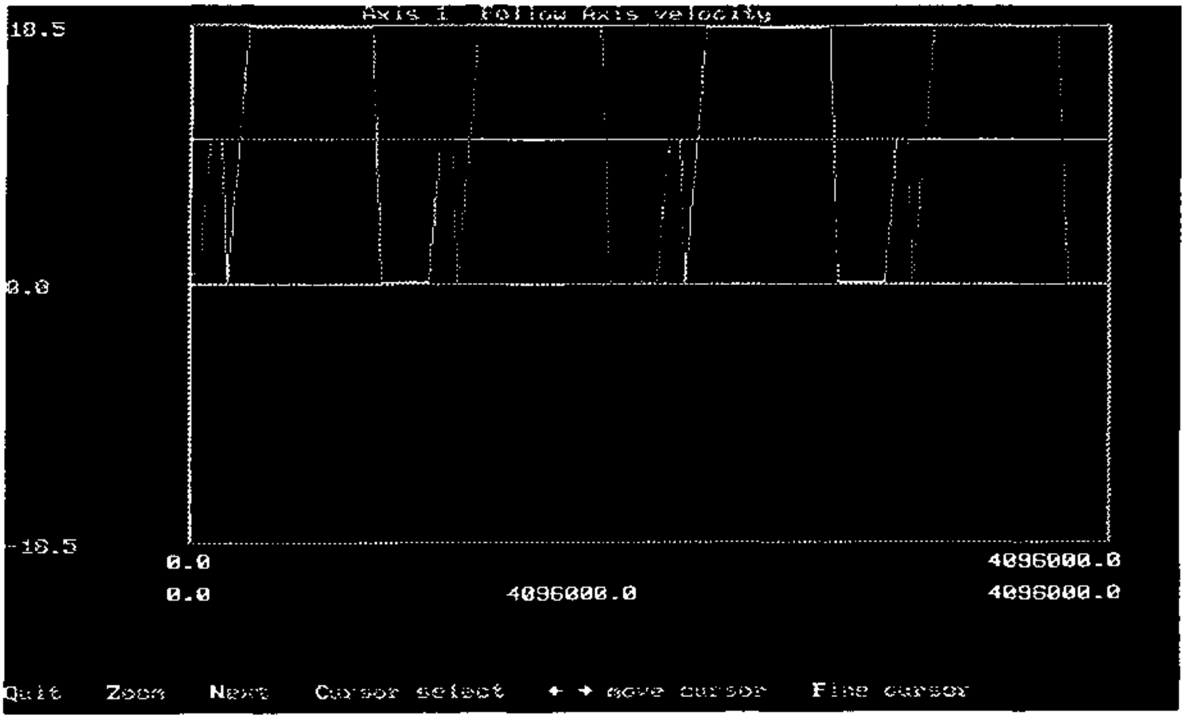
Signals: Master Velocity
Following Velocity
Following Target Position
Following Position

7.7 - EXAMPLE NUMBER 7 (MATERIAL EXTRUSION, ANALOG FOLLOWING)

The following is an example of a material extruding program using the Analog Following feature of the MX2000 software.

Material is to be extruded at a nominal rate of 10000 units/sec. A 20% adjustment to this velocity will be controlled by a +/-10-volt signal applied on board 1 Analog input. The extruding mechanism will be controlled by two motor and two clamps. The sequence of events for this cycle requires that an axis, from its starting position, match the velocity of the material, clamp the material and then pull then it approximately 10000 units. At this time the other axis must start and execute the same cycle. The first axis then unclamps from the material and returns to its starting position. This action will alternate between the two axes. The following program simulates this action and the logged results are shown below.

```
LOWSPD=3000,3000                     'following starting speed
ACCEL=200000,200000                  'ACCEL 200000 units/sec/sec
DECEL=400000,400000                  'DECEL 400000 units/sec/sec
ABSPOS=0,0                           'set starting position
POSMODE=1,1                          'absolute position mode
FOLMODE=0,0                          'velocity following mode
FOLTRIG=0,0                          'start motion on command
VELOCITY=10000                       'nominal master velocity 10000 units/sec
FOLDEVIATION=2000                    '20% deviation for a 10-volt signal
FOLRATIO=1,1                         'following ratio 100%
FOLMAXRATIO=2,2                      'following max speed allowed
FOLMINRATIO=.5,.5                    'following min speed
FOLANALOG=101,1,1                    'enable analog following
X=0                                  'init cycle count
DO
WAITDONE(1)                          'wait for axis 1 to complete motion
FOLRATIO(1)=1                        'axis 1 ratio 100%
JOGSTART(1)=-1                       'starts axis 1 jogging in - direction
DO:LOOP UNTIL FOLSYNC(1)=1           'wait for axis 1 to velocity sync with master
WAIT=.050                            'simulates axis 1 clamp and axis 2 unclamp
JOGSTOP(2)=0                         'stop axis 2
FOLRATIO(2)=1.8                      'set move velocity to 180% of master
MOVE(2)=0                            'move axis 2 to start position
DO:LOOP UNTIL DIST(1)< -10000        'wait for axis 1 to move 10000 units
WAITDONE(2)                          'wait for the axis to reach start position
FOLRATIO(2)=1                        'axis 2 following ratio 100%
JOGSTART(2)=-1                       'start axis 2 in - direction
DO:LOOP UNTIL FOLSYNC(2)=1           'wait for axis 1 to velocity sync with master
WAIT=.050                            'simulates axis 2 clamp and axis 1 unclamp
JOGSTOP(1)=0                         'stop axis 1
FOLRATIO(1)=1.8                      'set move velocity to 180% of master
MOVE(1)=0                            'move axis 1 to start position
DO:LOOP UNTIL DIST(2)<-10000         'wait for axis 2 to move 10000 units
X=X+1                                'increment count
LOOP UNTIL X> 3                      'loop until x>3
JOGSTOP(2)=0                         'stop axis 2
WAITDONE(2)                          'wait for axis 2 to stop jogging
MOVE(2)=0                            'move to starting position
FOLEND                               'disables analog following
FOLANALOG=0,0                        'end analog following
FOLEND
```



Signals:

Following Velocity axis 1
Following Velocity axis 2

SECTION 8 - TROUBLESHOOTING GUIDE

High voltages are present inside the unit. Always disconnect the power before performing any work on the unit. An electrical shock hazard exists that may cause serious injury or death if this unit is operated without its protective covers in place.

The status indicator lights (red LED's) on the front panel of the MX2000 provide an invaluable troubleshooting aid.

Power Led

The POWER indicator light is located on the Power Supply Card. When lit, it signifies the unit's power supply is energized. Should this light fail to come on, follow this procedure:

1. Check if the AC input power is applied; if not, apply power to the unit.

2. Check if the AC input power is within the operational range of 90 to 265 VAC, 50 to 60 Hz.

3. Check for an open fuse. There are two fuses (2A, 250 VAC, 3AG type, normal blow) located on the Power Supply card. If either of the fuses are open/blown, replace them and retest the unit. If the fuses are O.K., or if they fail after being replaced, an internal failure has occurred → contact Superior Electric. Do not apply power again.

Fault Led

The "FAULT" indicator light is located on the DSP controller card. When lit, it signifies a programming error, a processor error, or a motion error has occurred.

Busy Led

The "BUSY" indicator lights are located on the dual axis card. When lit, they signify the control has received or executed a motion command.

Serial Communications

If you are unable to establish serial communications between a host computer and the MX2000 controller:

1. Make sure that all hardware connections have been made properly, cable lengths do not exceed specified limits, and that power cables are isolated from the communications cables.

2. Make sure that a program is not being executed by the MX2000 while trying to establish communications. If a program is running, it can be stopped by pressing Ctrl-C or F2 from the terminal mode.

3. Make sure that the controller's baud rate matches that of the host computer and the correct communications protocol (RS232 or RS485) is selected. Also, ensure that the correct com port is chosen on the host computer.

The MX2000's baud rate and communications protocol is selected with a "BAUD" DIP switch located on the front panel of the DSP card. If this switch has to be reset, you must cycle power to the controller, since these switches are read only at power up.

The software command for setting the host computer's baud rate can be found in the MX2000 program editor under the menu item labelled System/Configure Com Port. The MX2000 serial communications format is 8 data bits, no parity, and 1 stop bit ("8-N-1").

If You Can Not Access Axis I/O:

Make sure that the polarity jumper on the Axis card is in the appropriate setting, sink or source, depending on your particular application. (Refer to section 5.11)

If a problem persists, contact Superior Electric's Motion Control Applications Engineering Department at 1-800-SUPELEC (1-800-787-3532), between the hours of 8:00 am and 5:00 pm EST.

(this page intentionally left blank)

SECTION 9 - GLOSSARY

ABSOLUTE MODE - Motion mode in which all motor movements are specified in reference to an electrical home position.

ABSOLUTE POSITION - A data register in the Controller which keeps track of the commanded motor position. When the value in this register is zero, the position is designated "Electrical Home".

ACCELERATION - The rate at which the motor speed is increased from its present speed to a higher speed (specified in units/second/second).

ACCURACY (of step motor) - The noncumulative incremental error which represents step to step error in one full motor revolution.

ALL WINDINGS OFF - Applying an average zero motor current at standstill to alleviate motor heating or eliminate holding torque.

AMBIENT TEMPERATURE - The temperature of the air surrounding the motor or drive.

ASCII - (American Standard Code for Information Interchange). A format to represent alphanumeric and control characters as seven-or eight-bit codes for data communications. A table of the ASCII codes appears on page 9-4.

ATTENTION CHARACTER - <nn, where "nn" is a unique integer from 1-99 (set by use of the unit ID# select switches) that is assigned to a Motion Controller arrayed in a multi-Controller system. The Attention Character directs the program command to the specified Motion Controller.

BASE SPEED - Starting speed for the motor (also known as low speed).

BAUD RATE - The rate of serial data communications expressed in binary bits per second.

BCD - (Binary Coded Decimal), a format to represent the digits 0 through 9 as four digital signals. Systems using thumbwheel switches may program commands using BCD digits. A BCD digit uses a standard format to represent the digits 0 through 9 as four digital signals.

The following table lists the BCD and complementary BCD representation for those digits. The Motion Controller uses the complementary BCD codes because the signals are active low.

BCD code table (0 = low state, 1=high state)

Digit	BCD Code	Complementary BCD Code
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	0100	1011
5	0101	1010
6	0110	1001
7	0111	1000
8	1000	0111
9	1001	0110

To represent numbers greater than 9, cascade the BCD states for each digit. For example, the decimal number 79 is BCD 0111:1001.

BOOST CURRENT - Increase of motor current during acceleration and deceleration to provide higher torques, which permits faster acceleration/deceleration times.

CLEAR - Input or Command to immediately halt all motor motion and program execution.

COLLECTORS (OPEN) - A transistor output that takes the signal to a low voltage level with no pull-up device; resistive pull-ups are added to provide the high voltage level.

CYCLE START - Command to initiate program execution.

CYCLE STOP - Command to stop program execution.

DAISY-CHAIN - A method to interface multiple Motion Controllers via RS485 to a single host using only one serial port.

DAMPING - A method of applying additional friction or load to the motor in order to alleviate resonance and ringout. Stepper motor shaft dampers are commercially available from several sources, including Superior Electric.

DECELERATION - The rate in which the motor speed is decreased from its present speed to a lower speed (specified in units/second/second).

DEVICE ADDRESS - A unique number used to assign which Motion Controller in a multi-drive stepper system is to respond to commands sent by a host computer or terminal. Device addresses from 1 - 99 are set by means of the ID # select switch. "00" is reserved to address all Motion Controllers in a system. Factory default is 01.

DWELL - See "WAIT".

ELECTRICAL HOME - The motor commanded position is zero (the Absolute Position register is zero).

FEEDRATE - The speed or velocity (in units per second) at which a move will occur.

FRICTION - Force that is opposite to the direction of motion as one body moves over another.

FULL-STEP - Position resolution in which 200 pulses corresponds to one motor revolution in a 200 step per revolution (1.8 degree) motor.

HALF-STEP - Position resolution in which 400 pulses corresponds to one motor revolution for a 200 step per revolution (1.8 degree) motor.

HANDSHAKE - A computer communications technique in which one computer's program links up with another's. The Motion Controller uses a software "Xon, Xoff" handshake method. See "XON" below.

HOST - The computer or terminal that is connected to the HOST serial port on the motion controller, and is responsible for primary programming and operation of the controller.

INCREMENTAL MODE - Motion mode in which all motor movements are specified in reference to the present motor position.

INDEXER - A Microprocessor-based programmable motion controller that controls move distance and speeds; possesses intelligent interfacing and input/output capabilities.

INDEX FROM RUN - See MARK REGISTRATION

INERTIA - Measurement of a property of matter that a body resists a change in speed (must be overcome during acceleration).

INERTIAL LOAD - A "flywheel" type load affixed to the shaft of a step motor. All rotary loads (such as gears or pulleys) have inertia. Sometimes used as a damper to eliminate resonance.

INSTABILITY - Also frequently called, "mid-range instability" or "mid-range resonance," this term refers to a resonance that occurs in the 500 - 1,500 steps/sec range. Mid-range instability is important because it refers to a loss of torque or a stalled motor condition at higher stepping rates. Since step motors do not start instantaneously above the mid-range resonance frequency, an acceleration scheme will have to be used to pass through the troublesome region.

JOG MOVE - moves the motor continuously in a specified direction.

LOAD - This term is used several ways in this and other manuals.

> **LOAD (ELECTRICAL)**: The current in Amperes passing through a motor's windings.
>
> **LOAD (MECHANICAL)**: The mass to which motor torque is being applied (the load being moved by the system).
>
> **LOAD (PROGRAMMING)**: Transmits a program from one commuter to another. "DOWNLOAD" refers to transmitting a program from a host computer (where a program has been written) to the Motion Controller where it will be used. "UPLOAD" refers to transmitting a program from a Motion Controller back to the host computer.

MARK REGISTRATION - A motion process (usually used in web handling applications) whereby a mark placed on the material is sensed (e.g., through the use of an optical sensor) and, following detection of this mark, the material is moved (indexed) a fixed length.

MECHANICAL HOME - The position where a switch input is used as a reference to establish electrical home.

MICROSTEPPING - A sophisticated form of motor control that allows finer resolution than full step (200 Pulses Per Revolution PPR) or half step (400 PPR) by adjusting the amount of current being applied to the motor windings. Microstepping up to 250 pulses per full step (50,000 PPR on a 200 step/rev or 1.8 degree motor) is supported. For 200 step per revolution motors, typical microstepping levels are 1/10-step and 1/125 step (2000 PPR and 25,000 PPR, respectively). Note: this is a DRIVE function.

MOVE TO MECHANICAL HOME - Function which allows the Motion Controller to move the motor and seek a switch to establish electrical home and set Absolute Position = zero.

NESTING - The ability of an active subroutine to call another subroutine. The Motion Controller can nest up to 16 levels.

NONVOLATILE MEMORY - Data storage device that retains its contents even if power is removed. Examples are EEPROM, flash memory, and battery-backed RAM.

OPTO-ISOLATION - The electrical separation of the logic section from the input/output section to achieve signal separation and to limit electrical noise. The two systems are coupled together via a transmission of light energy from a sender (LED) to a receiver (phototransistor).

PARITY -- An error checking scheme used in serial communications (via the RS-232 or RS-485 port) to ensure that the data is received by a Motion Controller is the same as the data sent by a host computer or terminal.

REDUCE CURRENT - Reduction of motor current during standstill to alleviate motor heating.

RESOLUTION - The minimum position command that can be executed. Specified in steps per revolution or some equivalent.

RINGOUT - The transient oscillatory response (prior to settling down) of a step motor about its final position. Note: a small wait or dwell time between moves can alleviate ringout problems.

RS232-C - EIA (Electronic Industries Association) communication standard to interface devices employing serial data interchanges. Single-wire connections for transmit and receive, etc.

RS-485 - EIA (Electronic Industries Association) communication standard to interface devices employing serial data interchanges. Two-wire connections (differential circuits) for transmit and receive, etc. Better than RS-232 for long wire runs and multi-drop circuits with many devices.

SINKING - An input that responds to, or output that produces, a "low" level (signal common or low side of the input/output power supply) when active.

SOURCING- An input that responds to, or output that produces, a "high" level (the voltage used for the input/output power supply) when active.

SUBROUTINE - A sequence of lines that may be accessed from anywhere in a program to preclude having to program those lines repetitively. This allows shorter, more powerful, and more efficient programs. See also NESTING.

TORQUE - Product of the magnitude of a force and its force arm (radius) to produce rotational movement. Units of measure are pound-inches, ounce-inches, newton-meters, etc.

TRANSLATOR- A motion control device (also called "translator drive") that converts input pulses to motor phase currents to produce motion.

WAIT - A programmed delay or dwell in program execution (specified in seconds).

XON / XOFF - A computer software "handshaking" scheme used by a Motion Controller.
The Motion Controller sends an XOFF character (ASCII Code 19) when it receives a command string with a Carriage Return and has less than 82 characters remaining in its host serial port buffer. The Controller sends an Xon when available buffer space reaches 100 characters or in response to an ID attention with adequate buffer space remaining. Since it is impossible for the host device to immediately cease transmissions, the next three characters (subject to the total serial buffer capacity of forty characters) received subsequent to the Motion Controller sending the XOFF character will be stored in the Motion Controller's serial buffer (a memory dedicated to store characters that are in the process of transmission).

Similarly, the Motion Controller will not transmit data if the host device has sent an XOFF character to the Controller; Motion Controller transmissions will resume when the Controller receives an XON character.

ASCII Table

ASCII Char	Dec Code	ASCII Char	Dec Code	ASCII Char	Dec Code	ASCII Char	Dec Code
Null	0	Space	32	@	64	`	96
SOH	1	!	33	A	65	a	97
STX	2	“	34	B	66	b	98
ETX	3	#	35	C	67	c	99
EOT	4	$	36	D	68	d	100
ENQ	5	%	37	E	69	e	101
ACK	6	&	38	F	70	f	102
BELL	7	‘	39	G	71	g	103
BS	8	(	40	H	72	h	104
HT	9	)	41	I	73	i	105
LF	10	*	42	J	74	j	106
VT	11	+	43	K	75	k	107
FF	12	,	44	L	76	l	108
CR	13	-	45	M	77	m	109
SO	14	.	46	N	78	n	110
SI	15	/	47	O	79	o	111
DLE	16	0	48	P	80	p	112
DC1	17	1	49	Q	81	q	113
DC2	18	2	50	R	82	r	114
DC3	19	3	51	S	83	s	115
DC4	20	4	52	T	84	t	116
NAK	21	5	53	U	85	u	117
SYNC	22	6	54	V	86	v	118
ETB	23	7	55	W	87	w	119
CAN	24	8	56	X	88	x	120
EM	25	9	57	Y	89	y	121
SUB	26	:	58	Z	90	z	122
ESC	27	;	59	[	91	{	123
FS	28	<	60	\	92	|	124
GS	29	=	61	]	93	}	125
RS	30	>	62	^	94	~	126
DEL	31	?	63		95	DEL	127

SECTION 10 - SPECIFICATIONS

MECHANICAL

MX2000-2	Size	(Inches):	5.34W x 10.63H x 7.48D	(See Figure 5.1)
		(Millimeters):	135.6W x 270H x 190D	
	Weight:		8.25 lbs (3.75 kg)	
MX2000-6	Size	(Inches):	9.34W x 10.63H x 7.48D	(See Figure 5.2)
		(Millimeters):	237.3W x 270H x 190D	
	Weight:		11.0 lbs (5.0 kg)	
MX2000-8	Size	(Inches):	19.0W x 10.63H x 7.54D	(See Figure 5.3)
		(Millimeters):	482.6W x 270H x 191.6D	
	Weight:		12.0 lbs (5.45 kg)	

ENVIRONMENTAL

Operating temperature:	+32° F to +122° F (0° C to +50° C)
Storage temperature	-40° F to +167° F (-40° C to +75° C)
Humidity	95% maximum, non-condensing
Altitude	10,000 feet (3048 meters) maximum

STEPPER DRIVE INTERFACE

There are two drive interfaces, one for each axis. Each interface has its own 7-position removable terminal strip connector. The pin assignments and signal characteristics are as follows:

Pin Assignments:

OPTO Supplies 5V power to the drive's opto isolators. A typical S.E. drive can draw 45 ma. maximum. (power)

PULSE 0 - 1.99 Mhz Square wave. (output)

DIR Motor direction control. (output)

AWO Signals drive to turn off motor current. (output)

RDCE Signals drive to reduce motor current (for S.E. drives, by 50%.) when motion is not occurring. (output)

BOOST Signals drive to increase motor current (for S.E. drives, by 50%.) during accel or decel. (output)

READY A high level applied to this input signifies that the drive is ready. A low level signifies either a drive failure or drive not powered. (input)

Output Signal Characteristics: (for outputs PULSE, DIR, AWO, RDCE and BOOST)

Open collector drivers (TTL types 7406 or 7407) are used The off-state voltage rating is 30V and the on-state current rating is 40 ma.

Input Signal Characteristics: (for READY signal)

Input signal loading :	10 K ohm.
High level input voltage:	3.5 - 5.0 V
Low level input voltage:	0 - .9 V

HOST SERIAL PORT

This serial port is used to program the unit or to interface to a host controller. There are two serial interfaces: RS232 and RS485. The RS232 interface is provided on a 9-pin D female connector. The RS485 interface is provided on a 6-position removable terminal strip. Units can be daisy chained using the RS485 interface. The unit that communicates with the host can do so using RS232 or RS485. This selection is made with the 485/232 dip switch on the front of the DSP card. The signal characteristics are in accordance with the EIA RS232, RS422, and RS485 standards. The RS422-RS485 standard allows for line length up to 4000 ft. The RS232 standard allows for line lengths up to 50 ft.

Proper cable termination is important when using the RS485 interface. Line termination resistors (120 ohms 1/4 W) should be installed at the ends of the wires connecting the units together. One terminating resistor is needed at the far end of the wire, which connects all the unit's RX terminals. The other resistor is placed at the near end of the wire, which connects all the unit's TX terminals. See Section 5.3 for further details.

The data format is: no parity, 8 data bits and 1 stop bit. The baud rate is switch-selectable, with settings for 9600, 19,200 and 38,400 baud. An ambiguous selection will default to 9600 baud; no selection (all switches to the right) gives 4800 baud. The baud rate switches are read upon powering up; therefore, any changes to them are only effective upon cycling power to the unit.

HOST RS485 Port pin assignments:

pin 1	GND	Signal ground reference (100 ohms to ground)
pin 2	RX+	Receive data input
pin 3	RX-	Receive data input
pin 4	TX+	Transmit data output
pin 5	TX-	Transmit data output
pin 6	SHLD	Shield drain-wire connection to case (chassis ground)

HOST RS232 Port pin assignments:

pin 1	GND	ground
pin 2	TX	Transmit data output
pin 3	RX	Receive data input
pin 4	GND	ground
pin 5	GND	ground
pins 6-9	---	no connection

AUXILIARY SERIAL PORT

This serial port is used to communicate to a user's control panel. Messages prompting an operator can be output to the panel's display, and keyboard responses can be read from the panel. It is configurable either for RS232 or RS485, via a jumper on the DSP card, located behind the port connector.

The data format is: no parity, 8 data bits and 1 stop bit. The baud rate is software-selectable via the HOST serial port, using the SETCOM command, with settings for 300 to 38,400 baud. The signal characteristics are in accordance with the RS422/RS485 standard (which allows for line length up to 4000 ft) or with RS232 (up to 50 ft), depending on the jumper setting.

When operating this port in RS485 mode, install a line termination resistor (120 ohms 1/4 W) across the RX terminals, and at the far end of the wire connected to the TX terminals. For RS232 operation, use the RX-, TX-, and GND pins.

A 6-position removable terminal strip is provided for this port, with the following pin assignments:

Auxiliary Serial Port pin assignments:

pin 1	GND	Signal ground reference (100 ohms to ground)
pin 2	RX+	Receive data input
pin 3	RX-	Receive data input
pin 4	TX+	Transmit data output
pin 5	TX-	Transmit data output
pin 6	SHLD	Shield drain-wire connection to case (chassis ground)

DSP CARD INPUTS

These inputs can be configured for either sinking or sourcing operation via a jumper on the card, behind the connector. CLR allows for starting and stopping ("Motion Clear") the program. Following a shutdown, the system can be restarted by cycling power or by receiving a "RUN" command over the HOST SERIAL PORT, provided the CLR line is active (CLR switch closed).

The SEL lines select the program to be automatically executed upon system power-up or following a serial port RESET command. *See Section 5.2.3.4 for complete details on the use of these inputs.*

A 6-position removable terminal strip is provided for this function, with the following pin assignments:

Pin Assignments:

pin 1	CLR	Used to start and stop the system. Tie low (to COM) when sinking or high (to +24V) when sourcing to allow program to execute and allow motion to occur in either Host or Program operating modes. Opening the connection causes operation to cease.
pin 2	SEL4	These 3 pins are used to select the program to be auto-executed upon power-up or following a RESET command over one of the Serial Ports
pin 3	SEL2	
pin 4	SEL1	
pin 5	+24V	This pin is connected to the internal 24 V power supply and is used for sourcing operation of these inputs
pin 6	COM	Signal common, for sinking operation of these inputs

Signal Characteristics:

Inputs - Sink Mode

On-state voltage range:	0 - 12 volts dc
Input current @ 12 V:	3.2 mA typical
input current @ 0 V:	6.8 mA typical

Inputs - Source Mode

On-state voltage range:	12 - 24 volts dc
Input current @ 12 V:	3.2 mA typical
input current @ 24 V:	6.8 mA typical

ENCODER INTERFACE

There are two incremental encoder interfaces. Encoders with differential or single-ended outputs can be used. Maximum count rate is 2 million counts/sec. The connector is a 9-position removable terminal strip.

Pin Assignment:

+5	Supplies 5V power to the encoder.
GND	5V return for encoder excitation.
SHLD	Tie point for shield, when using shielded cable. This terminal is connected to chassis ground.
A+	Encoder quadrature channel A. (not used with single ended encoders)
A-	Encoder quadrature channel A.
B+	Encoder quadrature channel B. (not used with single ended encoders)
B-	Encoder quadrature channel B.
I+	Encoder index (once/rev.) pulse. (not used with single ended encoders)
I-	Encoder index (once/rev.) pulse.

Signal Characteristics:

Hi level Input current	7.3 ma. typ. at Vin = 5V	(A+, B+, I+)
	0 ma. typ. at Vin = 5V	(A-, B-, I-)
Lo level Input current	-7.3 ma. typ. at Vin = 0 V	

ANALOG OUTPUTS

There are two 12 bit analog outputs, with an output voltage range of -10 to + 10 V. The outputs are provided on a 6-position removable terminal strip.

Pin Assignment:

OUT	Analog Output
AGND	Analog ground reference
EN+	Future function, not presently implemented
EN-	Future function, not presently implemented
RDY+	Future function, not presently implemented
RDY-	Future function, not presently implemented

Signal Characteristics:

Output loading:	5 ma. maximum. (2 K ohm)	
Output Accuracy:	Zero output error	± .03 V
	Full Scale output error	± .11 V

AXIS INPUTS

There are two sets of axis inputs, each contained on a 9-position removable terminal strip.

Pin Assignment:

24V or COM	24V internal power supply and return, used to wire limit switches, home switch, etc.
+LIM	Positive travel limit.
-LIM	Negative travel limit.
EVNT1	Input, typically connected to a home switch or mark registration signal.
EVNT2	Input, typically connected to a home switch or mark registration signal.
10V or AGND	10V power supply and return, intended use is to power a potentiometer connected to the analog inputs.
IN+	Analog differential input
IN-	Analog differential input
SHLD	Tie point for shield, when using shielded cable. This terminal is connected to chassis ground.

Signal Characteristics:

Signals +LIM, -LIM, EVNT1 and EVNT2 can be collectively configured as active low (sink) or active high (source) inputs. This selection is made with a jumper on the board.

SINK INPUT characteristics

On-state current	10.5 ma. with Vin = 0
On-state voltage	0 - 3V
Off-state voltage	24V

SOURCE INPUT characteristics

On-state voltage	12 - 24V
On-state current	10.5 ma. with Vin = 24V 4.5 ma. with Vin = 12 V
Off-state voltage	0 - 3V

EVNT1, EVNT2 response time	1.2 us. typ. 4 us. max.

10V OUTPUT

Voltage	10 ± .07 V
Current rating	0 - 20 ma. maximum

ANALOG INPUTS

Resolution	12 bits
Sample rate	1950 samples/sec.
Voltage range IN+ to IN-	-10 to +10 V
Input impedance IN+ or IN- to AGND	20 Kohm
Full scale input accuracy	±.1 V
Zero input accuracy	±.035 V

Note: The Digital I/O card, described in Appendix A, has the same specifications as the axis inputs defined above.

POWER SUPPLY

Ac power is applied on a fixed, three-terminal screw-clamp connector. The internal power supply produces the following output voltages +5, +15 , -15 and 24VDC. The +15 and -15V outputs are used internally. The 5V output is used internally, and it provides encoder excitation and stepper drive opto supply. The 24V isolated output is used with the system I/O; it is not used internally.

Input power

MX2000-2,-6	voltage	90 - 265 VAC, 50/60 Hz.
	current	<.65 A at 115 VAC
	fuse	2A (normal blow), 250 VAC, 3AG type (2 required)
MX2000-8	voltage	90 - 132 VAC or 175 - 264 VAC, 50/60 Hz.
	current	<3 A at 115 VAC
	fuse	2A (normal blow), 250 VAC, 3AG type (2 required)

24V output (system I/O)

voltage	24V ± 3V
current rating	750 ma.

+5V output (stepper drive & encoder)

voltage	5 ± 0.2 V
current rating	100 ma. per encoder

A Superior Electric Stepper Drive can draw 45 ma. from the 5V opto supply under the following signals all on: PULSE, DIR & one of these: (AWO or RDCE or BOOST).

EXPANSION I/O - BCD PORT

The EXPANSION I/O - BCD port has been designed to interface to BCD switches and/or to an "OPTO 22" module rack. The port consists of 24 bi-directional I/O points contained on a standard 50 pin header. Odd pins 1 - 47 are signal pins, even pins 2 - 50 are ground and pin 49 is not used. An I/O point, used as an output, sinks current to ground when active. The output load is connected between a positive voltage and the I/O pin. An I/O point, used as an input, recognizes a grounded input as active.

Input characteristics

On state input voltage	0 - 1.5V
On state input current	1 ma. max. (Vin = 0)
Off state input	2.9 - 30 V or open circuit

Output characteristics

Load voltage	30 V maximum
On state voltage	.5 V (15 ma. load current)
On state current	15 ma. maximum

Note: The Expansion I/O card, described in Appendix A, has the same specifications as the Expansion I/O port defined above.

INDEX

Subject **Page**

APPENDIX A - OPTIONAL CARDS

General Handling and Installation Instructions

The Optional cards described in this appendix are applicable to the MX2000-6 and MX2000-8 configurations only. Up to 4 of these plug-in cards can be installed in the MX2000-6,-8 enclosure in the combination which best suits the needs of your system. Blank panels are also available for unused slots

Handling Precautions

* High voltages are present inside this unit. Be sure to disconnect the ac power before opening the unit. An electrical shock hazard exists that may cause serious injury or death if this unit is operated without its protective covers in place.

* Do not exceed the voltage or current ratings of the various inputs and outputs; please read the electrical specifications in Section 10. This will protect the circuitry and components from accidental damage.

* Please follow good wiring practices and keep low-level signal lines away from power and motor wiring. It is best to use shielded, twisted-pair cables for signal lines, being sure to ground the shields at one end. Doing this will help to avoid electrical noise interference problems.

* When the unit is opened for installation or removal of the individual circuit cards, be sure to treat the cards as static-sensitive components to avoid damage due to electrostatic discharge (ESD). Work only in ESD protected areas, and it is best not to touch the circuit conductors or components unless you are wearing an ESD-protective grounding strap. Try to handle the cards only by their metal front panels.

Installation

NOTE: Refer to each card's detailed instructions for switch and jumper setting information; these should be set up prior to installing the cards in your system.

Be sure power to the MX enclosure is turned off before installing or removing any cards! Otherwise, they may be damaged. Place the card in the enclosure's card guides for the position you wish the card to occupy. Push the card in firmly to seat its DIN connector into the mating connector on the enclosure's backplane board. Then tighten the two thumbscrews at the top and bottom of the front panel to secure the card into the enclosure.

Dual-Axis Interface Card

Description

The two-axis drive and encoder interface card with axis I/O includes these features:

- two sets of optically-isolated stepper pulse & direction outputs, with boost & reduce
- two sets of analog outputs, which can be used to control servo or dc motors
- two sets of analog inputs, which can be configured for joystick operation
- axis I/O (limits, home, mark registration) - sink/source, optically-isolated, 12 to 24 Vdc
- two sets of quadrature encoder inputs, single-ended or differential, with index marker

Switch and Jumper Settings

Before installing the card in your MX2000 system, be sure to set the board identification number (ID) dip switch and the sink/source jumpers as described below. The board outline drawing shows their locations.

1. Board ID Number:

   a. Determine how many axis I/O boards are to be used in your system. Assign an identification (ID) number for each one, from 1 to 4. This number is used as the prefix in addressing the I/O points; see the programming examples below.

   b. Mark each axis # in the white square on the front panel, using an indelible marker.

   c. Set the Board ID dip switch (SW1 - see outline drawing below for location) for each board as follows:

Board ID #	SW1 Switch Position	Switch Setting ON	 OFF
1	C B A	ON ON ON	
2	C B A	ON ON	 OFF
3	C B A	ON ON	 OFF
4	C B A	ON	 OFF OFF

IMPORTANT NOTE: The card's address setting table shows that up to 8 axis cards may be installed in your MX system. This is to allow for possible future expansion. **However, at present, only 4 axis cards (total of 8 individual axes) may be installed.**

2. Sink/source jumper setting:

   a. For sourcing operation, place the jumper on the center pin and the right-hand pin labelled "SOURCE".

   b. For sinking operation, place the jumper on the center pin and the left-hand pin labelled "SINK".

Front-Panel Connections

All connections are made via the removable screw-clamp connectors on the front panel. See the figure below for locations of the various terminals.

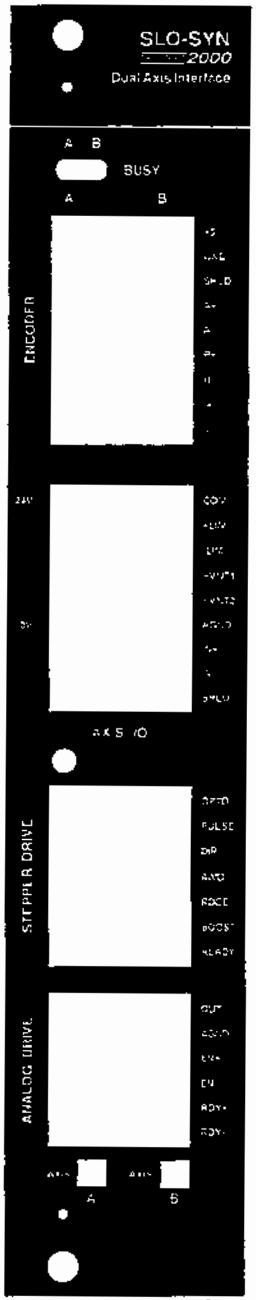
Note the small white square on the label; this is provided so that the axis numbers (e.g., 1, 2, 3, 4, etc.) can be written with an indelible marker for ease in keeping track of the various axes for programming purposes, and for connection to the motor drives.

Specifications

Complete electrical specifications for the various ports on this card can be found in Section 10 of this manual.

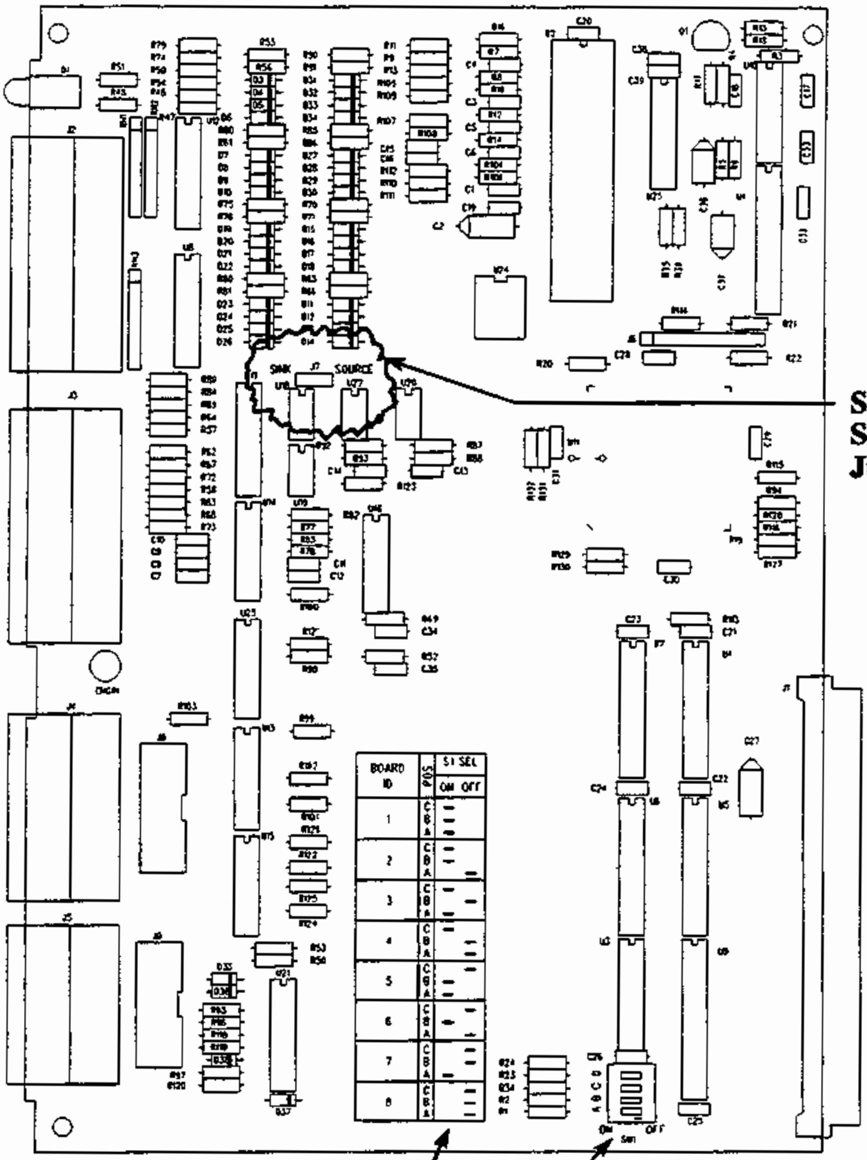
Dual-Axis Interface Card

Front Panel Layout



Unless otherwise noted the labels on the right side of the card refer to both axes.

Card Layout



Sink/Source Select Jumper

Board ID Setup Table

Board ID DIP Switch

Digital I/O Card

Description

This card provides 24 program-testable digital inputs and 16 program-settable digital outputs, all of which are optically-isolated and can be configured for either sinking or sourcing operation. An isolated 24 Vdc power supply output is also provided, therefore external power is not needed for the optocouplers.

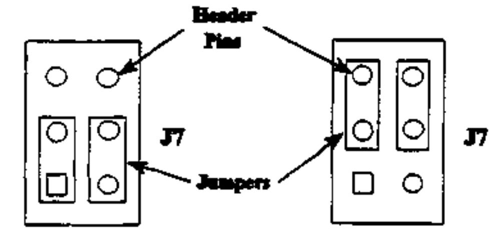
Switch and Jumper Settings

1. Board ID Number:
   a. Determine how many digital I/O boards are to be used in your system. Assign an identification (ID) number for each one, from 1 to 4. This number is used as the prefix in addressing the I/O points; see the programming examples below.
   b. Mark each board with its ID # in the white square on the front panel, using an indelible marker.
   c. Set the Board ID dip switch (S2 - see outline drawing below for location) for each board as follows:

Board ID #	S2	Switch Setting	
	Switch Position	ON	OFF
1	B		OFF
	A		OFF
2	B		OFF
	A	ON	
3	B	ON	
	A		OFF
4	B	ON	
	A	ON	

2. Sink/Source Jumpers:

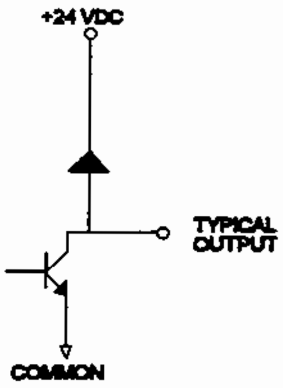
A set of pin headers (J7) and pair of jumpers is provided on the lower left corner of the board to set the I/O for sinking mode or sourcing mode. All the inputs and all the outputs are set as a group for either one mode or the other. Determine which mode is desired, then place the jumpers as shown below.



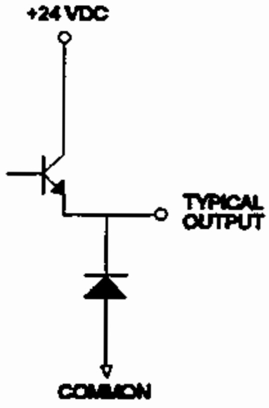
Jumper Positions for Sourcing I/O

Jumper Positions for Sinking I/O

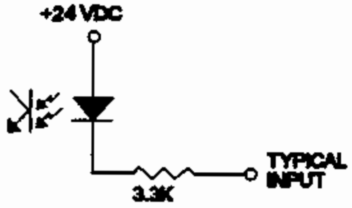
Equivalent Circuits



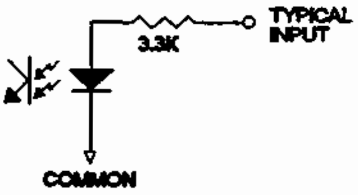
SINK OUTPUT



SOURCE OUTPUT



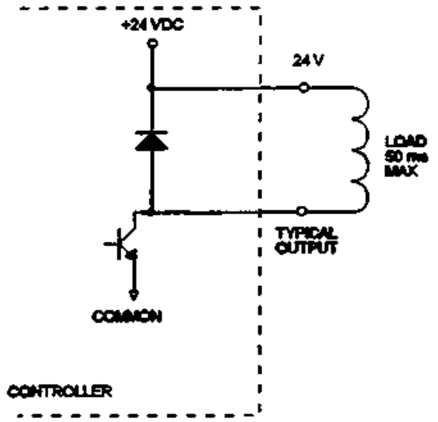
SINK INPUT



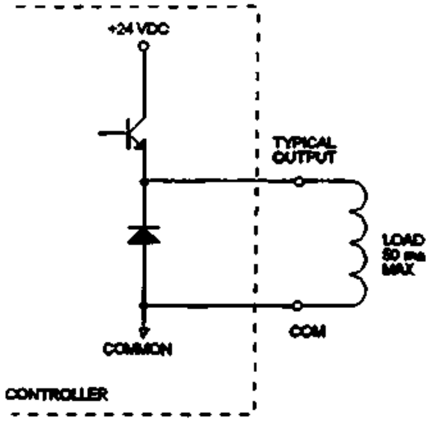
SOURCE INPUT

Note: All inputs and outputs are configured together for Sink or Source via the selection jumper (J7) as described on the previous page.

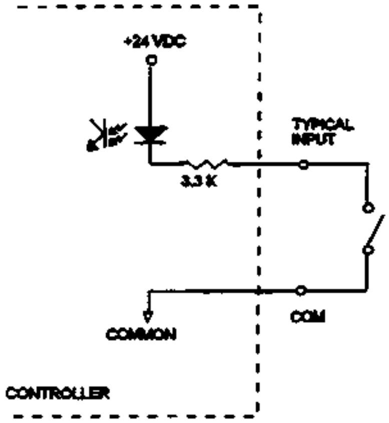
Sample Connections



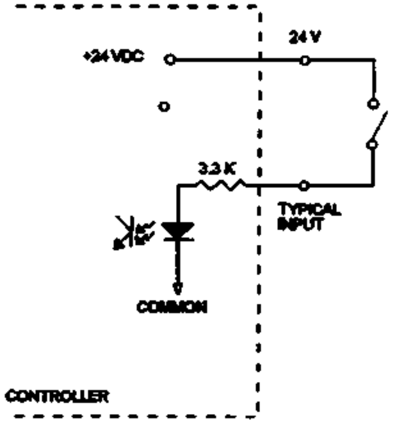
SINK OUTPUT



SOURCE OUTPUT



SINK INPUT



SOURCE INPUT

Notes:

1. +24 VDC and Common are the Power Supply output and common terminals respectively

2. All I/O's are configured together for sink or source mode via the sink/source selection jumper (J7) as described previously.

Front-Panel Connections

All connections are made via the removable screw-clamp connectors on the front panel. See the figure below for locations of the various terminals.

Specifications

Isolated Current-Limited Power Supply for Use with Single Point I/O)	24vdc (+/- 10%) At 0.75a
Inputs & Outputs:	
Single Point I/O Electrical Specs.	Outputs Shall Be Able to Drive a Shorted Load Indefinitely Without Damage. No Suppression Required for Inductive Loads.
Outputs (Sink Mode)	
Load Power Supply	Can Use Built-In 24 Vdc Supply or External 12 to 24 Vdc Supply
Current Rating	50 Ma
Voltage Rating	24 Vdc
On State Voltage @ 50 Ma	2.0 V Max.
Off State Leakage @ 24vdc	0.6 Ma Max.
Outputs (Source Mode) Current Rating	50 Ma
On State Voltage @ 50 Ma	20 V Min.
Off State Leakage	0.6 Ma Max.
Inputs (Sink Mode)	
On State Voltage Range	0 - 12 Volts
Input Current @ 12 V	2.3 Ma
Input Current @ 0 V	6.5 Ma
Inputs (Source Mode)	
On State Voltage Range	10 - 24 Vdc
Input Current @ 10 V	2.3 Ma
Input Current @ 24 V	6.5 Ma

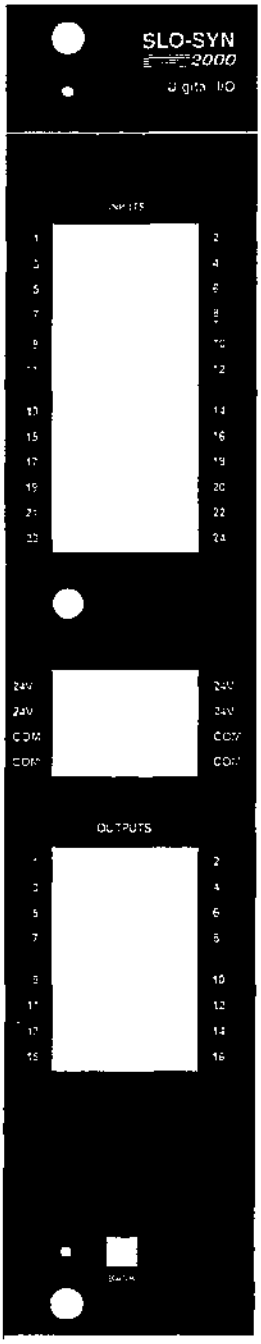
Programming Examples

Read the state of Input 3 on I/O board #1:	IN(103)
Read the state of Input 21 on I/O board #4:	IN(421)
Turn on Output 5 on I/O board #2:	OUT(205) = ON
Turn off Output 11 on I/O board #3):	OUT(311) = OFF

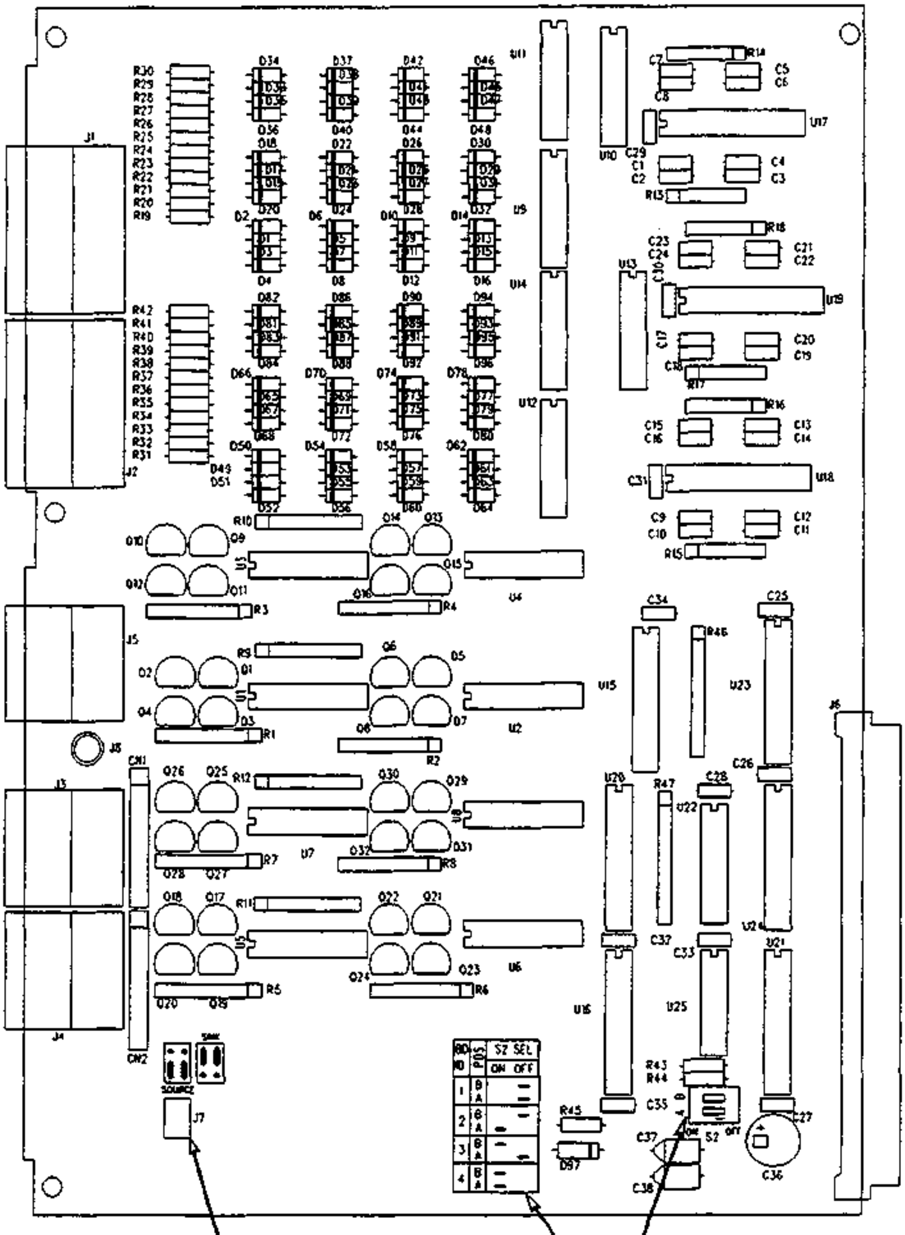
For more detailed programming instructions, see Section 6.0," Programming and Software Reference," of this manual.

Digital I/O Card

Front Panel Layout



Card Layout



Sink/Source Jumpers

Board ID Setting Table

Board ID DIP Switch

Expansion I/O & BCD Card

Description

This card provides two ports with 50-pin header connectors for attaching additional "OPTO-22"-style I/O module cards, or for using BCD switch banks. Each of the two ports can support up to 24 I/O modules (total of 48 per board), or up to 4 BCD banks of 7 digits plus sign (up to 8 banks per board). In addition, BCD's and I/O can be combined on the same port. Section 5.3.2.5 of this instruction manual gives complete information on the use of the expansion ports.

Switch and Jumper Settings

1. Board ID Number:

a. Determine how many Expansion I/O boards are to be used in your system and assign an identification (ID) number for each one as follows: The MX2000-6 may have up to 3 Expansion Boards; the I/O port on the power supply board uses Board ID#1, so the first Expansion Board should be set to ID#2, the second to ID#3 and third to ID#4. The MX2000-8 may have up to 4 Expansion Boards. Because it does not incorporate the Power Supply/Expansion board used in the MX2000-6 ID#'s 1 - 4 can be used. This number is used as the prefix in addressing the I/O points; see the programming examples below.

b. Mark each board with its ID # in the white square on the front panel, using an indelible marker.

c. Set the Board ID dip switch (S1 - see outline drawing below for location) for each board as follows:

Board ID #	S1 Switch Position	Switch Setting ON	 OFF
1	B		OFF
	A		OFF
2	B		OFF
	A	ON	
3	B	ON	
	A		OFF
4	B	ON	
	A	ON	

Front-Panel Connections

All connections are made via the removable screw-clamp connectors on the front panel. See the figure below for locations of the various terminals.

Specifications

Electrical specifications for the I/O - BCD ports on this card are found in Section 10 of this instruction manual.

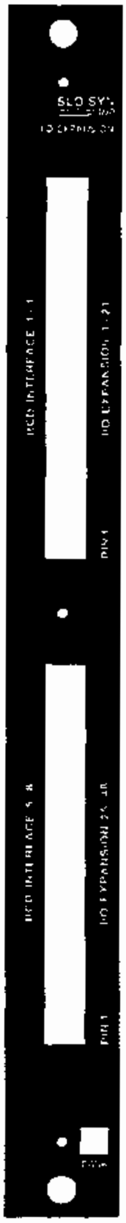
Programming Examples

Read the state of Input Module 3 on Expansion I/O board #1:	EXIN(103)
Read the state of Input Module 31 on Expansion I/O board #4:	EXIN(431)
Turn on Output Module 5 on Expansion I/O board #2:	EXOUT(205) = ON
Turn off Output Module 41 on Expansion I/O board #3:	EXOUT(341) = OFF

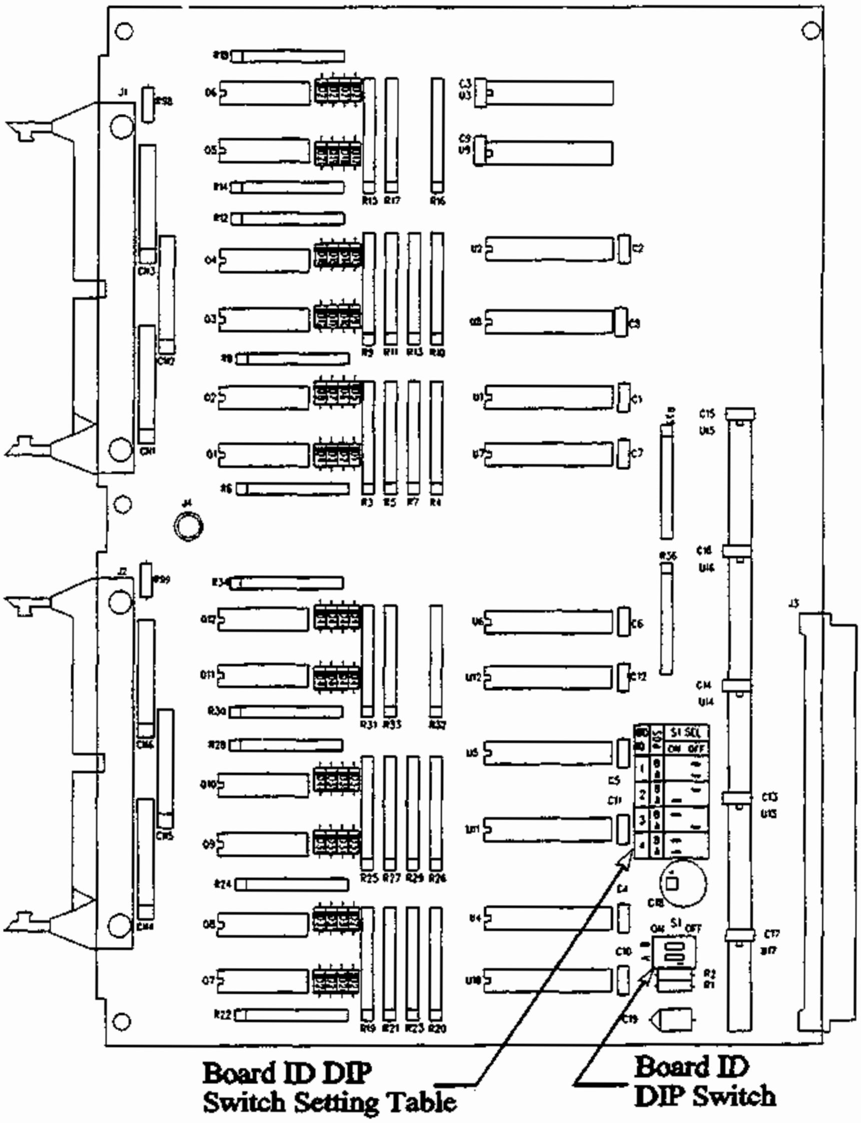
For more detailed programming instructions, see the Software Reference section of the MX2000 manual.

Expansion I/O & BCD Card

Front Panel Layout



Card Layout



WARRANTY AND LIMITATION OF LIABILITY

Warner Electric (the "Company"), Bristol, Connecticut, warrants to the first end user purchaser (the "purchaser") of equipment manufactured by the Company that such equipment, if new, unused and in original unopened cartons at the time of purchase, will be free from defects in material and workmanship under normal use and service for a period of one year from date of shipment from the Company's factory or a warehouse of the Company in the event that the equipment is purchased from the Company or for a period of one year from the date of shipment from the business establishment of an authorized distributor of the Company in the event that the equipment is purchased from an authorized distributor.

THE COMPANY'S OBLIGATION UNDER THIS WARRANTY SHALL BE STRICTLY AND EXCLUSIVELY LIMITED TO REPAIRING OR REPLACING, AT THE FACTORY OR A SERVICE CENTER OF THE COMPANY, ANY SUCH EQUIPMENT OF PARTS THEREOF WHICH AN AUTHORIZED REPRESENTATIVE OF THE COMPANY FINDS TO BE DEFECTIVE IN MATERIAL OR WORKMANSHIP UNDER NORMAL USE AND SERVICE WITHIN SUCH PERIOD OF ONE YEAR. THE COMPANY RESERVES THE RIGHT TO SATISFY SUCH OBLIGATION IN FULL BE REFUNDING THE FULL PURCHASE PRICE OF ANY SUCH DEFECTIVE EQUIPMENT. This warranty does not apply to any equipment which has been tampered with or altered in any way, which has been improperly installed or which has been subject to misuse, neglect or accident.

THE FOREGOING WARRANTY IS IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, and of any other obligations or liabilities on the part of the Company; and no person is authorized to assume for the Company any other liability with respect to equipment manufactured by the Company. The Company shall have no liability with respect to equipment not of its manufacture. THE COMPANY SHALL HAVE NO LIABILITY WHATSOEVER IN ANY EVENT FOR PAYMENT OF ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR INJURY TO ANY PERSON OR PROPERTY.

Written authorization to return any equipment or parts thereof must be obtained from the Company. The Company shall not be responsible for any transportation charges.

IF FOR ANY REASON ANY OF THE FOREGOING PROVISIONS SHALL BE INEFFECTIVE, THE COMPANY'S LIABILITY FOR DAMAGES ARISING OUT OF ITS MANUFACTURE OR SALE OF EQUIPMENT, OR USE THEREOF, WHETHER SUCH LIABILITY IS BASED ON WARRANTY, CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR OTHERWISE, SHALL NOT IN ANY EVENT EXCEED THE FULL PURCHASE PRICE OF SUCH EQUIPMENT.

Any action against the Company based upon any liability or obligation arising hereunder or under any law applicable to the sale of equipment, or the use thereof, must be commenced within one year after the cause of such action arises.

These products are subject to the standard Limitation of Liability and/or Warranty of the Company.

The right to make engineering refinements on all products is reserved. Dimensions and other details are subject to change

Distribution Coast-To-Coast and International

Superior SLO-SYN products are available worldwide through an extensive authorized distributor network. These distributors offer literature, technical assistance and a wide range of models off the shelf for fastest possible delivery and service.

In addition, Warner Electric sales engineers are conveniently located to provide prompt attention to customers' needs. Call the nearest office listed for ordering and application information or for the address of the closest authorized distributor.

In U.S.A. and Canada
383 Middle Street
Bristol, CT 06010
Tel: 860-585-4500
Fax: 860-589-2136
Customer Service: 1-800-787-3532
Product Application: 1-800-787-3532
Product Literature Request: 1-800-787-3532
Fax: 1-800-766-6366
Web Site: www.warnernet.com

In Europe
Warner Electric (Int.) Inc.
La Pierreire
CH-1029 Villars-Ste-Croix, Switzerland
Tel: 41 021 631 33 55
Fax: 41 021 636 07 04

DANA
WARNER ELECTRIC
MOTORS AND CONTROLS DIVISION
BRISTOL PLANT
383 MIDDLE STREET • BRISTOL, CT 06010
(860) 585-4500 FAX: (860) 589-2136
People Finding A Better Way