MM4006 8-Axis Motion Controller/Driver

Version 1.01 Firmware

USER'S MANUAL

Newport®

Warranty

Newport Corporation warrants this product to be free from defects in material and workmanship for a period of 1 year from the date of shipment. If found to be defective during the warranty period, the product will either be repaired or replaced at Newport's option.

To exercise this warranty, write or call your local Newport representative, or contact Newport headquarters in Irvine, California. You will be given prompt assistance and return instructions. Send the instrument, transportation prepaid, to the indicated service facility. Repairs will be made and the instrument returned, transportation prepaid. Repaired products are warranted for the balance of the original warranty period, or at least 90 days.

Limitation of Warranty

This warranty does not apply to defects resulting from modification or misuse of any product or part. This warranty also does not apply to fuses, batteries or damage from battery leakage.

This warranty is in lieu of all other warranties, expressed or implied, including any implied warranty of merchantability or fitness for a particular use. Newport Corporation shall not be liable for any indirect, special, or consequential damages.

This manual has been provided for information only and product specifications are subject to change without notice. Any changes will be reflected in future printings.

P.A. de Saint Guénault – 3 bis, rue J. Mermoz – BP 189 – 91006 – Evry Cedex – France
Tel.: 33 (0)1.60.91.68.68 – Fax: 33 (0)1.60.91.68.69

Table of Contents

Section 3 — Remote Mode

Section 4 — Motion Control Tutorial

Section 5 — Trajectory Functions Tutorial

Section 6 — Feature Descriptions Tutorial

Section 7 — Servo Tuning

Section 8 — Appendices

We declare that the accompanying product, identified with the “CE” mark, meets all relevant requirements of Directives:

- 73/23/EEC, for Low Voltage Compatibility.
- 89/336/EEC for Electromagnetic Compatibility.

Generic standard:

Emission: NF EN61326-1, for measurement, lab and control equipment.

Immunity: NF EN61326-1, for measurement, lab and control equipment.

Safety: EIC 1010-1, safety standards for measurement, lab and control equipment.

Newport Corporation shall not be liable for damages when using the product:

- Modification of the product.
- Using modified connector, or modified or not supplied cables.
- Connecting this product to non-CE equipment.

MM4006

4-Axis Motion Controller/Driver

Newport®

EC Declaration of Conformity

We declare that the accompanying product, identified with the "CE" mark, meets all relevant requirements of Directive:

- 73/23/CEE, for Low Voltage Compatibility.
- 89/336/EEC for Electromagnetic Compatibility.

Compliance was demonstrated to the following specifications:

EMISSION:

Radiated and Conducted Emission in accordance with relative prescription to the EMC, NF EN61326-1: Standards for measurement, lab and control equipment.

IMMUNITY:

Radiated and Conducted Emission in accordance with relative prescription to the EMC, NF EN61326-1: Standards for measurement, lab and control equipment.

SAFETY:

CEI 1010-1, safety standards for measurement, lab and control equipment.

Jean-Marc DELAHAYE
Quality Director
Zone Industrielle
45340 Beaune-la-Rolande, France

Table of Contents
Section 1 — Introduction

Newport

Section 1
Introduction

1.1 Safety Considerations

The following general safety precautions must be observed during all phases of operation of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture and intended use of this equipment.

Disconnect or do not plug in the power cord in the following circumstances:

- If the power cord or any other attached cables are frayed or damaged in any way.
- If the power plug or receptacle is damaged in any way.
- If the unit is exposed to rain, excessive moisture or liquids are spilled on it.
- If the unit has been dropped or the case is damaged.
- If you suspect service or repair is required.
- Whenever you clean the case.

To protect the equipment from damage and avoid hazardous situations, follow these recommendations:

- Do not open the equipment. There are no user serviceable or adjustable parts inside.
- Do not make any modifications or parts substitutions to the equipment.
- Return the equipment to Newport for any service and repair needs.
- Do not touch, directly or with other objects, live circuits inside the unit.
- Do not operate the unit in an explosive atmosphere.
- Keep all air vents free of dirt and dust.
- Do not block air vents with paper or other objects.
- Keep all liquids away from unit.
- Do not expose equipment to excessive moisture (>85% humidity).

WARNING

All attachment plug receptacles in the vicinity of this unit are to be of the grounding type and properly polarised.

Contact your electrician to check your receptacles.

CAUTION

This product is equipped with a 3-wire grounding type plug. Any interruption of the grounding connection can create an electric shock hazard. If you are unable to insert the plug into your wall plug receptacle, contact your electrician to perform the necessary alterations to assure that the green (green-yellow) wire is attached to earth ground.

CAUTION

This product operates with voltages that can be lethal. Pushing objects of any kind into cabinet slots or holes, or spilling any liquid on the product, may touch hazardous voltage points or short-circuit parts.

CAUTION

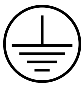
Opening or removing covers will expose you to hazardous voltages. Refer all servicing internal to this instrument enclosure to qualified service personnel who should observe the following precautions before proceeding:

- **Turn power OFF and unplug the unit from its power source**
- **Disconnect all cables**
- **Remove any jewellry from hands and wrists**
- **Use only insulated hand tools**
- **Maintain grounding by wearing a wrist strap attached to the instrument chassis.**

1.2 Conventions And Definitions

1.2.1 Symbols and Definitions

The following are definitions of safety and general symbols used on equipment or in this manual.



Chassis Ground. Indicates a connection to the equipment chassis which includes all exposed metal structure.

WARNING

Warning. Calls attention to a procedure, practice or condition which, if not correctly performed or adhered to, could result in injury or death.

CAUTION

Caution. Calls attention to a procedure, practice or condition which, if not correctly performed or adhered to, could result in damage to equipment.

NOTE

Note. Calls attention to a procedure, practice or condition which is considered important to remember.

Motor OFF

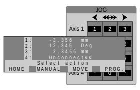
Menu Level (sample). Indicates the menu level from which to start a certain Quick front panel sequence.

STATUS

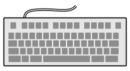
Function Key (sample). Represents one of the four function keys identified on the display's menu line with the indicated word.

UP

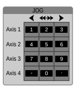
Function Key (sample). Represents one of the four function keys identified on the display's menu line with the indicated word that must be pressed multiple times.



Fast Front Panel Sequence. Indicates a quick key sequence description to get the described function. It is intended to be used by more experienced users as a quick reminder.



Remote Command. Indicates a remote command equivalent to the local function being described.



Numeric Keypad. Represents the numeric keypad on the front panel. Shown in a fast sequence, indicates a numeric entry on the keypad.

1.2.2 Terminology

The following is a brief description of terms specific to motion control and this instrument that are used in this manual.

Axis

A logical name for a motion device.

Controller

In this manual refers mostly to the MM4006 controller/driver.

Encoder

Displacement measuring device, term usually used for both linear and rotary models.

Function key

One of the four keys associated with the display; its function is determined by the current menu.

Home (position)

The unique point in space that can be accurately found by an axis, sometimes called origin.

Home search

A specific motion routine used to determine the home position.

Index pulse

A precision, encoder generated pulse, used in the home search algorithm.

Jog

Undetermined length motion initiated manually.

MM4006 controller

Refers to the MM4006 integrated controller/driver.

Motion device

An electro-mechanical motion device.

Move

Motion to a destination initiated manually or remotely.

Origin

Used sometimes instead of home.

Origin switch

A switch that determines an approximate point in space, used in the home search routine.

PID

Type of closed-loop control algorithm.

Remote

In this manual refers to the mode of operation where communication is performed over a computer interface link.

Stage

The most common type of motion device for the MM4006, used interchangeably in this manual for rotary and linear positioners.

1.3 General Description

The MM4006 is an advanced, stand-alone, integrated motion controller/driver. It can control and drive up to 4 axes of motion, in any stepper and DC motor combination. The MM4006 controller/driver (also referred to in this manual as “the controller”) was specifically designed to operate with Newport’s broad line of motion devices. This way, it significantly increases the user friendliness and raises the overall motion system performance. Using other motion devices is possible but not recommended for optimal system performance.

Fig. 1.1 shows a minimal system configuration. The MM4006 is used as a stand-alone unit to control and drive a motion device. The only components needed are a motion device, a connecting cable, the MM4006 and a power cord.

Fig. 1.1 — *Minimal system configuration.*

In this configuration all commands are received from the front panel. Programs can be generated and executed without using an additional computer.

A more common setup is shown in Fig. 1.2. The MM4006 drives multiple stages and is controlled by a remote computer.

Fig. 1.2 — *A common controller setup.*

A more complex configuration, shown in Fig. 1.3, could have up to 4 motion devices, digital and analog I/O signaling for motion Synchronisation, remote safety “motor off” switches and be part of a larger multi-axis system, controlled by a remote computer.

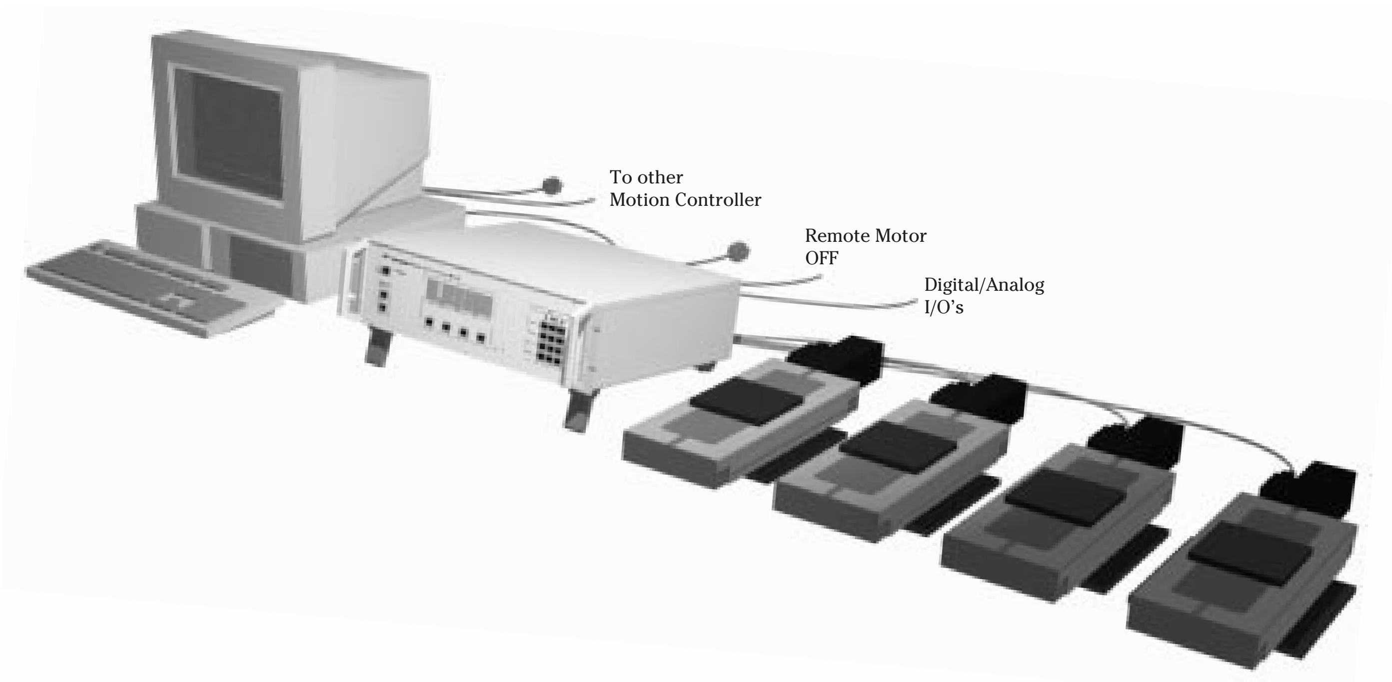


Fig. 1.3 — *A more complex controller configuration.*

To explore all capabilities of the MM4006 controller and identify the system configuration that best fits your application, you will have to read most of this manual or contact our applications support group for advice.

1.3.1 Features

Many advanced features make the MM4006 the preferred choice for precision applications:

- Integrated controller and driver design is more cost effective and a space saving solution.
- Compact, rack-mountable or bench-top enclosure.
- Allows any combination of motor types (stepper and DC) and sizes.
- Supports closed-loop operation of stepper motors.
- Feed-forward servo algorithm for smooth and precise motion.
- Velocity feedback motor drivers for best motion performance.
- Advanced multi-axis Synchronisation (linear interpolation).
- Powerful command set for the most demanding applications.
- Motion program storage and management capability.
- Advanced motion programming capabilities with up to 100 nested loops and complex digital and analog I/O functions.
- User-selectable displacement units.
- User-settable compensation for accuracy and backlash errors.
- Full-featured front panel with bright fluorescent backlit display, numeric/jog keypad, context-sensitive function keys, full motion selection and control capability and motion program creation and editing capability.
- Multilingual display capability — English or French.

1.3.2 Specifications

Function

- Integrated motion controller and driver.

Number of motion axes

- 1 to 8, can be used simultaneously or in any combination or order of stepper and DC motors.

Trajectory type

- Non-Synchronised motion.
- Multi-axis Synchronised motion (linear interpolation).
- S-Curve or Trapezoidal velocity profile for non-Synchronised and Synchronised motion. The default configuration is S-Curve velocity profile.
- 2D complex trajectory (XY linear and circular interpolation).

Motion device compatibility

- Entire family of motorised motion devices, using ether stepper or DC motors.

CPU type

- AMD5x86/133 Processor.

DC motor control

- 16 bit DAC resolution.
- 25 MHz maximum encoder input frequency.
- PID with velocity feed-forward servo loop.
- Digital servo cycle: 0.3 ms, up to 4 axes.
0.4 ms, from 5 to 8 axes.

Stepper motor control
- 1 MHz maximum pulse rate.
- Full-, half-, mini- and micro-step capability.
- Open or closed-loop operation.
- PID with velocity feed-forward closed-loop mode.
- Digital servo cycle: 0.3 ms, up to 4 axes.
  0.4 ms, from 5 to 8 axes.

Computer interfaces
- RS-232-C.
- IEEE-488.
- RS-485.

Utility interface
- 8-bit opto-coupled digital inputs.
- 8-bit open-collector digital outputs.
- 4 analog inputs, 12-bits resolution programmable input range (0-5 V, 0-10 V, ±5 V, ±10 V).
- Synchronisation pulse output on position or on time.
- Remote motor off input (interlock).

Operating modes
- Local mode: stand-alone operation, executing commands from the front panel.
- Remote mode: execution of commands received over one of the computer interfaces.
- Program execution mode: execution of a stored program, initiated either remotely or from the front panel, or execution of a program at power-on.
- Trajectory execution.

Programming
- Remotely via the computer interface.
- From the front panel.

Program memory
- 30 KB (30 760 bytes), non-volatile.

Display
- LCD with LED back light.
- 40 mm x 130 mm, 6 lines by 30 characters.
- Displays position, status, utility menus, program viewing and editing screens and setup screens.

Dimensions
- 5.28 (3U) H x 19 W x 15.6 D in. (134 x 483 x 395 mm).

Power requirements
- Power supply with PFC (Power Form Corrector) 90 to 264 V, 50/60 Hz.
- Motors off: 100 VA max.
- Motors on: 570 VA max.

Fuses
- AC line only.
  Line Voltage Fuse Type: T10A 250 VAC.

Operating conditions

- Temperature: 10 °C to 40 °C, non condensing.
- Humidity: < 85%.
- Altitude: < 1000 m.

Storage conditions

- Temperature: -15 °C to 45 °C.
- Humidity: < 85%.
- Altitude: < 1000 m.

Weight

- 18 lb. max. (8 kg max.).

1.3.3 Modes of Operation

LOCAL Mode

In LOCAL mode, the MM4006 is operated through the keys on the front panel. The display and function keys allow the selection of menus and operations that can be performed without using an external computer.

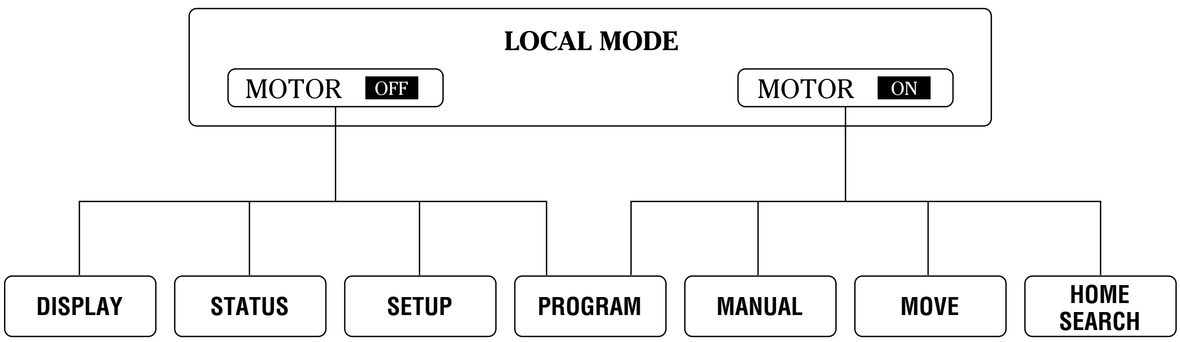


Fig. 1.4 — *Functions available in LOCAL mode.*

Operations that can be performed from the front panel depend on whether the power to the motors is turned on or off. A motion, for instance, cannot be performed when the motors are turned off and a general controller setup should not be done when the motors are on.

DISPLAY switches the display and front panel controls for the next or previous 4-axis.

STATUS valids or inhibits the display of signals for: home position, end-of-run and home done.

SETUP can be activated only from LOCAL mode, Motor Off. In this mode, the user can set up the general operation of the controller and the parameters specific to every motion axis and motion device.

The **PROGRAM** mode can be activated in LOCAL Mode while motors are on or off. In programming mode, a motion program can be created or modified.

MANUAL allows to move the axis in "+" or "–" direction, using the numeric keyboard.

MOVE is a general mode of operation in which an axis is commanded to move. The most complex motions result from executing a program. The other two cases are when a manual JOG or a point-to-point MOVE is executed.

HOME SEARCH is discussed separately because it is an important procedure that deserves special attention. In this mode, the controller is executing a home search algorithm on one or more axes. A home search cycle should not be interrupted. The controller will exit this mode automatically upon task completion.

The controller displays a set of hierarchical menus to navigate the various controller modes. It can be viewed as a "glue logic" between all the other modes.

REMOTE Mode

To operate in REMOTE mode, the controller must be connected through one of its interfaces (RS-232-C, RS-485 or IEEE-488) to a computer or terminal. In this mode, all commands are received remotely and the controller executes them as directed. The MM4006 command language consists of 129 commands which are described in chapter 3.

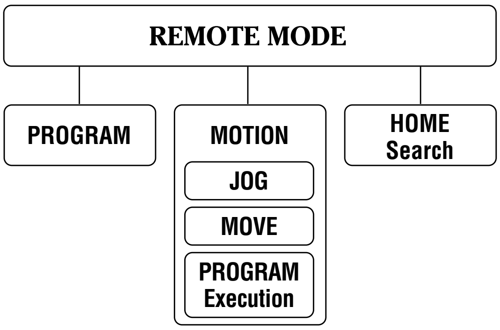


Fig. 1.5 — *The functions available in REMOTE mode.*

The functions available in REMOTE mode are similar to the ones in LOCAL mode. The main difference is that the MOTOR OFF / MOTOR ON cases are handled by the command interpreter so there is no need to distinguish between them. The controller will refuse to execute motion commands when the motors are turned off and will set the appropriate error flag.

Another difference between LOCAL and REMOTE is that the SETUP mode is not available remotely. Some SETUP parameters can be changed but the controller cannot be placed remotely into a setup mode.

PROGRAMMING mode is enabled and disabled by specific commands. All valid commands sent in this mode are not executed immediately but stored as part of a motion program.

MOTION is a general mode of operation in which an axis is commanded to move. The most complex motions result from program execution. Other types of motion include manual JOG and a point-to-point MOVE.

HOME Search mode has the same meaning and functionality as in LOCAL mode. A home search cycle should not be interrupted. The controller will exit this mode automatically on task completion.

Immediate Mode

This is not an operating mode in which the controller can be placed. Rather, the term merely differentiates the way the controller responds to remote commands. If a command is not being sent as part of a program, it is executed "immediately" in immediate mode.

Remote Commands In LOCAL Mode

The controller may be operated in LOCAL mode when connected to a remote computer. The LOCAL mode has many screen and menu combinations and most REMOTE commands are ignored when not received at the top level menu. For this reason, always keep in mind the following recommendations:

- In LOCAL mode, avoid sending REMOTE commands when not at the top menu level.
- When not at the LOCAL mode top menu level, restrict the use of remote commands to those that read information or stop motion.
- Do not send REMOTE commands when in LOCAL PROGRAMMING or SETUP modes.
- Do not send REMOTE commands when in an Intermediate Local mode (for instance when entering the value of a move).
- Do not interfere with a HOME Search cycle, including read commands.
- The preferred remote operation is the REMOTE mode, obtained by using the appropriate command.

1.3.4 Rear Panel Description

Before attempting to operate the MM4006, it must first be properly connected and configured. Carefully unpack the controller and place it on a flat surface. Save all packing materials.

Begin by familiarising yourself with the connectors and controls on the rear panel (Fig. 1.6).

NOTE

For complete connector description and pin-outs see Appendix B, Connector Pin-Outs.

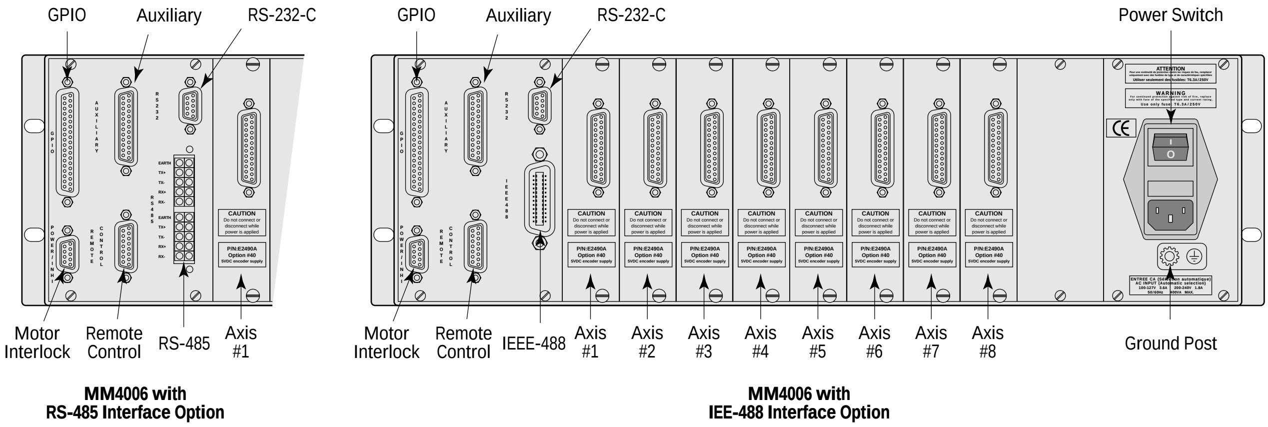


***Fig. 1.6** — Rear panel of the MM4006.*

Axis Modules

The MM4006 can accommodate up to eight motor driver cards. Each motor driver card has a 25-pin D-Sub connector, mounted on a small panel visible from the rear of the controller, for attaching the motion device. Uninstalled axes have a blank panel with no connector.

Each driver module has an identification label which clearly specifies the model and the type of motor it is configured to drive.

CAUTION

Carefully read the labels on the driver cards and make sure the specifications (motor type, voltage, current, etc.) match those for one the motion devices you intend to connect. Serious damage could occur if a stage is connected to the wrong driver card.

GPIO Connector

This 37-pin D-Sub connector is used for general purpose digital Input/Output signals. The MM4006 offers two separate 8-bit digital ports, one for input and one for output. A variety of commands are available for control and interface using these ports from within a motion program.

Power Inhibition Connector

This 9-pin D-Sub connector provides remote motor power interlock capability. One or more external switches can be wired to remotely inhibit the motor power in a way similar to the MOTOR OFF button on the front panel. The controller is shipped with a mating 9-pin connector installed that provides the necessary wiring to enable proper operation without an external switch.

Auxiliary Connector

This 25-pin D-Sub connector has two active lines. One is for motor power status indication and the other for frequency generator output. The frequency generator is controlled by the motion program and has a frequency range of 0.01 to 5000 Hz.

Remote Control Connector

This 15-pin D-Sub connector provides two functions. The first is similar to the Power Inhibition connector. The two active pins must be short-circuited for the motor power to be enabled.

The connector's second function is to provide inputs for the two analog ports. These ports are two independent 8 bit analog-to-digital converters. Programming commands allow the user to read and manipulate the information provided by these ports.

The controller is shipped with a mating 15-pin connector installed that provides the necessary wiring to enable the activation of motor power.

RS-232-C Connector

This 9-pin D-Sub connector provides an RS-232-C interface to a host computer or terminal. The port has a three-line configuration using a software (XON/XOFF) handshake. The pinout enables the use of an off-the-shelf, pin-to-pin cable. The port provides internally the necessary jumpers to bypass the hardware handshake, if needed.

IEEE-488 Connector

This is a standard 24-pin IEEE-488 connector.

RS-485 Connectors

Two identical RS-485 connectors are available. Both are connected in parallel, so you can make the connections on each. These connectors are provided to facilitate the construction of a network with several MM4006 controllers connected to one computer through a serial communication port.

Power Switch/Entry Module

The power entry module include a standard IEC 320 inlet combined with a line filter, fuse box and main power switch. The main power switch turns power on and off to the entire unit, including the stand-by circuit.

NOTE

The MM4006 senses the line voltage and automatically switches between 110 V and 220 V operation. The acceptable voltage ranges are 95 to 132 V or 195 to 263 V at 48 to 63 Hz.

While familiarising yourself with the rear panel and its components, leave the main power switch in the OFF position. Always make certain the power switch is in the OFF position before plugging in the power cord.

Ground Post

The ground post provides an additional chassis ground connection when needed. The MM4006 controller chassis can be externally grounded, in addition to or instead of the grounding supplied through the AC cord.

1.3.5 Blank Front Panel

A general view of the blank front panel is shown in figure below. There are three distinct areas, from left to right: power controls, a display and function keys, and a keypad.

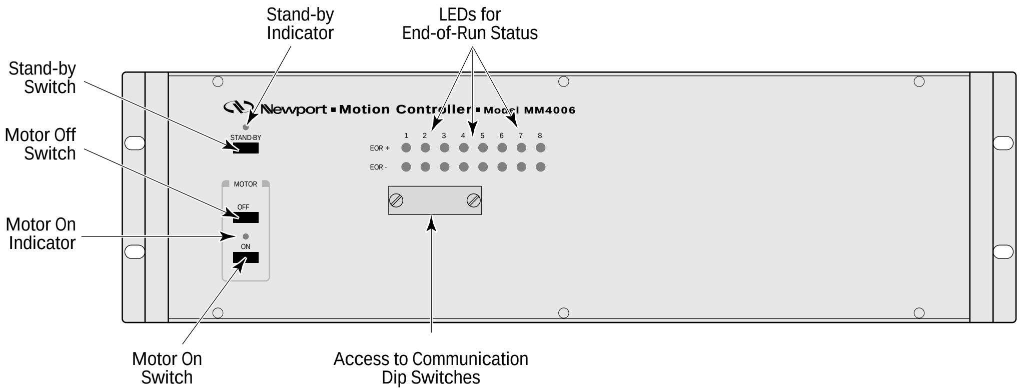


***Fig. 1.7A** — MM4006 blank front panel.*

Power STAND-BY

Use this button for your everyday controller power ON/OFF switching. Power is switched through a relay, not directly as it is through the main power switch on the rear panel. For this reason, a low power, low voltage (12 V) auxiliary power supply is always on when the main power switch in the back is ON.

To differentiate from the rear main power switch, this button is called “STAND-BY ”.

The power stand-by switch has a red LED indicator on top, indicates that the controller is powered OFF but the rear power switch is ON. This is the “STAND-BY” mode.

Motor ON/OFF

For convenience and safety reasons, the power to the motors can be controlled separately. This is done from the front panel through two buttons labelled MOTOR OFF and MOTOR ON. For easier identification, the MOTOR OFF button has a red bezel.

A green LED on top of the MOTOR ON provides a quick visual indication of the motor Power ON condition.

NOTE

The MOTOR OFF button is a normally closed switch wired in series with the two Motor Interlock switch connections on the rear panel. For the motors to turn on, the entire circuit must be closed.

End-of-Run Status

For each axis connected to the controller, 2 LEDs display the status of its ends-of-run (EOR+ & EOR-). Axes are numbered from 1 to 8, axis #1 is on the left.

The upper LEDs lign represents the positive end-of-run status (EOR+). The lower LEDs lign represents the negaitive end-of-run status (EOR-).

When a LED is lighting green, the axis is not in end-of-run, and its status is “inactive”. When a LED is lighting red, the end-of-run is reached and its status is “active”.

If both EOR+ and EOR- of an axis are lighting red in a same time, ends-of-run status are active in a same time. This is a fault condition. If this condition is detected at the controller power-up, the axis is considered as unconnected and is not active. All commands sent for this axis will be ignored by the MM4006 controller.

Access to Communication Dip Switches

This window gives an acces to twelve slide dip switches. Thes switches are use to configure the communication parameters of the MM4006 controller.

1.3.6 Equipped Front Panel

A general view of the equipped front panel is shown in figure below. There are three distinct areas, from left to right: power controls, a display and function keys, and a keypad.

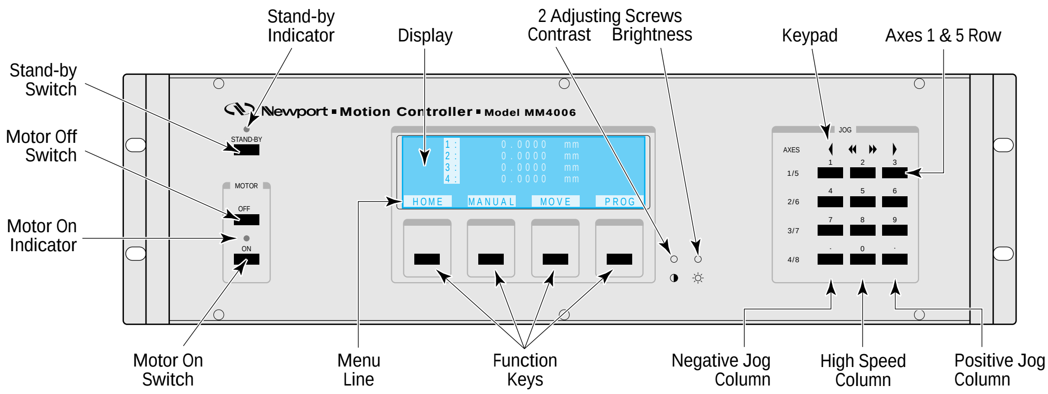


Fig. 1.7B — MM4006 equipped front panel.

Power STAND-BY

Use this button for your everyday controller power ON/OFF switching. Power is switched through a relay, not directly as it is through the main power switch on the rear panel. For this reason, a low power, low voltage (12 V) auxiliary power supply is always on when the main power switch in the back is ON.

To differentiate from the rear main power switch, this button is called “STAND-BY ▬”.

The power stand-by switch has a red LED indicator on top, indicates that the controller is powered OFF but the rear power switch is ON. This is the “STAND-BY” mode.

Motor ON/OFF

For convenience and safety reasons, the power to the motors can be controlled separately. This is done from the front panel through two buttons labelled MOTOR OFF and MOTOR ON. For easier identification, the MOTOR OFF button has a red bezel.

A green LED on top of the MOTOR ON provides a quick visual indication of the motor Power ON condition.

NOTE

The MOTOR OFF button is a normally closed switch wired in series with the two Motor Interlock switch connections on the rear panel. For the motors to turn on, the entire circuit must be closed.

Numeric Keypad

On the right hand side of the front panel there is a 12-button numeric keypad. Depending on the mode the controller is in, this keypad can be used for numerical data entry or controlling the manual JOG mode.

For details on using the keypad for jog control see Section 2, Local Mode.

Function Keys / Display

The central part of the front panel is occupied by a large display and four function keys. The display is a six-line back-lit LCD which shows both menu and status information.

Below the display are four function keys. Their context-sensitive functions are always given on the bottom line of the display window.

The contents of the display window are described in detail in Section 2.

1.3.7 Display Configuration

Display Organisation

The display has six lines with a maximum of 30 characters each. For better visibility, the characters are bright on a dark background. Information is highlighted using dark characters on a bright background.

On the right of the function keys, 2 screws permit to adust the contrast and the brightness of the display.

CAUTION

Saturation brightness reduces the display lifetime.

The bottom line of the display, line number 6, is reserved exclusively for defining the four function keys.

The next line up, line number 5, is primarily used to display messages, definitions or other helpful information. It generally displays information in reverse mode.

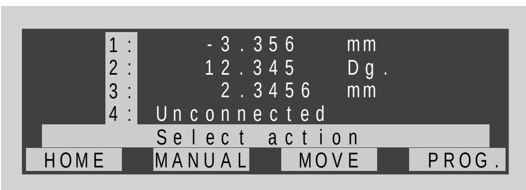


***Fig. 1.8** — Typical display contents.*

In the above example, line number 6 displays the current function of the function keys and line number 5 informs us that the controller is idle, waiting for the operator to select an action.

Lines number 1, 2, 3 and 4 identify the axis number and display the current position of each. Note that in the example, the controller detected that there is no motion device connected to axis number 4 and displays the message Unconnected.

When the controller is in some modes (**SETUP** , **PROG.** , etc.), the first four lines will display specific information while the fifth one will be reserved for helpful messages.

Menu Structure

A wide range of functions can be performed from the front panel. To fully explore its capabilities, carefully read Section 2, Local Mode, and experiment with the controller. This paragraph gives only a brief introductory description of the menu structure.

The bottom line on the display (line 6) is dedicated to the four function keys. An option description field will appear above each key if it has an active function in the current menu. Pressing a key will perform the selected command or will change the display to a new menu level. This capability to navigate between a number of menu levels to get to the desired command is the basis of the LOCAL mode operation.

Common Function Keys

Some of the function keys have the same definition in different menus. The following descriptions list the most common keys and their functions.

DISPL. Switches the display and front panel controls for the next or previous 4-axis.

QUIT Terminates the current operation and returns to the menu one level up. In most cases, any unsaved entries are ignored.

VALID Appears when an entry is required. It accepts the selected value and advances the display to the next menu.

MODIFY Activates a lower-level menu that enables the user to make changes to the currently displayed parameter.

UP Scrolls the display up through a list of parameters.

DOWN Scrolls the display down through a list of parameters.

DELETE Is used when a numeric entry is required. It deletes the last character entered. Note that for a value to be modified, it must first be activated and the ▶ symbol must precede it.

NEXT Scrolls the display through a number of choices in the same menu level.

Status Display

Pressing the STATUS function key activates the display to provide additional axis information. It does not change the menu level.

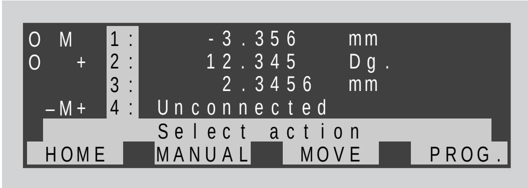


***Fig. 1.9** — Axis Status.*

To the left of the axis identifiers, as shown in Figure 1.9, there are four characters that can appear depending on the status of each axis:

O Will appear only if a Home Search routine has been performed successfully on that axis. It means that a mechanical origin has been found.

– Indicates that the negative direction (usually left) limit switch has been activated (tripped).

M Appears when the mechanical origin switch is in "high" state. As a stage moves from one end of travel to the other, you will see this indicator appear and disappear. This means that the stage has moved from one side of the switch to the other. The state of this indicator does not affect the normal operation of the motion device. For a complete description of the home search algorithm see Section 4, Motion Control Tutorial.

\+ Indicates that the positive direction (usually right) limit switch has been activated (tripped).

NOTE

If both - and + appear, the motion device is either disconnected or a hardware failure exists. On power-up sequence, the controller checks every axis for this case. If found, it assumes that no motion device is present. The axis is marked with Unconnected on the display and all commands for it will be ignored.

At the end of each axis information line an OK is displayed if no error has been detected. If a problem is detected on one of the axes, the message ERROR will appear.

1.3.8 Display Structure

This section describes the most common menus and display functions. Only local mode menus will be addressed since they represent the vast majority of the front panel operations.

As described in Section 1.3.3 and illustrated in Fig. 1.4, the local mode is divided into two sections: MOTOR power OFF and MOTOR power ON.

MOTOR OFF Menu

When motor power is turned off — the controller "power-on" default mode — the display function keys are as shown:

DISPL. STATUS PROG. SETUP

This is the "top level MOTOR OFF" menu.

Each function is defined as follows:

DISPL. Switches the display and front panel controls for the next or previous 4-axis.

STATUS Toggles the display for additional status information.

PROG. Activates the motion program management and generation environment. This mode can be activated from both MOTOR OFF and MOTOR ON top level menus. When selected, the function keys change to:

CREAT. MODIFY QUIT

The creation and modification of a program section is addressed extensively in the Programming In Local Mode section of the Local Mode chapter.

SETUP Is described in detail in Section 2.2, Controller Configuration. A brief introductory description is provided here.

The top level setup menu (after pressing SETUP) offers the choice of two different setup categories and looks similar to Fig. 1.10.

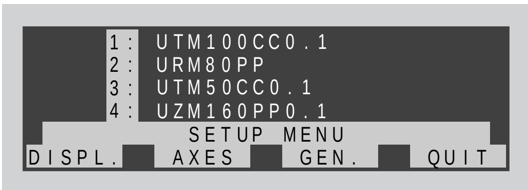


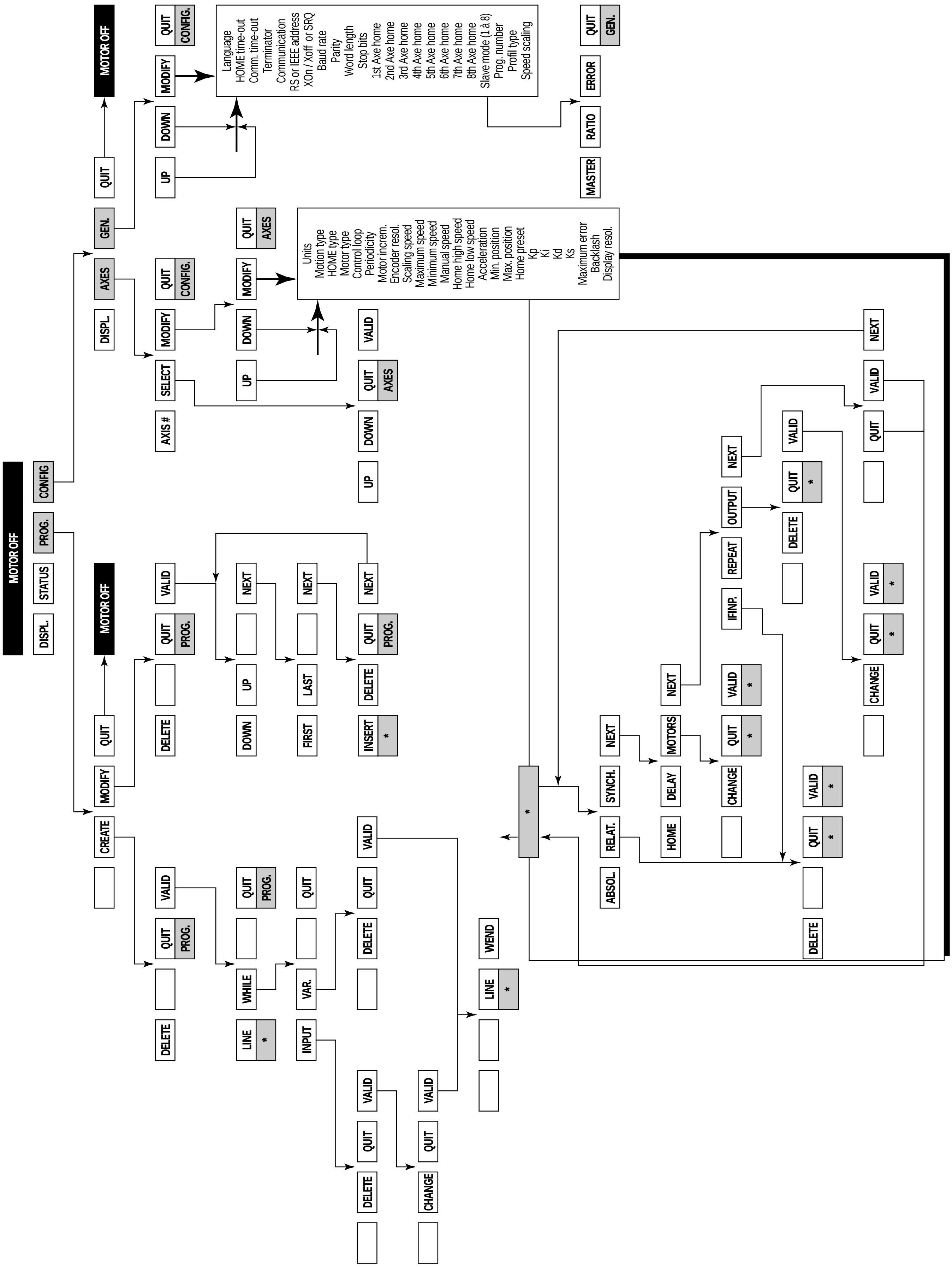
***Fig. 1.10** — Top level* SETUP *menu.*

Note the changes in the first four lines on the display. The axis positions have been replaced by the motion device types the controller thinks are connected to it. This is important because any attempt to first power on the controller should be preceded by a verification of the proper setup.

AXES Selecting AXES activates a menu to set up each motion device connected and its parameters.

GEN. By choosing GEN. you activate the General Setup mode in which the general controller parameters (language, communication ports, etc.) are defined

MOTOR OFF Menus Synoptic Table



MOTOR ON Menu

When motor power is turned on, the four function keys are defined as follows:

HOME MANUAL MOVE PROG.

This is the "top level MOTOR ON" menu. These four choices can be grouped into three important categories: home search, motion commands and program management.

DISPL. Switches the display and front panel controls for the next or previous 4-axis.

HOME Activates the home search setup menu in which one or more axes can be selected to perform a sequential home search cycle.

MANUAL Is a motion function which allows the user to manually jog each axis using the numeric keypad.

MOVE Is a motion function which activates a lower-level menu that offers position "zeroing", manual jog and go-to-position functions.

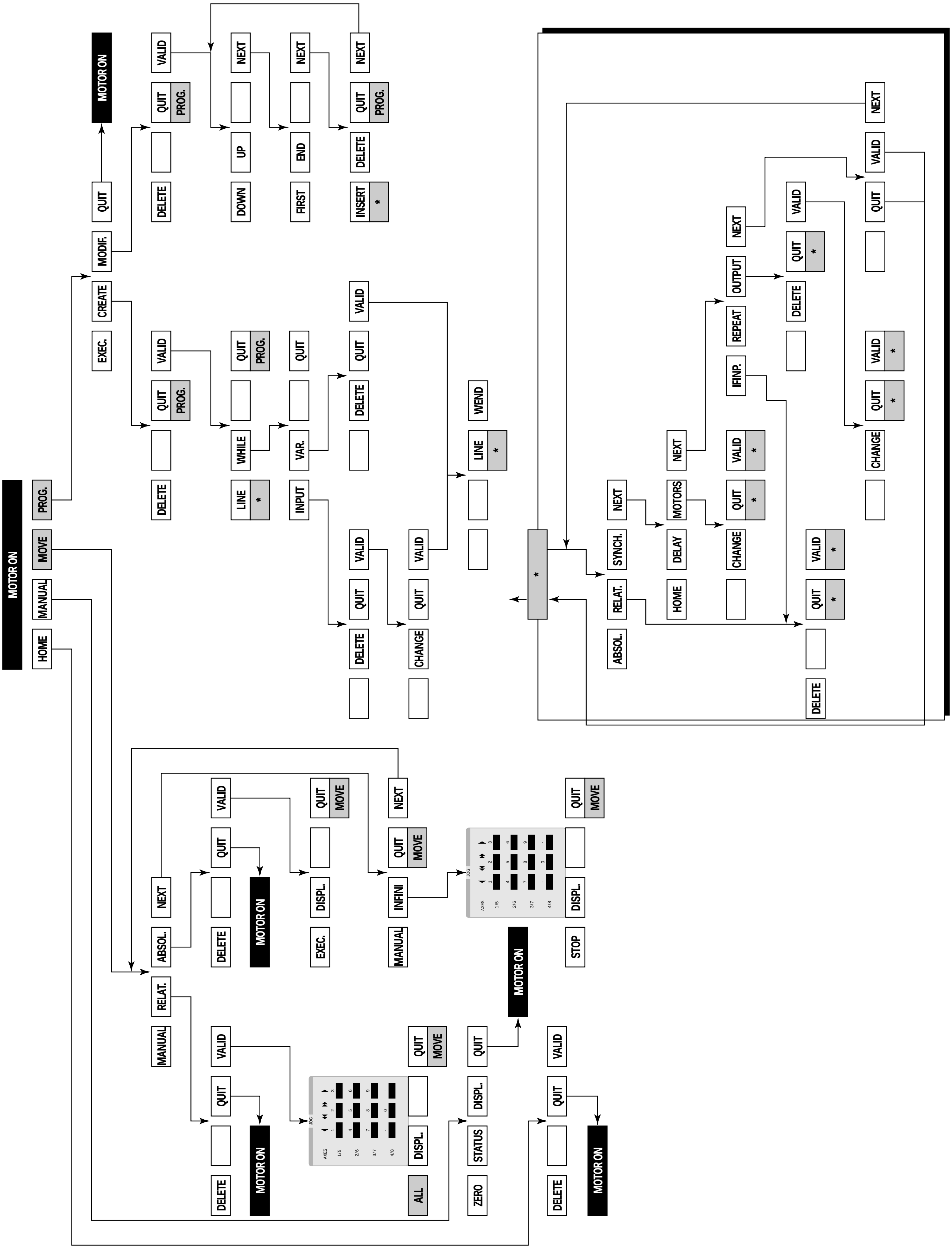
PROG. Activates the motion program management and generation environment. This mode can be activated from both MOTOR OFF and MOTOR ON top level menus. When selected, the next screen shows the following function choices:

CREAT. MODIFY QUIT

EXEC. Compared to the MOTOR OFF initiated menu, the MOTOR ON menu adds the EXEC. function which executes stored motion programs.

The creation and modification of programs is covered in detail in the Programming In Local Mode section of the Local Mode chapter.

MOTOR ON Menus Synoptic Table



Newport

1.4 System Setup

This section covers motion control system set up and preparing use it. First all necessary cables must be connected and the controller must be properly configured. This set up procedure configures a minimal system, similar to Fig. 1.11.

***Fig. 1.11** — A minimal control system.*

NOTE

If you have not already done so, carefully unpack and visually inspect the controllers and stages. Please save the packing material, in case you have to ship the controller in the future.

Place all components on a flat and clear work surface. Check visually for any sign of damage and if found, report immediately to the carrier.

NOTE

The front two "legs" of the chassis have a tab that, if rotated 90° forward, place the controller in a slightly angled position. To return the controller to horizontal position, lift the front side, pull on the tabs and return them to the original position.

CAUTION

No cables should be connected yet to the controller.

1.4.1 Connecting Motion Devices

If you purchased a standard motion control system, you should have received all necessary hardware to set it up.

First connect the motion device (stage) interface cables. These are 10-ft-long (3-m) cables with 25-pin to 25-pin D-Sub connectors. Insert them gently as you would do with any computer cable, both into the stage and the appropriate driver card and secure them with the locking thumb-screws.

CAUTION

Carefully read the labels on the driver cards to be sure the specifications (motor type, voltage, current, etc.) match those for the motion devices you are connecting. Serious damage could occur if a stage is connected to the wrong driver card.

1.4.2 First Power On

Once all stages have been properly connected, you are ready to proceed with the power connection.

CAUTION

Make sure the main power switch on the power entry module is turned off before connecting the controller to the AC line.

Verify that the main power switch on the rear panel and the stand-by power switch on the front panel are turned off.

Plug the AC line cord in the power entry module on the rear panel.

Plug the AC line cord in the AC outlet.

NOTE

At this point, no lights should appear on the front panel.

Turn the main power switch on the rear panel on.

The red LED indicator on the front panel marked STAND-BY should come on and stay on. At this point, the low power stand-by power supply is energised.

Finally, press the red STAND-BY button once to turn the controller on.

The red LED goes off and the green one comes on, the front panel display turns dark blue and the controller makes a slight ticking sound. This is normal.

After a short delay, a welcome screen with the Newport logo flashes for a few seconds, showing you the firmware version in use.

NOTE

Any time you call for technical support, the firmware version is one piece of information you need to supply. It is displayed every time the controller power is turned on.

Now, the display shows the main power off menu, similar to Fig. 1.12.

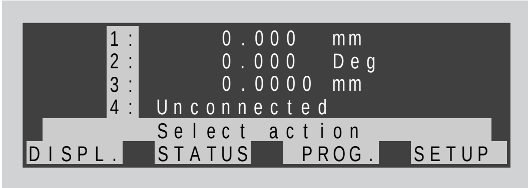


Fig. 1.12 — Display after initial power up.

NOTE

If, instead of a screen similar to Fig. 1.12 you see a different message, this means that the controller has detected an error.

See Appendix A, Error Messages and Appendix E, Troubleshooting Guide.

NOTE

If the display looks like Fig. 1.16 but in a wrong language, follow these steps:

1 **Assume the following labelling convention for the function keys:**

1 2 3 4

2 **From the top level MOTOR OFF menu, press the function keys in this sequence:**

4 → 3 → 3

3 **Press function key 2 until the desired language appears.**

4 **Press function key 4 three times to return to the top level menu.**

1.4.3 Verifying Default Devices

Before powering the motors, verify that the controller is configured for the actual motion devices it is supposed to drive.

From the main motor off menu, press the **SETUP** function key.

The top-level setup menu will indicate on the first four lines the type of motion device each axis is configured for. The display should look similar to Fig. 1.13. Depending on your system configuration, different models will be listed.

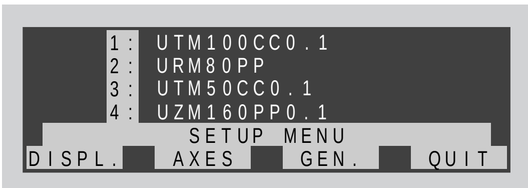


Fig. 1.13 — Typical display slowing connected devices.

If the components listed match with the actual motion devices installed, you are ready for the first motion test.

On the other hand, if there is a discrepancy, it must be corrected immediately. In this case, you should perform the following steps:

1 From the main MOTOR OFF menu, select the SETUP key.

2 In the main setup menu press AXES function key.

3 From the next menu press AXIS # function key. This will let you select which axis you want to modify. (Note the ▶ symbol on the first line, in front of the existing axis number.) Using the numerical keypad, enter the axis number to be corrected and then press VALID key to accept the selection and return to the previous screen.

NOTE

Remember that any time a numerical entry on the keypad must be corrected, the DELETE function key erases the last digit entered.

---

4 Now press the YES key. This enters the product family selection screen.

5 Use the UP or DOWN keys scroll through the product families until you find the one you need.

6 Press the VALID key to accept the product family currently on the display. The next menu level consists of product models is the chosen product family.

7 Use the UP or DOWN keys to scroll through the different product models of the chosen family.

8 Press the VALID key to accept the product model on the display and to advance to the next menu.

9 The next two screens are for changing the default axis parameters, but do not attempt to do at this point. Press the VALID key to pass through these screens without making any modifications.

10 When the display returns to a screen similar to Fig. 1.14, observe the axis specified on the first line and the component on line two. They should correspond to the selections you made and to the motion device used on that axis.

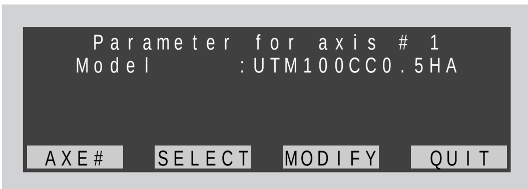


***Fig. 1.14** — Axis/Device Assignment.*

11 If you need to modify another axis, repeat all steps starting with number 3.

12 When all necessary modifications are completed, from the screen shown in Fig. 1.14, press the QUIT key. If modifications to any axis have been made, the next screen will ask if you want to save the changes (Fig. 1.15).

***Fig. 1.15** — Save screen for axis modifications.*

13 Press the YES key to save the changes and return to the main setup menu.

NOTE

If no changes have been made, the screen in Fig. 1.15 will not appear.

---

14 Press QUIT to return to the main MOTOR OFF menu.

Now, with all axes configured for the proper motion devices, we are ready to use the motion devices.

Section 1
Introduction

Table of Contents
Section 2 — Mode local

Section 2
Local Mode

2.1 Quick Start

After reading the Introduction Section you are now prepared to turn the motors on and command the stages to execute motions. The following paragraphs will guide you through a quick tour of the LOCAL mode motion commands.

CAUTION

You should at least read the System Setup Section of the Introduction before attempting to turn on the controller or the motors. Serious damage could occur if the system is not properly configured.

2.1.1 Motor On

After first turning the controller on as described in the previous section, you are ready to turn the motors on.

Be sure that the motion devices are placed on a flat surface and that their full travel will not be obstructed.

CAUTION

Be prepared to quickly turn the motor power off if you observe any abnormal operation.

Press the MOTOR ON button on the front panel. You may hear a small relay click inside the controller as the green LED indicator on the button lights. If no errors are detected, the green LED will stay lit. The display switches to the top level motor-on menu:

HOME MANUAL MOVE PROG.

The motion system is ready for a command.

2.1.2 Home Motion Devices

As a general practice, before executing any motion, always home the motion devices. As described in detail in the Motion Control Tutorial section, homing a motion device means executing a special routine that locates a dedicated origin switch and an encoder index pulse and establishes an absolute position zero.

Finding the home position of a motion device is important for two reasons. First, after each power off/on cycle, you must position the stage accurately in space. This means that the controller must find a zero position that is always in the same point in space, relative to the base. Secondly, in order to prevent the motion device from running into the limits and possibly causing damage, the controller must determine its position on power-up.

From the top level motor-on menu press the **HOME** function key. The display will ask you to select an axis for the execution of a home search routine. Use the keypad to enter a number and then press **VALID**. If you want to perform a home search on all axes, leave the default 0 and press **VALID**. The axis will start moving, the function keys will be disabled and the display will indicate the progress of the routine.

When all selected axes complete the home search cycle, the display returns to the top level motor-on menu.

The stages are ready for a move.

2.1.3 First Jog

From the top level MOTOR ON menu press the **MANUAL** function key.

The display switches to the manual jog screen and menu. As the message on line number 5 instructs you to do, use the keypad to jog any installed motion device.

Since this is the first time you are using the keypad for jogging, some clarifications are needed. The keypad is a 3-column by 4-row matrix (Fig. 2.1).

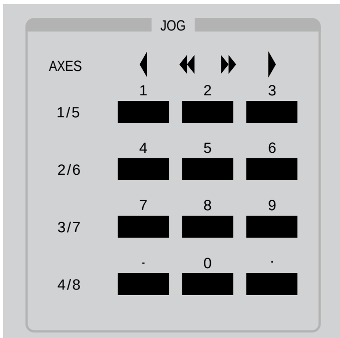


***Fig. 2.1** — Using the numeric keypad to JOG.*

Concentrate first on the column convention. For better identification, we can label them ❶, ❷ and ❸ (Fig. 2.2).

Newport

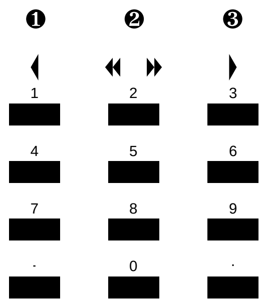


Fig. 2.2 — The columns of the numeric keypad indicate the direction and speed of a JOG.

If a key in column ❶ is pressed, the selected axes will move slowly in the negative direction.

If a key in column ❸ is pressed, the selected axes will move slowly in the positive direction.

Column ❷ is used for high-speed jogging. If a key in column ❷ is pressed simultaneously with one in column ❶ or ❸, the axis will jog at high speed in the selected direction.

Now, lets take a look at the keypad row convention (Fig. 2.3).

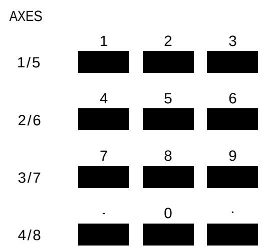


Fig. 2.3 — The rows of the numeric keypad indicate the axis that will JOG.

Each key row controls one axis, as indicated by the label on the left.

To summarise, if you want to jog fast axis number 2 or 6 in the positive direction, press simultaneously the keys numbered 5 and 6.

NOTE

The DISPL. key switches the display and front panel controls for the next or previous 4-axis.

To exit the manual jog mode press the QUIT key.

2.1.4 First Move

From the top level MOTOR ON menu press the MOVE function key. The display will offer you the following menu:

MANUAL RELAT. ABSOL. NEXT

You can start by pressing the ABSOL. key to command an absolute motion. The next screen will look similar on the top four lines, with the exception of the ▶ symbol in front of the first active axis position. As you recall, this means that a numerical input is required for the marked value.

Use the keypad to enter a desired destination (absolute position). To correct the entry use the DELETE key. When finished, press the VALID key to accept the value.

If there is more than one axis connected to the system, the ► symbol now jumps to the next axis position displayed. Repeat the desired destination entry procedure or press VALID until all connected axes have been confirmed and the following menu appears:

EXEC.			QUIT

Press the EXEC. key and observe the motion devices. They will rapidly move to the requested destinations and when motion is complete, the display will return to the motion selection menu:

MANUAL	RELAT.	ABSOL.	QUIT

To execute a relative motion, select the RELAT. function key. The same position entry screen appears as for the absolute motion, with the exception that all position values are zero, rather than the current absolute position. This is because motion is made relative to the current position instead of absolute home.

Enter desired relative motion values as described earlier. When complete, the display changes to the following menu:

ALL			QUIT

Pressing the ALL key will start the relative motion on all axes. The difference from the absolute motion is that, when the relative motion on all axes is finished, the display returns to the same menu. This means that you can repeat the relative motion again and again by pressing the ALL key.

If you entered relative motion values on multiple axes, but only need to move one, use the same JOG keypad convention and press a key from column ❶ or ❸ that corresponds to the axis you want to move.

One special note about the keypad in this mode. If you enter a negative value for a relative move and you press a key in column ❸, the move will be in the negative direction. If a key in column ❶ is pressed, the move will be in the positive direction. In other words, pressing a key in column ❶ will initiate a relative move in the opposite direction than requested.

To exit the relative move mode press the QUIT key.

To exit the move mode and return to the top level motor-on menu, press the QUIT key again.

Now that you know how to JOG and MOVE motion devices, experiment with front-panel-initiated motions to become familiar with the controller and the local motion modes.

NOTE

Remember that only motions with destinations inside the software travel limits are allowed. Any entry outside these limits will be ignored.

2.2 Controller Configuration

Now that you have had the chance to perform some basic motion commands in LOCAL mode more details on the controller's operation are in order. The first aspect is the controller configuration.

Though some parameters can also be changed with remote commands, the primary environment for configuring the MM4006 controller is the SETUP section of the LOCAL mode.

The SETUP mode can only be initiated from the top level MOTOR OFF menu:

DISPL. STATUS PROG. SETUP

Pressing the SETUP function key will enter the setup mode and display the main setup screen and menu (Fig. 2.4).

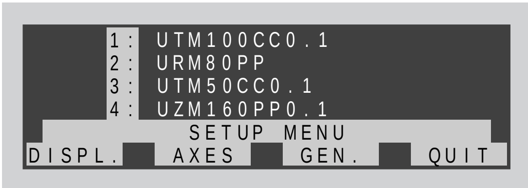


Fig. 2.4 *— Typical main setup menu.*

There are two types of setup functions: Axis Setup and General Setup. The following sections will describe both in detail.

2.2.1 General Setup

General Setup is entered by pressing the GEN. function key in the top level setup menu. The display will change to the one illustrated in Fig. 2.5.

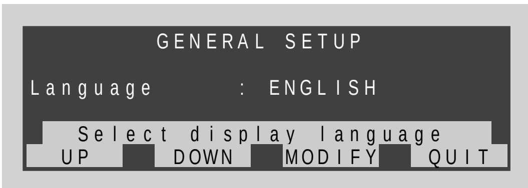


Fig. 2.5 — GEN. *setup menu.*

The top display line (number 1) indicates the setup mode (or level). Line number 3 displays one parameter at a time and its current value. Line number 5 displays an operator prompt.

The function key definition line (number 6) displays a menu that is common for most setup screens at this level. The first two functions, UP and DOWN, perform scrolling through a list of parameters.

Pressing the MODIFY key enters a lower level menu that allows modification of the currently displayed parameter. If the value needed is provided from a short list, the new menu looks like this:

[ ] CHANGE QUIT VALID

The CHANGE key scrolls through the list, QUIT exits this level without recording any modification and VALID also returns to the previous screen (level) but the displayed value is stored as the new entry for the selected parameter.

If a parameter requires a numerical value, the menu level that allows modifications will have the following choices:

DELETE [ ] QUIT VALID

The numerical value displayed will have a ▶ sign in front, indicating that a numerical entry from the keypad is expected. For simple editing, pressing the DELETE key erases the last digit of the numerical entry. The VALID key accepts the value for the selected parameter and returns to the previous menu. The QUIT key returns to the previous menu without keeping any modifications.

NOTE

This manual contains detailed descriptions, mostly with the first time reader in mind. To help the more experienced user looking for a quick memory refresher, each operation description is also accompanied by a quick front panel key sequence and, if appropriate, the remote command that accomplishes the same function.

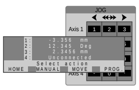
Language Selection

The first parameter displayed from the General Setup list is the display language. This sets the language the controller uses to communicate with the operator, especially through the front panel.

The MM4006 can use two languages at this time: English and French. To change the language, from the display shown in Fig. 2.5 press the MODIFY key. The new menu displayed is

[ ] CHANGE QUIT VALID

Press the CHANGE key until the desired language is displayed. Press the VALID key to accept the selection and return to the previous menu. The display will now use the new selected language.



SETUP → GEN. → MODIFY → CHANGE → VALID → QUIT → QUIT

Emergency Language Reset

In case the controller has been set to operate in a language you do not understand, use the following procedure to reset the controller:

1 Assume the following labelling convention for the function keys:

1   2   3   4

2 From the top level motor-off menu (power-on default screen), press the function keys in this sequence:

4 → 3 → 3

3 Press function key 2 until the desired language appears.

4 Press function key 4 three times to return to the top level menu.

Command Language Set

The second parameter in the General Setup menu (selected by pressing the UP key once) is the Command Language Set, labelled as Controller. This is the only parameter that you must not change. It selects the command set the controller will respond to. The selection exists only to assure compatibility with future controller models.

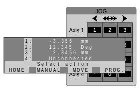
Always leave the setting on STANDARD. Press the UP key to advance to the next parameter.

Speed Scaling

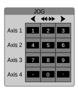
The Speed Scaling parameter offers a feature not present in many high-end controllers. It allows an user to execute a motion program at a reduced speed to more easily observe its operation. This feature is a great help in troubleshooting complex programs.

To change speed scaling from the General Setup menu, press the UP key until the speed scaling parameter is selected in the display. To change the value, press the MODIFY key. The display will prompt you to change the

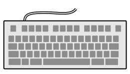
existing value by preceding it with the ► sign. Use the keypad to enter the desired% value. The maximum value is 100%, meaning that the controller will run with the actual programmed velocities. To reduce the program execution speed to half of the programmed value, enter 50. This means that all velocities in a program will be reduced to 50%.



Motor OFF → SETUP → GEN. → UP → MODIFY →



→ VALID → QUIT → QUIT



SD — Set scaling speed.

NOTE

In the following paragraphs it is assumed that you need to modify only the mentioned parameter. For that reason, the key sequence description starts from the General Setup menu and the quick key sequence starts and stops at the top level MOTOR OFF menu.

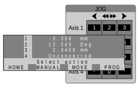
If you need to modify more than one parameter, advance through the list with the UP key, without returning each time to the top menu.

Communication Time-out

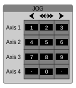
This parameter represents the time duration the controller will wait when expecting an input. It is a general communication parameter that applies to RS-232-C, RS-485 and IEEE-488 interfaces.

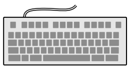
To change the existing value, from the General Setup menu, press UP until the parameter appears on the display. Press MODIFY and then enter the desired value on the keypad. Press the VALID key to accept the entry and return to the previous menu.

The default value is 1 seconds.



Motor OFF → SETUP → GEN. → UP → MODIFY →

 → VALID → QUIT → QUIT



CMOxx — Set communication setup.

HOME Time-out

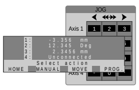
This parameter sets the time duration for which the controller will wait for each axis to complete a home search routine. Since the controller cannot be interrupted during a home search, this parameter provides a time-out in case of malfunction.

Use this parameter with discretion. A small value will cause the controller to falsely report an error when the stage starts a home search cycle from an extreme distance and does not have enough time to complete it. A large value prevents the controller from detecting a real problem, when the home search cycle takes an excessive amount of time.

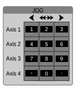
An ideal home time-out value is about 20% over the time it takes the slowest stage installed to perform a home search. The longest time is usually when the stage starts from the farthest point away from the origin switch.

To change the existing value, from the General Setup menu, press UP until the parameter appears on the display. Press MODIFY and then enter the desired value on the keypad. Press the VALID key to accept the entry and return to the previous menu.

The default value is 90 seconds.



Motor OFF → SETUP → GEN. → UP → MODIFY →

 → VALID → QUIT → QUIT

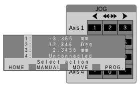
Terminator

This parameter defines the terminator to be used in communication with a host computer or a terminal. As described in the Remote Mode section, the controller responds to command lines, not characters. In order for a command to be acted upon, it must be followed by the terminator.

The MM4006 controller offers a choice of four command line terminators which are combinations of line-feeds and carriage-returns: LF, CR, CR/LF and LF/CR.

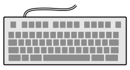
To change the existing selection from the General Setup menu, press **UP** until the Terminator parameter appears on the display. Press **MODIFY** and then the **CHANGE** key until the desired terminator is selected. Press the **VALID** key to accept the entry and return to the previous menu.

The factory default terminator is LF.



Motor OFF → **SETUP** → **GEN.** → **UP** → **MODIFY** →

CHANGE → **VALID** → **QUIT** → **QUIT**

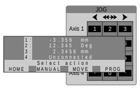


CMTxx — Set communication terminator.

Communication

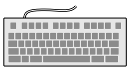
This setting selects the communication port to be used with a host computer. The MM4006 controller can use one of RS-232-C, RS-485 or IEEE-488 interfaces, but only one at a time. The selection can be made only through this setup. Default mode is RS-232-C.

To change the existing selection from the General Setup menu, press **UP** until the Communication parameter appears on the display. Press **MODIFY** and then the **CHANGE** key to select a new communication port. Press the **VALID** key to accept the entry and return to the previous menu.



Motor OFF → **SETUP** → **GEN.** → **UP** → **MODIFY** →

CHANGE → **VALID** → **QUIT** → **QUIT**

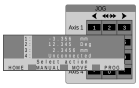


CMMxx — Set communication mode.

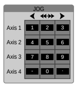
IEEE-488 or RS-485 Address

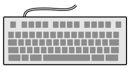
The IEEE-488 standard requires each connected instrument (device) to have an address. The RS-485 used in Multi-Drop mode requires also an adress setting. If not used, ignore the selection by pressing the UP key and advancing to the next parameter. Default address is 2.

To change the existing address from the General Setup menu, press UP until the IEEE Address appears on the display. Press MODIFY and then enter the desired address on the keypad. Press the VALID key to accept the entry and return to the previous menu.



Motor OFF → SETUP → GEN. → UP → MODIFY →

 → VALID → QUIT → QUIT

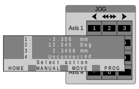


CMAxx — Set communication address.

IEEE-488 SRQ Used

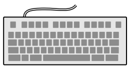
The SRQ line is an IEEE-488 handshake that ensures proper transmission of large files (trace data, large programs, etc.) Default is NO.

To enable or disable the use of the SRQ from the General Setup menu, press UP until the IEEE SRQ Used appears on the display. Press MODIFY and then the CHANGE key to select a new setting. Press the VALID key to accept the entry and return to the previous menu.



Motor OFF → SETUP → GEN. → UP → MODIFY →

CHANGE → VALID → QUIT → QUIT



CMQxx — Set IEEE communication SRQ mode.

Baud Rate

This parameter applies to RS-232-C and RS-485 interfaces. It sets the communication speed to be used on this port. The valid range is from 1200 to 115200. The factory default is 9600 baud rate.

If the IEEE interface is used, ignore the selection by pressing the UP key and advancing to the next parameter.

To set the baud rate from the General Setup menu, press UP until the Baud Rate parameter appears on the display. Press MODIFY and then the CHANGE key to select a new value. Press the VALID key to accept the entry and return to the previous menu.

Motor OFF → SETUP → GEN. → UP → MODIFY →

CHANGE → VALID → QUIT → QUIT

CMBxx — Set serial communication baud rate.

XON/XOFF Mode

XON/XOFF mode synchronises work between the transmitter and the receiver. In this mode, XON and XOFF characters are automatically generated (XOFF if the receipt buffer is almost full and risks to be erased, XON if the receipt buffer is sufficiently emptied to receive new characters).

To change the setting, from the General Setup menu, press UP until the XON/XOFF Mode appears on the display. Press MODIFY and then the CHANGE key to select YES. Press the VALID key to accept the entry and return to the previous menu.

The factory default is NO.

Motor OFF → SETUP → GEN. → UP → MODIFY →

CHANGE → VALID → QUIT → QUIT

CMXxx — Set serial communication XON/XOFF mode.

Parity

Parity must be set correctly for the RS-232-C or RS-485 communication to work properly. As the standard suggests, when a word length of less than 8 is used, the parity bit can be set to Odd or Even. Both communicating devices must use the same setting.

The possible settings are Odd, Even and None. The factory default is None.

To change the setting, from the General Setup menu, press UP until the Parity parameter appears on the display. Press MODIFY and then the CHANGE key to select a new parity. Press the VALID key to accept the entry and return to the previous menu.

Motor OFF → SETUP → GEN. → UP → MODIFY →

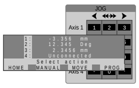
CHANGE → VALID → QUIT → QUIT

CMPxx — Set serial communication parity.

Word Length

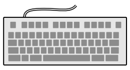
The word length refers to the word size to be used with the RS-232-C and RS-485 interfaces. The MM4006 controller is designed to accept either 7- or 8-bit words. The factory preset word size is 8 bits.

To change the RS-232-C word length from the General Setup menu, press UP until the Word Length parameter appears on the display. Press MODIFY and then the CHANGE key to select a new value. Press the VALID key to accept the entry and return to the previous menu.



Motor OFF → SETUP → GEN. → UP → MODIFY →

CHANGE → VALID → QUIT → QUIT



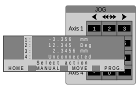
CMLxx — Set serial communication data length.

Stop Bits

The stop bits must also be set to the same value on both the controller and remote computer for RS-232-C or RS-485 communication.

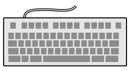
The possible options are 1 and 2 bits. The factory default is set at 1 bit.

To change the RS-232-C stop bits setting from the General Setup menu press UP until the Stop Bits parameter appears on the display. Press MODIFY and then the CHANGE key to select a new value. Press the VALID key to accept the entry and return to the previous menu.



Motor OFF → SETUP → GEN. → UP → MODIFY →

CHANGE → VALID → QUIT → QUIT



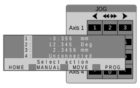
CMSxx — Set serial communication Stop Bit number.

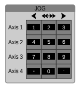
Axis HOME Sequence

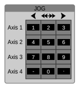
The Axis Home sequence performs a home search cycle on multiple axes. When a home search routine is invoked, you may execute it on one or all axes. If you choose to perform a home search on all axes, you may specify the order in which axis home sequences are executed. For example, the controller may be configured to first perform a home search on axis number 3, then on axis number 2, 4 and 1.

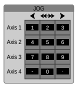
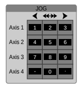
To change the Axis Home sequence you must set the priority of each axis. If, for instance, you want axis number 3 to execute first, you must set its priority (order) to 1.

To change the order of the home search, from the General Setup menu press UP until the 1st Axis HOME appears on the display. Press MODIFY and then enter the desired priority (order) number on the keypad. Press the VALID key to accept the entry and return to the previous menu. Press UP again to display 2nd Axis HOME. Press MODIFY and then enter the desired order (priority) number on the keypad. Press the VALID key to accept the entry and return to the previous display.



Motor OFF → SETUP → GEN. → UP → MODIFY →  →

VALID → UP → MODIFY →  → VALID → UP →

MODIFY →  → VALID → UP → MODIFY →  → VALID →

QUIT → QUIT

If you followed the general setup procedure up to this point, pressing the UP key will bring you back to the Language parameter, the first one covered at the beginning of this section. Press QUIT to return to the top level SETUP menu.

To exit the setup menu press the QUIT key again.

Master-Slave Mode Definition

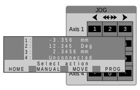
Master-Slave mode defined the relation between the master and slave axis in the master hierarchy system. The slave axis number is displayed with small character inverted video.

For axis # xx:

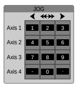
- Master axis nn = 0: axis # xx is independent
- Master axis nn = between 1 and 8 (≠ xx): axis #xx is independent.

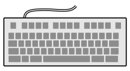
Press the UP key until the Master-Slave mode appears and then press the UP key to select the Slave axis. To change the Master axis, press the MODIFY key and then the MASTER key to modify the axis number with the numeric keypad. To accept the new Master axis number (must be different from the Slave axis number), press the VALID key.

The factory default is NO.



Motor OFF → SETUP → GEN. → UP → MODIFY →

MASTER →  → VALID → QUIT → QUIT



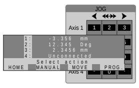
SS — Set Master-Slave mode.

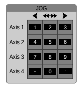
Program Automatical Execution on Power On

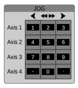
No program will be executed on power on.

Press the UP key until the program number appears. Press MODIFY and then select the program number with the numeric keypad. To accept this program number, press the VALID key. To select the number of time to execute this program, enter a value with the numeric keypad and then press the VALID keypad to confirm this entry.

The program number default is 0.



Motor OFF → SETUP → GEN. → UP → MODIFY →  →

VALID →  → VALID → CHANGE → VALID → QUIT → QUIT

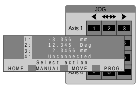
Profile Type

This menu defined the type of velocity profile:

TRAPEZOID or S-CURVE

The S-curve type avoids brutal changes of the speed in the course of axis displacement, consequently, it improves the stage quality of movement.

Press the UP key until the profile type appears. Press MODIFY and then the CHANGE key to select S-CURVE or TRAPEZOID. To accept this entry, press the VALID key.



Motor OFF → SETUP → GEN. → UP → MODIFY →

CHANGE → VALID → QUIT → QUIT

2.2.2 Axis Setup

The Axis Setup is entered by pressing the AXES function key in the top level setup menu. When activated, the display changes to one similar to Fig. 2.6.

***Fig. 2.6** — Axis setup menu.*

This menu is the top level Axis Setup menu, labelled Axis Setup in the quick key sequence listing.

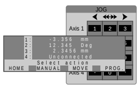
The first line displays the axis number to be reviewed. The second shows the motion device (stage) connected to the selected axis. Your display will show the motion device connected to axis number 1.

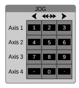
Newport

On the menu line, all four function keys are active. QUIT exits this level and returns to the previous menu. The other functions are discussed in the following paragraphs.

Axis Number Selection

As mentioned earlier, line number 1 of the top level Axis Setup menu displays the axis number. The default is number 1. To select a different axis to display (and change) press the AXIS # key. In front of the axis number, at the end of the first line, the ▶ symbol appears, indicating that the controller is expecting a numerical entry from the front panel's numerical keypad. Enter the desired axis number and then press the VALID key. If you need to correct the entry, use the DELETE key to erase the number or, if you changed your mind, exit by pressing the QUIT key.



Axis Setup → AXIS # →  → VALID

Motion Device Selection

One of the advanced features of the MM4006 controller is that it has stored in its firmware all necessary parameters for all compatible motion devices supplied by Newport.

To avoid scrolling through over 100 components, the selection is made in two steps, first the family and then the component model.

From the top-level Axes Setup menu press the SELECT key. This activates a screen to change the Product Family. A family represents a group of motion devices with the same prefix. Use the UP or DOWN keys to scroll the family list. When the desired product family is displayed, press the VALID key to accept the entry and advance to the next menu, the model selection.

NOTE

Notice that the first product family displayed is DEFAULT. This is to allows the user to define the parameters of a custom device or one that is not manufactured by Newport.

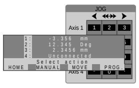
Once a family is selected, the controller prompts you to pick a product model from the selected family. Use the UP or DOWN keys to scroll the model list. When the desired product model is displayed, press the VALID key to accept the entry and advance to the next menu.

Pressing the QUIT function key will return the display to the top level Axes Setup menu.

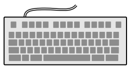
NOTE

Part of the component selection procedure is setting the PID parameters. It is strongly recommended that these parameters should not be changed from this menu. They are displayed here only for factory setup purposes.

After a component model has been selected, the next menu displays the Kp parameter. Do not make any changes to the value; press VALID. Do the same for the following Ki and Kd parameters. This will return the display to the top level Axes Setup menu.



Axis Setup → SELECT → UP → VALID → UP → VALID → VALID → VALID



TA — Read motion device.

SF — Set motion device.

Modifying Axis Parameters

Once a new motion component has been defined for an axis, you can review its default parameters. The following discussion assumes that you want to see them all and will not exit after each one is displayed.

NOTE

If you just want to change one parameter, you are probably familiar with the controller's operation and need just some pointers. For this case we included the quick key sequence and, where appropriate, the related remote commands. For simplicity, we start the quick key sequence from the top level Axes Setup menu, assuming that you already selected the axis number you want to make the change to.

From the top level Axes Setup menu, press the MODIFY key to view (select) the first axis parameter. To scroll the parameter list you can use the UP or DOWN keys. For consistency, in the following descriptions we will use only the UP key.

Units

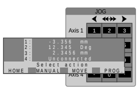
Units is the first axis parameter displayed. It represents the displacement units the controller will use for display and reporting. The available units are *mm*, *µm*, *In*, *mIn*, *µIn* and *Inc* for translation mechanical families and *Deg*, *Grd*, *Rad*, *mRad*, *µRad* and *Inc* for rotary mechanical families.

CAUTION

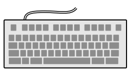
If you change the displacement units, you must revise all other parameters that are affected. All velocities, accelerations, travel limits, etc. must be scaled to the new units.

To change the displacement units, press the MODIFY key. Next press the UP or DOWN key to select new units. Press VALID to accept the new units and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → MODIFY → UP → VALID → QUIT → QUIT → YES → QUIT



TN — Read displacement units.

SN — Set displacement units.

Motion Type

The Motion Type parameter should not be changed by the user. It selects between real and simulated motion. The real motion is the normal mode of operation. The simulated motion is a mode in which the motion commands are not actually performed and is intended to be used only by the factory personnel for testing purposes.

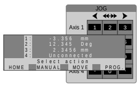
HOME Type

All standard motion devices offered with the MM4006 have an origin (home) switch and they can all perform a home search cycle. In cases where the controller is used with a device that does not have a home switch, the controller must know not to look for it.

Use the HOME Type parameter to tell the controller if the home switch is real (for standard devices with an origin switch) or if it should be simulated (for non-standard devices without a home switch). The simulated home switch (sometimes called floating) is assumed to be at the current position where the device is when the home search command is received.

If you need to change the HOME Type, press the **MODIFY** key when the parameter is displayed. Press the **CHANGE** key to select a new home switch type. Press **VALID** to accept the new type and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → **MODIFY** → **UP** → **MODIFY** → **CHANGE** → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**

Motor Type

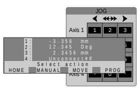
This parameter selects the type of motor to be used with the motion device. The two choices are Stepper and DC.

CAUTION

The motor type configured in the setup mode must match the actual motor and driver installed on the specific axis.

If the displayed motor type is incorrect for the selected axis, press the **MODIFY** key to change the setting. Press the **CHANGE** key to select a new motor type. Press **VALID** to accept the new selection and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → **MODIFY** → **UP** → **MODIFY** → **CHANGE** → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**

Control Loop

The MM4006 controller has the capability to operate both DC and stepper motors in closed loop or open loop configurations. This is an important feature, especially for the stepper motors. As described in the tutorial section, when operating a stepper motor in a mini- or micro-stepping mode, the actual position can vary a few steps when under load. If the servo loop in closed with an encoder, position errors are corrected both during the motion and at stop.

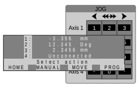
All stages offered with the MM4006 are equipped with an encoder, including those driven by a stepper motor. The recommendation is to always use the closed-loop mode.

NOTE

It is not recommended to operate a DC motor in open loop. This mode is used only at the factory to output a constant DC voltage to the motor driver for testing and calibration purposes.

If you need to change the control loop type, press the MODIFY key to modify the current setting. Press the CHANGE key to select a different loop type. Press VALID to accept the new loop type and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY → CHANGE → VALID →

QUIT → QUIT → YES → QUIT

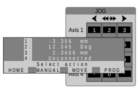
Periodicity

This mode enables to display periodically (for example 0 to 360°) for certain of rotary mechanical families (default, RTM, URM).

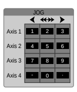
Press the UP key to advance to the next parameter. Press MODIFY and then the CHANGE key to select YES or NO. To accept this entry, press the VALID key. If periodicity is YES, enter the displacement period with the numeric keypad and then press the VALID key to accept this value. Press QUIT two times and then the YES key to save changes.

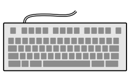
NOTE

Processing of an axis infinite movement is possible only if a periodicity has been defined, and only for rotary stages.



Axis Setup → MODIFY → UP → MODIFY → CHANGE → VALID →

 → VALID → QUIT → QUIT → YES → QUIT



CD — Set periodical display mode.

Motor Increment

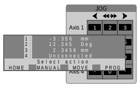
This parameter is used only for stepper motors and tells the controller how much the motion device will travel for each motor increment. By increment we mean one pulse going to the stepper driver, not necessarily a full motor step. Depending on the type of stepper driver, the motor increment could be a full step, a half step or a micro-step.

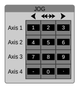
The Motor Increment parameter should reflect the actual stage/driver combination. A wrong setting will cause inaccurate closed-loop operation.

For the PP families, the Motor and the Encoder Increment (resolution) can be changed separately, on condition that the new value of Motor Increment would be inferior of the actual value of Encoder Increment.

If you need to change the motor increment setting, press the **MODIFY** key to modify the current value. Use the numeric keypad to enter the correct value. Press **VALID** to accept the new setting and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → **MODIFY** → **UP** → **MODIFY** →  → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**

Encoder Increment

This parameter defines the physical travel of the motion device that corresponds to one encoder count. It represent the resolution of the system and must reflect the real physical value (theoretical value, excluding all errors).

NOTE

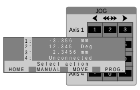
This parameter can be used to correct for a linear error in the motion device's absolute position accuracy. See the Motion Control Tutorial section for more details.

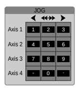
For the PP families, the Motor and the Encoder Increment can be changed separately, on condition that:

- If encoder resolution ≥ motor resolution: OK.
- If encoder resolution < motor resolution: OK but Motor Resolution = new encoder resolution.

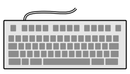
If you need to correct the encoder increment setting, press the **MODIFY** key to modify the current value. Use the numeric keypad to enter the correct value. Press **VALID** to accept the new setting and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → **MODIFY** → **UP** → **MODIFY** →  → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**



TU — Read encoder resolution.

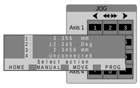
Scaling Speed

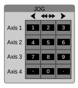
Scaling Speed is a hardware calibration parameter and is not intended to be used by the operator. It represents the approximate velocity the motion device will move if the maximum control voltage is sent to the driver (DC motor case). In other words, it is the velocity for a saturated DAC. It is a hardware calibration factor and the default value should not be modified.

For stepper motors it has a similar meaning but represents the stage velocity corresponding to the maximum acceptable motor speed.

If you need to set up a non-standard motion device that has no default parameters, after determining the correct value (motion control expertise is required), press the **MODIFY** key to set the Scaling Speed. Use the numeric keypad to enter the correct value. Press **VALID** to accept the new setting and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

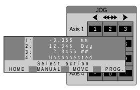
QUIT → QUIT → YES → QUIT

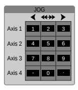
Maximum Speed

This is the maximum velocity allowed for a given motion device. No programmed velocities are allowed to exceed this value. It also represents the default velocity if no other value has been set previously (by a remote command). It can be changed by the user but it should never exceed the default value. Use the Speed Scaling parameter instead to temporarily reduce the motion velocities.

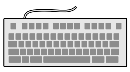
If you are setting up a new motion device that has no default parameters and have defined the Scaling Speed, press the **MODIFY** key to set the Maximum speed. Use the numeric keypad to enter a value that is about 80% of the Scaling Speed. Press **VALID** to accept the new setting and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



DV — Read desired velocity.

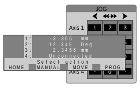
VA — Set velocity.

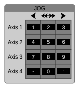
Manual Speed

This parameter defines the high velocity of the manual jog mode (using front panel or joystick). The default value is 50% of the Maximum Speed, but you can change it to suit your needs. The slow speed manual jog is one tenth of the high speed.

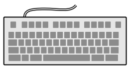
To change the manual jog high speed, press the MODIFY key when the Manual Speed parameter is displayed. Use the numeric keypad to enter a new value. Press VALID to accept the new setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



MH — Set manual velocity.

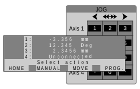
DM — Read manual velocity.

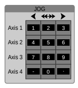
HOME Speed

This parameter sets the value of the home search high velocity portion. It is recommended that this value not be altered.

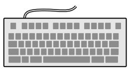
If you are setting up a new motion device that has no default parameters, press the MODIFY key to set the HOME speed. Use the numeric keypad to enter a value that is equal to 50% of the Maximum Speed. Press VALID to accept the setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



OH — Set home search velocity.

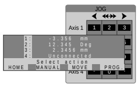
DO — Read home search velocity.

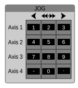
Acceleration

This parameter defines the maximum acceleration/deceleration value to be allowed in all programmed or commanded point-to-point motions. No remote or local commanded acceleration can exceed this value. The only motion that is not affected by this setting is the home search routine which uses its own acceleration values. The manual jog uses an acceleration ten times smaller than the value set with this parameter.

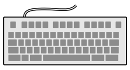
To change the maximum acceleration, press the MODIFY key. Use the numeric keypad to enter a value and then press VALID to accept the setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



AC — Set acceleration.

DA — Read desired acceleration.

Minimum Position

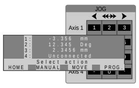
This parameter defines the negative (usually left) software travel limit. No motion will be allowed to exceed this position in the negative direction.

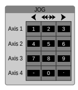
CAUTION

Do not set a value for this parameter more negative than the default value, otherwise the hardware limit switch will be tripped.

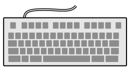
To change the negative software travel limit, press the MODIFY key. Use the numeric keypad to enter a new value and then press VALID to accept the setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



SL — Set left travel limit.

TL — Read left travel limit.

Maximum Position

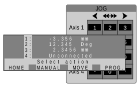
This parameter defines the positive (usually right) software travel limit. No motion will be allowed to exceed this position in the positive direction.

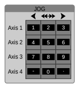
CAUTION

Do not set a higher value for this than the default value, otherwise the hardware limit switch will be tripped.

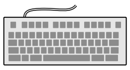
To change the positive software travel limit, press the MODIFY key. Use the numeric keypad to enter a new value and then press VALID to accept the setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



SR — Set right travel limit.

TR — Read right travel limit.

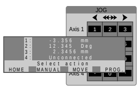
HOME Preset

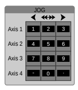
This feature is used to set the zero location according to the application's needs. This parameter defines the value that will be loaded into the position register when the motion device's home is found. The factory default is zero, meaning that at the home location the position is zero. If, for instance, this parameter is set to 12.3 mm, at the home location the controller reports position 12.3 mm.

Do not set a value for the HOME Preset parameter that is outside the software travel limits.

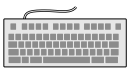
To change the HOME Preset parameter, press the MODIFY key. Use the numeric keypad to enter a new value and then press VALID to accept the setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



SH — Set home preset position.

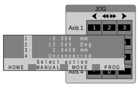
XH — Read home preset position.

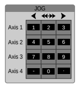
Kp

This parameter is the proportional gain factor of the digital PID filter. The valid range is between 0 and 1. All standard motion devices offered with the MM4006 have a set of conservative PID parameters stored in the controller's firmware. To change them you will need some knowledge of motion control loops and the help of a software utility. For some general guidelines read the Servo Tuning section.

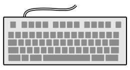
To change the proportional gain factor Kp, press the **MODIFY** key. Use the numeric keypad to enter a new value and then press **VALID** to accept the setting and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → **MODIFY** → **UP** → **MODIFY** →  → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**



KP — Set proportional gain.

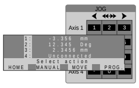
XP — Read proportional gain factor.

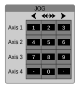
Ki

This parameter is the integral gain factor of the digital PID filter. The valid range is between 0 and 1. All standard motion devices offered with the MM4006 have a set of conservative PID parameters stored in the controller's firmware. To change them you need some knowledge of motion control loops and the help of a software utility. For some general guidelines read the Servo Tuning section.

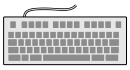
To change the integral gain factor Ki, press the **MODIFY** key. Use the numeric keypad to enter a new value and then press **VALID** to accept the setting and return to the previous menu.

Press the **UP** key to advance to the next parameter.



Axis Setup → **MODIFY** → **UP** → **MODIFY** →  → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**



KI — Set integral gain.

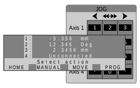
XI — Read integral gain factor.

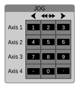
Kd

This parameter is the derivative gain factor of the digital PID filter. The valid range is between 0 and 1. All standard motion devices offered with the MM4006 have a set of conservative PID parameters stored in the controller's firmware. To change them you will need some knowledge of motion control loops and the help of a software utility. For some general guidelines read the Servo Tuning section.

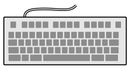
To change the derivative gain factor Kd, press the MODIFY key. Use the numeric keypad to enter a new value and then press VALID to accept the setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



KD — Set derivative gain.

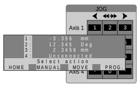
XD — Read derivative gain factor.

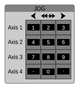
Ks

This parameter is the saturation gain factor of the PID filter integrator. The valid range is between 0 and 1. All standard motion devices offered with the MM4006 have a set of conservative PID parameters stored in the controller's firmware. To change them you will need some knowledge of motion control loops and the help of a software utility. For some general guidelines read the Servo Tuning section.

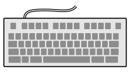
To change the proportional gain factor Ks, press the MODIFY key. Use the numeric keypad to enter a new value and then press VALID to accept the setting and return to the previous menu.

Press the UP key to advance to the next parameter.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



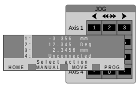
KS — Read proportional gain factor.

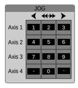
Maximum Error

This parameter represents the maximum allowed following error. If, at any time, the following error exceeds this value, the controller stops all motions in progress and turns the motor power off. Use good judgment when setting this parameter. A small value will cause premature fault and a large value will not protect the system from a real problem.

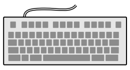
To change the Maximum Error parameter, press the **MODIFY** key. Use the numeric keypad to enter a new value and then press **VALID** to accept the setting and return to the previous menu.

If you have been following the Axes Setup procedure from the beginning of the section, pressing the **UP** key will bring you back to the first parameter that was discussed. Exit the Axes Setup by pressing the key.



Axis Setup → **MODIFY** → **UP** → **MODIFY** →  → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**



FE — Set max. following error.

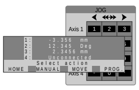
XF — Read max. following error.

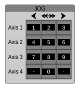
Backlash

This parameter represents the mechanical hysteresis of motion device. Use good judgment when setting this parameter.

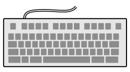
To change the backlash parameter, press the **MODIFY** key. Use the numeric keypad to enter a new value and then press **VALID** to accept the setting and return to the previous menu.

If you have been following the Axes Setup procedure from the beginning of the section, pressing the **UP** key will bring you back to the first parameter that was discussed. Exit the Axes Setup by pressing the **QUIT** key.



Axis Setup → **MODIFY** → **UP** → **MODIFY** →  → **VALID** →

QUIT → **QUIT** → **YES** → **QUIT**



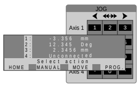
XB — Read mechanical backlash.

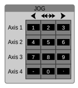
BA — Set mechanical backlash.

Display Resolution

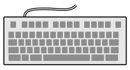
This menu modified axis position (digit number after decimal point). The controller used this parameter to adjust exactly the mechanical displacement.

To change the Display Resolution, press the MODIFY key. Use the numeric keypad to enter a new value and then press VALID to confirm the setting and return to the previous menu.



Axis Setup → MODIFY → UP → MODIFY →  → VALID →

QUIT → QUIT → YES → QUIT



NP — Set display resolution.

2.3 Operating In Local Mode

In addition to the SETUP mode, the other two types of operations that can be performed from the front panel of the MM4006 controller are motion-related commands and creating or editing motion programs.

The most common use of the Local Mode is to initiate motion and motion-related commands from the front panel. The following paragraphs describe this in detail.

From the top level Motor OFF menu (the power-on default menu) press the MOTOR ON button. The display will change to one similar to Fig. 2.7. We will call this the top level motor-on menu.

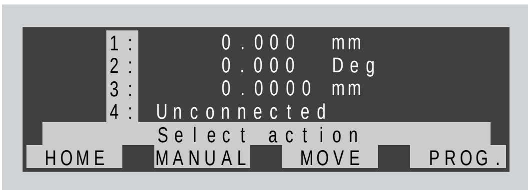


***Fig. 2.7** — Top level MOTOR ON menu.*

NOTE

It is possible to press the MOTOR ON button while in any menu. To avoid confusion and unexpected controller behaviours, it is strongly recommended to turn the motor power on only when in the top level Motor OFF menu.

NOTE

It is possible to press the MOTOR OFF button at any time. To avoid confusion, use this capability only for emergencies. During normal operation, turn the motor power off only when in the top level Motor ON menu.

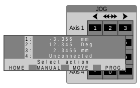
2.3.1 HOME Search

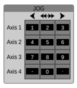
The HOME Search routine is a sequence of high and low speed motion segments through which the controller determines the exact location of a home (origin) switch and an encoder index pulse. A detailed description of the algorithm can be found in the Motion Control Tutorial section.

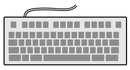
NOTE

It is strongly recommended that the user perform a home search routine after each controller power-on or reset. The controller must know the exact initial position of the motion device not only to accurately repeat a motion sequence (program) but also to prevent it from hitting the travel limits (limit switches). A limit switch detection is interpreted as a major fault and the motor power is turned off immediately.

To perform a home search routine, press the **HOME** function key from the top level power-on menu. The display will prompt you to select which axis should execute the home search. Use the keypad to indicate an axis number. If you enter or accept the default number 0, the controller will execute the home search routine sequentially on all installed axes, in the order specified in the General Setup.



MOTOR ON → **HOME** →  → **VALID**



OR — Search for home.

NOTE

The position value assumed at the home position is defined in the axis setup using the HOME Preset value or through the SH command.

2.3.2 Manual Jog

Manual Jog is a commonly used Local Mode front panel function. The selected axis will move at a pre-defined velocity. This type of motion is known as a JOG.

The MM4006 controller implements this function on the numeric keypad. The Manual Jog mode can be enabled either from the top level **MOTOR ON** menu or using the Move menu. In both cases, the calling function key is labelled **MANUAL** and functionality is identical.

In the Manual Jog mode, the display looks similar to Fig. 2.8.

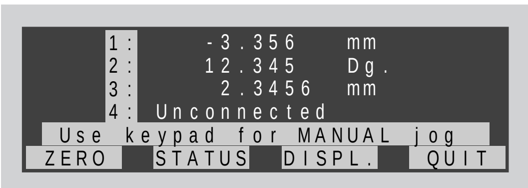
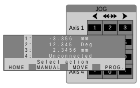


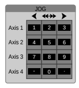
***Fig. 2.8** — Using the JOG menu.*

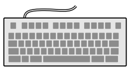
Line number 5 indicates that the keypad can be used to initiate a manual jog. As described in paragraph 2.1.3 First Jog, each keypad row controls one axis. The keys on the left initiate a jog in the negative direction and the keys on the right in the positive direction. To jog at a high speed, simultaneously press the corresponding middle key with one of the jog direction keys.

NOTE

The high speed manual jog velocity is set in the Axis Setup mode or by using the MH command. The low speed manual jog velocity is 10% of the high speed value.



MOTOR ON → MANUAL →  → QUIT



MC — Set manual mode.

To exit the Manual Jog mode press the QUIT key.

There are two more function keys defined in the manual jog menu. The ZERO key is described in the next paragraph. The STATUS key is described in the Introduction section, Display Configuration - Status Display paragraph.

The STATUS key displays a number of status indicators in front of each axis identifier and a general axis condition message at the end of each axis line (Fig. 2.9).

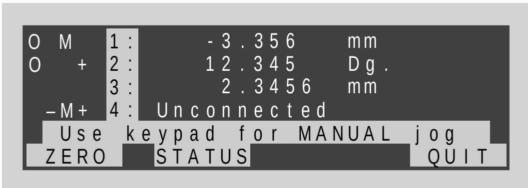


***Fig. 2.9** — Status display.*

The four status indicators are **0**, **–**, **M** and + have the following meaning:

O Will appear only if a “home search” routine has been performed successfully on that axis. It indicates that a mechanical “origin” has been found.

– Indicates that the negative direction (left) limit switch has been activated (tripped).

M Appears when the mechanical origin switch is in “high” state. As a stage moves from one end to the other, you will see this indicator appear and disappear. This means that the stage has moved from one side of the switch to the other. The state of this indicator does not affect the normal operation of the motion device. For a complete description of the home search algorithm see Section 4, Motion Control Tutorial.

\+ Indicates that the positive direction (right) limit switch has been activated (tripped).

NOTE

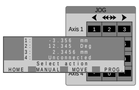
If both – and + appear, the motion device is either disconnected or a hardware failure exists. On power-up sequence, the controller checks every axis for this case. If found, it assumes that no motion device is present. The axis is marked with Unconnected on the display and all commands for it will be ignored.

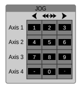
At the end of each axis information line an OK is displayed if no error has been detected. If a problem is detected on one of the axes, the message ERROR will appear.

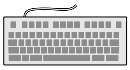
2.3.3 Zero Display

During operation in Local Mode, the need may arise to force the current position to become zero. This means that, without moving, the current position is displayed and reported as zero. Any subsequent motion will be referenced to this new zero location and the software limits will be recalculated to reflect the change while keeping their absolute position relative to the stage.

To activate this function from the Manual Jog menu, press the ZERO key. Using the keypad, enter the axis number you want to select or leave the 0 default to zero all active axes. Press the VALID key to execute the command.



MOTOR ON → MANUAL → ZERO →  → VALID → QUIT



ZP — Zero position.

2.3.4 Relative Moves

A move is defined as a point-to-point motion. The initial point is the current position and the ending point is the destination, or desired position.

There are two types of moves: relative and absolute. In this section we discuss the relative moves.

A relative move is defined as a move for which the destination is specified as an incremental distance from the current position. Repeating a 1 mm relative move command, for instance, will advance the motion device 1 mm at a time. For this reason, the relative motion is sometimes called incremental motion.

From the top level MOTOR ON menu press the MOVE key. This will activate the first Move Menu:

MANUAL RELAT. ABSOL. NEXT

Select NEXT to activate the second move menu:

MANUAL INFINI QUIT NEXT

Select NEXT to come back in the first move menu or select QUIT to quit this menu.

Select the RELAT. function key and the display will prompt you to enter the desired relative motion on the first active axis (Fig. 2.10).

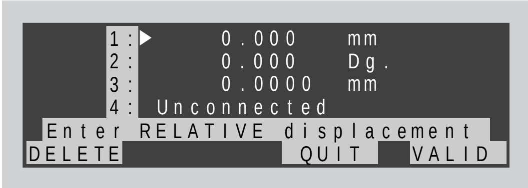


Fig. 2.10 — Relative motion menu.

All displayed positions become zero and the ▶ symbol indicates which numerical value will be changed with the keypad. The position display becomes zero because the values entered are relative motions. A zero relative motion, the default value, means that the motor will not move.

Enter a desired positive or negative relative motion. Press the **VALID** key to accept the value, edit the entry using the **DELETE** key or exit this mode by pressing the **QUIT** key.

Once the **VALID** key is pressed, the ▶ symbol moves to the next connected axis. Enter a numerical value or press **VALID** to accept the zero default.

When the **VALID** key is pressed on the last active axis, the display changes to a move execution menu. The two active function keys are **ALL** and **QUIT** .

The **QUIT** key will exit this mode and return to the top level motor-on menu.

The **ALL** key will start a relative motion on all axes using the values entered.

If relative motion values were entered on multiple axes but now you require only one axis to move, use the numerical keypad to select the axis and start the motion. Each keypad row controls one axis and the first or last key in the row determines the direction. The middle key has no effect.

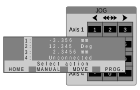
For example, if axis number 1 is to be moved, press the 3 key to start the motion in the specified direction or the 1 key to move it in the opposite direction.

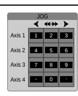
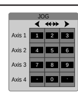
NOTE

When the relative motion is complete, the display does not return to the top level menu. This feature allows the user to repeat relative motions an unlimited number of times.

Press the **QUIT** key to terminate the relative move mode and return to the Move menu.

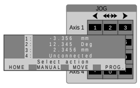
Single Axis Relative Move

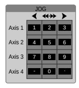
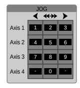
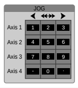
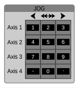


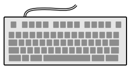
MOTOR ON → MOVE → RELAT. →  → VALID →  →

QUIT → QUIT

Multiple Axes Relative Move



MOTOR ON → MOVE → RELAT. →  → VALID →  → VALID →  → VALID →  → VALID → ALL → QUIT



PR — Move to relative position.

2.3.5 Absolute Moves

Absolute moves initiate motion to a destination specified by a value relative to the zero (home) position rather than the current position used by the relative move command. Repeated identical absolute move commands therefore are not productive because once at the destination, the current position becomes the desired position.

To activate the Absolute Move mode, from the Move menu press the ABSOL. function key. The display will prompt you to enter a destination value for the first active axis (Fig. 2.11).

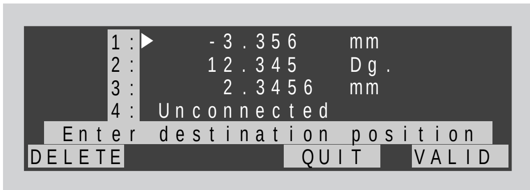


***Fig. 2.11** — Absolute move menu.*

If an axis is not to be moved, you must enter the current position as the desired position and press the VALID key. The controller will recognise this as a zero displacement motion and not issue any motion command for that axis.

Pressing the VALID key after a numerical entry will shift the ▶ symbol to the next active axis. Repeat the operation for each installed axis.

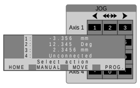
When all destinations are defined, pressing the VALID key on the last active axis will change the menu on the display to:

EXEC.			QUIT

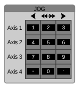
Pressing the EXEC. key will start the absolute motion on all axes. When motion on all axes is complete, the display returns to the Move menu.

The display returns to the Move menu if the QUIT key is pressed without executing the absolute motion.

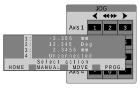
Newport



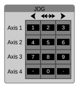
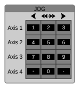
Single Axis Absolute Move

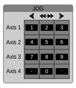
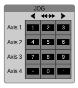
MOTOR ON → MOVE → ABSOL. →  → VALID →

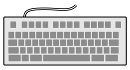
EXEC. → QUIT



Multiple Axes Absolute Move

MOTOR ON → MOVE → ABSOL. →  → VALID →  →

VALID →  → VALID →  → VALID → EXEC. → QUIT



PA — Move to absolute position.

2.3.6 Program Execution

The most complex motion that can be initiated from the front panel is the execution of a motion program. In this mode, an existing program in memory is called and executed a specified number of times.

NOTE

A program must exist in the controller's non-volatile memory in order to be executed. See the Programming In Local Mode section on creating programs from the front panel or the Remote Mode section for downloading programs to the controller.

To execute a program from the top-level MOTOR ON menu, press the PROG. key. The controller enters the Program Mode and displays the following menu:

EXEC. CREATE MODIFY QUIT

For now we are interested only in the EXEC. and QUIT keys.

QUIT will return the display to the top level motor-on menu.

Pressing the EXEC. key will enter the Program Execution mode. In the first screen, the user is asked to select the program number to be executed. Use the keypad to enter a valid, existing program number.

NOTE

Valid program numbers are from 1 to 100. It is the user's responsibility to remember what programs are loaded and what they do. Stored programs can be viewed locally or remotely but logging the stored program list is the best approach to motion program management.

Press the VALID key to accept the program number. If the specified program does not exist, the controller will inform you and remain in the same menu until a valid program number is entered or the QUIT key is pressed.

Once a valid program is selected, the controller prompts you for the number of times to repeat program execution. Enter the desired number on the keypad and press **VALID** .

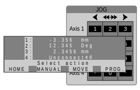
The next menu confirms your choice to execute the selected program the specified number of times or allows you to exit without execution by pressing the **QUIT** key.

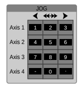
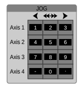
To start the program sequence press the **EXEC.** key. The display informs you on line number 5 that a program is executing and no function keys are available.

NOTE

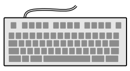
The only way to stop a program or a sequence of programs from the front panel is to turn the motor power off. Use this method for an emergency stop.

When the program sequence is finished, the controller returns to the top level **MOTOR ON** menu.



MOTOR ON → **PROG.** → **EXEC.** →  → **VALID** →  →

VALID → **EXEC.**



EX — Execute a program.

2.3.7 Axis Infinite Movement

The infinite movement is realised with the **INFINI** menu to move one or some axis eternally (none stop).

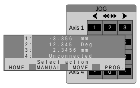
Press the **MOVE** key on **MOTOR ON** menu. Press the **NEXT** key to activate the second Move menu. Press the **INFINI** key and then use the numeric keypad to start motion. Press **QUIT** to exit the **INFINI** menu.

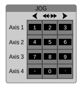
NOTE

Processing of an axis infinite movement is possible only if a periodicity has been defined, and only for rotary stages.

Now, if one key of numeric keypad is pressed, one infinite displacement will be activated or stopped:

- Key **1**: Infinite displacement of axis #1 or #5 in negative direction.
- Key **2**: Stop the infinite displacement of axis #1 or #5.
- Key **3**: Infinite displacement of axis #1 or #5 in positive direction.
- Key **4**: Infinite displacement of axis #2 or #6 in negative direction.
- Key **5**: Stop the infinite displacement of axis #2 or #6.
- Key **6**: Infinite displacement of axis #2 or #6 in positive direction.
- Key **7**: Infinite displacement of axis #3 or #7 in negative direction.
- Key **8**: Stop the infinite displacement of axis #3 or #7.
- Key **9**: Infinite displacement of axis #3 or #7 in positive direction.
- Key **–**: Infinite displacement of axis #4 or #8 in negative direction.
- Key **0**: Stop the infinite displacement of axis #4 or #8.
- Key **.**: Infinite displacement of axis #4 or #8 in positive direction.



MOTOR ON → MOVE → NEXT → INFINI →  →

QUIT → QUIT

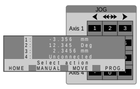
2.3.8 Stop Axis Infinite Movement

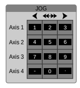
The infinite movement can be stopped at any time, thanks to the INFINI menu.

Press the MOVE key on MOTOR ON menu. Press the NEXT key to activate the second Move menu. Press the INFINI key and then use the numeric keypad to stop motion. Press QUIT to exit the INFINI menu.

Now, if one key of numeric keypad is pressed, one infinite displacement will be activated or stopped:

- Key 2: Stop the infinite displacement of axis #1 or #5.
- Key 5: Stop the infinite displacement of axis #2 or #6.
- Key 8: Stop the infinite displacement of axis #3 or #7.
- Key 0: Stop the infinite displacement of axis #4 or #8.
- Key STOP: Stop all infinite displacements and quit the menu.



MOTOR ON → MOVE → NEXT → INFINI →  or STOP →

QUIT → QUIT

2.4 Programming in Local Mode

The MM4006 controller allows the user to create and edit programs from the front panel. This makes it a true stand-alone unit, capable of executing most motion and motion-related functions without the help of an external computer.

NOTE

Though very versatile, the front panel programming capabilities of the MM4006 controller are intended to be used only for smaller, simple motion programs. For larger, more sophisticated programs, the use of a computer with a powerful editing environment is still recommended.

The Program mode can be invoked from both top-level Motor OFF or MOTOR ON menus. The only difference is that, when starting from the top level MOTOR ON menu, an additional program execution function key is available. Both functions that are of interest for this section, program creation and program editing, are the same regardless how they have been activated.

2.4.1 General Concepts

To communicate with the MM4006 controller, a language is needed that both user and controller can understand. When communicating remotely we use a motion control language that is described in the Remote Mode section. A program downloaded remotely is stored in non-volatile memory, as is a program created locally.

Any program in memory can be read and edited both locally and remotely. For this reason, to create a program in local mode we need a way to enter alpha-numeric commands from the front panel. Since the number of keys available on the front panel is limited, the MM4006 controller uses a special convention to enter motion program commands. Taking advantage of the context-sensitive menus, commands are assigned to function keys and numerical values are entered on the numerical keypad.

Because only four function keys are available, only three commands can be viewed at a time (the fourth key is reserved to advance or exit the menu). To avoid scrolling through the entire list, commands are grouped by categories.

When entering or editing a command line, the controller will display it on the first four lines.

A command line can have up to 110 characters. The display has only 30 characters per line, so long command lines will take up several lines. For this reason, an asterisk (*) will identify the beginning of each logical command line.

NOTE

To save display space when wrapping around a command line, the controller does not look for command boundaries (separators). The result is that commands and numbers will be split without any restriction.

Once a command line has been entered and terminated, it will disappear from the display to make room for a new one. To scroll through the program and view different command lines, the controller must be in the Program Editing mode. The Program Creation mode does not allow you to view program lines other than the one being written or edited.

2.4.2 Creating a Program

To start creating a program, first enter the Program mode by pressing the PROG. key (from the top-level Motor OFF or MOTOR ON menu) and then press the CREATE key to enable the program creation mode.

Since the controller can store up to 100 different programs, the first screen will ask which program number you want to create. You can consider this number as the program name. When retrieving a program, you will call it by its assigned number.

Enter a program number on the numeric keypad and press VALID. If a program with this number (name) already exists, a warning screen will appear. In this case, press YES to overwrite it or NO to return to the Program mode and start over again.

Once a valid program number is accepted, the controller enters the Program Creation mode. As mentioned earlier, the commands are grouped in categories. To select a particular command, the user must navigate through a number of screens (menus). An important controller characteristic to remember is that it responds to command lines. This means that, when commands are entered, they will be placed on the same command line until the line is terminated.

NOTE

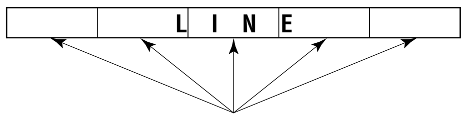
The controller is always able to create, store or modify programs for all four axes, even if all axes are not installed.

The first screen separates the program entries into two categories: simple lines and while loops. The menu has the following look:

LINE	WHILE		QUIT

The QUIT function key exits the program creation mode.

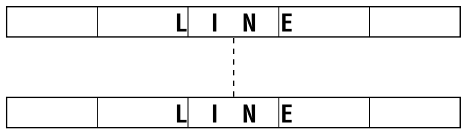
The LINE key will start entering a simple command line, composed of motion and I/O commands:



Commands

The WHILE key will start a special while loop creation mode that lets you enter command lines inside a while loop:

Start WHILE Loop



End WHILE Loop

Command Line Creation

Start by first looking at the options offered for the simple command line. There are nine different types of commands available. They are separated in four different screens:

ABSOL.	RELAT.	SYNCH.	NEXT
HOME	DELAY	MOTORS	NEXT
IFINP.	REPEAT	OUTPUT	NEXT
	QUIT	VALID	NEXT

This menu level is the Line Entry menu.

The actual commands are on the first three menus. The fourth is used to accept and terminate the line entry by pressing the VALID key or to abort the current line being worked on by pressing QUIT. The last function key of each screen, NEXT, and advances the display to the next menu. It can be viewed as the scrolling key. When the NEXT key is pressed on the last screen, the display returns to the first menu of the group.

The nine available program functions can be entered by pressing the appropriate function key in the menu and have the following meanings:

ABSOL.

Start an absolute point-to-point motion.

When this key is pressed, the controller asks for the axis number the motion is to be performed on. Use the keypad to enter a valid axis number. If 0 is entered, the controller will assume that you want to perform a simultaneous motion on all axis.

Pressing **VALID** will cause the screen to display the usual position information for all four axes. The ▶ symbol appears for the first axis and the user can enter the desired destination for it. Press **VALID** to accept the entry and advance the ▶ symbol to the next axis. If a single axis was selected or the entry is made on the last axis, the command is stored and the display returns to the Line Entry menu.

Command generated

PA — Move to absolute position.

NOTE

A simultaneous motion is not a synchronous, a linear interpolated motion. The motion is not truly synchronised, because there are one or more servo cycles delay between axes. For most applications this causes an imperceptible error.

RELAT.

Start a relative point-to-point motion.

When this key is pressed, the controller asks for the axis number the motion is to be performed on. Use the keypad to enter a valid axis number. If 0 is entered, the controller will assume that you want to perform a simultaneous motion on all axis.

Pressing **VALID** will cause the screen to display the usual position information for all four axes. The ▶ symbol appears for the first axis and the user can enter the desired relative travel for it. Press **VALID** to accept the entry. If a single axis was selected or the entry is made on the last axis, the command is stored and the display returns to the Line Entry menu.

Command generated

PR — Move to relative position.

NOTE

The controller recognises zero-travel relative motions as no-motions and does not issue a command for them.

SYNCH.

synchronise motion sequence commands.

This function will add to the program a command that causes the controller to wait for a motion to be complete before executing the next command.

Depending on the selection, the controller can wait for all or one axis to complete motion. When prompted, enter the axis number to wait for, or just press **VALID** to accept the default 0 and wait for all axes.

Command generated

WS — Wait for motion stop.

HOME Perform a home search sequence.

Use this function to initiate a home search sequence on one or all axes. Press **VALID** or enter a 0 for all axes or select an axis number on the keypad. Pressing **VALID** will add the command to the command line and return to the Line Entry menu.

Command generated

OR — Search for home.

DELAY Introduce a delay in the program execution.

This command, when added to a program, causes the controller to wait for a specified amount of time. Use the numeric keypad to specify the delay and then press **VALID** to accept the value and return to the Line Entry menu.

Command generated

PR — Wait.

MOTORS Turn motor power on or off.

Use this function to turn the power to the motors on or off When the program is executing it will have the same effect as the front panel MOTOR OFF / MOTOR ON buttons. To add a command that forces the motor power to a certain state, press the **MOTORS** key and then use the **CHANGE** key to select the desired action. When done, press **VALID** to accept the entry and return to the Line Entry menu.

Commands generated

MO — Motor power on.

MF — Motor power off.

IFINP. Conditionally execute a line on I/O input port.

This function should be placed only at the beginning of a command line to control its execution. It will allow the execution of the following commands on the line only if the specified I/O input bit has the requested state. If the condition is not met at the time of evaluation, the rest of the command line is ignored and the program execution continues with the next line.

After pressing the **IFINP.** key, the display asks you to select an input bit to be tested. Enter a number between 1 and 8 on the numeric keypad and then press the **VALID** key. Next, press the **CHANGE** key to specify the high or low state of the bit and then the **VALID** key to accept the entry and return to the Line Entry menu.

Command generated

IE — If I/O input is equal.

REPEAT repeat a command line a number of times.

Use this function only at the end of a command line to repeat its execution a number of times. When selected, enter on the keypad the number of time you want to repeat the line and then press **VALID** to accept the entry and return to the Line Entry menu.

Command generated

RP — repeat command line.

OUTPUT Set a bit on the I/O output port.

This function will generate a command that sets an I/O output bit to a specified state. Use the numeric keypad to enter a number between 1 and 8 to select a bit or enter 0 to set all bits and then press **VALID** to accept the selection. Next, press the **CHANGE** key to specify the operation to be performed on the bit: set high, set low or toggle. Press the **VALID** key to accept the entry and return to the Line Entry menu.

Commands generated

CB — Clear I/O output bit.

SB — Set I/O output bit.

TG — Toggle I/O output bit.

When all entries have been made on a command line, use the **NEXT** key to find the screen in the Line Entry menu that has the **VALID** function key and press it to save the line in memory and advance to a new one.

WHILE Loop Creation

As mentioned earlier, the Program Creation menu offers the choice of creating simple while loops.

NOTE

Programs created from the front panel can have only simple while loops. Remote programs could have up to 100 nested loops.

From the Program Creation menu press the **WHILE** key. The next selection you have to make is the type of while loop. The controller can do a check on an I/O input bit or a variable, thus the two choices are **INPUT** and **VAR.** .

Pressing the **INPUT** key will start a loop that repeats while an I/O input bit has a specified state. First enter a bit number between 1 and 8 on the keypad, press **VALID** and then using the **CHANGE** key select the bit state. When done, press the **VALID** key to accept the entry.

If you want to create a loop that repeats a specified number of times, at the while selection menu press the **VAR.** key. The next choice you have to make is the number of times you want to repeat the loop. Enter the number on the keypad and press **VALID** . The controller will write the necessary commands to initialise a new variable, to increment it every time the loop executes and to verify that it reached the specified number.

After defining the initial loop parameters, the display shows the WHILE Loop menu with two choices:

		LINE	WEND

Pressing **LINE** will enter a Line Entry menu identical to the one described in the Command Line Creation paragraph. Use it to create command lines that will be part of the while loop. Enter command lines as described in the previous paragraph. When a line is terminated, the display returns to the WHILE Loop menu. To enter a new line, press the **LINE** key again and repeat the operation.

To close the while loop, press the **WEND** key. This will add the appropriate command to close the loop and return the display to the Program Creation menu.

With the functions described up to this point we can create a simplified flowchart of the Program Creation menu (Fig. 2.12).

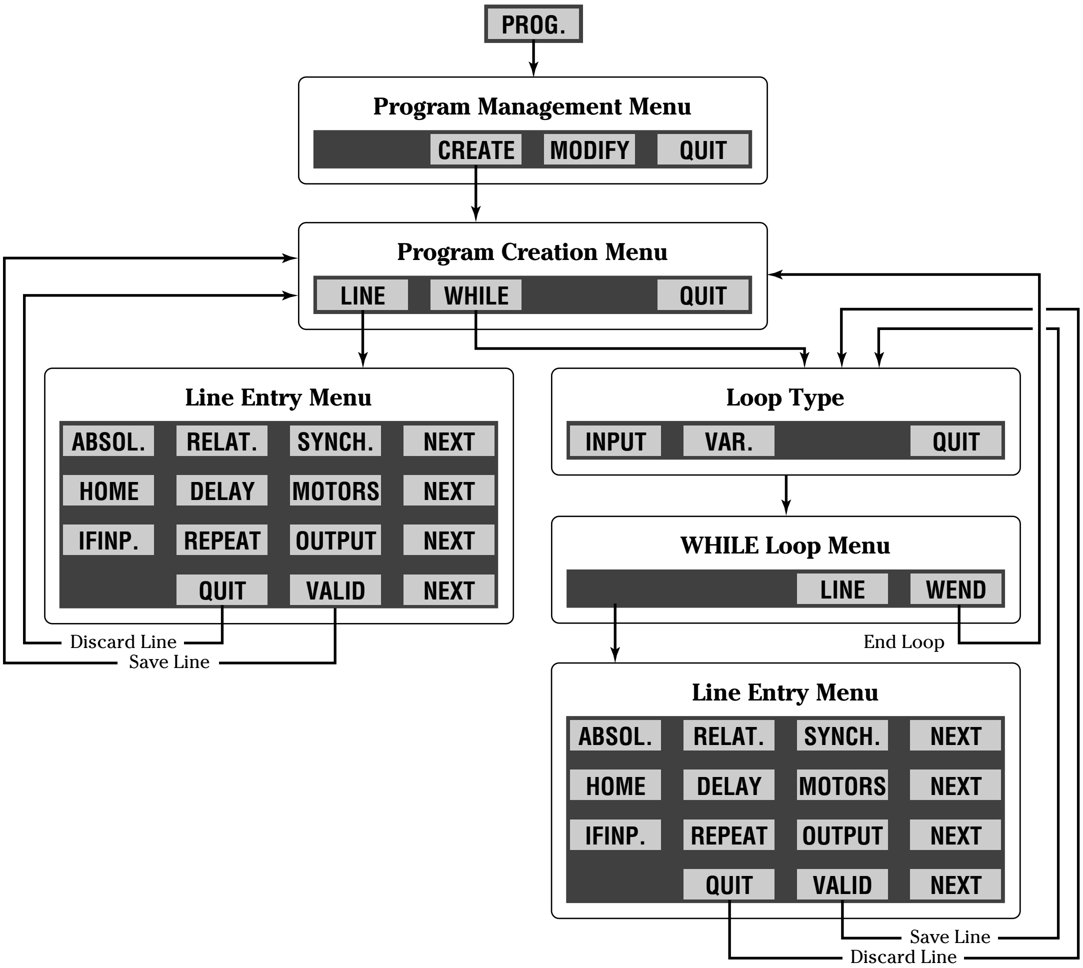


Fig. 2.12 *— Program Creation Menu Flow Chart.*

2.4.3 Modifying a Program

The Program Creation mode does not have the capability to edit commands or command lines already entered in a program. To modify anything in a program you must enter the Program Editing mode by pressing the MODIFY key from the Program mode.

When this option is selected, the controller asks for the program number (name) to be modified. Enter the desired number on the keypad and press the VALID key. Next, the controller will show the top portion of the selected program on the first four lines of the display.

NOTE

Selecting an empty (non-existent) program is allowed. In this case, a new program with the specified number will be created if commands are added with the Insert feature.

There are more than three possible options in the Program Editing mode, thus the need again to split the functions in a number of screens:

DOWN	UP		NEXT
FIRST	LAST		NEXT
INSERT	DELETE	QUIT	NEXT

This menu level is the Program Editing menu.

The NEXT key advances to the next line of the menu. When at the last menu line, pressing it will display again the first line of the menu.

The six editing-specific commands have the following meanings:

UP

Scroll the program listing up.

When the key is pressed, the program listing is scrolled up by one display line.

DOWN

Scroll the program listing down.

When the key is pressed, the program listing is scrolled down by one display line.

FIRST

Display first line of the program listing.

When the key is pressed, the program listing on the display will start with the first line of the program and the display changes to the first Program Editing menu line:

UP	DOWN		NEXT

LAST

Display the last line of the program listing.

When the key is pressed, the display shows the first blank line after the last program line and then the display changes to the top of the Program Editing menu:

UP	DOWN		NEXT

To see the last program line you must press the DOWN key.

INSERT

Insert a new program line.

Pressing this function key allows you to add a command line to an existing program. The new line is inserted before the first line currently displayed.

When the INSERT key is selected, the controller activates the Line Entry menu. Following the descriptions in the Command Line Creation paragraph, create a new command line and, when done, press the key to terminate it and return to the Program Editing menu.

DELETE

Delete a program line.

Program lines cannot be edited. They can only be erased and new ones created.

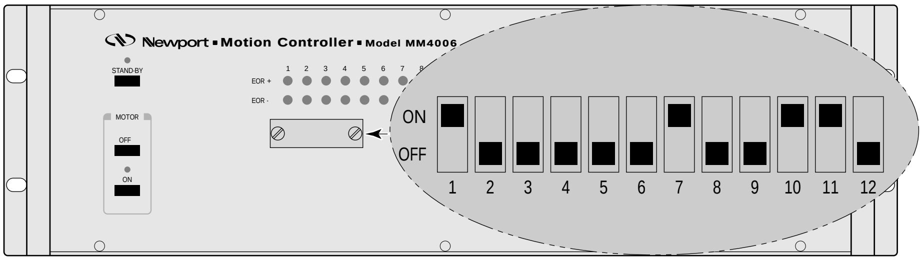
To delete a program line, use the UP, DOWN, FIRST, and LAST keys to scroll through the listing until the line to be deleted is the first one on the display. Pressing the DELETE key will erase the line and the display returns to the Program Editing menu.

When all the modifications have been made, press the QUIT key. The controller will ask if you want to save the changes. Press the YES key to accept the modifications and return to the Program menu.

2.5 Operating in Blind Front Panel

2.5.1 Dip Switches to Configure the Communication

A window in the front panel give access to twelve slide dip switches, these dip switches are used to configure the communication setup of the controller. These dip switches must be set accordingly to the communication parameters you want to use to enable your computer to communicate with the controller. The following tables will help you to properly setup these switches.



The controller only reads these switches at its initialisation. So, to change the configuration you should switch the controller MOTOR OFF (STAND-BY) make the new switches setup and switch the controller MOTOR ON.

2.5.2 RS-232-C Communication Setup

Front panel DIP switches setting

1	2	3	4	5	6	7	8	9	10	11	12	Description
ON												RS-232-C Communication
	OFF	OFF	OFF	OFF	OFF							Not Used
						ON						Disable XON/XOFF Mode
						OFF						Enable XON/XOFF Mode
							ON	ON				Terminator: LF
							ON	OFF				Terminator: CR/LF
							OFF	ON				Terminator: LF/CR
							OFF	OFF				Terminator: CR
									ON	ON	ON	9600 baud
									OFF	ON	ON	1200 baud
									ON	OFF	ON	2400 baud
									OFF	OFF	ON	4800 baud
									ON	ON	OFF	19200 baud
									OFF	ON	OFF	38400 baud
									ON	OFF	OFF	57600 baud
									OFF	OFF	OFF	115200 baud

Default Parameters
Parity: NONE
Stop bits: 1
Word length: 8 bits

2.5.3 IEEE-488 Communication Setup

Front panel DIP switches setting												
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Description**
OFF												IEEE-488 Communication
	OFF	ON	ON	ON	ON							Adress 1
	ON	OFF	ON	ON	ON							Adress 2
	ON	OFF	ON	ON	ON							Adress 3
	ON	ON	OFF	ON	ON							Adress 4
	OFF	ON	OFF	ON	ON							Adress 5
	ON	OFF	OFF	ON	ON							Adress 6
	OFF	OFF	OFF	ON	ON							Adress 7
	ON	ON	ON	OFF	ON							Adress 8
	OFF	ON	ON	OFF	ON							Adress 9
	ON	OFF	ON	OFF	ON							Adress 10
	OFF	OFF	ON	OFF	ON							Adress 11
	ON	ON	OFF	OFF	ON							Adress 12
	OFF	ON	OFF	OFF	ON							Adress 13
	ON	OFF	OFF	OFF	ON							Adress 14
	OFF	OFF	OFF	OFF	ON							Adress 15
	ON	ON	ON	ON	OFF							Adress 16
	OFF	ON	ON	ON	OFF							Adress 17
	ON	OFF	ON	ON	OFF							Adress 18
	OFF	OFF	ON	ON	OFF							Adress 19
	ON	ON	OFF	ON	OFF							Adress 20
	OFF	ON	OFF	ON	OFF							Adress 21
	ON	OFF	OFF	ON	OFF							Adress 22
	OFF	OFF	OFF	ON	OFF							Adress 23
	ON	ON	ON	OFF	OFF							Adress 24
	OFF	ON	ON	OFF	OFF							Adress 25
	ON	OFF	ON	OFF	OFF							Adress 26
	OFF	OFF	ON	OFF	OFF							Adress 27
	ON	ON	OFF	OFF	OFF							Adress 28
	OFF	ON	OFF	OFF	OFF							Adress 29
	ON	OFF	OFF	OFF	OFF							Adress 30
	OFF	OFF	OFF	OFF	OFF							Adress 31
						ON						Disable SRQ Mode
						OFF						Enable SRQ Mode
							ON	ON				Terminator: LF
							ON	OFF				Terminator: CR/LF
							OFF	ON				Terminator: LF/CR
							OFF	OFF				Terminator: CR
									OFF	OFF	OFF	Not Used

2.5.4 RS-485 Point-to-Point Communication Setup

This is the mode for single computer to a single MM4006 controller in long distance or noisy environment high speed communication.

Front panel DIP switches setting

1	2	3	4	5	6	7	8	9	10	11	12	Description
OFF												RS-485 Communication
	ON	ON	ON	ON	ON							Adress must be Zero
						ON						Disable XON/XOFF Mode
						OFF						Enable XON/XOFF Mode
							ON	ON				Terminator: LF
							ON	OFF				Terminator: CR/LF
							OFF	ON				Terminator: LF/CR
							OFF	OFF				Terminator: CR
									ON	ON	ON	9600 baud
									OFF	ON	ON	1200 baud
									ON	OFF	ON	2400 baud
									OFF	OFF	ON	4800 baud
									ON	ON	OFF	19200 baud
									OFF	ON	OFF	38400 baud
									ON	OFF	OFF	57600 baud
									OFF	OFF	OFF	115200 baud

Default Parameters
Parity: NONE
Stop bits: 1
Word length: 8 bits

2.5.5 RS-485 Network Communication Setup

This RS-485 setup meets the requirements for a multi-drop communication network up to 31 MM4006 controllers.

Front panel DIP switches setting

1	2	3	4	5	6	7	8	9	10	11	12	Description
OFF												RS-485 Communication
	OFF	ON	ON	ON	ON							Adress 1
	ON	OFF	ON	ON	ON							Adress 2
	ON	OFF	ON	ON	ON							Adress 3
	ON	ON	OFF	ON	ON							Adress 4
	OFF	ON	OFF	ON	ON							Adress 5
	ON	OFF	OFF	ON	ON							Adress 6
	OFF	OFF	OFF	ON	ON							Adress 7
	ON	ON	ON	OFF	ON							Adress 8
	OFF	ON	ON	OFF	ON							Adress 9
	ON	OFF	ON	OFF	ON							Adress 10
	OFF	OFF	ON	OFF	ON							Adress 11
	ON	ON	OFF	OFF	ON							Adress 12
	OFF	ON	OFF	OFF	ON							Adress 13
	ON	OFF	OFF	OFF	ON							Adress 14
	OFF	OFF	OFF	OFF	ON							Adress 15
	ON	ON	ON	ON	OFF							Adress 16
	OFF	ON	ON	ON	OFF							Adress 17
	ON	OFF	ON	ON	OFF							Adress 18
	OFF	OFF	ON	ON	OFF							Adress 19
	ON	ON	OFF	ON	OFF							Adress 20
	OFF	ON	OFF	ON	OFF							Adress 21
	ON	OFF	OFF	ON	OFF							Adress 22
	OFF	OFF	OFF	ON	OFF							Adress 23
	ON	ON	ON	OFF	OFF							Adress 24
	OFF	ON	ON	OFF	OFF							Adress 25
	ON	OFF	ON	OFF	OFF							Adress 26
	OFF	OFF	ON	OFF	OFF							Adress 27
	ON	ON	OFF	OFF	OFF							Adress 28
	OFF	ON	OFF	OFF	OFF							Adress 29
	ON	OFF	OFF	OFF	OFF							Adress 30
	OFF	OFF	OFF	OFF	OFF							Adress 31
						ON						Disable XON/XOFF Mode
						OFF						Enable XON/XOFF Mode
							ON	ON				Terminator: LF
							ON	OFF				Terminator: CR/LF
							OFF	ON				Terminator: LF/CR
							OFF	OFF				Terminator: CR
									ON	ON	ON	9600 baud
									OFF	ON	ON	1200 baud
									ON	OFF	ON	2400 baud
									OFF	OFF	ON	4800 baud
									ON	ON	OFF	19200 baud
									OFF	ON	OFF	38400 baud
									ON	OFF	OFF	57600 baud
									OFF	OFF	OFF	115200 baud

Default Parameters
Parity: NONE
Stop bits: 1
Word length: 8 bits

Section 2
Local Mode


Table of Contents
Section 4 — Motion Control Tutorial

Section 4
Motion Control Tutorial

4.1 Motion Systems

A schematic of a typical motion control system is shown in Fig. 4.1.

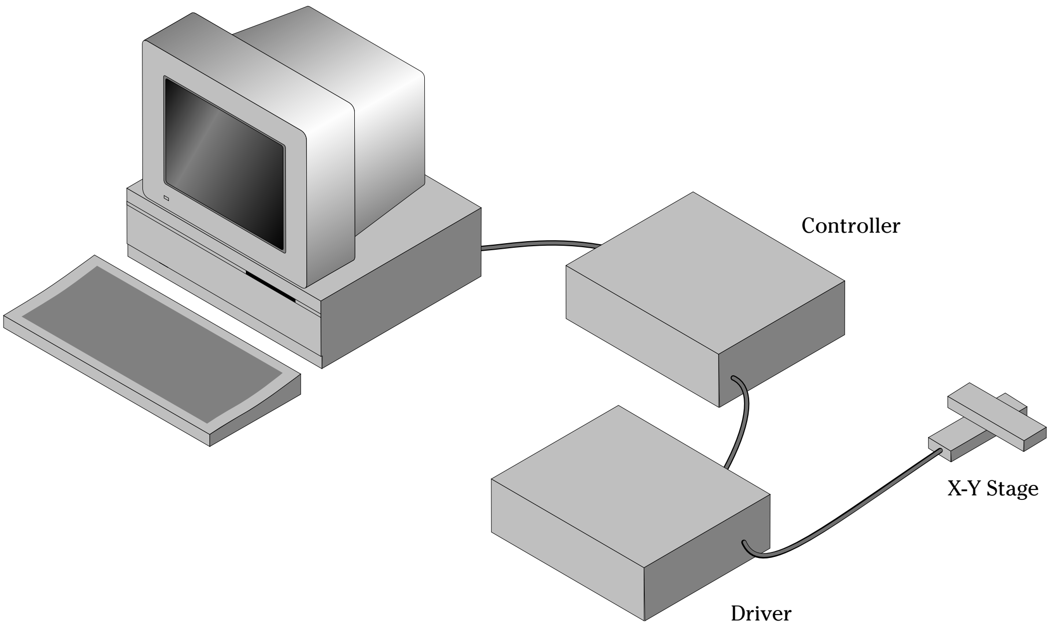


***Fig. 4.1** — Typical Motion Control System.*

Its major components are:

Controller

An electronic device that receives motion commands from an operator directly or via a computer, verifies the real motion device position and generates the necessary control signals.

Driver

An electronic device that converts the control signals to the correct format and power needed to drive the motors.

Motion device

An electro-mechanical device that can move a load with the necessary specifications.

Cables

Needed to interconnect the other motion control components.

If you are like most motion control users, you started by selecting a motion device that matches certain specifications needed for an application. Next, you chose a controller that can satisfy the motion characteristics required. The chances are that you are less interested in how the components look or what their individual specs are, but want to be sure that together they perform reliably according to your needs.

We mentioned this to make a point: A component is only as good as the system lets (or helps) it to be.

For this reason, when discussing a particular system performance specification, we will also mention which components affect performance the most and, if appropriate, which components improve it.

4.2 Specification Definitions

People mean different things when referring to the same parameter name. To establish some common ground for motion control terminology, here are some general guidelines for the interpretation of motion control terms and specifications.

- As mentioned earlier, most motion control performance specifications should be considered system specifications.
- When not otherwise specified, all error-related specifications refer to the position error.
- The servo loop feedback is position-based. All other velocity, acceleration, error, etc. parameters are derived from the position feedback and the internal clock.
- To measure the absolute position, we need a reference, a measuring device, that is significantly more accurate than the device tested. In our case, dealing with fractions of microns (0.1 µm and less), even a standard laser interferometer becomes unsatisfactory. For this reason, all factory measurements are made using a number of high precision interferometers, most of them connected to a computerises test station.
- To avoid unnecessary confusion and to more easily understand and troubleshoot a problem, special attention must be paid to avoid bundling discrete errors in one general term. Depending on the application, some discrete errors are not significant. Grouping them in one general parameter will only complicate the understanding of the system performance in certain applications.

4.2.1 Following Error

The Following Error is not a specifications parameter but, because it is at the heart of the servo algorithm calculations and of other parameter definitions, it deserves our attention.

As will be described later in the Control Loops paragraph, a major part of the servo controller’s task is to make sure that the actual motion device follows as close as possible an ideal trajectory in time. You can imagine having an imaginary (ideal) motion device that executes exactly the motion profile you are requesting. In reality, the real motion device will find itself deviating from this ideal trajectory. Since most of the time the real motion device is trailing the ideal one, the instantaneous error is called Following Error.

To summarise, the Following Error is the instantaneous difference between the actual position as reported by the position feedback device and the ideal position, as seen by the controller. A negative following error means that the load is trailing the ideal motion device.

4.2.2 Error

Error has the same definition as the Following Error with the exception that the ideal trajectory is not compared to the position feedback device (encoder) but to an external precision measuring device.

In other words, the Following Error is the instantaneous error perceived by the controller while the Error is the one perceived by the user.

4.2.3 Accuracy

The Accuracy of a system is probably the most common parameter users want to know. Unfortunately, due to its perceived simplicity, it is also the easiest to misinterpret.

The Accuracy is a static measure of a point-to-point positioning error. Starting from a reference point, we command the controller to move a certain distance. When the motion is completed, we measure the actual distance travelled with an external precision measuring device. The difference (the Error) represents the positioning Accuracy for that particular motion.

Because every application is different, we need to know the errors for all possible motions. Since this is practically impossible, an acceptable compromise is to perform the following test.

Starting from one end of the travel, we make small incremental moves and at every stop we record the position Error. We perform this operation for the entire nominal travel. When finished, the Error data is plotted on a graph similar to Fig. 4.2.

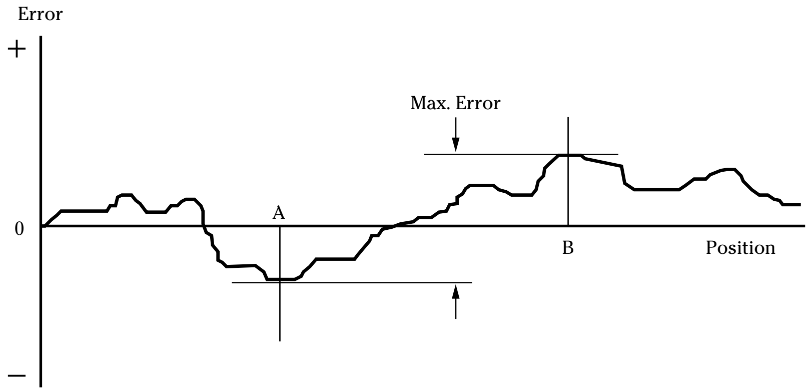


***Fig. 4.2** — Position Error Test.*

The difference between the highest and the lowest points on the graph is the maximum possible Error that the motion device can have. This worst-case number is reported as the positioning Accuracy. It guaranties the user that for any application, the positioning error will not be greater than this value.

4.2.4 Local Accuracy

For some applications, it is important to know not just the positioning Accuracy over the entire travel but also over a small distance. To illustrate this case, Fig. 4.3-a and Fig. 4.3-b show two extreme cases.

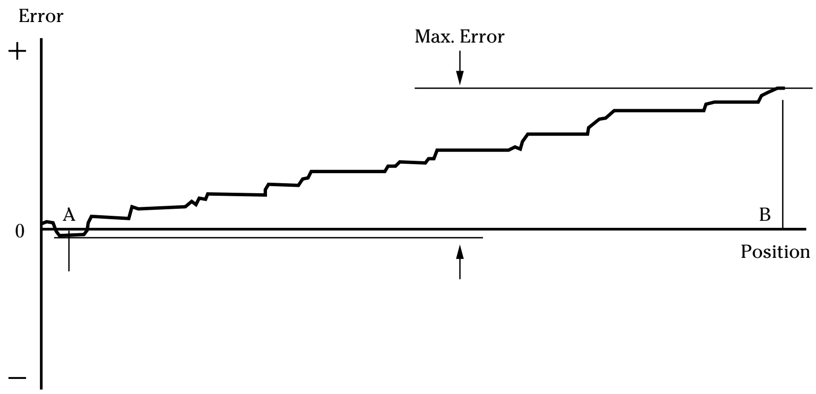


Fig. 4.3-a *— High Accuracy for Small Motions.*

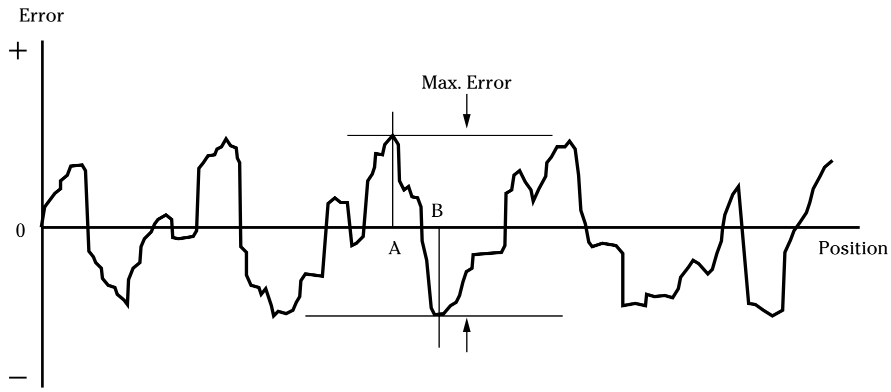


Fig. 4.3-b *— Low Accuracy for Small Motions.*

Both error plots from Fig. 4.3-a and Fig. 4.3-b have a similar maximum Error. But, if you compare the maximum Error for small distances, the system in Fig. 4.3-b shows significantly larger values. For application requires high accuracy for small motions, the system in Fig. 4.3-a is definitely preferred.

"Local Error" is a relative term that depends on the application; usually no Local Error value is given with the system specifications. The user should study the error plot supplied with the motion device and determine the approximate maximum Local Error for the specific application.

4.2.5 Resolution

Resolution is the smallest motion that the controller attempts to make. For all DC motor and all standard stepper motor driven stages supported by the MM4006, this is also the resolution of the encoder.

Keeping in mind that the servo loop is a digital loop, the Resolution can be also viewed as the smallest position increment that the controller can handle.

4.2.6 Minimum Incremental Motion

The Minimum Incremental Motion is the smallest motion that a device can reliably make, measured with an external precision measuring device. The controller can, for instance, execute a motion equal to the Resolution (one encoder count) but in reality, the load may not move at all. The cause for this is in the mechanics.

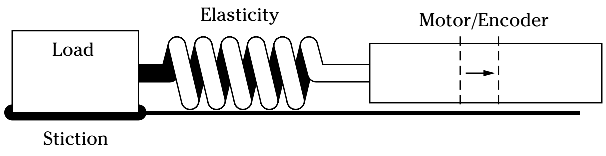


***Fig. 4.4** — Effect of Stiction and Elasticity on Small Motions.*

Fig. 4.4 shows how excessive stiction and elasticity between the encoder and the load can cause the motion device to deviate from ideal motion when executing small motions.

The effect of these two factors has a random nature. Sometimes, for a small motion step of the motor, the load may not move at all. Other times, the accumulated energy in the spring will cause the load to jump a larger distance. The error plot will be similar to Fig. 4.5.

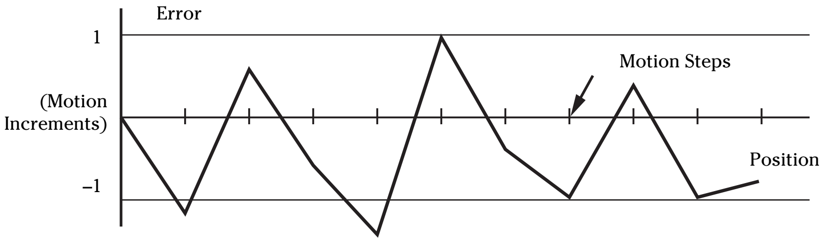


***Fig. 4.5** — Error Plot.*

Once the Minimum Incremental Motion is defined, the next task is to quantify it. This is more difficult for two reasons: one is its random nature and the other is in defining what a completed motion represents.

Assume that we have a motion device with a 1 µm resolution. If every time we command a 1 µm motion the measured error is never greater than 2%, we will probably be very satisfied and declare that the Minimum Incremental Motion is better than 1 µm. If, on the other hand, the measured motion is sometimes as small as 0.1 µm (a 90% error), we could not say that 1 µm is a reliable motion step. The difficulty is in drawing the line between acceptable and unacceptable errors when performing a small motion step. The most common value for the maximum acceptable error for small motions is 20%, but each application ultimately has its own standards.

One way to solve the problem is to take a large number of measurements (a few hundred at minimum) for each motion step size and present them in a format that an operator can use to determine the Minimum Incremental Motion by its own standards.

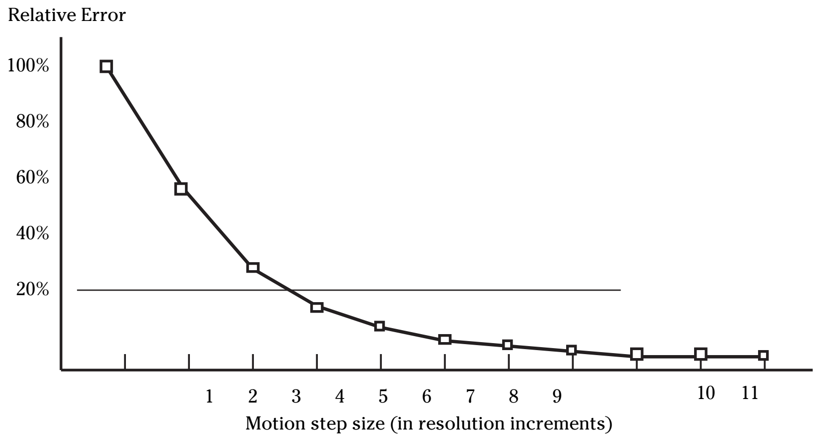


***Fig. 4.6** — Error vs Motion Step Size.*

Fig. 4.6 shows an example of such a plot. The graph represents the maximum relative error for different motion step sizes. In this example, the Minimum Incremental Motion that can be reliably performed with a maximum of 20% error is one equivalent to 4 resolution (encoder) increments.

4.2.7 Repeatability

Repeatability is the positioning variation when executing the same motion profile. Assuming that we have a motion sequence that stops at a number of different locations, the Repeatability is the maximum variation in position all targets when the same motion sequence is repeated a large number of times. It is a relative, not absolute, error between identical motions.

4.2.8 Backlash (Hysteresis)

For all practical purposes, Hysteresis and Backlash have the same meaning for typical motion control systems. The term Hysteresis has an electromagnetic origin while Backlash comes from mechanical engineering. Both describe the same phenomenon: the error caused by approaching a point from a different direction.

All parameters discussed up to now that involve the positioning Error assumed that all motions were performed in the same direction. If we try to measure the positioning error of a certain target (destination), approaching the destination from different directions could make a significant difference.

In generating the plot in Fig. 4.2 we said that the motion device will make a large number of incremental moves, from one end of travel to the other. If we command the motion device to move back and stop at the same locations to take a position error measurement, we would expect to get an identical plot, superimposed on the first one. In reality, the result could be similar to Fig. 4.7.

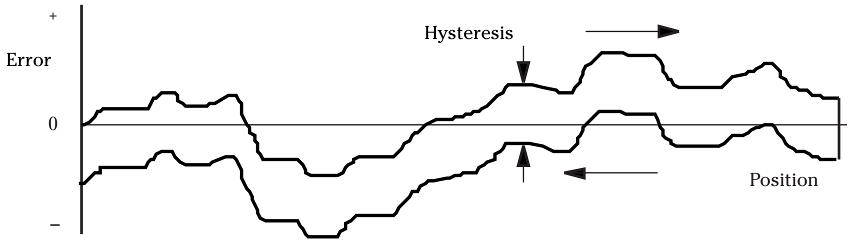


Fig. 4.7 — Hysteresis Plot.

The error plot in reverse direction is identical with the first one but seems to be shifted down by a constant error. This constant error is the Hysteresis of the system.

To justify a little more why we call this error Hysteresis, lets do the same graph in a different format (Fig. 4.8). Plotting the real versus the ideal position will give us a familiar hysteresis shape.

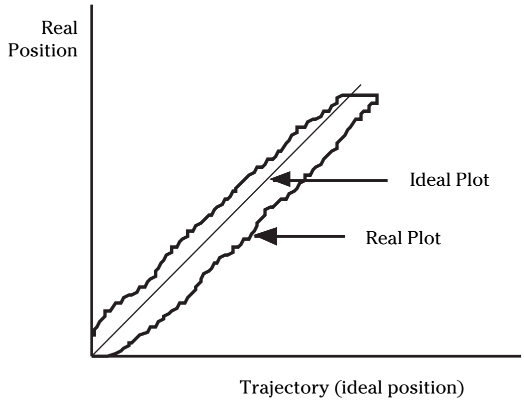


Fig. 4.8 — Real vs Ideal Position.

4.2.9 Pitch, Roll and Yaw

These are the most common angular error parameters for linear translation stages. They are pure mechanical errors and represent the rotational error of a stage carriage around the three axes. A perfect stage should not rotate around any of the axes, thus the Pitch, Roll and Yaw should be zero.

The commonly used representation of the three errors is shown in Fig. 4.9. Pitch is rotation around the Y axis, Roll is rotation around the X axis and Yaw around the Z axis.

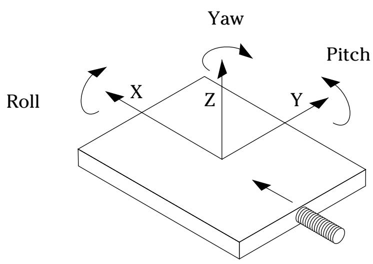


Fig. 4.9 — Pitch, Yaw and Roll Motion Axes.

The problem with this definition is that, though correct, it is difficult to remember. A more graphical representation in presented in Fig. 4.10. Imagine a tiny carriage driven by a giant leadscrew. When the carriage rolls sideways on the lead screw, we call it a Roll. When it rides up and down on the lead screw pitch, we call that Pitch. And, when the carriage deviates left or right from the straight direction (on an imaginary Y trajectory), we call it Yaw.

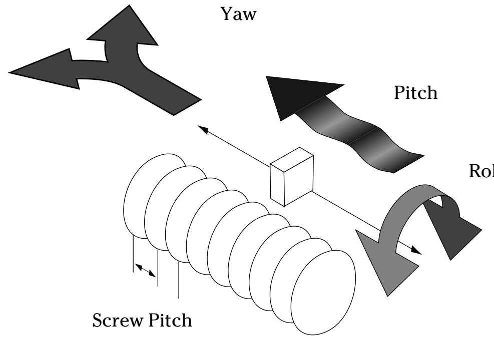


***Fig. 4.10** — Pitch, Yaw and Roll.*

4.2.10 Wobble

This parameter applies only to rotary stages. It represents the deviation of the axis of rotation during motion. A simple form of Wobble is a constant one, where the rotating axis generates a circle (Fig. 4.11).

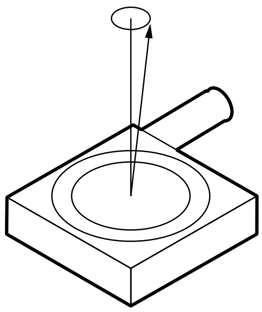


***Fig. 4.11** — Wobble.*

A real rotary stage may have a more complex Wobble, where the axis of rotation follows a complicated trajectory. This type of error is caused by the imperfections of the stage machining and/or ball bearings.

4.2.11 Load Capacity

There are two types of loads that are of interest for motion control applications: static and dynamic loads.

The static Load Capacity represents the amount of load that can be placed on a stage without damaging or excessively deforming it. Determining the Load Capacity of a stage for a particular application is more complicated than it may first appear. The stage orientation and the distance from the load to the carriage play a significant role. For a detailed description on how to calculate the static Load Capacity, please consult the motion control catalogue tutorial section.

The dynamic Load Capacity refers to the motor's effort to move the load. The first parameter to determine is how much load the stage can push or pull. In some cases the two values could be different due to internal mechanical construction.

The second type of dynamic Load Capacity refers to the maximum load that the stage could move with the nominal acceleration. This parameter is more difficult to specify because it involves defining an acceptable following error during acceleration.

4.2.12 Maximum Velocity

The Maximum Velocity that could be used in a motion control system is determined by both motion device and driver. Usually it represents a lower value than the motor or driver are capable of. In most cases and in particular for the MM4006, the default Maximum Velocity should not be increased. The hardware and firmware are tuned for a particular maximum velocity that cannot be exceeded.

4.2.13 Minimum Velocity

The Minimum Velocity usable with a motion device depends on the motion control system but also on the acceptable velocity regulation. First, the controller sets the slowest rate of motion increments it can make. The encoder resolution determines the motion increment size and then, the application sets a limit on the velocity ripple.

To illustrate this, take the example of a linear stage with a resolution of 0.1 µm. If we set the velocity to 0.5 µm/sec, the stage will move 5 encoder counts in one second. But a properly tuned servo loop could move the stage 0.1 µm in about 20 ms. The position and velocity plots are illustrated in Fig. 4.12.

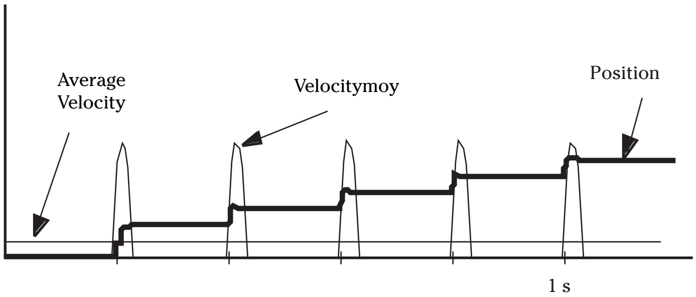


***Fig. 4.12** — Position, Velocity and Average Velocity.*

The average velocity is low but the velocity ripple is very high. Depending on the application, this may be acceptable or not. With increasing velocity, the ripple decreases and the velocity becomes smoother.

This example is even more true in the case of a stepper motor driven stage. The typical noise comes from a very fast transition from one step position to another. The velocity ripple in that case is significantly higher.

In the case of a DC motor, adjusting the PID parameters to get a softer response will reduce the velocity ripple but care must be taken not to negatively affect other desirable motion characteristics.

4.2.14 Velocity Regulation

In some applications, for example scanning, it is important for the velocity to be very constant. In reality, there are a number of factors besides the controller that affect the velocity.

As described in the Minimum Velocity definition, the speed plays a significant role in the amount of ripple generated, specially at low values.

Even if the controller does a perfect job by running with zero following error, imperfections in the mechanics (friction variation, transmission ripple, etc.) will generate some velocity ripple that can be translated to Velocity Regulation problems.

Depending on the specific application, one motor technology can be preferred over the other.

As far as the controller is concerned, the stepper motor version is the ideal case for a good average Velocity Regulation because the motor inherently follows precisely the desired trajectory. The only problem is the ripple caused by the actual stepping process.

The best a DC motor controller can do is to approach the stepper motor's performance in average Velocity Regulation, but it has the advantage of significantly reduced velocity ripple, inherently and through PID tuning. If the DC motor driver implements a velocity closed loop through the use of a tachometer, the overall servo performance increases and one of the biggest beneficiary is the Velocity Regulation. Usually only higher end motion control systems use this technology and the MM4006 is one of them. Since having a real tachometer is very expensive and in some cases close to impossible to implement, the MM4006 can both use or simulate a tachometer through special circuitry and obtains the same result.

4.2.15 Maximum Acceleration

The Maximum Acceleration is a complex parameter that depends as much on the motion control system as it does on application requirements. For stepper motors, the main concern is not to loose steps (or synchronisation) during the acceleration. Besides the motor and driver performance, the load inertia plays a significant role.

For DC motor systems the situation is different. If the size of the following error is of no concern during the acceleration, high Maximum Acceleration values can be entered. The motion device will move with the highest natural acceleration it can (determined by the motor, driver, load inertia, etc.) and the errors will be just a temporary larger following error and a velocity overshoot.

In any case, special consideration should be given when setting the acceleration. Though in most cases no harm will be done in setting a high acceleration value, avoid doing so if the application does not require it. The driver, motor, motion device and load undergo maximum stress during high acceleration.

4.2.16 Combined Parameters

Very often a user looks at an application and concludes that he needs a certain overall accuracy. This usually means that he is combining a number of individual terms (error parameters) into a single one. Some of these combined parameters even have their own name, even though not all people mean the same thing by them: Absolute Accuracy, Bi-directional Repeatability, etc. The problem with these generalisations is that, unless the term is well defined and the testing closely simulates the application, the numbers could be of little value.

The best approach is to carefully study the application, extract from the specification sheet the applicable discrete error parameters and combine them (usually add them) to get the worst-case general error applicable to the specific case. This method not only offers a more accurate value but also gives a better understanding of the motion control system performance and helps pinpoint problems.

Also, due to the integrated nature of the MM4006 system, many basic errors can be significantly corrected by an other component of the loop. Backlash, Accuracy and Velocity Regulation are just a few examples where the controller can improve motion device performance.

4.3 Control Loops

When talking about motion control systems, one of the most important questions is the type of servo loop implemented. The first major distinction is between open and closed loops. Of course, this is of particular interest when driving stepper motors. As far as the DC servo loops, the PID type is by far the most widely used.

The MM4006 implements a PID servo loop with velocity feed-forward for both DC and stepper-motor motion devices. It is not just a static closed loop, when the motion is stopped, but a fully dynamic one.

The basic diagram of a servo loop is shown in Fig. 4.13. Besides the command interpreter, the main two parts of a motion controller are the trajectory generator and the servo controller. The first generates the desired trajectory and the second one controls the motor to follow it as closely as possible.

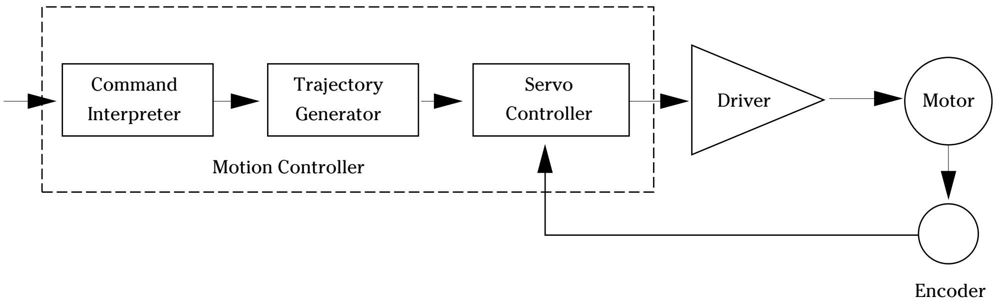


***Fig. 4.13** — Servo Loop.*

4.3.1 PID Servo Loops

The PID term comes from the proportional, integral and derivative gain factors that are at the basis of the control loop calculation. The common equation given for it is:

$$K_p \bullet e + K_i \int e\,dt + K_d \bullet \frac{de}{dt}$$

where K_p = Proportional gain factor.

K_i = integral gain factor.

K_d = derivative gain factor.

e = instantaneous following error.

The problem for most users is to get a feeling for this formula, specially when trying to tune the PID loop. Tuning the PID means changing its three gain factors to obtain a certain system response, task quite difficult to achieve without some understanding of its behaviour.

The following paragraphs explain the PID components and their operation.

P Loop

Lets start with the simplest type of closed loop, the P (proportional) loop. The diagram in Fig. 4.14 shows its configuration.

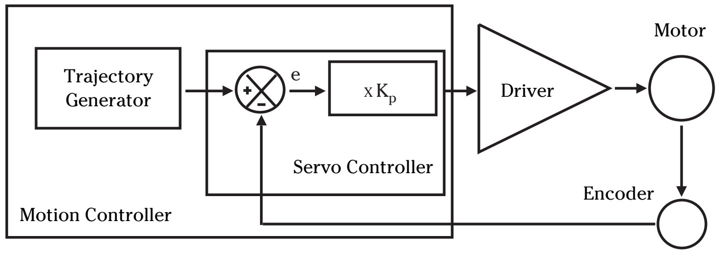


Fig. 4.14 *— P Loop.*

Every servo cycle, the actual position, as reported by the encoder, is compared to the desired position generated by the trajectory generator. The difference e is the positioning error (the following error). Amplifying it (multiplying it by K_p) generates a control signal that, converted to an analog signal, is sent to the motor driver.

There are a few conclusions that could be drawn from studying this circuit:

- The motor control signal, thus the motor voltage, is proportional to the following error.
- There must be a following error in order to drive the motor.
- Higher velocities need higher motor voltages and thus higher following errors.
- At stop, small errors cannot be corrected if they don’t generate enough voltage for the motor to overcome friction and stiction.
- Increasing the K_p gain reduces the necessary following error but too much of it will generate instabilities and oscillations.

PI Loop

To eliminate the error at stop and during long constant velocity motions (usually called steady-state error), an integral term can be added to the loop. This term integrates (adds) the error every servo cycle and the value, multiplied by the K_i gain factor, is added to the control signal (Fig. 4.15).

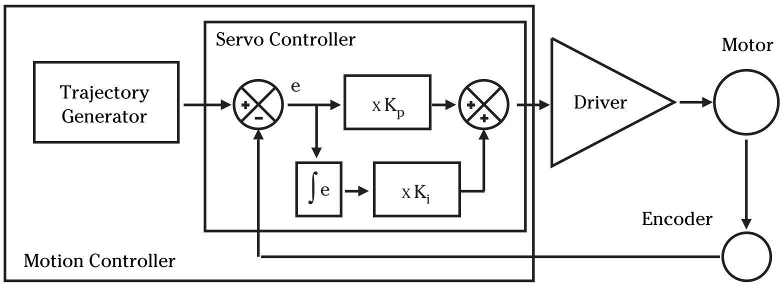


***Fig. 4.15** — PI Loop.*

The result is that the integral term will increase until it drives the motor by itself, reducing the following error to zero. At stop, this has the very desirable effect of driving the positioning error to zero. During a long constant-velocity motion it also brings the following error to zero, an important feature for some applications.

Unfortunately, the integral term also has a negative side, a severe de-stabilising effect on the servo loop. In the real world, a simple PI loop is usually undesirable.

PID Loop

The third term of the PID loop is the derivative term. It is defined as the difference between the following error of the current servo cycle and of the previous one. If the following error does not change, the derivative term is zero.

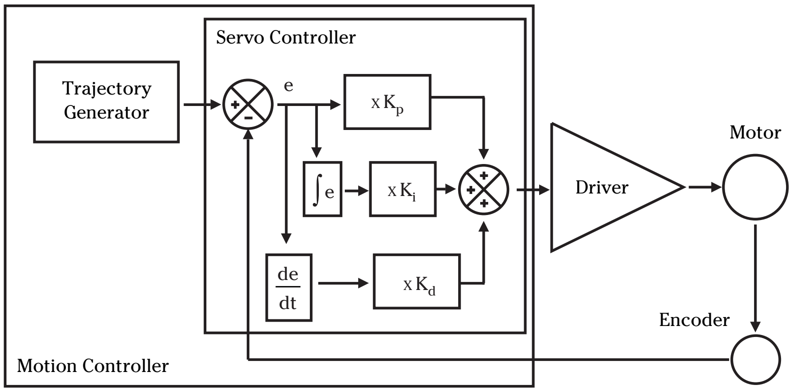


***Fig. 4.16** — PID Loop.*

Fig. 4.16 shows the PID servo loop diagram. The derivative term is added to the proportional and integral one. All three process the following error in their own way and, added together, form the control signal.

The derivative term adds a damping effect which prevents oscillations and position overshoot.

4.3.2 Feed-Forward Loops

As described in the previous paragraph, the main driving force in a PID loop is the proportional term. The other two correct static and dynamic errors associated with the closed loop.

Taking a closer look at the desired and actual motion parameters and at the characteristics of the DC motors, some interesting observations can be made. For a constant load, the velocity of a DC motor is approximately proportional with the voltage. This means that for a trapezoidal velocity profile, for instance, the motor voltage will have also a trapezoidal shape (Fig. 4.17).

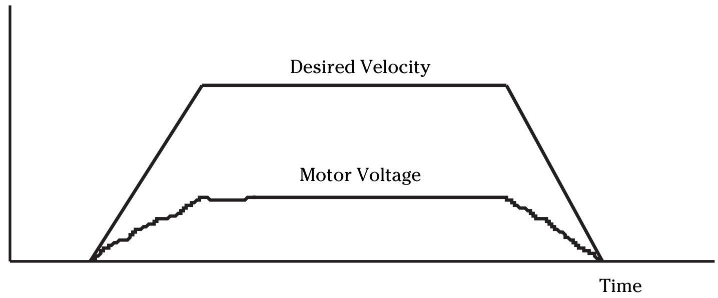


***Fig. 4.17** — Trapezoidal Velocity Profile.*

The second observation is that the desired velocity is calculated by the trajectory generator and is known ahead of time. The obvious conclusion is that we could take this velocity information, scale it by a Kvff factor and feed it to the motor driver. If the scaling is done properly, the right amount of voltage is sent to the motor to get the desired velocities, without the need for a closed loop. Because the signal is derived from the velocity profile and it is being sent directly to motor driver, the procedure is called velocity feed-forward.

Of course, this looks like an open loop, and it is (Fig. 4.18). But, adding this signal to the closed loop has the effect of significantly reducing the “work” the PID has to do, thus reducing the overall following error. The PID now has to correct only for the residual error left over by the feed-forward signal.

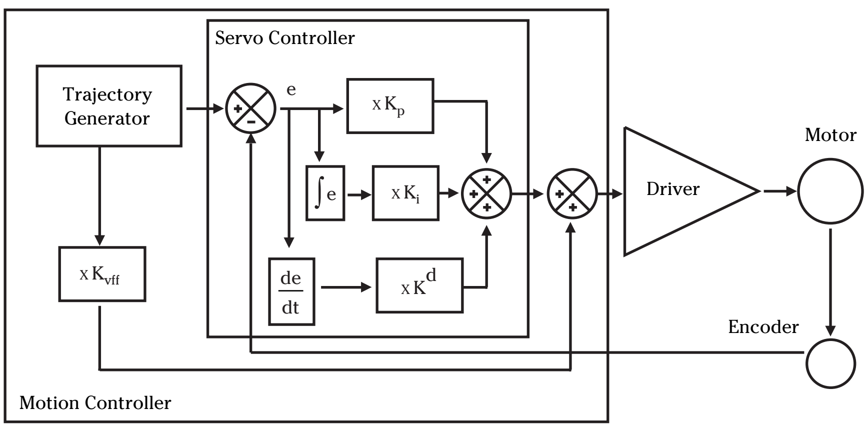


***Fig. 4.18** — PID Loop with Feed-Forward.*

There is an other special note that has to be made about the feed-forward method. The velocity is approximately proportional to the voltage and only for constant loads. but this is true only if the driver is a simple voltage amplifier or current (torque) driver. A special case is when the driver has its own velocity feedback loop from a tachometer (Fig. 4.19).

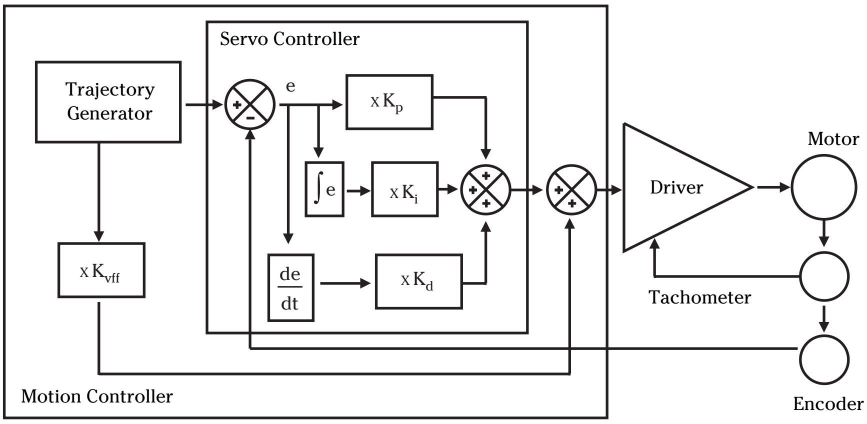


***Fig. 4.19** — Tachometer-driven PIDF Loop.*

The tachometer is a device that outputs a voltage proportional with the velocity. Using its signal, the driver can maintain the velocity to be proportional to the control signal. If such a driver is used with a velocity feed-forward algorithm, by properly tuning the Kvff parameter, the feed-forward signal could perform an excellent job, leaving very little for the PID loop to do.

The MM4006 uses this type of velocity control driver to get the highest performance possible. In addition, since tachometers are expensive and many times impractical or even impossible to use, the driver has a special circuitry tuned to each individual motor type that can “calculate” the velocity. The results are similar to a tachometer feedback but at a fraction of a cost. The drawback is that each motor type needs a specially tuned driver card but, because it is designed to work in a pre-defined system using known motion devices, its operation is totally transparent to the user. All driver cards are pre-tuned and clearly labelled and no adjustments are required (or allowed).

4.4 Motion Profiles

When talking about motion commands we refer to certain strings sent to a motion controller that will initiate a certain action, usually a motion. There are a number of common motion commands which are identified by name. The following paragraphs describe a few of them.

4.4.1 Move

A move is a point-to-point motion. On execution of a move motion command, the motion device moves from the current position to a desired destination. The destination can be specified either as an absolute position or as a relative distance from the current position.

When executing a move command, the motion device will accelerate until the velocity reaches a pre-defined value. Then, at the proper time, it will start decelerating so that when the motor stops, the device is at the correct position. The velocity plot of this type of motion will have a trapezoidal shape (Fig. 4.20). For this reason, this type of motion is called a trapezoidal motion.

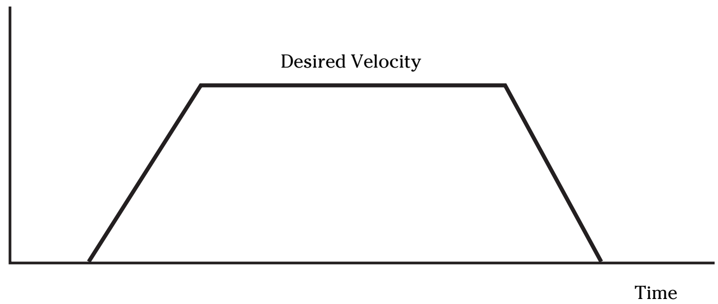


***Fig. 4.20** — Trapezoidal Motion Profile.*

The position and acceleration profiles relative to the velocity are shown in Fig. 4.21.

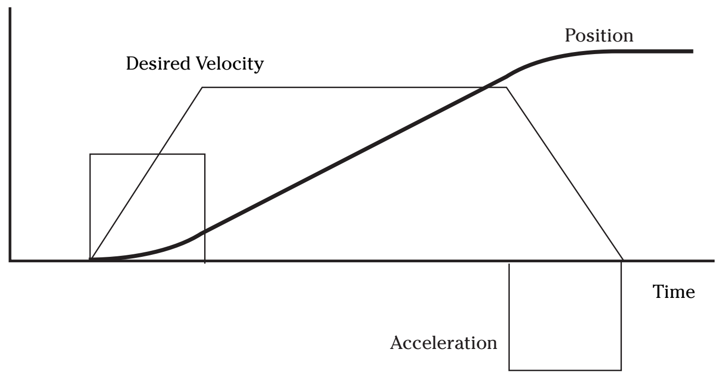


***Fig. 4.21** — Position and Acceleration Profiles.*

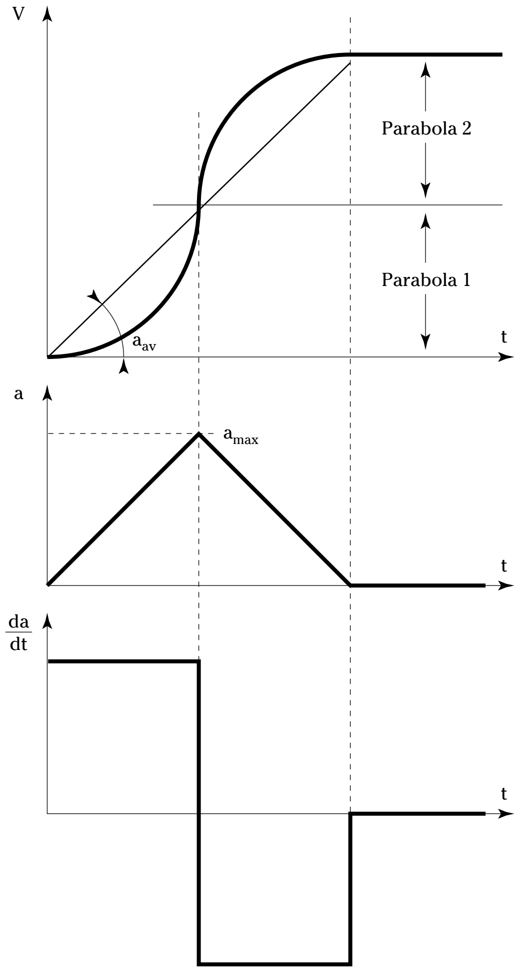
Besides the destination, the acceleration and the velocity of the motion (the constant portion of it) can be set by the user before every move command. Advanced controllers like the MM4006 allow the user to change them even during the motion. However, the MM4006 always verifies that a parameter change can be safely performed. If not, the command is ignored and the motion continues as initially defined.

4.4.2 Default Mode: S-CURVE Profile

Two types of profiles exist: TRAPEZOID and S-CURVE.

TRAPEZOID or S-CURVE

The Scurve type avoids abrupt variations in speed during axis movement, consequently it improves the movement quality.



a: Temporary acceleration

a_{max}: Maximum acceleration

a_{av}: Average acceleration value set from the front panel of the controller (Axis Setup → MODIFY → Acceleration menu)

$$a_{av} = a_{max}/2$$

Advantage

Gives smooth acceleration in the acceleration phase (start) and in deceleration (stop), thereby avoiding severe jolts to the mechanisms during these events (start/end).

4.4.3 Jog

When setting up an application, it is often necessary to move a devices manually while observing motion. The easy way to do this without resorting to specialised input devices such as joysticks or track-wheels is to use simple push-button switches. This type of motion is called a jog. When a jog button is pressed the selected axis starts moving with a pre-defined velocity. The motion continues only while the button is pressed and stops immediately after its release.

The MM4006 offers two jog speeds. The high speed is programmable and the low speed is ten times smaller. The jog acceleration is also ten times smaller than the programmed maximum acceleration values.

4.4.4 Home Search

Home search is a specific motion routine that is useful for most types of applications. Its goal is to find a specific point in travel relative to the mounting base of the motion device very accurately and repeatably. The need for this absolute reference point is twofold. First, in many applications it is important to know the exact position in space, even after a power-off cycle. Secondly, to protect the motion device from hitting a travel obstruction set by the application (or its own travel limits), the controller uses programmable software limits. To be efficient though, the software limits must be placed accurately in space before running the application.

To achieve this precise position referencing, the MM4006 motion control system executes a unique sequence of moves.

First, lets look at the hardware required to determine the position of a motion device. The most common (and the one supported by the MM4006) are incremental encoders. By definition, these are encoders that can tell only relative moves, not absolute position. The controller keeps track of position by incrementing or decrementing a dedicated counter according to the information received from the encoder. Since there is no absolute position information, position “zero” is where the controller was powered on (and the position counter reset).

To determine an absolute position, the controller must find a “switch” that is unique to the entire travel, called a home switch or origin switch. An important requisition is that this switch must be located with the same accuracy as the encoder pulses. If the motion device is using a linear scale as position encoder, the home switch is usually placed on the same scale and read with the same accuracy.

If, on the other hand, a rotary encoder is used, the problem becomes more complicated. To have the same accuracy, a mark on the encoder disk could be used (called index pulse) but because it repeats itself every revolution, it does not define a unique point over the entire travel. An origin switch, on the other hand, placed in the travel of the motion device is unique but not accurate (repeatable) enough. The solution is to use both, following a search algorithm.

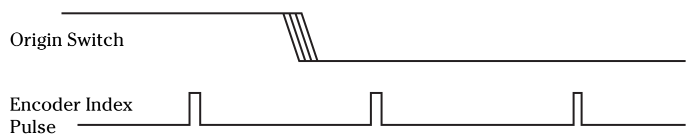


***Fig. 4.22** — Origin switch and encoder index pulse.*

An origin switch (Fig. 4.22) separates the entire travel in two areas: one for which it has a high level and one for which it is low. The most important part of it is the transition between the two areas. Also, looking at the origin switch level, the controller knows on which side of the transition it currently is and which way to move to find it.

The task of the home search routine is to identify one unique index pulse as the absolute position reference. This is done by first finding the origin switch transition and then the very first index pulse (Fig. 4.23).

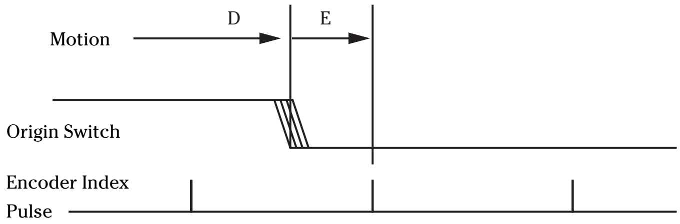


***Fig. 4.23** — Slow-Speed Origin Switch Search.*

So far, we can label the two motion segments D and E. During D the controller is looking for the origin switch transition and during E for the index pulse. To guarantee the best accuracy possible, both D and E segments are performed at a very low speed and without a stop in-between. Also, during E the display update is suppressed to eliminate any unnecessary overhead.

The routine described above could work but has one problem. Using the low speeds, it could take a very long time if the motion device happens to start from the opposite end of travel. To speed things up, we can have the motion device move fast in the vicinity of the origin switch and then perform the two slow motions, D and E. The new sequence is shown in Fig. 4.24.

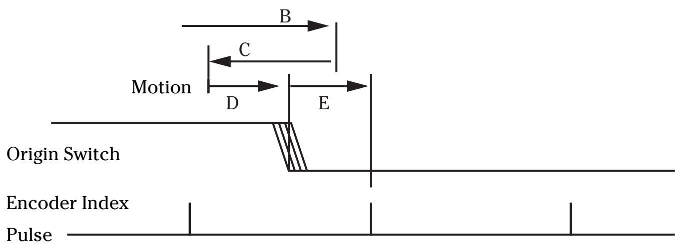


***Fig. 4.24** — High/Low-Speed Origin Switch Search.*

Motion segment B is performed at high speed, with the pre-programmed home search speed. When the origin switch transition is encountered, the motion device stops (with an overshoot), reverses direction and looks for it again, this time with half the velocity (segment C). Once found, it stops again with an overshoot, reverses direction and executes D and E with one tenth of the programmed home search speed.

In the case when the motion device starts from the other side of the origin switch transition, the routine will look like Fig. 4.25.

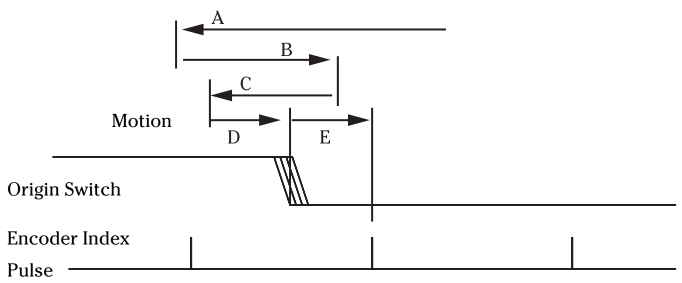


***Fig. 4.25** — Origin Search From Opposite Direction.*

The MM4006 moves at high speed up to the origin switch transition (segment A) and then execute B, C, D and E.

All home search routines are run so that the last segment, E, is performed in the positive direction of travel.

CAUTION

The home search routine is a very important procedure for the positioning accuracy of the entire system and it requires full attention from the controller. Do not interrupt or send other commands during its execution, unless it is for emergency purposes.

Section 4

Motion Control Tutorial

Table of Contents
Section 5 — Trajectory Functions Tutorial

Section 5
Trajectory Functions Tutorial

5.1 Definition of Terms

5.1.1 Trajectory

A continuous multi-dimensional motion path. In the MM4006 case, the trajectory is defined in a two-dimensional X-Y plane. The major requirement in executing a trajectory is to maintain a constant vector velocity throughout the entire path, with the exception of the acceleration and deceleration periods.

5.1.2 Trajectory Element

A segment of a trajectory that can be defined by a simple geometric shape, in our case a **line** or an **arc** of circle.

5.1.3 Trajectory Vector

The tangent to the trajectory in any particular point.

5.1.4 Vector Velocity

The linear velocity (the speed) along the trajectory during its execution.

5.1.5 Vector Acceleration

The tangential linear acceleration used to start and end a trajectory. (Acceleration and deceleration are equal by default).

5.2 Trajectory Description and Conventions

When defining and executing a trajectory, a number of rules must be followed. For the current MM4006 version, these are the conventions that guide the contouring implementation:

- Multiple trajectories can be defined in a program but only one is active at a time. This means that the controller can have only one trajectory ready to be executed.
- Once one trajectory is started, it executes in background allowing the other axes and peripherals to work independently and simultaneously.
- Each trajectory must have a beginning and an end. “Endless” (infinite) trajectories are not allowed.
- The size of a trajectory is limited to 100 trajectory elements. This value is arbitrary and should satisfy most complex applications.
- The trajectory definition process must ensure a continuous motion path to avoid any excessive accelerations and shocks that could damage the stages.
- The line segments are true linear interpolations:

$$y = Ax + B$$

- The arc segments must be true arc of circles:

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

- A trajectory is always defined relative to the pre-defined stage units. To avoid confusion, it is recommended to use same units of displacement on both axes.
- Each trajectory is defined relative to its starting point. Thus, every starting point has the coordinates 0, 0.
- All trajectories start executing from the current X and Y positions. To execute a trajectory from a desired location, the two axes corresponding to X and Y must be moved using the standard point-to-point commands (PA, PR).
- Before executing a trajectory, the controller verifies if its definition does infringe on any pre-defined motion rules (excessive tangent discontinuity, excessive acceleration, travel limits).
- Trajectories can be defined in both IMMediate and PRoGram mode.

5.3 Geometric Conventions

- The coordinate system is an X-Y orthogonal system.
- Any valid motion axis can be assigned to be the X or Y axis.
- After executing a trajectory, new axes can be assigned to X or Y axis.
- The origin of the X-Y coordinate system is in the lower left corner, with positive values up and to the right.
- All angles are measured in degrees, represented as floating points numbers.
- Angle origin and sign follow the trigonometric convention: positive angles are measured counter-clockwise.

5.4 Defining Trajectory Elements

Trajectories can be defined in many different ways. There is no universal standard and most manufacturers of motion controllers use some degree of custom conventions. For the MM4006, the guiding principal was to be as user friendly as possible. Line and arc elements can be defined in more than one way to offer the best solution for each application. The elements are “seamed” together automatically and the entire trajectory is verified before execution to guarantee its definition conforms to all rules.

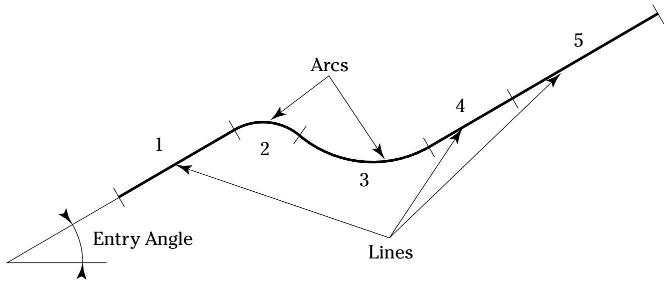


***Fig. 5.1** — Trajectory example.*

Figure 5.1 shows a trajectory example. Every trajectory must have an “entry angle” defined. If the first element is an arc of circle, the entry angle is the tangent to the first point of the arc.

Each element defined is identified by a number, starting from 1. The references for synchronising external events with the trajectory execution are the starting and ending points of these elements.

Line and arc elements can be sequenced in any order. Arcs can be followed by arcs or lines and lines by arcs or other lines. An arc is automatically placed by the controller such that its “entry angle” corresponds to the “exit angle” of the preceding element to insure the continuity of the trajectory. But, when defining a line by its X-Y end point, this responsibility falls on the user. The end coordinates of the new line must be chosen such that the angle it defines is identical to the “exit angle” of the previous trajectory element. Since we are dealing with a coordinate system with finite resolution - the encoder resolution - getting a perfect match of the two angles is not always possible. For this reason, a window of acceptable angle mismatch is defined, called “maximum angle discontinuity”. This new parameter is measured in degrees and has a range of 0.001° to 10°. A trajectory can thus theoretically be build out of straight lines that have less than 10° angle difference, as shown in Figure 5.2.

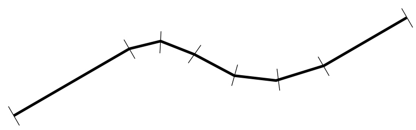


***Fig. 5.2** — Contouring with lines only.*

This practice is not recommended since each angle of discontinuity corresponds to an instantaneous velocity change on both axes, which represents an infinite acceleration. The result is a shock (jerk) felt by the stages and the load and a temporary following error pulse. The larger the angle of discontinuity, the larger the jerk and the following error will be. Special consideration must be given to both of these effects when increasing the maximum discontinuity angle from its 0.001° default value.

To eliminate the burden of calculating the angle matching, use as much as possible the commands that define a straight line by one coordinate, X or Y, and by the entry angle (also referred to as the tangent). This simplifies the user's programming task and lets the controller find the best fit for the trajectory elements.

5.4.1 Defining Lines

There are two ways to define a line of a trajectory. The first one is to specify the X-Y end coordinates (the starting point is always the end point of the previous element). This is the most common procedure found in the industry (Fig. 5.3).

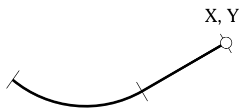


***Fig. 5.3** — Line to X-Y.*

As described previously, when using this method the user must make an extra effort in making sure the maximum discontinuity angle is not exceeded.

A second mode of defining a straight line in a trajectory is illustrated in Figure 5.4.

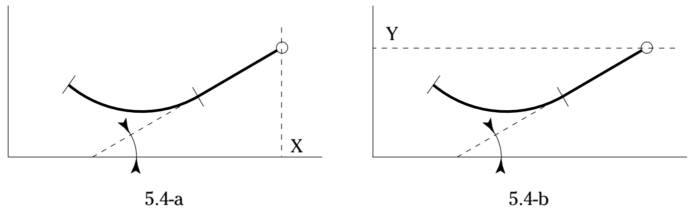


***Fig. 5.4** — Line to X or Y.*

Using the previous element's exit angle (tangent), the controller can execute a line to the specified X coordinate (Fig. 5.4-a) or Y coordinate (Fig. 5.4-b). This method simplifies the programming job and guaranties the best trajectory elements fit.

5.4.2 Defining Arcs

Arcs can also be defined in two different ways. The first one is more conventional, where a radius and the sweep angle will define the arc (Fig 5.5).

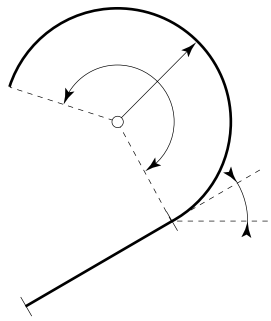


***Fig. 5.5** — Arc defined with radius and angle.*

Both radius and sweep angle are expressed in double precision floating point numbers. It is particularly interesting to mention that the sweep angle has a range of 1E-12 to 1.7E304, allowing execution of arcs from a fraction of a degree to a practically infinite number of overlapping circles.

The second method of defining an arc is to specify the X-Y coordinates of the end point. Using the exit angle of the previous element, the controller will determine the unique arc that fits the parameters (Fig. 5.6).

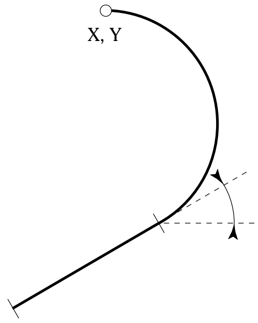


Fig. 5.6 *— Arc defined with end point.*

This automatic mode of describing an arc can simplify the process of geometrically defining a trajectory, significantly reducing the programming time.

A particular application is in approximating an irregular path (Fig. 5.7).

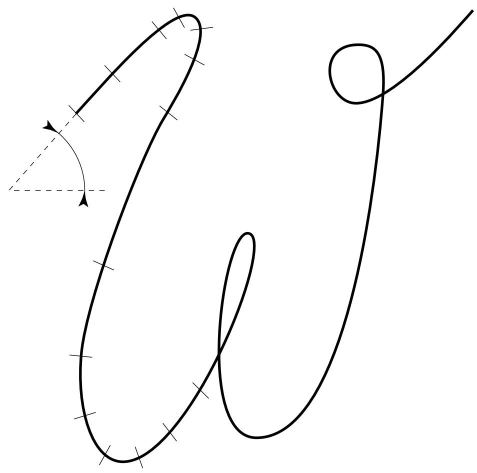


Fig. 5.7 *— Contouring with arcs.*

By specifying an entry angle and a number of X-Y coordinates, the controller will automatically perform a circular interpolation that closely approximates the desired trajectory.

5.5 Programming a Trajectory

The following list describes the few rules that govern the trajectory programming process and gives some examples:

- A trajectory must be first defined and then executed.

NT	*Start new trajectory definition.*
…	*Define trajectory.*
…	
ET	*Execute trajectory.*

- Trajectory definition commands and other controller commands can be intermixed. Even though the controller will extract the appropriate commands to build the trajectory, for the clarity of the program this practice is not recommended.

NT	*Start new trajectory definition.*
…	*Define trajectory.*
…	
…	*Other commands.*
…	
…	*Define trajectory.*
…	
…	*Other commands.*
…	
ET	*Execute trajectory.*

- A trajectory can be defined once and executed any number of times. To allow this feature, all trajectories are defined relative to the starting point.

NT	*Start new trajectory definition.*
…	*Define trajectory.*
…	
ET	*Execute trajectory.*
…	
xxPAnn	*Move to new trajectory start location.*
ET	*Execute trajectory.*

- During the trajectory execution, the designated axes are unavailable for point-to-point commands.

- Once a trajectory is defined, it can be edited by deleting the last element and inserting or appending new elements.

NT	*Start new trajectory definition.*
…	*Define trajectory.*
…	
ET	*Execute trajectory.*
…	
EL	*Erase last trajectory element.*
LXnn	*Add new trajectory element (line to nn).*
xxPAnn	*Move to new trajectory start location.*
ET	*Execute same trajectory at new location.*

- When defining a trajectory, start by assigning the two motion axes to the X and Y coordinates. These axis can still be used when the trajectory is not executing.

NT	*Start new trajectory definition.*
2AX	*Assign axis #2 to the X coordinate.*
4AY	*Assign axis #4 to the Y coordinate.*
…	*Define trajectory.*
…	
xxPAnn	*Move axis #2 to absolute position 10.*
ET	*Execute trajectory.*

- Before executing a trajectory, the controller verifies, among other things, if the defined geometry will cause, at any time, any axes to exceed the individual maximum allowed velocities or accelerations. If so, it will calculate the highest acceptable vector velocity and use it during the execution. The actual vector velocity that will be used can be queried remotely.

NT	*Start new trajectory definition.*
VV20	*Set vector velocity to 20 units/s.*
…	*Define trajectory.*
…	
ET	*Execute trajectory.*
XV	*Read actual vector velocity in use.*
XV12.736	*Controller returns actual vector velocity used.*

- Only one trajectory can be defined and be active at a time. NT command erases any old trajectory and starts defining a new one. Any new trajectory-specific command will be added or affect the existing defined trajectory and will be active at the next execution.

5.6 Trajectory Element Parameters

Both line and arc parameters can be entered using different commands. Most differences are in the type and number of parameters used to describe one trajectory element. Many commands require only the minimum number of commands that geometrically define one element.

The controller always calculates and keeps track of all element parameters. Using the LT command, the complete set of trajectory element parameters can be listed. This is an excellent tool in developing and debugging complex trajectories.

5.7 Trajectory-Specific Commands

These are the commands needed to support the contouring feature. They are fully compatible with the general description of all other commands and will follow the same protocol.

5.7.1 Trajectory Setup Commands

	AD nn	Define the maximum allowed angle of discontinuity.
xx	**AX**	Assign a physical axis as X geometric axis.
xx	**AY**	Assign a physical axis as Y geometric axis.
	FA nn	Define the tangent angle for the first point.
	NT	Start definition of a new trajectory.

5.7.2 Trajectory Elements Definition Commands

CA nn	Define sweep angle and build an arc of circle = *f* (CR, CA).
CR nn	Define radius for an arc of circle = *f* (CR, CA).
CX nn	Define X position to reach with an arc of circle = *f* (CX, CY).
CY nn	Define Y position to reach and build an arc of circle = *f* (CX, CY).
EL	Erase the last element of trajectory.
LX nn	Define X position and build a line segment = *f* (LX, tangent).
LY nn	Define Y position and build a line segment = *f* (LY, tangent).
MX nn	Define X position for a line segment = *f* (MX, MY).
MY nn	Define Y position and build a line segment = *f* (MX, MY).

5.7.3 Reporting Commands

	AT	Tell the element number under execution.
xx	**LT**	Extended list of the trajectory.
	XA	Tell the current maximum allowed angle of discontinuity.
	XE	Tell the current element.
	XT	Tell number of elements in the trajectory.
	XU nn	Tell the vector acceleration on trajectory (trajectory acceleration).
	XV nn	Tell the vector velocity on trajectory (trajectory velocity).

5.7.4 Trajectory Synchronisation Commands

NB nn	Set trajectory element where the generation of pulses starts.
NE nn	Set trajectory element where the generation of pulses ends.
NI nn	Set step (curvi-linear distance) between synchronisation pulses.
NN nn	Set number of synchronisation pulses to generate.
NS	Allow generation of trajectory.
WI nn	Wait for a trajectory (curvi-linear) length.
WN nn	Wait for a element of trajectory.

5.7.5 Execution of a Trajectory

EL	Erase the last element of trajectory.
ET	Execution of trajectory.
VS nn	Define the vector acceleration on trajectory (trajectory acceleration).
VV nn	Define the vector velocity on trajectory (trajectory velocity).

Section 5

Trajectory Functions Tutorial

Table of Contents
Section 6 — Feature Descriptions Tutorial

Section 6
Feature Descriptions Tutorial

6.1 Synchronizing Events to Motion

6.1.1 Pulses Synchronized to One Axis

Certain applications require an output pulse (≈ 5 μsec) synchronized with the motion of an axis. This signal is triggered not by a timer but by the specified axis crossing a pre-defined position. It is available on the Auxiliary connector (25-pin D-Sub) at pin 11.

Four commands are available to define and use this signal.

xx **PB** nn		Defining and reading the signal starting position.
xx **PE** nn		Defining and reading the signal ending position.
xx **PI** nn		Defining and reading the step of the synchronizing signal.
xx **PS**		Start the signal generation.

Where:

xx — Axis number (from 1 to 4).

nn — Position (absolute), in displacement units.

The PB, PE and PI commands define the synchronized signal while the PS initiates it. The best location for the PS command is just before PA or PR. The signal is terminated at the position indicated by PE or at the end of the motion when the effect of the PS command is canceled.

The necessary conditions for generating the signal, verified by PS, are:

- The start position defined by PB must be reached before the stop position defined by PE.
- The value set with PI must be greater or equal to the smallest servo step. This is the displacement made by an axis during one servo sampling period: smallest servo step = velocity * T_{base} where T_{base} = 0.25 or 0.3 msec, depending on the processor. If not, PS will not generate a signal and will return an error. In this case, the smallest servo step can be determined by using the command xxPI?. If PS command is executed successfully, xxPI? will return the previously entered value, not the smallest servo step.

If any of these conditions is not satisfied, PS command is not executed and it will return an error code.

NOTE

To know the exact value of the servo sampling period (T_{base}) of the controller, send the command SQ0 and the, the command SQ?. The controller will return SQ value. This value will be the exact sampling period of the controller, in seconds.

NOTE

The starting position and the ending position of the axis must be outside the interval defined with PB and PE commands without forget acceleration and deceleration ranges.

Example

Generate a signal synchronized by axis 1. It should start when the axis crosses position -10 (current units) and end when it reaches position 10. The step should be 1 (current units).

1PB-10, 1PE10, 1PI1	*Defining the signal.*
1PA-20	*Motion without any signal generation.*
1PS, 1PR40	*Motion with signal generation between position -10 and 10, with a step of 1 units.*
TT	*Position reading for each pulse generated.*
1PR60	*Motion without any signal generation.*

NOTE

During a motion with synchronized signal generation, the real and theoretical position corresponding to each pulse is recorded in the position buffer. This information can be read back using the TT command. To make the buffer available and avoid any conflicts, the PS command terminates and clears any settings made by a TM command. To use the trace mode after a motion with synchronized signals, the TM command must be re-initiated.

A pulse is generated when the selected axis reaches the specified position, as follows:

$$\text{Axis position} = PB + (n * PI), \text{ where Axis position} \in [PB\ldots PE].$$

The position accuracy of the generated pulse (the difference between the theoretical position PB + n * PI and the real position) depends on the velocity of the selected axis as follows:

$$MaxError = \pm(T_{base} * \text{Velocity})/2$$

Obviously, this value cannot be smaller than the encoder resolution.

Thus, if the calculated MaxError < Encoder resolution,

then MaxError = Encoder resolution.

Example

For a velocity of 20 mm/sec and T_{base} = 300 µsec:

$$MaxError = \pm (3 * 10^{-4} * 20)/2 = \pm 3\ \mu m$$

This pulse location uncertainty will exist for every pulse generated but it is not cumulative.

NOTE

The pulse location uncertainty is no longer significant if the pulse interval (PI) is set to greater than ten times this error. Thus, the recommended value for PI is:

$$\mathbf{PI_{min\ recommended} = 5 * T_{base} * Velocity}$$

6.1.2 Pulses Synchronized to a Trajectory

Equally spaced pulses can also be generated synchronous with a trajectory (at pin 12 of the 25-pin D-Sub Auxiliary connector). The NB, NE, NI, NN and NS commands are used as follows:

NB nn Defining and reading the trajectory element number where the signal should start. The pulses are generated immediately when this element starts executing.

NE nn Defining and reading the trajectory element number where the signal should stop. The pulses will stop immediately when this element finishes executing.

NI nn Defining and reading the distance (the step) on the trajectory between synchronizing pulses.

NN nn Defining and reading the number of pulses (number of steps + 1) that are generated in a symetric geometric fashion.

NS Start generating the signal.

The NB, NE, NI and NN commands provide the necessary data to define the signal generation while the NS command enables it. The correct location for the NS command is right before the ET command. The pulses are terminated at the location specified by the NE command or at the end of the trajectory where the NS command's effect ends automatically.

The necessary conditions (verified by ET) to generate the signal are:

- Values defined by NB and NE must be less than or equal to the total number of trajectory elements.
- The value of NB must be less than that of NE.
- The number of pulses to generate must be greater than 2 and less than or equal to the Maximum Pulse Number. If not, ET will replace the desired number of pulses with the Maximum Pulse Number and return the appropriate error code. In this case, the Maximum Pulse Number value can be read by the NI? command. If the desired number of pulses is smaller than the Maximum Pulse Number, NI? returns the specified value.

The Maximum Pulse Number (MPN) is defined as follows:

MPN = CPTL/[max (2 * max (Encoder resolution of axis X and axis Y)
and (Trajectory velocity * T_{base})]

CPTL (Curvilinear Pulsed Trajectory Length) = Sum of all trajectory element lengths between NB and NE.

NOTE

The starting position and the ending position of the axis must be outside the interval defined with NB and NE commands without forget acceleration and deceleration ranges.

Example:

Generate 11 pulses on a trajectory starting with element number 2 and ending with element number 3:

NT	*Start defining a new trajectory.*
LX10	*Create element #1.*
CR10, CA90	*Create element #2.*
LY20	*Create element #3.*
CX10, CY30	*Create element #4.*
NB2, NE3, NI10.1, NN11	*Start pulses on element #2, end on element #3 generate 11 pulses (10 steps), each 0.1 unit.*

NS	|	*Enable the signal generation.*
ET	|	*Execute the trajectory.*
TQ	|	*Read position of every pulse generated.*

During the execution of a trajectory with such synchronized signals, each time a pulse is generated, the real and theoretical position of all axes is recorded in the global position buffer that could be read with the TQ command. The NS command thus terminates the effect of a previously entered GQ command. To enable the global trace mode after a trajectory with synchronized pulses, the GQ command must be re-issued.

A pulse is generated automatically as soon as the trajectory execution reaches position:

$$\text{Pulse position} = \text{Pos(NB)} + (n * \text{Step}) \text{ with stage position} \in [\text{NB} \ldots \text{NE}]$$

$$\text{where: Step} = \text{CPTL}/(\text{NI} - 1)$$

The position accuracy of the generated pulse (the difference between the theoretical position Pos(NB) + n * Step and the real position where the pulse is generated) depends on the trajectory velocity as follows:

$$\text{MaxError} = \pm\, 0.707 * (T_{base} * \text{Trajectory velocity})$$

This value cannot be smaller than the encoder resolution of X or Y axis.

If	MaxError < encoder resolution of X axis.
then	MaxError = encoder resolution of X axis.
If	MaxError < encoder resolution of Y axis.
then	MaxError = encoder resolution of Y axis.

Example

For a trajectory velocity of 20 mm/sec and T_{base} = 300 µsec:

$$\text{MaxError} = \pm 0.707 * (3 * 10^{-4} * 20) = \pm 4.24\ \mu\text{m}$$

This position uncertainty exists for every pulse generated but is not cumulative.

6.1.3 Synchronizing Events to Trajectory Elements

Controller operations and functions can be synchronized to the execution of a trajectory element. This is achieved by using the WNnn command.

The nn parameter represents the trajectory element number to synchronize with. At the beginning of this element, one or more secondary controller activities could be initiated.

Example

Increase the trajectory velocity starting with element number 2 and reduce it with element number 4.

1XX	|	*Erase program #1 (if exists).*
1EP	|	*Start program entry mode.*
NT	|	*Start new trajectory definition.*
LX10	|	*Trajectory element #1.*
CR10, CA90	|	*Trajectory element #2.*
LY20	|	*Trajectory element #3.*
CX10, CY30	|	*Trajectory element #4.*
LX0	|	*Trajectory element #5.*
CX0, CY0	|	*Trajectory element #6.*
VV5	|	*Set trajectory velocity to 5 mm/sec.*
ET	|	*Execute trajectory.*

WN2, VV10	|	*Starting with element #2 set velocity to 10 mm/sec.*
WN4, VV5	|	*Starting with element #4 set velocity to 5 mm/sec.*
QP	|	*End program entry mode.*
1SM	|	*Save program in non-volatile RAM.*
1EX	|	*Execute program #1.*

6.1.4 Synchronizing Events to Trajectory Position

Controller operations and functions can also be synchronized to the trajectory position. This is achieved by using the WInn command.

Here, nn represents the trajectory position to synchronize with. When the trajectory length executed reaches the value specified by nn, one or more secondary controller activities could be initiated.

Example

Increase the trajectory velocity when the trajectory reaches position 5 and reduce the velocity when it reaches position 24.

2XX	|	*Erase program #2 (if exists).*
2EP	|	*Start program entry mode.*
NT	|	*Start new trajectory definition.*
LX10	|	*Trajectory element #1.*
CR10, CA90	|	*Trajectory element #2.*
LY20	|	*Trajectory element #3.*
CX10, CY30	|	*Trajectory element #4.*
LX0	|	*Trajectory element #5.*
CX0, CY0	|	*Trajectory element #6.*
VV5	|	*Set trajectory velocity to 5 mm/sec.*
ET	|	*Execute trajectory.*
WI5, VV10	|	*Starting with trajectory position 5 set velocity to 10 mm/sec.*
WI24, VV5	|	*Starting with trajectory position 24 set velocity to 5 mm/sec.*
QP	|	*End program entry mode.*
2SM	|	*Save program in non-volatile RAM.*
2EX	|	*Execute program #2.*

6.2 Synchronized Axes (Electronic Gearing)

Certain applications require to synchronize the motion of two or more axes. In this case, one or more axis precisely follow the motion of another one. To safely define and operate such a motion control system, the following rules must be observed:

- Each axis of the MM4006 has an identity: Master (default) or Slave. By default, all axes are configured as masters, meaning that all can execute independent motion commands.
- In a group of synchronized axes there is only one master and one or more slaves. The slaves always follow the motion of the master.
- All commands to a group of synchronized axes (from the front panel, through commands or through programs) is done by addressing only the master axis. No communication with the slave axes is allowed.
- Determining the master-slave relationship can be done on the front panel (Motor OFF → SETUP → GEN.), through remote commands (xxSSnn command) or through a program.
- A master axis is defined as an independent axis. It could have one or more slave axes or, as a particular case, none (default).
- A slave axis belongs to a unique master axis, in effect losing its identity. It will duplicate the behavior of its master. Consequently, two master axes cannot have the same slave.
- By default (standard MM4006 configuration) all axes are declared masters. However, each time a master-slave system is defined, its characteristics are saved in the non-volatile memory. On each consequent power-on, the controller will remember the latest configuration.
- The motion of a master axis is limited by its own travel limits. A slave axis is limited both by its own and its master's limits. If in the course of the motion a slave axis encounters its own travel limits, the emergency stop procedure is initiated and all motion will stop.

The following three commands are needed to define and operate a master-slave motion system:

xx **SS** nn Defining and reading the master-slave status of an axis.

xx **GR** nn Defining and reading the electronic gear ratio between the master and the slave (by default = 1.0), using the following formula:

Displacement of the slave axis =
GR * Displacement of the master axis.

GR can be a positive or negative number but not zero.

xx **FF** nn Defining and reading the maximum master-slave tracking error. If this tracking error is exceeded, the emergency stop procedure is initiated and all motors are turned OFF.

The tracking error (Tk_Err) is calculated as follows:

Tk_Err = Absolute value (Pos_Err_Master - (Pos_Err_Slave/GR))

Where:

Pos_Err_Master — Position error of the master axis.
Pos_Err_Slave — Position error of the master axis.
GR — Electronic gear ratio.

Newport

6.3 Joystick

The MM4006 lets you use a joystick to manually manipulate axes remotely.

The MM4006 joystick has four buttons: MOT. ON, MOT. OFF, ORIG, MANU., three slide switches PX, PY, PXY, two LEDs L1, L2 and a two-dimensional potentiometer PO. The joystick is connected to the MM4006 via the 15-pin D-Sub remote control output (the joystick connector replaces the 15-pin D-Sub short circuit connector on the rear panel of the MM4006).

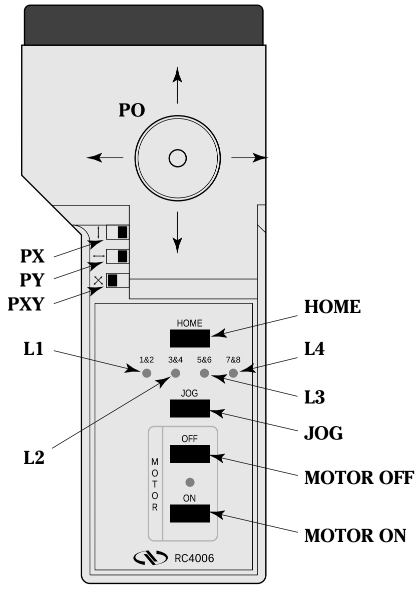


***Fig. 6.1** — MM4006 Joystick.*

PO The potentiometer to manipulate one or two axes simultaneously. The further the potentiometer is moved from its center the more rapidly the axis/axes move.

PX Slide switch to reverse the X axis direction.

PY Slide switch to reverse the Y axis direction.

PXY Slide switch to exchange the X and Y axes directions.

HOME Button to start an origin search cycle.

JOG Button to select two axes that will be linked to the action of the joystick.

- Initial state: No axis is chosen (LEDs **L1**, **L2**, **L3** and **L4** are off).
  If the button is pressed, the joystick goes to state 1.
- State 1: Axes 1 and 2 are chosen (LED **L1** is on).
  If the button is pressed again, the joystick goes to state 2.
- State 2: Axes 3 and 4 are chosen (LED **L2** is on).
  If the button is pressed again, the joystick goes to state 3.
- State 3: Axes 5 and 6 are chosen (LED **L3** is on).
  If the button is pressed again, the joystick goes to state 4.
- State 4: Axes 7 and 8 are chosen (LED **L4** is on).
  If the button is pressed, the joystick returns to the initial state (the LEDs are off).

MOTOR OFF Equivalent to the MOTOR OFF button on the MM4006.

MOTOR ON Equivalent to the MOTOR ON button on the MM4006.

The corresponding messages are displayed when the joystick goes to state 1 (axes 1 and 2) and state 2 (axes 3 and 4). In state 1 or 2, the potentiometer can be used to manipulate the axes.

NOTE

In remote mode (MR command), using of the joystick is not permitted by default. To use it in remote mode, send MC command after MR command.

6.4 Executing Sub-routines in a Program: EX Command

The MM4006 is capable of executing complex programs containing sub-routines.

The sub-routines are blocks of commands that do not contain the EX command. They are called by the main program.

Example

	********* Program 1 (Main Program) *********
1EP	*Enter program 1.*
1PA10, 2PA10	*Two-axis movement.*
2EX	*Execute program 2.*
3EX	*Execute program 3.*
4EX	*Execute program 4.*
OR	*Origin search on all axes.*
QP	*Quit main program.*
	**************** Program 2 ****************
2EP	*Enter program 2.*
SB	*Set bits.*
1AS"This "	*Define string # 1.*
2AS"is "	*Define string # 2.*
1CS$S2	*Concatenate string # 1 and string # 2.*
DS$S1"a string"	*Display on screen.*
WT3000	*Wait for 3 seconds.*
QP	*Quit program 2.*
	**************** Program 3 ****************
3EP	*Enter program 3.*
3AS"a value: "	*Define string # 3.*
101YS99.99	*Define value # 101.*
3CS$Y101	*Concatenate string # 3 and value # 101.*
DS$S1$S3 " !"	*Display on screen.*
WT3000	*Wait for 3 seconds.*
QP	*Quit program 3.*
	**************** Program 4 ****************
4EP	*Enter program 4.*
1PR-20,WS	*Axis 1 movement.*
2PR-20,WS	*Axis 2 movement.*
CB	*Clear bits.*
WT1000	*Wait for 1 second.*
QP	*Quit program 4.*
THIS IS A STRING	*Display on controller screen.*
THIS IS A VALUE: 99.99 !	*Display on controller screen.*

Section 6

Feature Descriptions Tutorial

Table of Contents
Section 7 — Servo Tuning

Section 7
Servo Tuning

7.1 Servo Tuning Principles

The MM4006 controller uses a PID servo loop with feed forward. Servo Tuning sets the Kp, Ki and Kd, and Feed-Forward parameters of the digital PID algorithm, also called the PID filter.

Tuning PID parameters requires a reasonable amount of closed-loop system understanding. You should first review the Control Loops paragraph in the Motion Control Tutorial Section and, if needed, consult additional servo control theory books.

Always start the tuning process using the default values supplied with the MM4006 for each motion device type, or for the generic Default type. These values are usually very conservative, favouring safe, oscillation-free operation for a tighter, more responsive system that minimises following error. To achieve the best dynamic performance possible, the system must be tuned for your specific application. Load, acceleration, stage orientation and performance requirements all affect how the servo loop should be tuned for best results.

7.1.1 Hardware Requirements

Tuning is best accomplished when the system response can be measured. This can be done with external monitoring devices but this can introduce errors.

The MM4006 controller avoids this problem by offering a Trace capability. When Trace mode is activated, the controller can record real and desired positions simultaneously. These are the basic pieces of information that the controller uses to calculate the PID filter. The sample interval can be as fast as the servo update cycle (0.0005s) and the total number of samples can be up to 4000 points.

With these powerful capabilities, there is no need for additional hardware to perform servo tuning.

7.1.2 Software Requirements

The MM4006 controller offers two types of trace capabilities. One is a single axis Trace mode supported by the SP, XS, TM, XN and TT commands and the other is a Global Trace mode in which all axes are sampled. This is controlled by the SQ, XQ, GQ, NQ and TQ commands. The two modes are completely independent.

Performance data for tuning can be acquired in two ways: you could write custom software using the commands mentioned or use the NMCServo NEWPORT software that has all the necessary functions, including plotting performance and saving the results.

For a detailed description of the NMCServo software and its operation please review NMC Softwares User’s Manual.

7.2 Tuning Procedures

Servo tuning is usually performed to achieve better motion performance (such as reducing the following error statically and/or dynamically) or because the system is malfunctioning (oscillating and/or shutting off due to excessive following error).

NOTE

Remember that all three PID gain factors are normalised, meaning that they take a value between 0 and 1. If the value is 1, the parameter has the highest gain possible. If the value is 0, the specified parameter is disabled.

Acceleration plays a significant role in the magnitudes of the following error and the overshoot, especially at start and stop. Asking the controller to change the velocity instantaneously amounts to an infinite acceleration which, since it's physically impossible, causes large following errors and overshoot. Use the smallest acceleration the application can tolerate to reduce overshoot and make tuning the PID filter easier.

NOTE

In the following descriptions, it is assumed that some kind of NMCServo software is being used to capture the response of the servo loop during a motion step command and to visualise the results.

7.2.1 Axis Oscillation

If the axis oscillates, this indicates that the gain Kp may be too large. Start by reducing the proportional gain factor Kp by one order of magnitude (e.g. 0.2 to 0.02) and making Ki and Kd equal to zero.

NOTE

Remember that the default values are conservative enough to guarantee oscillation-free operation. You can always reload them through in the Axis SETUP menu on the front panel by re-selecting the motion device you are using.

If the oscillation does not stop, reduce Kp again.

NOTE

The first step should be sufficient to eliminate the oscillation. If not, it may indicate the existence of other problems, usually with the hardware (wiring, etc.).

When the axis stops oscillating, the system response is probably very soft. The following error may be quite large during motion and non-zero at stop. You should continue tuning the PID with the steps described in the next paragraph.

7.2.2 Increasing Performance

If your system is stable and you want to improve the performance, start with the current parameters. The goal is to reduce the following error during motion and to eliminate it at stop.

Depending on the performance starting point and the desired outcome, here are some guidelines for further tuning.

Following Error Too Large

This is the case of a soft loop. It is especially common if you just performed the steps in 7.2.1. The proportional gain Kp is probably too low and Ki and Kd are zero.

Start by increasing Kp by a factor of 1.5 to 2. Continue this operation while monitoring the following error until it starts to exhibit excessive ringing characteristics (more than 3 cycles after stop.) To reduce the ringing, add some damping by increasing the Kd parameter.

Start with a Kd value one order of magnitude smaller than Kp. Increase it by a factor of 2 while monitoring the following error. As Kd is increased, the overshoot and the ringing decrease almost to zero.

NOTE

Remember that if the acceleration is set too high, the overshoot cannot be completely eliminated with Kd.

If Kd is further increased, at some point the oscillation will reappear, usually at a higher frequency. Avoid this by keeping Kd at a high enough value, but not so high as to reintroduce oscillations.

Next add more gain. Increase the Kp value by 50% at a time until signs of excessive ringing appear again.

Alternatively increase Kd and Kp until Kd cannot eliminate the overshoot and ringing at stop. This indicates Kp is larger than its optimal value and should be reduced.

Ultimately, optimal values for Kp and Kd depend on the stiffness of the loop and how much ringing the application can tolerate.

Errors At Stop (Not In Position)

If you are satisfied with the dynamic response of the PID loop but the motion device does not always stop accurately, modify the integral gain factor Ki. As described in the Motion Control Tutorial Section, this term of the PID reduces the following error to near zero. Unfortunately it can also contribute to oscillation and overshoot. Always change this parameter carefully and in conjunction with Kd.

NOTE

Ks (0 to 1) controls the saturation level of Ki integral factor of the PID position closed loop. A excessive value of Ks implies the delayed effect on the controller reaction towards processes to command. Conversely, a too little value eliminates the integrator action. The optimal value varies between 0.5 and 0.9.

Start, if possible, with a value for Ki that is at least two orders of magnitude smaller than Kp. Increase its value by 50% at a time and monitor the overshoot and the final position at stop.

If intolerable overshoot develops, increase the Kd factor. Continue increasing Ki and Kd alternatively until an acceptable loop response is obtained. If oscillation develops, immediately reduce the Ki.

Remember that any finite value for Ki will eventually reduce the error at stop. It is simply a matter of how much time is acceptable for your application. In most cases it is preferable to wait a few extra milliseconds to stop in position rather than have overshoot or run the risk of oscillations.

Following Error During Motion

This is caused by a Ki value that is too low. Follow the steps in the previous paragraph, keeping in mind that it is desirable to increase the integral gain factor as little as possible.

7.2.3 Points to Remember

- The MM4006 controller uses a servo loop based on the PID with velocity feed-forward algorithm.
- Special servo design makes the velocity feed-forward only motor-dependent, not load-dependent. It is factory-set and not accessible to the user.
- Use the lowest acceleration the application can tolerate. Smaller acceleration generates less overshoot.
- Use the default values provided with the system for all standard motion devices as a starting point.
- Use the minimum value for Ki that gives acceptable performance. The integral gain factor can cause overshoot and oscillations.

Section 7
Servo Tuning

Table of Contents
Section 8 — Appendices

A — Error Messages

The MM4006 controller continually verifies the actions of the motion control system and the operator. When an error is detected, the controller stores it in an error register. To avoid communication and application conflicts, the MM4006 does not automatically report the error. It is the user's responsibility to periodically query the error status, particularly during the development phase of an application.

To better understand error-handling, keep in mind the following points:

- Reading the error with TE or TB clears the error buffer.
- The controller stores only the last error encountered.
- Once an error is detected, it is stored until read or replaced by a new error.
- The error read represents an error that could have happened at any time since the last read.
- For faster communication throughput, use the TE command to read only the error code.
- Use the TB command to read an existing error or to translate an error code.

Error List

The following is a list of all error message codes and their descriptions:

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.
E — Incorrect I/O channel number.
F — Program number incorrect.
G — Program does not exist.
H — Calculation overflow.
I — Unauthorized command in programming mode.
J — Command authorized only in programming mode.
K — Undefined label.
L — Command not at the beginning of a line.
M — Program is too long.
N — Incorrect label number.
O — Variable number out of range.
P — Number of WE commands does not match the number of open loops.
Q — Unauthorized command.
R — Command cannot be at the beginning of a line.
S — Communication time-out.
T — Error during home search cycle.
U — Failure while accessing the EEPROM.
V — Too long trajectory.
W — Trajectory: to big discontinuity angle.

X — Trajectory: first angle definition error.
Y — Trajectory: Line (x, y) Line expected.
Z — Trajectory: Line (x, y) too big discontinuity.
[— Trajectory: Line (x, θ) or Line (y, θ) impossible.
**** — Trajectory: Arc expected.
] — Trajectory: Arc (r, θ) radius is too small.
^ — Trajectory: Arc (r, θ) radius is too big.
_ — Trajectory: Arc (r, θ) sweep angle is too small.
` — Trajectory: Arc (x, y) circle is too small.
a — Trajectory: Arc (x, y) Circle is impossible.
b — Trajectory: trajectory is empty.
c — Unit not translational or incorrect.
d — Unit not rotationnal or incorrect.
e — Trajectory: Units not translationnal or not identical.
f — sync. pulses generation impossible.
g — mechanical familly name incorrect.
h — Trajectory: execution exceeds physical or logical limits.

Besides the standard screens available on the front panel display, there are a number of error screens that appear only in special error conditions.

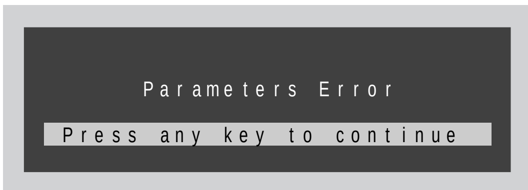


***Fig. A.1** — Error screen (English).*

***Fig. A.2** — Error screen (French).*

The screen in Fig. A.1 (English version) or Fig. A.2 (French version) appears if the battery-backed non-volatile memory is corrupted. This will result in a loss of all data in this memory and the controller will request the operator to perform a complete setup procedure on the front panel.

NOTE

Under certain conditions, you may need to erase the non-volatile memory and load the default parameters. This is accomplished simultaneously pressing the minus key "–" and the period key "." on the keypad during the power-up sequence. This will initiate a setup procedure.

The error message shown in Fig. A.3 appears on power-up if the IEEE488 is detected to be malfunctioning. Under this condition, only the RS-232 interface can be used.

Fig. A.3 *— Error screen, IEEE488.*

The error message in Fig. A.4 appears if one of the function keys or keypad keys are detected being pressed (or stuck) during power-up. The X indicates which key is detected, function keys being labeled from A to D, from left to right.

Fig. A.4 *— Error screen, depressed key during start-up.*

During program creation or modification, the screen shown in Fig. A.5 could appear if the command line being edited exceeds the 110 character limit. The last command entered will be lost but the rest of the line is retained and can be saved. (The XXXX… represents the actual command line being edited).

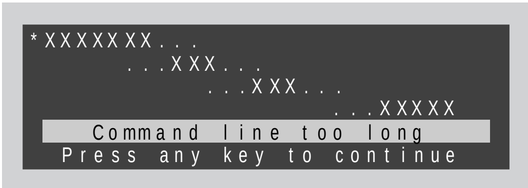


Fig. A.5 *— Error screen, command line too long.*

The second type of error message that is available during program creation or modification is shown in Fig. A.6. It will appear when the non-volatile memory allocated to program storage becomes full. The last line entered (XXXX…) will be lost but the rest of the program is saved.

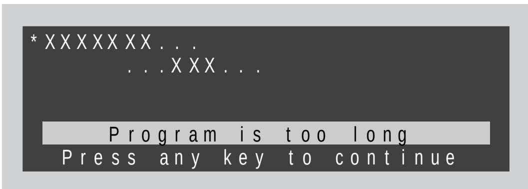


Fig. A.6 *— Error screen, program memory full.*

B — IEEE-488 Link Characteristics

NOTE

In order to meet FCC emission limits for a Class B device, you must use a double shielded IEEE-488 cable. Operating this equipment with a single shielded cable may cause interference to radio and television reception in residential areas.

NOTE

Comply to IEEE Standard Digital Interface for Programmable Instrumentation.

ANSI/IEEE Std. 488 - 1978. This norm is commonly called IEEE-488.

IEEE-488 Functions Supported by MM4006 Controller

Mnemonic	Definition	Support
ATN	Attention	Yes
DCL	Device Clear	Yes
EOI	End or Identify	Yes
EOL	End of Line	Yes
GET	Group Execute Trigger	No
GTL	Go to Local	No
IFC	Interface Clear	Yes
LAD	Listen Address	Yes
LLO	Local Lockout	No
OSA	Other Secondary Address	No
PPC	Parallel Pol Configure	No
PPD	Parallel Poll Disable	No
PPE	Parallel Poll Enable	No
PPU	Parallel Poll Unconfigure	No
REN	Remote Enable	No
SDC	Selected Device Clear	Yes
SPD	Serial Poll Disable	No
SPE	Serial Poll Enable	Yes
SRQ	Service Request	Yes
TAD	Talk Address	Yes
TCT	Take Control	No
UNL	Unlisten	Yes
UNT	Untalk	Yes

IEEE-488 Function Subsets

This controller support the many GPIB function subsets, as listed bellow. Some of the listings described subsets that the controller does not support.

C0 (Controller). The MM4006 can not control other devices.

T5 (Talker). The MM4006 becomes a Talker when the CIC (Controller In Charge) sends its TAD (Talker Address) with the ATN (Attention) line asserted. It ceases to be a talker when the CIC (Controller In Charge) sends another device's TAD (Talker Address) with ATN (Attention) asserted.

L4 (Listener). The MM4006 becomes Listener when the CIC (Controller In Charge) sends its LAD (Listener Address) with the ATN line asserted. The MM4006 does not have Listen Only capability.

SH1 (Source Handshake). The MM4006 can transmit multiline messages accros the GPIB.

AH1 (Acceptor Handshake). The MM4006 can receive multiline messages accros the GPIB.

SR1 (Service Request). The MM4006 asserts SRQ (Serial Request) line to notify the CIC (controller In Charge) when it requires service.

RL0 (Remote/Local). The MM4006 does not support the GTL (Go To Local) and LLO (Local Lock Out) functions.

PP0 (Parralel Poll). The MM4006 has no Parallel Poll capability. It does not respond to the following interface messages: PPC, PPD, PPE and PPU. The MM4006 does not send out a message when the ATN (Attention) and EOI (End or Identify) line are asserted.

DC1 (Device Clear). The MM4006 responds to the DCL (Device Clear) and, when made Listener, the SDC (Selected Device Clear) interface message.

DT0 (Device Trigger). The MM4006 does not support GET (Group Execute Trigger) interface message.

E2 (Electrical). The MM4006 uses tristate buffers to provide optimal high-speed data transfer.

SRQ Using

The NI488.2 User Manual for Windows from National Instruments, in the GPIB Programming Techniques chapter describes the use of Serial Polling as follow (page 7-5):

Serial Polling

You can use serial polling to obtain specific information from GPIB devices when they request service. When the GPIB SRQ line is asserted, it signals the Controller that a service request is pending. The controller must then determine which device asserted the SRQ line and respond accordingly. The most common method for SRQ detection and servicing is serial poll. This section describes how you can set up your application to detect and respond to service requests from GPIB devices.

Service Requests from IEEE-488 Devices

IEEE-488 devices request service from the GPIB Controller by asserting the GPIB SRQ line. When the Controller acknowledge the SRQ, it serial polls each open device on the bus to determine which device requested service. Any device requesting service returns a status byte with bit 6 set and then unasserts the SRQ line. Devices not requesting service return a status byte with bit 6 cleared. Manufacturers of IEEE-488 devices use lower order bits to communicate the reason for the service request or to summarize the state of the device.

Service Requests from IEEE-488.2 Devices

The IEEE-488.2 standard redefined the bit assignments in the status byte. In addition to setting bit 6 when requesting service, IEEE-488.2 devices also use two other bits to specify their status. Bit 4, the Message Availiable Bit (MAV), is set when the device is ready to send previously queried data. Bit 5, the Event Status Bit (ESB), is set if one or more of the enabled IEEE-488.2 events occurs. These events include power-on, user request, command error, execution error, device-dependant error, querry error, request control and operation complete. The device can assert SRQ when ESB or MAV is set, or when a manufacturer-defined condition occurs.

Also on page 7-7, National instruments give an example on how to conduct a serial poll:

SRQ and Serial Polling with NI-488 Device Functions…

The following example illustrates the use of the ibwait and ibrsp functions in a typical SRQ servicing situation when automatic serial polling is enabled.

```
#include "decl.h"

char GetSerialPollResponse (int DeviceHandle)
{
char SerialPollResponse = 0;

ibwait (DeviceHandle, TIMO | RQS);
if (ibsta & RQS)
{
printf ("Device asserted SRQ.\n");
/* Use ibrsp to retrieve the serial poll response. */
ibrsp (DeviceHandle, &SerialPollResponse);
}
return (SerialPollResponse);
}"
```

The MM4006 Controller is an IEEE-488 device in which the SRQ is always enable. It will respond accordingly to the National Instruments example. When the queried data will be ready, the MM4006 will assert the SRQ line and, in the serial poll response bit 6 will be set (Requesting service) and bit 7 (manufacturer-defined) will be set (Message Availiable). After that you can use the ibrd command to retreive the data from the MM4006.

C — Connector Pinouts

Labeling Conventions

All pinout diagrams in this section use the following labeling convention:

AGND ⇒ Analog ground.
DGND ⇒ Digital ground.
N.C. ⇒ Not connected.
UTIL ⇒ Test/utility signal. **DO NOT USE; MAY BE ENERGIZED.**
I ⇒ Input.
O ⇒ Output.

WARNING

The company assumes no responsability for the use of any UTIL labelled pin.

Power Inhibition Connector (9-Pin D-Sub)

This connector is provided for the wiring of one or more remote Emergency Stop switches or Start switches. They will have the same effect as the front panel MOTOR OFF or MOTOR ON buttons.

The minimum rating for the switches should be 50 mA at 24 V and the maximum contact resistance should be less than 100 Ω.

Pin #		Description	
1	—	**N.C.**	
2	—	**UTIL**	Start, switches must be self release push buttons. Wire the switch contacts normally opened. The other side of the switch should be connectd to DGND. If more than one switch is installed, they should be connected in parallela.
3	—	**I**	Emergency Stop, must always be connected to DGND during normal controller operation. An open circuit is equivalent to pressing MOTOR OFF on the front panel. Wire the switch contacts normally closed. If more than one switch is installed, they should be connected in series
4	—	**N.C.**	
5	—	**N.C.**	
6	—	**DGND**	
7	—	**DGND**	
8	—	**DGND**	
9	—	**N.C.**	

Remote Control Connector (15-pin D-Sub)

This connector should only be used with the NEWPORT RC4000 remote Controller.

The connector also provides an Emergency Stop switch input with identical operation to the one in the Power Inhibition connector. If no remote controller are used, the pins must be shorted.

Pin #		Description	
1	—	DGND	
2	—	I	For normal operation connect pins 2 and 3 together.An open circuit is equivalent to pressing the MOTOR OFF on the front panel.
3	—	O	
4	—	UTIL	
5	—	UTIL	
6	—	UTIL	
7	—	UTIL	
8	—	UTIL	
9	—	DGND	
10	—	DGND	
11	—	UTIL	
12	—	UTIL	
13	—	UTIL	
14	—	UTIL	
15	—	UTIL	

WARNING

NEWPORT assumes no responsability for the use of any other Remote Controller.

Auxiliary Connector (25-Pin D-Sub)

This connector is used for the MOTOR OFF indicator, the frequency generator output, the analog inputs and outputs and the synchronisation pulses.

The analog outputs are only available in option.

The logic outputs are open-collector type and are rated for maximum 30 V and 40 mA (Fig. C.2). To drive logic input, they require a pull-up resistor.

The analog inputs and outputs have 12 bits resolution.

The analog inputs are multi-range, software programmable. The available ranges are ±10V, ±5V, 0-10V, 0-5V. See the RA and AM commands for more programmation details. In all cases, analog inputs must be below ±10 V. The impedance of the converter inputs is typically 10kOhms. The maximum input current is ±300µA. The maximum offset error is ±10 LSB, and the maximum gain error is ±10 LSB. The input characteristics of the analog inputs are in Fig. C.1.

The value of 1 LSB depends of the used range:

- 1 LSB is: 20 V/4096 ≈ 5 mV for the ±10 V range.
- 1 LSB is: 10 V/4096 ≈ 2.5 mV for the ±5 V range and 0-10 V range.
- 1 LSB is: 5 V/4096 ≈ 1.25 mV for the 0-5 V range.

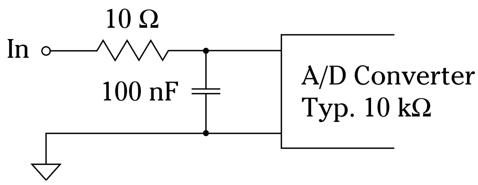


***Fig. C.1** — Equivalent circuit of an analog input.*

The analog outputs range is ±10 V. The maximum offset error is ±200 mV, and the maximum gain error is ±10 LSB. The output setting time is typically 6 µsec. These outputs are voltage outputs (output current less than 1 mA), so to use them properly, they must be connected to an impedance higher than 10 kW. 1 LSB is: 20 V/4096 ≈ 5 mV.

Pin #		**Description**	
1	**—**	**DGND**	
2	**—**	**N.C.**	
3	**—**	**UTIL**	
4	**—**	**UTIL**	
5	**—**	**UTIL**	
6	**—**	**UTIL**	
7	**—**	**UTIL**	
8	**—**	**N.C.**	
9	**—**	**N.C.**	
10	**—**	**O**	A LOW signal indicates that Motor Power is ON.
11	**—**	**O**	Pulse synchronized to one AXIS, see PB, PE, PI and PS commands.
12	**—**	**O**	Pulse synchronized to a trajectory, see NB, NE, NI, NN and NS commands.
13	**—**	**DGND**	
14	**—**	**I**	Analog Input 1.
15	**—**	**I**	Analog Input 2.
16	**—**	**I**	Analog Input 3.

17	**—**	**I**	Analog Input 4.
18	**—**	**DGND**	
19	**—**	**O**	Analog Output 1.
20	**—**	**O**	Analog Output 2.
21	**—**	**O**	Analog Output 3.
22	**—**	**O**	Analog Output 4.
23	**—**	**DGND**	
24	**—**	**O**	Output frequency, defined by the FT command.
25	**—**	**DGND**	

NOTE

Remember that an I/O output bit "set" means that the transistor is conducting, thus appearing to be "low".

GPIO Connector (37-Pin D-Sub)

This connector is dedicated to the digital I/O ports.

All outputs are open-collector type and are rated for maximum 30V and 40mA (Fig. C.2). To drive a logic input, they require a pull-up resistor.

All inputs are optocoupled and are configured as a LED in series with a 1 kΩ resistor connected to the +12 V line (Fig. C.2).

Pin #	Description	Pin #	Description
1 —	**External +12 V/Internal +12 V** [(1)]	**20 —**	**DGND** [(2)]
2 —	**+12 V, 25 mA**	**21 —**	**DGND** [(2)]
3 —	**+5 V,100 mA**	**22 —**	**DGND** [(2)]
4 —	**I** Digital port Input 1.	**23 —**	**DGND** [(2)]
5 —	**I** Digital port Input 2.	**24 —**	**DGND** [(2)]
6 —	**I** Digital port Input 3.	**25 —**	**DGND** [(2)]
7 —	**I** Digital port Input 4.	**26 —**	**DGND** [(2)]
8 —	**I** Digital port Input 5.	**27 —**	**External Ground/Internal Ground** [(2)]
9 —	**I** Digital port Input 6.	**28 —**	**DGND** [(2)]
10 —	**I** Digital port Input 7.	**29 —**	**DGND** [(2)]
11 —	**I** Digital port Input 8.	**30 —**	**DGND**
12 —	**O** Digital port Output.1.	**31 —**	**DGND**
13 —	**O** Digital port Output.2.	**32 —**	**DGND**
14 —	**O** Digital port Output.3.	**33 —**	**DGND**
15 —	**O** Digital port Output.4.	**34 —**	**DGND**
16 —	**O** Digital port Output.5.	**35 —**	**DGND**
17 —	**O** Digital port Output.6.	**36 —**	**DGND**
18 —	**O** Digital port Output.7.	**37 —**	**DGND**
19 —	**O** Digital port Output.8.		

1) *If optocoupling is not activated, pin #1 outputs +12 VDC.*

If optocoupling is activated, external +12 VDC must be supplied to pin #1. Needs factory service to be activated.

2) *If optocoupling is not activated, pin #20 to pin #29 are tied to the internal DGND.*

If optocoupling is activated, pin 20 to pin 29 are not tied to the internal ground and must be tied to the ground of the external +12 V power supply. Needs factory service to be activated.

Logical Inputs

Parameter	Symbol	Min.	Max.	Units
Low Level Input Voltage	V_{il}	0	5	V
High Level Input Voltage	V_{ih}	11	12	V
Input Current LOW	I_{il}	-5	-10	mA
Pulse Width (1)		1		Servo Cycle
Input low to high	TP_{lh}		10	µsec
Input high to low	Tp_{hl}		10	µsec

1) *Optoisolated logical inputs:*

These inputs works with current driven into the led. If there is no current, input is read as a 1, if there is current through the LED, input is read as a 0.

To drive current through the LED, you can tie the input to ground or drive it by an open collector. This way, the logic level seen at the input, is the same as the one given by the RB command.

To ensure good performances, when current is present its value must be between 5mA and 10 mA.

To be taken into account, one pulse on the input must be larger than one servo-cycle.

Logical Outputs

Parameter	Symbol	Min.	Max.	Units
Low Level Output Voltage	V_{ol}	0	1	V
High Level Output Voltage	V_{oh}		30	V
Output Current LOW	I_{il}		-40	mA
Pulse Width (2)		1		Servo Cycle
Output low to high	TP_{lh}	1		µsec
Output high to low	TP_{hl}	1		µsec

2) *The minimum width on an output pulse cannot be smaller than one servo-cycle.*

To assure good use and performances of the MM4006, respect these maximum ratings.

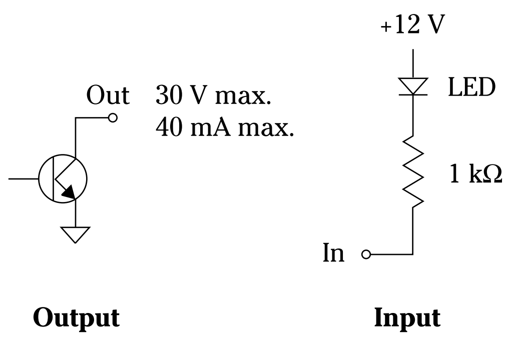


***Fig. C.2** — Equivalent circuits for the digital input and output ports.*

RS-232C Interface Connector (9-Pin D-Sub)

The RS-232 C interface uses a 9-pin Sub-D connector.

The back panel connector pinout is shown in Fig. C.3.

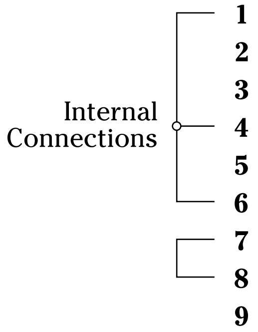


***Fig. C.3** — RS-232C connector pinout.*

RS-232C Interface Cable

The reason some pins are jumpered in the controller as described in Fig. C.3 is to override the hardware handshake when an of-the-shelf cable is used for the RS-232C interface. This guaranties proper communication even when the handshake cannot be controlled from the communication software.

Fig. C.4 shows a simple pin-to-pin cable with 9 conductors.

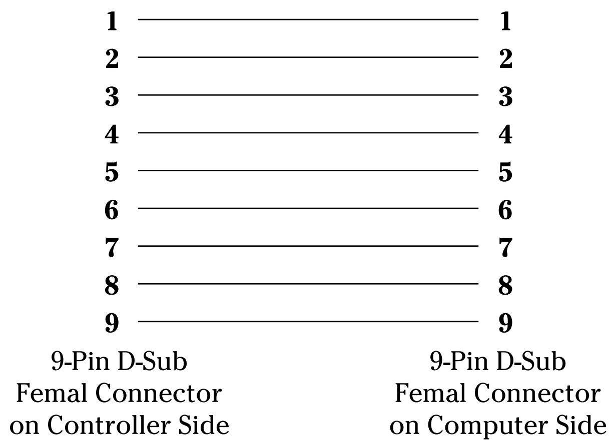


***Fig. C.4** — Conductor, pin-to-pin RS-232C interface cable.*

If you want to use a three conductor cable, you must use a cable configured as in Fig. C.5 to get the same hardware handshake override.

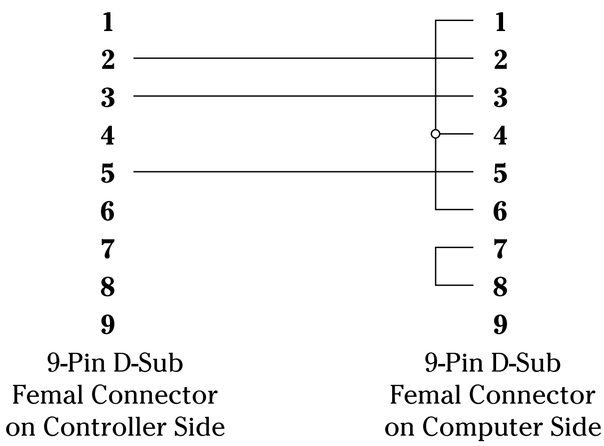


***Fig. C.5** — Conductor RS-232C interface cable.*

If your computer or terminal uses a 25-pin connector for the RS 232C interface, you can use an off-the-shelf 25 to 9-pin adapter and one of the two cables described above.

If you do not wish to add an adapter, you can use an off-the-shelf 9 to 25-pin RS-232C cable or build one like in Fig. C.6.

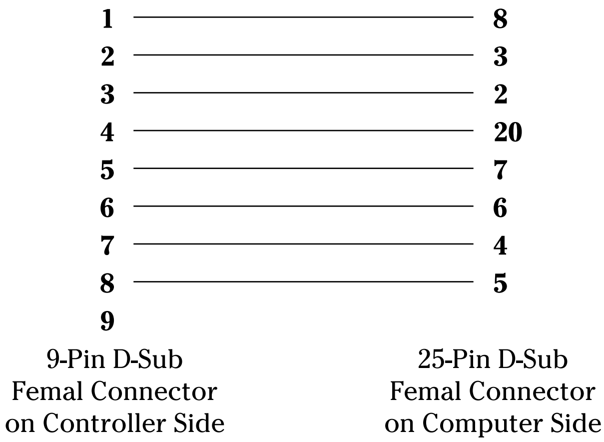


***Fig. C.6** — 9-pin to 25-pin RS-232C interface cable.*

To build a three conductor cable with a 25-pin RS-232C connector, use the wiring diagram in Fig. C.7.

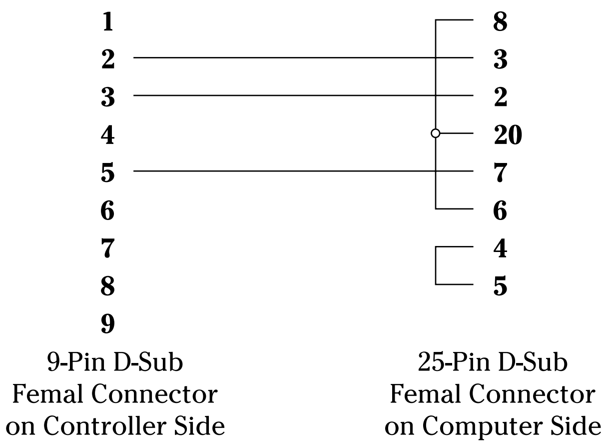


***Fig. C.7** — 3-conductor, 9-pin to 25-pin RS-232C interface cable.*

IEEE488 Interface Connector (24-Pin)

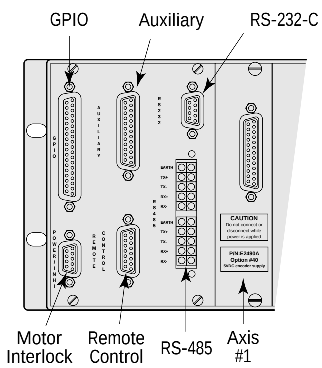
The IEEE488 connector has a standard configuration, shown in Fig. C.8.

	Pin #		
DIO1	1	13	DIO5
DIO2	2	14	DIO6
DIO3	3	15	DIO7
DIO4	4	16	DIO8
EOI	5	17	REN
DAV	6	18	GND
NRFD	7	19	GND
NDAC	8	20	GND
IFC	9	21	GND
SRQ	10	22	GND
ATN	11	23	GND
SHIELD	12	24	SIG. GND

Fig. C.8 — IEEE488 connector definition.

RS-485 Interface Connector (5-Pin)

Two identical RS-485 connectors are available. Both are connected in parallel, so you can make the connections on each.



MM4006 with RS-485 Interface Option

	Pin #	
EARTH	1	1
TX+	2	2
TX-	3	3
RX-	4	4
RX+	5	5

EARTH	1	1
TX+	2	2
TX-	3	3
RX-	4	4
RX+	5	5

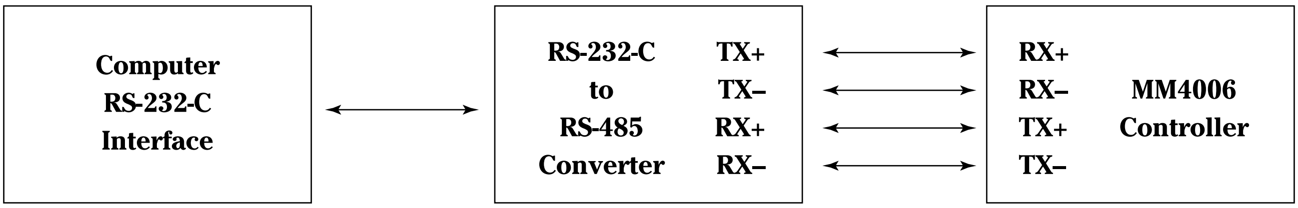
Connecting RS-232-C to a Protocol Converter

To use this communication protocol from a computer equipped with an RS-232-C serial port you must connect a RS-232-C to RS-485 protocol converter. A large choice of those converters can be found from the shelf. The following one are very popular ones and are not a limiting list: ROLINE IC-485S, ROLINE IC-485SI, Burr-Brown LDM485S. Refer to the protocol converter's to properly configure it and check it's connection to a RS-232-C interface. The above figure gives the standard RS-232-C pin-out and interconnection.

Computer RS-232-C Connector				**RS-232-C/RS-485 Converter**		
25 D-Sub Male	9 D-Sub Male	Pin Name		Pin Name	25 D-Sub Femal	25 D-Sub Femal
3	**2**	**TX**	→	**RX**	**2**	**3**
2	**3**	**RX**	←	**TX**	**3**	**2**
5	**8**	**RTS**	→	**CTS**	**5**	**4**
4	**7**	**CTS**	←	**RTS**	**4**	**5**
7	**5**	**GND**	—	**GND**	**7**	**7**

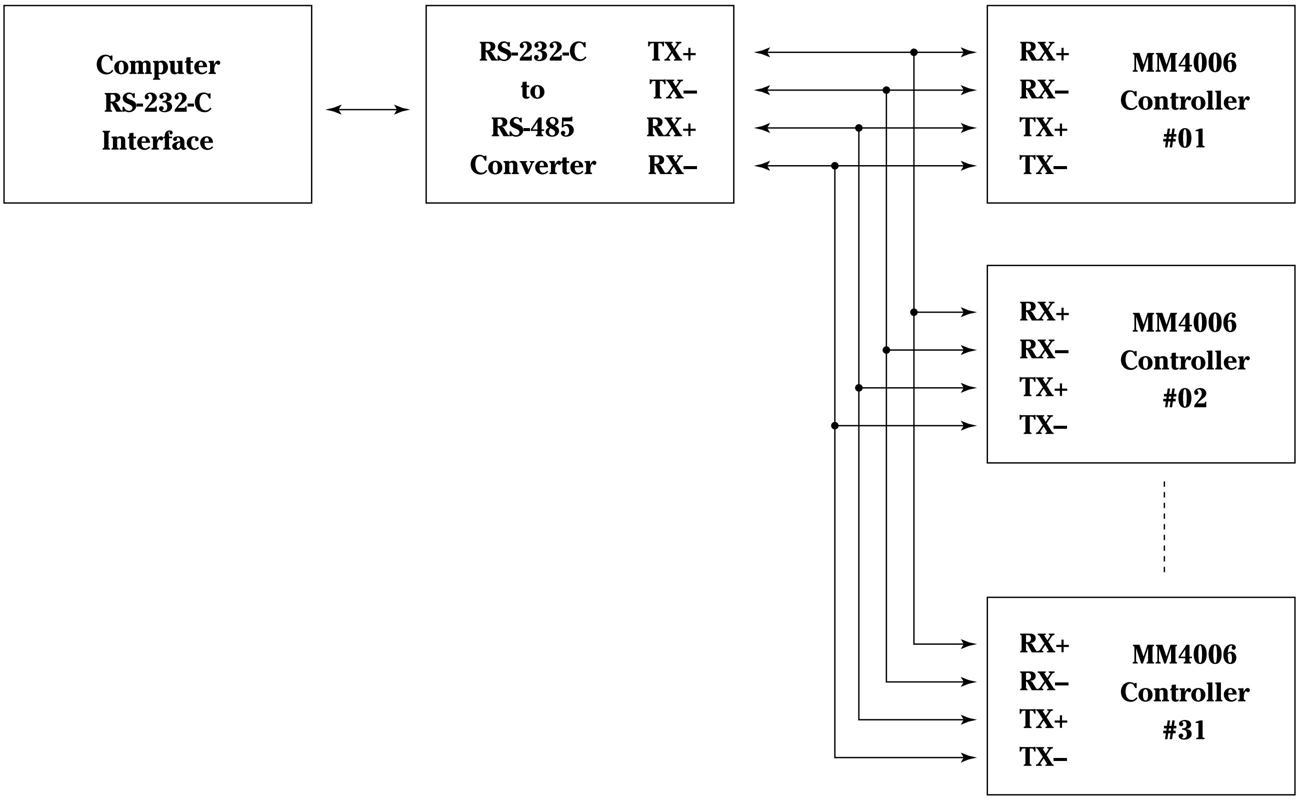
Point-to-Point Four Wires Full Duplex

This is the mode for single computer to a single MM4006 controller in long distance or noisy environment high speed communication. The following figure shows the how to connect your computer, or protocol converter to the MM4006 controller.



Multidrop Four Wires Full Duplex

This feature enables you to connect up to 31 MM4006 controllers to one serial communication port. As a network, each MM4006 controller will have its own address to identify the commands that are sent to it. The following figure shows the how to connect your computer, or protocol converter to several MM4006 controllers.



In this mode of communication each controller must have a single address, from 1 to 31, this address must be different for each controller.

In this network communication, if some controllers are switched off, the others will still continue to work. And, it is no delay in the communication, all controllers receive the commands at the same moment. This is more efficient than daisy chaining, in daisy chaining the computer send command to the first controller who repeat that command to the next one, who repeat to the next one and so on. The daisy chaining puts a lot of traffic on the communication line, introduce repeater delays and will not work if any of the controllers is switched off.

The standard command set of the MM4006 controller is directly usable with the following changes:

- Each command must be initiated with the string address:: to be understood by the right controller. For example for a single computer you send a command like 1OR (home axis 1), this same command will be 1::1OR (controller 1, home axis 1).
- For commands to which the controller has to respond, e.g.: 1TP tell position of axis 1, you should operate with care to avoid any collision on the communication lines. Only one controller should be asked to respond at a time and the computer must wait the reception of the response before interrogating an other controller. So to avoid some of the possible collisions, in this mode commands without axis number to which the controller has to respond will be ignored by the controller. For example commands like TP tell position of all axes will be ignored. To do the same the computer should issue these commands axis per axis and wait the response each time before issuing the next one.

Motor Interface Connector (25-Pin D-Sub)

This connector interfaces to the motion device. Depending on the type of driver and motor, some pins have different meanings. If not otherwise specified, this description is valid for all cases.

	Stepper Motors			DC Motors	
Pin #	UE16PP	UE16PPSC	UE31PP, UE41PP, UE41UP UE62PP, UE63PP	UE16CC, UE17CC, UE31CC, UE33CC, UE35CC, UE404S, UE404S2, UE511S	UE404CC, UE511CC, UE611CC
1	+ Phase 1	+ Phase 1	+ Phase 1	N.C.	+ Tacho Generator
2	N.C.	N.C.	+ Phase 1	N.C.	+ Tacho Generator
3	– Phase 1	– Phase 1	– Phase 1	N.C.	– Tacho Generator
4	N.C.	N.C.	– Phase 1	N.C.	– Tacho Generator
5	+ Phase 2	+ Phase 2	+ Phase 2	+ Motor	+ Motor
6	N.C.	N.C.	+ Phase 2	+ Motor	+ Motor
7	– Phase 2	– Phase 2	– Phase 2	– Motor	– Motor
8	N.C.	N.C.	– Phase 2	– Motor	– Motor
9	N.C.	N.C.	Middle Point [3] Phase 1	N.C.	N.C.
10	N.C.	N.C.	N.C.	N.C.	N.C.
11	N.C.	N.C.	Middle Point [3] Phase 2	N.C.	N.C.
12	N.C.	N.C.	N.C.	N.C.	N.C.
13	Mechanical Zero	Mechanical Zero	Mechanical Zero	Mechanical Zero	Mechanical Zero
14	Shield Ground	Shield Ground	Shield Ground	Shield Ground	Shield Ground
15	Index Pulse I Forcing (Level 1)	Index Pulse I Forcing (Level 1)	Index Pulse I	Index Pulse I [1]	Index Pulse I
16	0 V Logic	0 V Logic	0 V Logic	0 V Logic	0 V Logic
17	+ End-of-Run	N.C.	+ End-of-Run	+ End-of-Run	+ End-of-Run
18	– End-of-Run	N.C.	– End-of-Run	– End-of-Run	– End-of-Run
19	Encoder Phase A	N.C.	Encoder Phase A	Encoder Phase A	Encoder Phase A
20	Encoder Phase B	N.C.	Encoder Phase B	Encoder Phase B	Encoder Phase B
21	+5 V Encoder	N.C.	+5 V Encoder	+5 V Encoder	+5 V Encoder
22	0 V Encoder	N.C.	0 V Encoder	0 V Encoder	0 V Encoder
23	Encoder Phase $\overline{A}$	N.C.	Encoder Phase $\overline{A}$	Encoder Phase $\overline{A}$	Encoder Phase $\overline{A}$
24	Encoder Phase $\overline{B}$	N.C.	Encoder Phase $\overline{B}$	Encoder Phase $\overline{B}$	Encoder Phase $\overline{B}$
25	Index Pulse $\overline{I}$ Forcing (Level 0)	Index Pulse $\overline{I}$ Forcing (Level 0)	Index Pulse $\overline{I}$	Index Pulse $\overline{I}$ [2]	Index Pulse $\overline{I}$

1) *For UE16CC and UE17CC motors, the pin #15 is connected: Index Pulse I Forcing (Level 1).*

2) *For UE16CC and UE17CC motors, the pin #25 is connected: Index Pulse $\overline{I}$ Forcing (Level 0).*

3) *Except UE41UP motor: N.C.*

Pass-Through Board Connector (25-Pin D-Sub)

WARNING

This pass-through board connector takes the place of the motor interface connector only if this axis is connected to an external motor driver.

Pin #		Designation	
1	—	Ground	
2	—	5 V Encoder Supply	
3	—	**I**	Mechanical Zero
4	—	**I**	– End-of-Travel
5	—	**I**	+ End-of-Travel
6	—	**I**	Driver Fault Signal
7	—	**I**	Encoder Phase A
8	—	**I**	Encoder Phase B
9	—	**I**	Index Pulse I
10	—	**O**	Pulse Command (1)
11	—	**O**	Direction Command (1)
12	—	**O**	±10 V Analog Output (2)
13	—	**N.C.**	
14	—	0 V Encoder Supply	
15	—	**O**	Driver Inhibition Command
16	—	**N.C.**	
17	—	**N.C.**	
18	—	**N.C.**	
19	—	**I**	Encoder Phase $\overline{A}$
20	—	**I**	Encoder Phase $\overline{B}$
21	—	**I**	Index Pulse $\overline{I}$
22	—	0 V logic	
23	—	0 V logic	
24	—	**N.C.**	
25	—	**O**	Reference for ±10 V Analog Output

1) Stepper Motor Driver.

2) DC Motor Driver.

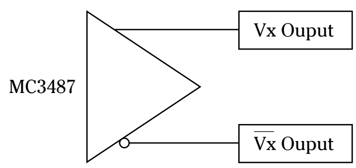


Fig. C.9 *— DiFF. Output Type.*

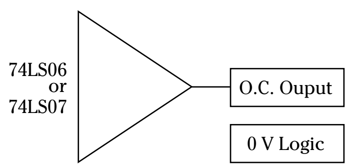


Fig. C.10 *— Open Collector Output Type.*

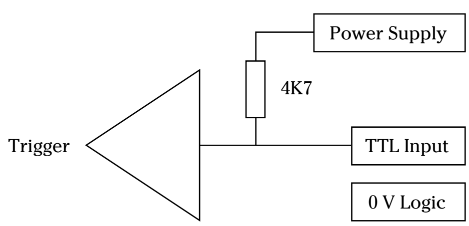


Fig. C.11 *— TTL Input Type.*

D — Motion Program Examples

When learning a new computer language, there is no substitute for actually writing some real programs. The motion controller's command set is a specialized language that needs to be mastered in order to be able to create complex applications. To help you familiarize yourself with MM4006 programming structure and language, this appendix contains a few examples that you can read and copy.

Example 1

The first example is a simple two-axes program that will generate the triangle shown in Fig. D.1.

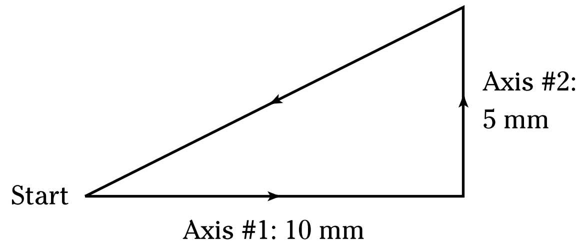
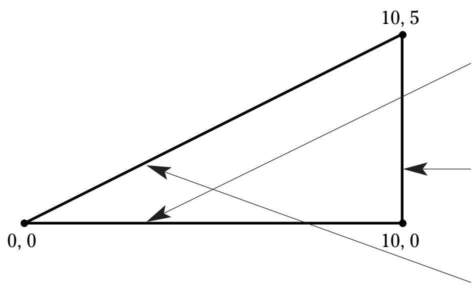


Fig. D.1 *— Triangle Pattern.*

Make sure there is no other program in memory with the same name (number). If you are operating the controller from a remote computer, start by issuing the XX command for that program number. Then, enter the programming mode by using the EP command. If you enter the program from the front panel, ignore these two and the QP commands.



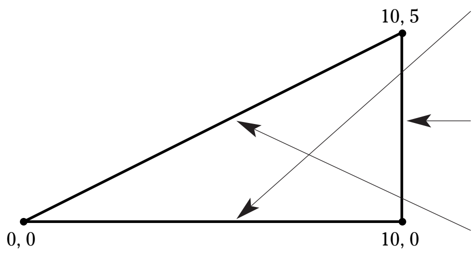
1XX	*Erase program #1, if it exists.*
1EP	*Enter programming mode and store all entries as program #1.*
1VA4	*Set velocity of axis #1 to 4 mm/sec.*
1PA10,1WS	*Move axis #1 to absolute position 10 mm; wait for axis #1 to complete motion.*
2VA4	*Set velocity of axis #2 to 4 mm/sec.*
2PA5,2WS	*Move axis #2 to absolute position 5 mm; wait for axis #2 to complete motion.*
2VA2	*Change velocity of axis #2 to 2 mm/sec.*
1PA0,2PA0	*Move axis #1 to absolute position 0 mm and axis #2 to absolute position 0 mm.*
1QP	*End of program; quit programming mode.*

Example 2

In the previous example, to generate the diagonal line (the third motion segment) both axes must move simultaneously. This is achieved by taking two special precautions: the commands are placed on the same line to insure a good start synchronization and the velocities are modified such that the motions will end in the same time.

But, if you would measure very accurately the precision of this diagonal line, you would notice some errors due to imperfect start synchronization and an incorrect acceleration ratio. In other words, we achieved this dual-axes motion with two independent single-axis motions.

To eliminate these motion errors, we need to use the axes synchronization (linear interpolation) feature. The improved program will have the following listing:



2XX	*Erase program #2, if it exists.*
2EP	*Enter programming mode and store all entries as program #2.*
1VA4	*Set velocity of axis #1 to 4 mm/sec.*
1PA10,1WS	*Move axis #1 to absolute position 10 mm; wait for axis #1 to complete motion.*
2VA4	*Set velocity of axis #1 to 4 mm/sec.*
2PA5,2WS	*Move axis #2 to absolute position 5 mm; wait for axis #2 to complete motion.*
1SY1,2SY1	*Declare axes #1 and #2 synchronized.*
1PA0,2PA0,SE,WS	*Set axis #1 destination to 0 mm and axis #2 destination to 0 mm; start synchronous motion; wait for motion to complete.*
1SY0,2SY0	*Declare axes #1 and #2 non-synchronized.*
2QP	*End of program #2; quit programming mode.*

Notice that there is no need to set the velocities before the synchronized (interpolated) motion. The controller automatically calculates them to get the best accuracy possible, without exceeding the pre-set individual velocities.

Also, when finished with an interpolated motion, always return the axes to the non-synchronized mode.

Example 3

The MM4006 does not offer true circular interpolation but in many cases less demanding applications can be successfully implemented.

Take the example of dispensing glue on the pattern shown in Fig. D.2.

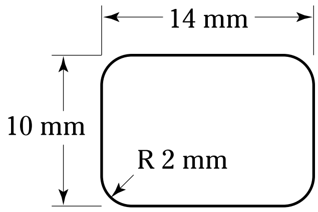


***Fig. D.2** — Glue Dispensing Pattern.*

Notice that there is no need to set the velocities before the synchronized (interpolated) motion. The controller automatically calculates them to get the best accuracy possible, without exceeding the pre-set individual velocities.

Also, when finished with an interpolated motion, always return the axes to the non-synchronized mode.

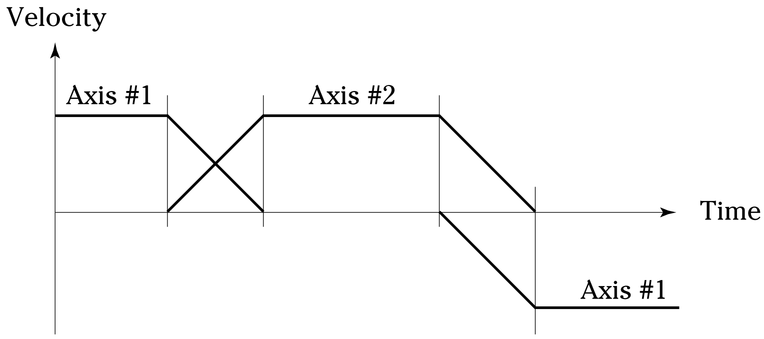


***Fig. D.3** — Overlapping Axis Acceleration/Deceleration.*

Assuming that the desired velocity is 4 mm/sec, we need to calculate the acceleration and the positions where one axis starts decelerating and the other accelerating.

We know that an axis must travel 2 mm before reaching a velocity of 4 mm/sec.

$$\text{Velocity} = \frac{\Delta\,\text{Distance}}{\text{Time}} \quad\Rightarrow\quad \text{Time} = \frac{\Delta\,\text{Distance}}{\text{Velocity}}$$

$$\text{Acceleration} = \frac{\Delta\,\text{Velocity}}{\text{Time}} = \Delta\,\text{Velocity} \bullet \frac{\text{Velocity}}{\Delta\,\text{Distance}}$$

Since the velocity starts from zero, Δ Velocity = Velocity.

$$\text{Acceleration} = \frac{\text{Velocity}^2}{\Delta\,\text{Distance}} = \frac{42}{2} = 8\ \text{mm/sec}^2$$

Before starting to write the actual program, we need to consider one more thing: to assure a good result, the glue must start being dispensed while the motion is in progress. Thus, we have to start the motion first and then turn on the dispenser.

The motion we decide to perform is shown in Fig. D.4.

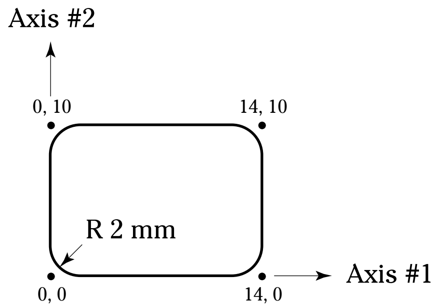
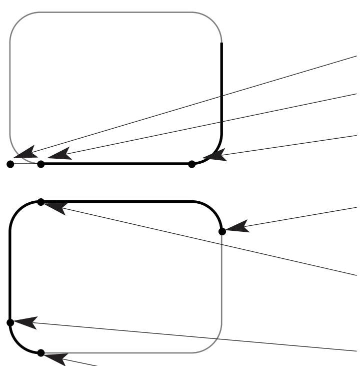


Fig. D.4 *— Desired Motion Result.*

The program will have the following listing:



3XX	*Erase program #3, if it exists.*
3EP	*Enter programming mode and store all entries as program #3.*
CB	*Clear all output I/O bits; set all bits to zero.*
1PA0,2PA0,WS	*Move axes #1 and #2 to absolute position 0 mm; wait for all axes to complete motion.*
1VA4,2VA4	*Set velocity of axes #1 and #2 to 4 mm/sec.*
1AC8,2AC8	*Set acceleration of axes #1 and #2 to 8 mm/s^2.*
1PA14	*Move axis #1 to absolute position 14 mm.*
1WP2,3SB	*Wait for axis #1 to reach position 2 mm; set bit #3.*
1WP12,2PA10	*Wait for axis #1 to reach position 12 mm; start axis #2 and move to position 10 mm.*
2WP8,1PA0	*Wait for axis #2 to reach position 8 mm; start axis #1 and move to position 0 mm.*
1WP2,2PA0	*Wait for axis #1 to reach position 2 mm; start axis #2 and move to position 0 mm.*
2WP2,1PA4	*Wait for axis #2 to reach position 2 mm; start axis #1 and move to position 4 mm.*
1WP2,3CB	*Wait for axis #1 to reach position 2 mm; clear bit #3.*
3QP	*End of program #2; quit programming mode.*

Example 4

Lets assume we want to write the from the Newport logo. We have a X-Y table and a 0.5 mm plotter pen (or a laser beam) controlled by a TTL line. One possibility is to scan the symbol with a 0.5 mm spacing and fill it in with 0.5 mm lines. The result will be similar to Fig. D.5.

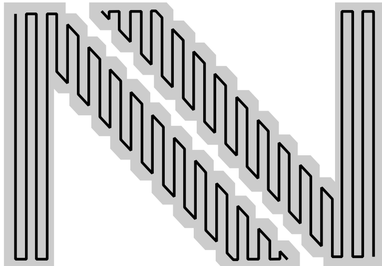


Fig. D.5.

The solid lines show the actual pen trajectory.

Next, we need to select a coordinate system. For simplicity, lets make the lower left corner of the trajectory the origin (zero), as shown in Fig. D.6.

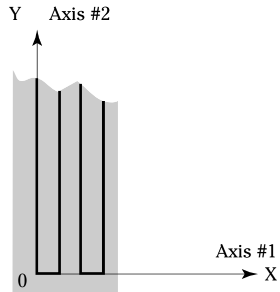
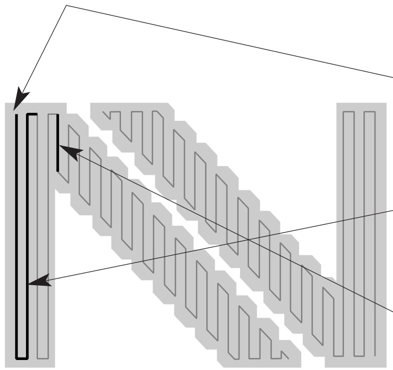


Fig. D.6.

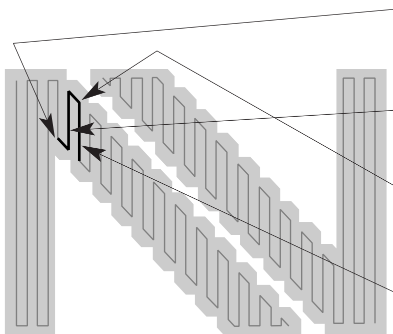
We decide to make the symbol 13 mm high and 17.5 mm wide. But, using a pen with a 0.5 mm wide tip, the actual trajectory must be shrunk to 12.5 ¥ 17 mm. To control the pen up and down we will use bit #8 of the I/O output port, where logic high means pen down.

First, we need to make sure that there is no other program in memory with the same name (number). We do this by listing the program number selected or just by erasing it with the XX command.

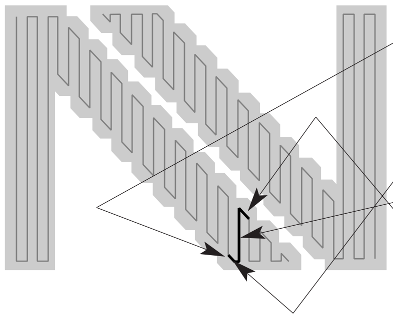
Assuming that this program is being edited on a computer and then downloaded to the controller, we also need to send the commands to enter and terminate the programming mode.



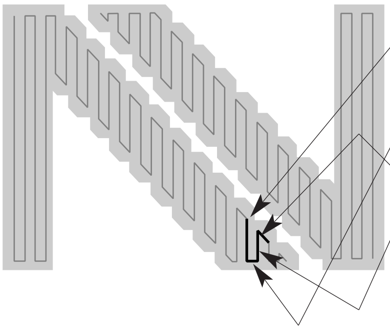
4XX	*Erase program #4, if it exists.*
4EP	*Store all following entries as program #1.*
CB	*Clear all output I/O bits; set all bits to zero.*
1PA0,2PA12.5,WS	*Move axis #1 to 0 mm and axis #2 to 12.5 mm, wait for all motion to complete.*
8SB	*Set I/O bit #8 high; this brings the pen down.*
2PR-12.5,WS,1PR0.5,WS,2PR12.5,WS,1PR0.5,WS,RP2	*Make four relative motions by sequentially incrementing axis #1 and #2; wait for each motion to stop; repeat the cycle (command line) two times.*
2PA10,WS	*Move axis #2 to 10 mm and wait for motion complete.*
1YS0	*Initialize variable #1; set its value to zero.*
1SY1,2SY1	*Declare axes #1 and #2 synchronized.*
1WL8	*Start a while loop; repeat the following commands while variable #1 is less than 8.*



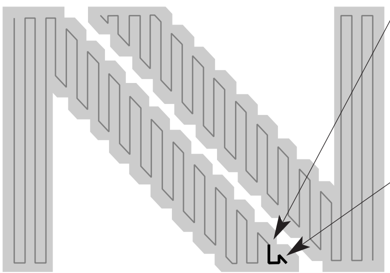
1PR0.5,2PR-0.596,SE,WS	*Set relative destination of axis #1 at 0.5 mm and of axis #2 at -0.596 mm away from current position; start synchronous motion; wait for motion to complete.*
2PR3,SE,WS	*Set relative destination of axis #2 3 mm away from current position; start motion on the synchronized axis; wait for motion to complete.*
1PR0.5,2PR-0.596,SE,WS	*Set relative destination of axis #1 at 0.5 mm and of axis #2 at -0.596 mm away from current position; start synchronous motion; wait for motion to complete.*
2PR-3,SE,WS	*Set relative destination of axis #2 -3 mm away from current position; start motion on the synchronized axis; wait for motion to complete.*
1YA1	*Increment variable #1 by 1.*
WE	*End while loop.*



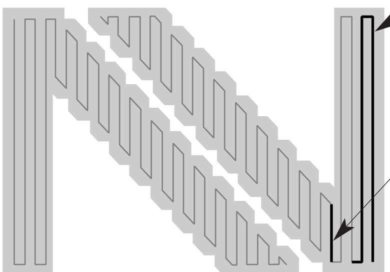
1PA10.35,2PA0,SE,WE	*Set destination of axis #1 to 10.35 mm and of axis #2 to 0 mm; start synchronous motion; wait for motion to complete.*
1PA10.5,SE,WS	*Set destination of axis #1 to 10.5 mm; start synchronized axis; wait for motion to complete.*
2PA2.979,SE,WS	*Set destination of axis #2 to 2.979 mm; start synchronized axis; wait for motion to complete.*
1PR0.5,2PR-0.596,SE,WS	*set relative destination of axis #1 at 0.5 mm and of axis #2 at -0.596 mm away from current position; start motion; wait for motion to complete.*



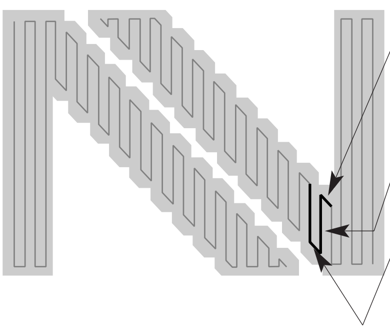
2PA0,SE,WS	*Set destination of axis #2 to 0 mm; start synchronized axis; wait for motion to complete.*
1PA11.5,SE,WS	*Set destination of axis #1 to 11.5 mm; start synchronized axis; wait for motion to complete.*
2PA1.788,SE,WS	*set destination of axis #2 to 1.788 mm; start synchronized axis; wait for motion to complete.*
1PR0.5,2PR-0.596,SE,WS	*set relative destination of axis #1 at 0.5 mm and of axis #2 at -0.596 mm away from current position; start synchronous motion; wait for motion end.*



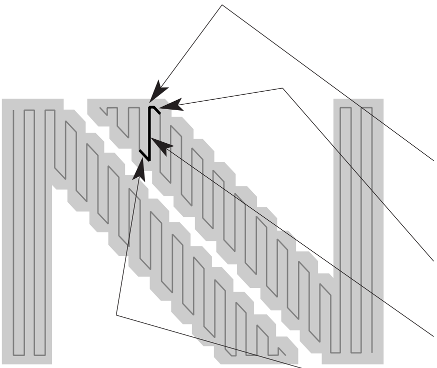
2PA0,SE,WS	*set destination of axis #2 to 0 mm; start synchronized axis; wait for motion to complete.*
1PA12.5,SE,WS	*Set destination of axis #1 to 12.5 mm; start synchronized axis; wait for motion to complete.*
2PA0.596,SE,WS	*Set destination of axis #2 to 0.596 mm; start synchronized axis; wait for motion to complete.*
1PA13,2PA0,SE,WS	*Set destination of axis #1 to 13 mm and of axis #2 to 0 mm; start motion; wait for motion to complete.*
1SY0, 2SY0	*Declare axes #1 and #2 non-synchronized.*
8CB	*Set I/O bit #8 low; this will lift the pen up.*
1PA17,WS	*Move axis #1 to 17 mm; start synchronized axis; wait for motion to complete.*
8SB	*Set I/O bit #8 high; this brings the pen down.*

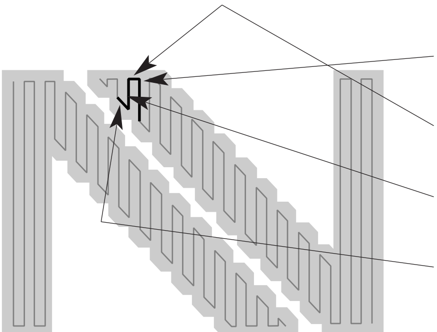


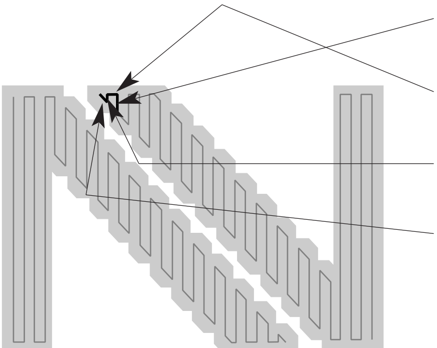
2PR12.5,WS,1PR-0.5,WS,2PR-12.5,WS,1PR-0.5,WS,RP2	*Make four relative motions by sequentially incrementing axis #1 and #2; wait for each motion to stop; repeat the cycle (command line) two times.*
2PA2.5,WS	*Move axis #2 to 2.5 mm and wait for motion complete.*
1YS0	*Initialize variable #1; set its value to zero.*
1SY1, 2SY1	*Declare axes #1 and #2 synchronized.*
1WL8	*Start a wile loop; repeat the following commands while variable #1 is less than 8.*



1PR-0.5,2PR0.596,SE,WS	*Set relative destination of axis #1 at -0.5 mm and of axis #2 at 0.596 mm away from current position; start motion; wait for motion to complete.*
2PR-3,SE,WS	*Set relative destination of axis #2 -3 mm away from current position; start motion on the synchronized axis; wait for motion to complete.*
1PR-0.5,2PR0.596,SE,WS	*Set relative destination of axis #1 at -0.5 mm and of axis #2 at 0.596 mm away from current position; start synchronous motion; wait for motion to complete.*







2PR3,SE,WS	*Set relative destination of axis #2 3 mm away from current position; start motion on the synchronized axis; wait for motion to complete.*
1YA1	*Increment variable #1 by 1.*
WE	*End while loop.*
1PA6.65,2PA12.5,SE,WS	*Set destination of axis #1 to 6.65 mm and of axis #2 to 12.5 mm; start synchronous motion; wait for motion to complete.*
1PA6.5, SE,WS	*Set destination of axis #1 to 6.5 mm; start synchronized axis; wait for motion to complete.*
2PA9.521,SE,WS	*Set destination of axis #2 to 9.521 mm; start synchronized axis; wait for motion to complete.*
1PR-0.5,2PR0.596,SE,WS	*Set relative destination of axis #1 at -0.5 mm and of axis #2 at 0.596 mm away from current position; start synchronous motion; wait for motion to complete.*
2PA12.5,SE,WS	*Set destination of axis #2 to 12.5 mm; start synchronized axis; wait for motion to complete.*
1PA5.5,SE,WS	*Set destination of axis #1 to 5.5 mm; start synchronized axis; wait for motion to complete.*
2PA10.712,SE,WS	*Set destination of axis #2 to 10.712 mm; start synchronized axis; wait for motion to complete.*
1PR-0,5,2PR0.596,SE,WS	*Set relative destination of axis #1 at -0.5 mm and of axis #2 at 0.596 mm away from current position; start synchronous motion; wait for motion to complete.*
2PA12.5,SE,WS	*Set destination of axis #2 to 12.5 mm; start synchronized axis; wait for motion to complete.*
1PA4.5,SE,WS	*Set destination of axis #1 to 4.5 mm; start synchronized axis; wait for motion to complete.*
2PA11.904,SE,WS	*Set destination of axis #2 to 11.904 mm; start synchronized axis; wait for motion to complete.*
1PA4,2PA12.5,SE,WS	*Set destination of axis #1 to 4 mm and of axis #2 to 12.5 mm; start synchronous motion; wait for motion to complete.*
8CB	*Set I/O bit #8 low; this will lift the pen up.*
1SY0,2SY0	*Declare axes #1 and #2 non-synchronized.*
4QP	*End of program; quit programing mode.*

E — Troubleshooting Guide

Remember that there are no user-serviceable parts or adjustments to be made inside the controller or any other component. Contact Newport for any repair or other hardware corrective action.

Most of the time, a blown fuse or an error reported by the controller is the result of a more serious problem. Fixing the problem should include not only correcting the effect (blown fuse, limit switch. etc.) but also the cause of the failure. Analyze the problem carefully to avoid repeating it in the future. The following is a list of the most probable problems and their corrective actions. Use it as a reference but keep in mind that in most cases a perceived error is usually an operator error or has a simple solution.

Problem	Cause	Corrective Action
Stand-By red LED does not come on	Rear power switch turned off	Turn on the main power switch located on the power entry module in the rear of the unit.
	No electrical power	Verify with an adequate tester or another electrical device (lamp, etc.) that the power is present in the outlet. If not, contact an electrician to correct the problem.
	Unplugged power cord	Plug the power cord in the appropriate outlet. Observe all caution notes and procedures described in the System Setup section.
	Blown fuse	Replace the line fuse as described in the System Setup section. Beware that the fuse blows only when a serious problem arises. If fuse blows again, contact Newport for service.
A physically present axis is declared unconnected	Bad connection	Turn power off and verify the motion device cable connection.
	Bad component	Turn power off and swap motor cable with another axis (if cables are identical) to locate the problem. Contact Newport for cable replacement or motion device service.

Problem	Cause	Corrective Action
The MOTOR ON green LED does not stay on	Limit switch tripped	Execute a home search routine or move the axis in manual mode (jog). Make sure that the limit switch was not tripped by a serious problem.
	Executive following error	Verify that teh motion device installed is connected to the proper driver card.
		Verify that all setup parameters correspond to the actual motion device installed.
		Verify that the load specifications for the motion device are not being exceeded.
The axis does not move	Incorrect connection	Verify that the motion device is connected to the correct driver card, as specified by the labels.
	Incorrect parameters	Verify that all relevant parameters (PID, velocity, etc.) are set properly.
System performance below expectations	Incorrect connection	Verify that the motion device is connected to the correct driver card, as specified by the labels.
	Incorrect parameters	Verify that all relevant parameters (PID, velocity, etc.) are set properly.
Motor excessively hot	Incorrect connection	Verify that the motion device is connected to the correct driver card, as specified by the labels.
Move command not executed	Software travel limit	The software travel limit in the specified direction was reached. If limits are set correctly, do not try to move past them.
	Incorrect parameters	Verify that all relevant parameters (PID, velocity, etc.) are set properly.
Home search not completed	Time-out too short	Verify the home search time-out is set correctly. If the home search velocity was changed, the time-out must be increased.
	Faultry origin or index signals	Carefully observe and record the motion sequence by watching the manual knob rotation, if available. With the information collected, call Newport for assistance.

Problem	Cause	Corrective Action
No remote communication	Wrong line	Make sure that the computer and the controller use the same line terminator.
	Wrong communication port	Verify that the controller is set to communication on the left port RS-232-C or IEEE-488.
	Wrong communication parameters	Verify that all communication parameters match between the computer and the controller.

NOTE

Many other type of problems are detected by the controller and reported on the display and/or in the error register. Consult appendix A for a complete list and description.

F — Decimal/ASCII/Binary Conversion Table

Some of the status reporting commands return an ASCII character that must be converted to binary. To aid with the conversion process, the following table converts all character used and some other common ASCII symbols to decimal and binary. To also help in working with the I/O port related commands, the table is extended to a full byte, all 256 values.

Number (decimal)	ASCII Code	Binary Code
0	*null*	00000000
1	*soh*	00000001
2	*stx*	00000010
3	*etx*	00000011
4	*eot*	00000100
5	*enq*	00000101
6	*ack*	00000110
7	*bel*	00000111
8	*bs*	00001000
9	*tab*	00001001
10	*lf*	00001010
11	*vt*	00001011
12	*ff*	00001100
13	*cr*	00001101
14	*so*	00001110
15	*si*	00001111
16	*dle*	00010000
17	*dc1*	00010001
18	*dc2*	00010010
19	*dc3*	00010011
20	*dc4*	00010100
21	*nak*	00010101
22	*syn*	00010110
23	*etb*	00010111
24	*can*	00011000
25	*em*	00011001
26	*eof*	00011010
27	*esc*	00011011
28	*fs*	00011100
29	*gs*	00011101
30	*rs*	00011110
31	*us*	00011111
32	*space*	00100000
33	**!**	00100001
34	**“**	00100010
35	**#**	00100011
36	**$**	00100100
37	**%**	00100101
38	**&**	00100110
39	**‘**	00100111
40	**(**	00101000
41	**)**	00101001
42	*****	00101010
43	**+**	00101011
44	**,**	00101100
45	**-**	00101101
46	**.**	00101110
47	**/**	00101111
48	**0**	00110000
49	**1**	00110001
50	**2**	00110010
51	**3**	00110011
52	**4**	00110100
53	**5**	00110101
54	**6**	00110110
55	**7**	00110111
56	**8**	00111000
57	**9**	00111001
58	**:**	00111010
59	**;**	00111011
60	**<**	00111100
61	**=**	00111101
62	**>**	00111110
63	**?**	00111111
64	**@**	01000000
65	**A**	01000001
66	**B**	01000010
67	**C**	01000011
68	**D**	01000100
69	**E**	01000101
70	**F**	01000110
71	**G**	01000111

Number (decimal)	ASCII Code	Binary Code	Number (decimal)	ASCII Code	Binary Code
72	**H**	01001000	120	**x**	01111000
73	**I**	01001001	121	**y**	01111001
74	**J**	01001010	122	**z**	01111010
75	**K**	01001011	123	**{**	01111011
76	**L**	01001100	124	**|**	01111100
77	**M**	01001101	125	**}**	01111101
78	**N**	01001110	126	**~**	01111110
79	**O**	01001111	127		01111111
80	**P**	01010000	128		10000000
81	**Q**	01010001	129		10000001
82	**R**	01010010	130		10000010
83	**S**	01010011	131		10000011
84	**T**	01010100	132		10000100
85	**U**	01010101	133		10000101
86	**V**	01010110	134		10000110
87	**W**	01010111	135		10000111
88	**X**	01011000	136		10001000
89	**Y**	01011001	137		10001001
90	**Z**	01011010	138		10001010
91	**[**	01011011	139		10001011
92	****	01011100	140		10001100
93	**]**	01011101	141		10001101
94	**^**	01011110	142		10001110
95	**_**	01011111	143		10001111
96	**‘**	01100000	144		10010000
97	**a**	01100001	145		10010001
98	**b**	01100010	146		10010010
99	**c**	01100011	147		10010011
100	**d**	01100100	148		10010100
101	**e**	01100101	149		10010101
102	**f**	01100110	150		10010110
103	**g**	01100111	151		10010111
104	**h**	01101000	152		10011000
105	**i**	01101001	153		10011001
106	**j**	01101010	154		10011010
107	**k**	01101011	155		10011011
108	**l**	01101100	156		10011100
109	**m**	01101101	157		10011101
110	**n**	01101110	158		10011110
111	**o**	01101111	159		10011111
112	**p**	01110000	160		10100000
113	**q**	01110001	161		10100001
114	**r**	01110010	162		10100010
115	**s**	01110011	163		10100011
116	**t**	01110100	164		10100100
117	**u**	01110101	165		10100101
118	**v**	01110110	166		10100110
119	**w**	01110111	167		10100111

Number (decimal)	ASCII Code	Binary Code	Number (decimal)	ASCII Code	Binary Code
168		10101000	212		11010100
169		10101001	213		11010101
170		10101010	214		11010110
171		10101011	215		11010111
172		10101100	216		11011000
173		10101101	217		11011001
174		10101110	218		11011010
175		10101111	219		11011011
176		10110000	220		11011100
177		10110001	221		11011101
178		10110010	222		11011110
179		10110011	223		11011111
180		10110100	224		11100000
181		10110101	225		11100001
182		10110110	226		11100010
183		10110111	227		11100011
184		10111000	228		11100100
185		10111001	229		11100101
186		10111010	230		11100110
187		10111011	231		11100111
188		10111100	232		11101000
189		10111101	233		11101001
190		10111110	234		11101010
191		10111111	235		11101011
192		11000000	236		11101100
193		11000001	237		11101101
194		11000010	238		11101110
195		11000011	239		11101111
196		11000100	240		11110000
197		11000101	241		11110001
198		11000110	242		11110010
199		11000111	243		11110011
200		11001000	244		11110100
201		11001001	245		11110101
202		11001010	246		11110110
203		11001011	247		11110111
204		11001100	248		11111000
205		11001101	249		11111001
206		11001110	250		11111010
207		11001111	251		11111011
208		11010000	252		11111100
209		11010001	253		11111101
210		11010010	254		11111110
211		11010011	255		11111111

G — Factory Service

Introduction

This section contains information regarding factory service for the MM4006. The MM4006 contains no user-serviceable parts. The user should not attempt any maintenance or service of this instrument and/or accessories beyond the procedures outlined in the Troubleshooting Guide, Appendix E. Any problem that cannot be resolved should be referred to Newport Corporation or your Newport representative for assistance.

Obtaining Service

To obtain information about factory service, contact Newport Corporation or your Newport representative. Please have the following information available:

1 Instrument model number (MM4006).

2 Instrument serial number.

3 Firmware version number.

4 Description of the problem.

If the instrument is to be returned for repair, you will be given a Return Authorization Number, which you should refer to in your shipping documents. Please fill out the service form on the next page and return the completed form with your system.

Service Form

Your Local Representative
Tel.: ______________________
Fax: ______________________

Name: ______________________ Return Authorization #: ______________________
(Please obtain prior to return of item)

Compagny: ______________________

Adress: ______________________ Date: ______________________

Country: ______________________ Phone Number: ______________________

P.O. Number: ______________________ Fax Number: ______________________

Item(s) Being Returned:

Model #: ______________________ Serial #: ______________________

Description: ____

Reasons of return of goods (please list any specific problems): ______________________

Newport®

Section 8
Appendices

Table of Contents
Section 3 — Remote Mode

Section 3
Remote Mode

3.1 Remote Interfaces

In this manual, Remote Interface refers to the three communication interfaces that the controller can use to communicate with a computer or a terminal via commands in ASCII format. It is not called a Computer Interface since any device capable of sending ASCII characters can be interfaced with the controller.

The Remote Interface should not be confused with the analog and digital I/Os. These interfaces communicate with the controller via discrete lines, with specific functions, without using any motion commands. They are used to synchronize external events in complex motion systems.

Selecting the interface

The MM4006 controller is equipped with RS-232-C, RS-485 or IEEE-488 interface. Selecting the interface and setting the parameters is done through the General SETUP menu on the front panel.

NOTE

For more details on setting up communication parameters see the Controller Configuration paragraph of the Local Mode chapter.

3.1.1 RS-232-C Interface

Hardware configuration

The serial (RS-232-C) communication port is a 9-pin D-Sub connector located on the rear panel. The pinout is designed to interface directly with an IBM PC or compatible computer, using a one-to-one cable. No special adapters are required.

Appendix C shows the pinout of the RS-232-C connector and different cable types that may be used to interface to a computer.

Communication protocol

The RS-232-C interface must be properly configured on both devices communicating. A correct setting is one that matches all parameters (baud rate, number of data bits, number of stop bits, parity type and handshake type) for both devices.

RS-232-C communication parameters are set through the General SETUP menu on the front panel. To make changes, follow the instructions in the Controller Configuration paragraph of the Local Mode chapter.

3.1.2 IEEE-488 Interface

NOTE

See Appendix B: “IEEE-488 Link Characteristics”.

Hardware configuration

The IEEE-488 interface has a well defined hardware configuration. The MM4006 conforms to the standard so you simply need to connect the proper cable to the clearly identified connector on the back panel.

Communication protocol

The IEEE-488 interface is implemented on the MM4006 somewhat differently than on a typical instrument. The standard IEEE-488 command set and command format are inadequate for a complex motion controller. Since the MM4006 has its own language and command set, the IEEE-488 is used only as a communication port. The extended protocol is not supported. The only exception is the use of the SRQ line, which permits more reliable data transfer, especially when downloading large amounts of data (trace data, large programs, etc.) The SRQ can be enabled or disabled from the General Setup menu on the front panel.

The main setup requirement for an IEEE-488 device is to select the proper address. This identifies the unit to the other devices connected to the system.

To change the address or the SRQ usage, follow the instructions in the Controller Configuration section of the Local Mode chapter.

3.1.3 RS-232-C Interface

The RS-485 interface permits:

- The communication between a single computer to a single MM4006 controller in long distance (several hundred-meters) or noisy environment high speed communication.
- To meet the requirements for a multi-drop communication network up to 31 MM4006 controllers.

Hardware configuration

The RS-48 communication port is a 10-pin connector. The MM4006 controller is equipped with 2 identical connectors connected in parallel, located on the rear panel. The pinout is designed to interface directly with an IBM PC or compatible computer, using a one-to-one cable. No special adapters are required.

Appendix C shows the pinout of the RS-485 connector.

Communication protocol

The RS-485 interface must be properly configured on both devices communicating. A correct setting is one that matches all parameters (adress, baud rate, number of data bits, number of stop bits, parity type and handshake type) for both devices.

3.2 Softwares

In order to communicate with the controller, the user must have a terminal or a computer capable of communicating with external devices via a RS-232-C, RS-485 or IEEE-488 interface.

One approach is to use a communications software that can emulate a terminal. An other solution is to use available NEWPORT MOTION Suite 32 softwares.

3.2.1 MOTION Suite 32

MOTION Suite 32 is a set of Windows™ applications for the MM4006 Motion Controller.

MOTION Suite 32 requires the following minimum configuration:

- An IBM PC or 100% compatible.
- A VGA monitor.
- An 80486 or later processor.
- 16 MB of available memory.
- A CD-ROM drive.
- A hard disk.
- A Windows™ environment.

This software only accepts the following interfaces:

- RS-232-C: COM1 or COM2 standard serial port.
- RS-485 interface port.
- IEE-488: GPIB National Instrument® board, model AT-GPIB/TNT.

ATTENTION

Before uisng NEWPORT MOTION Suite 32 softwares, the IEEE-488 board listed above must be into the initial configuration of the constructor, and installed in accordance with its recommended procedure.

Before uisng NEWPORT MOTION Suite 32 softwares, we advise you to use softwares utilities supplied with the IEEE-488 board to check that the installation is completed successfully.

MOTION Suite 32 is a set of 3 softwares:

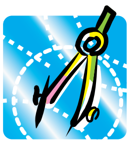
- MM4006 Terminal.
- MM4006 Servo.
- MM4006 XY Draw.
- MOTION Prog.

3.2.2 MM4006 Terminal

MM4006 Terminal is an application which permits to communicate with the Newport MM4006 Controller. It offers the possibility to change communication configuration, and send commands to the controller directly, or since a file or a file containing a MM4006 program.

3.2.3 MM4006 Servo

MM4006 Servo is an application which permits to set PID servo loop parameters of mechanical axes controlled by the Newport MM4006 Controller. It automatically controls the MM4006, reads and calculates all important dynamic motion parameters. Results are plotted on graphs corresponding to position, velocity or following error.



3.2.4 MM4006 XY Draw

MM4006 XY Draw is an application which permits to perform the linear and the circular interpolation of the Newport MM4006 controller. It presents you a draw area where you can draw complex trajectory with lines and arcs. It converts the drawing in MM4006 specific commands.

NOTE

A complete description of operatings and features of these programs can be found in the MOTION Suite 32 Getting Started.

Section 3

Remote Mode

3.3 MOTION Suite 32 Quick Start

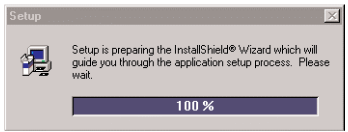
3.3.1 Installation of the MOTION Suite 32

MOTION Suite 32 is a package of three softwares:

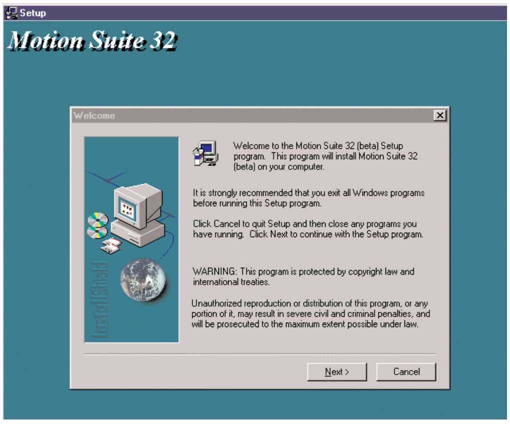
- MM4006 Terminal.
- MM 4006 Servo.
- MM 4006 XY Draw

MOTION Suite 32 is supplied on an installation CD-ROM.

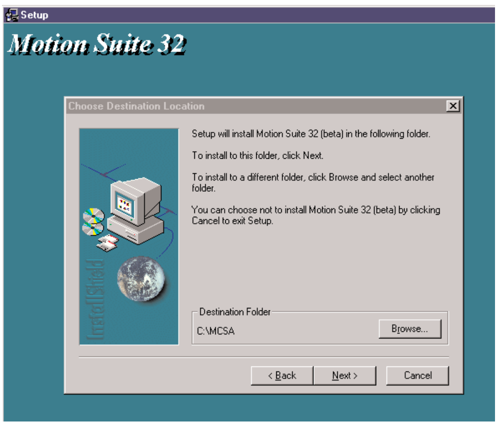
1) Forinstallation of softwares, execute SETUP.EXE from the directory ENGLISN/DISK 1 of CD-ROM.



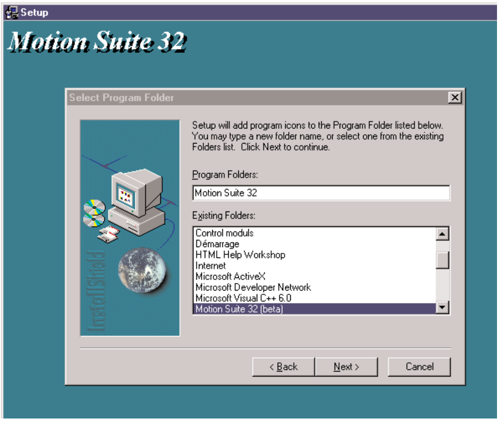
2) Follow the set up procedure stepwise.



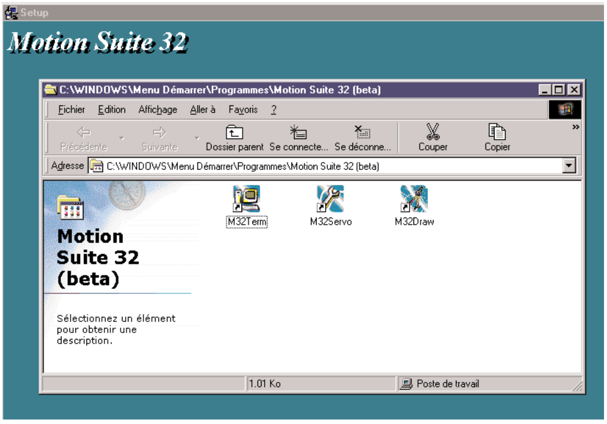
3) Follow the set up procedure stepwise.



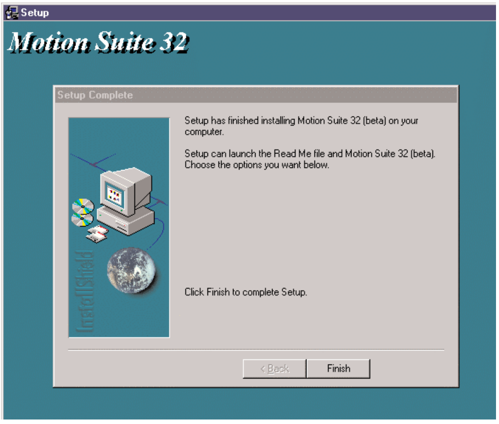
4) Adding programme icons and folder names.



5) After the file set up, the MOTION Suite 32 files will appear as icons.



6) Set up completed successfully.



3.3.2 PC/Controller Communication

Connecting the controller to a PC

Connection is made with a RS-232-C cable (see Appendix C).

Connect one end of the cable to the RS-232-C connector of the MM4006 controller and the other end to the serial port #1 of the PC.

Configuring controller communication

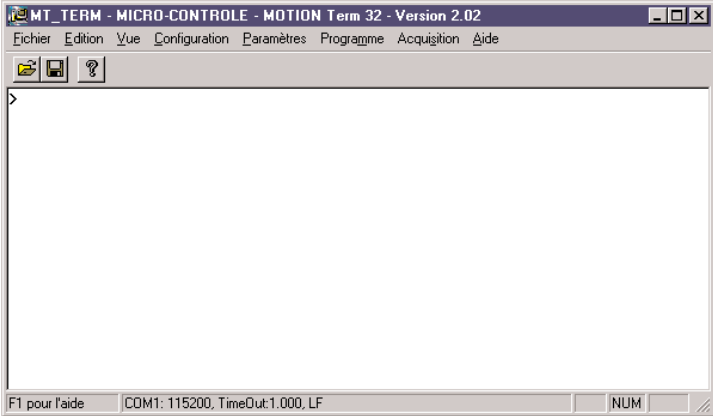
- Switch ON the controller with the power switch in rear panel and the STAND-BY button in front panel.
- Switch OFF the motor with the MOTOR OFF button.
- Select the CONFIG menu then the GEN. menu.
- Scroll the general parameters with the help of UP or DOWN keys.
- To change a parameter select the MODIFY menu.
- To change the value of the parameter, select the CHANGE key.
- To save the new value, select the VALID key.

Example:

- Press on DOWN key until the **Terminator** parameter appears.
- Select the MODIFY menu.
- Select the CHANGE key until the terminator is at **LF**.
- Save the selection with the VALID key.
- Configure the communication parameters of the MM4006 as follows:
  - Terminateur: LF
  - Communication: RS232
  - Mode XON/XOFF: NON
  - Speed: 19200
  - Parity: None
  - Length: 8
  - Stop Bits: 1
- When the communication parameters are configured, press on the QUIT key.
- To go back to the main scren, press on the QUIT key.

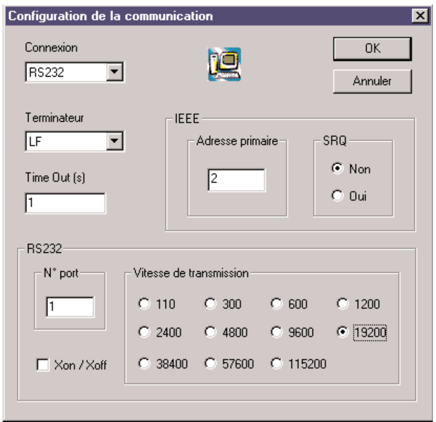
PC communication configuration

Running **MM4006 Terminal** under Windows environment.



Select the **Configuration** menu to set the PC communication parameters. They shouls be the same as for the controller:

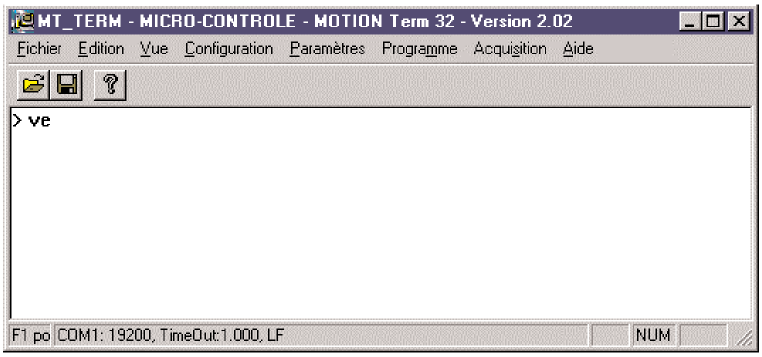
- Connection: **RS232**
- Terminator: **LF**
- Port #: **1**
- XON/XOff: **NO**
- Transmission speed: **19200**



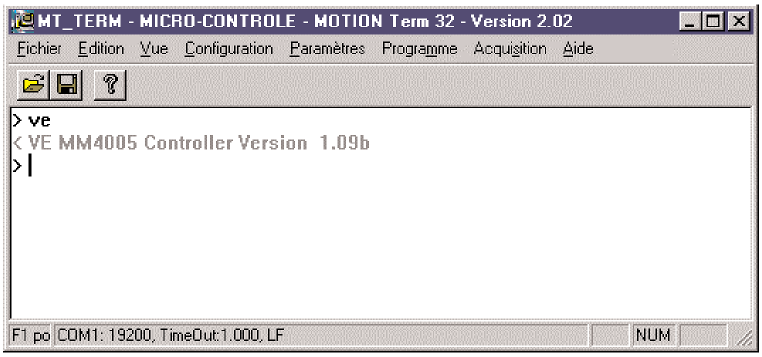
Press on **OK** button to confirm the configuration.

Sending an instruction

Enter **VE** command in the dialogue box of the Terminal and press **ENTER** key to send the instruction towards the controller.



Once the instruction is sent and if the link is established, the controller acknowledges receipt and sends a message to the PC. The message appears in adifferent colour in the dialogue box.



3.3.3 Axis Motion Control

Axis configuration

- Switch OFF the controller with the power switch in rear panel and the STAND-BY button in front panel.
- Plug in the electronic card into axis slot #1 of the controller.
- Switch ON the controller with the power switch in rear panel and the STAND-BY button in front panel.
- Switch OFF the motor with the MOTOR OFF button.
- Select the CONFIG menu then the AXES menu.
- Press on the AXIS# key to select the axis to be configued.
- Press on key 1 of the keypad of the controller and confirm with the VALID key.
- Press the SELECT key to select the type of product to be driven.
- With UP or DOWN keys, scroll through the list of product types.
- To select a particular family of product, select the VALID key.
- Go through the list of models available with the help of the UP or DOWN keys.
- To select a particular model, press the VALID key.
- Validate the Pos. Origine, KP, KI, KD, KS parameters with the help of the VALID key.
- Press QUIT.
- Select the MODIFY menu to pour change the parameters of the selected axis.
- Go through the axis parameters with the help of the UP or DOWN keys.
- To change a parameter, select the MODIFY menu.
- Press QUIT.
- Select the YES key to save all axis #1 parameters in the EEPROM.
- To go back to the main menu press QUIT.

Axis motion control

Run **MM4006 Terminal** and configure the communication paramùeters as indicated previously.

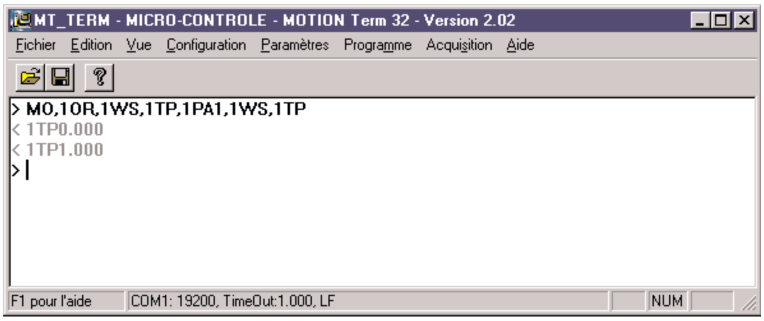
Enter the following instructions:

MO,1OR,1WS,1TP,1PR1,1WS,1TP

To execute the instructions press **ENTER**.

Instructions carried out:

MO	: Switch on motor.
1OR	: Search origin on axis #1.
1WS	: Wait until end of motion on axis #1.
1TP	: Read position on axis #1.
1PR1	: Cary out relative motion on axis #1.
1WS	: Wait until end of motion on axis #1.
1TP	: Read position on axis #1.



3.3.4 Acquisition

Acquisition for all axes

Run **MM4006 Terminal**, and configure the communication parameters as indicated previously.

Enter the instructions below and press **ENTER** each time:

SQ0.001

GQ10,1PR-1,1WS,NQ

TQ

Instructions carried out:

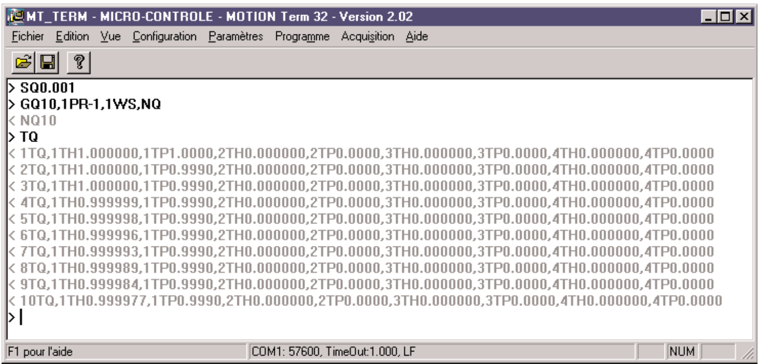
SQ0.001 : Set sampling rate in seconds.

GQ20 : Start a series of 20 acquisitions on all axes.

1PR-1 : Starts relative motion on axis #1.

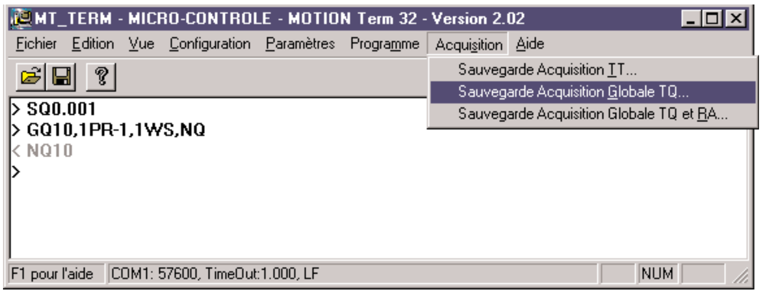
NQ : Reads the number of acquisitions done.

TQ : Read data acquisition buffer TQ.



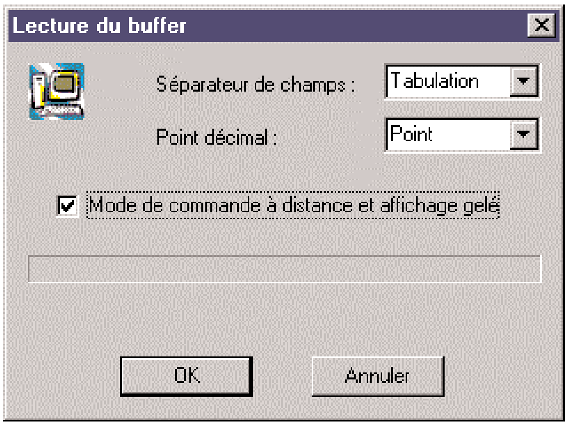
Save all data capture in a text file

To read the buffer **TQ** and to save the content in a text file, select **Acquisition** menu. Then select the sub-menu **Global Acquisition TQ Record...**



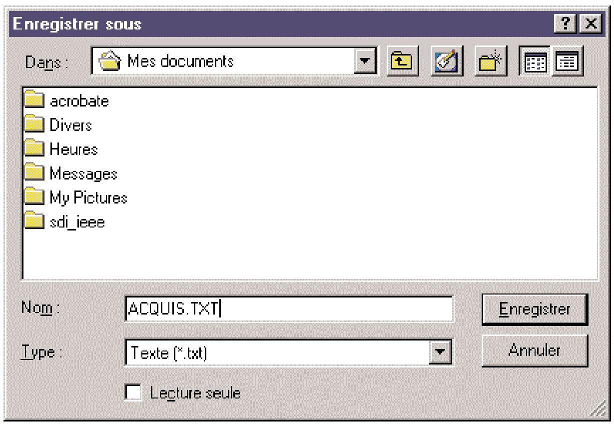
Select filed separator (tabulation or space) and decimal point indicator (“.” or “,”).

Activate the remote command mode and the conntroller display during buffer reading. Press **OK** to start buffer reading.

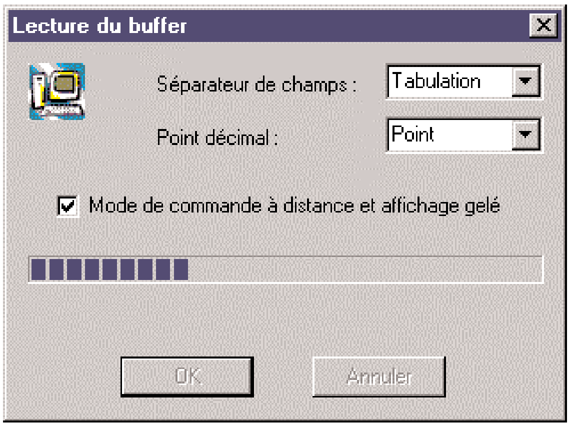


Select name and path of the text file in which all acquisitions were saved.

Press on **SAVE** to save all data.



Progress of buffer reading is indicated.



Abstract of the text file:

10	0.00090011566	0	0	0	0	0	0
1	1	0	0	0	0	0	0
1	0.999	0	0	0	0	0	0
1	0.999	0	0	0	0	0	0
0.999999	0.999	0	0	0	0	0	0
0.999998	0.999	0	0	0	0	0	0
0.999996	0.999	0	0	0	0	0	0
0.999993	0.999	0	0	0	0	0	0
0.999989	0.999	0	0	0	0	0	0
0.999984	0.999	0	0	0	0	0	0
0.999977	0.999	0	0	0	0	0	0

3.4 Communication Principles

The MM4006 controller follows simple conventions when interfacing with a computer or terminal. Please read them carefully since they are the basis of the remote mode operation.

RS-232-C or IEEE-488?

The MM4006 always listens to one of the two remote interfaces but never to both in the same time. This is done to avoid potential conflicts that could occur if two computers are trying to control a motion device at the same time.

Command lines

The MM4006 responds only to command line instructions. This means that no single or multiple character command is executed until a line terminator is received. Section 3.4.1 describes in detail the rules associated with the command format.

Controller responses

The MM4006 does not send any data out over the communication line unless asked to do so. Even in the case of an error, the controller does not send anything back. If an error is suspected, the user must query the controller, usually with the TE command. This is particularly useful when designing complex programs using custom environments. There is no need to constantly check the communication buffer if no transmission request was made. During the application development, the error buffer can be continuously checked. When the program is finished and everything works fine, the error queries can be eliminated to reduce unnecessary overhead.

Communication buffer

The controller has a separate input buffer and output buffer, each 4096 characters wide. A single command line, however, may not exceed 110 characters.

3.4.1 Command Syntax

Command format

yy:	xx	AA	nn

yy — Controller adress: 0 to 31.

NOTE

Use the "yy" controller adress only in RS-485 communication mode.

xx — Optional or required preceding.

AA — Command code.

nn — Parameter can be represented by:

- A value

or

- An interrogation **"?"** (for certains commands);

or

- A variable: **$Ypp** or **$Saa** (for certains commands).

  pp [int]: value variable number
  1 to 100 : integers
  101 to 120 : floats

  aa [int]: String variable number
  1 to 8 : strings

The general format of a command is a two character mnemonic (**AA**). Both upper and lower case are accepted. Depending on the command, it could also have optional or required preceding (**xx**) and/or following (**nn**) parameters.

Blank spaces

Blank spaces are allowed and ignored in any position, including inside a numerical value. For the clarity of the program and memory saving considerations, use blank spaces with restraint. The following two commands are equivalent:

2P A1.43 6

2PA1.436

but the first example is very confusing and uses more than twice the memory.

Command line

Commands are executed line by line. A line can consist of one or a number of commands. The controller will interpret the commands in the order they are received and then they are executed, usually within a few microseconds. This means that commands issued on the same line are executed significantly closer to each other than if they would be if issued on separate lines. The maximum number of characters allowed on a command line is 110.

Separator

Commands issued on the same line must be separated by a comma (,) or semicolons (;).

Terminator

Each command line, to be executed or accepted in a program, must end with a line terminator. The terminator must have the format defined in the GENERAL SETUP mode. The controller supports all combinations of line feed (LF) and carriage return (CR) combinations: LF, CR, LF/CR and CR/LF.

3.5 Command Summary

The MM4006 controller understands 194 commands. The following two tables list them all, sorted first by category and then in alphabetical order. The tables also show the modes in which each command can be used. The mode mnemonics used in the tables have the following meaning:

IMM	**IMM**ediate mode	Controller is idling and the commands are executed immediately.
PGM	**P**ro**G**ra**M** mode	Controller does not execute but stores all commands as part of a program. **EP** activates this mode and **QP** exits it.
MIP	**M**otion **I**n **P**rogress	Controller executes a motion on the specified axis.

For the Command Description section, an empty box in front of the mode designator indicates the command not being valid in that particular mode of operation.

Parameters in brackets (e.g. [xx]) indicate optional parameters.

3.5.1 Command List by Category

Command	Description	IMM	PGM	MIP
	General mode selection			
xx **CD** nn	Set cycle value and activate periodic display mode	■	■	□
CM [nn]	Change communication mode	■	■	□
MC	Set manual mode	■	■	□
[xx] **MF**	Motor OFF	■	■	■
ML	Set local mode	■	■	□
MO	Motor ON	■	■	□
MR	Set remote mode	■	■	□
QW	Save general parameters	■	■	□
	Motion and position control			
AB	Abort motion	■	□	■
[xx] **DH**	Define home	■	■	□
xx **MT** nn	Move to travel limit switch	■	■	□
[xx] **OR** [nn]	Search for home	■	■	□
xx **PA** nn	Move to absolute position	■	■	□
xx **PR** nn	Move to relative position	■	■	□
SE	Start synchronized motion	■	■	□
[xx] **ST**	Stop motion	■	■	■
[xx] **ZP**	Zero position	■	■	□
	Trajectory definition parameters			
xx **AC** nn	Set acceleration	■	■	■
xx **DA** pp	Read desired acceleration	■	■	■
[xx] **DF**	Read following error	■	■	■
[xx] **DP**	Read desired position	■	■	■
xx **DV** pp	Read desired velocity	■	■	■
xx **MV** + or -	Infinite movement	■	■	■
SD nn	Speed scaling	■	■	□
[xx] **TH**	Read theoretical position	■	■	■
[xx] **TP**	Read actual position	■	■	■
xx **VA** nn	Set velocity	■	■	■
xx **VB** nn	Set base velocity (Stepper motor only)	■	■	■
	Special motion parameters			
xx **AU** nn	Set maximum acceleration	■	■	□
xx **DM**	Read manual velocity	■	■	■
xx **DO**	Read home search velocity	■	■	■
xx **HT** nn	Set home type	■	■	□
xx **MH** nn	Set manual velocity	■	■	□
xx **OA** nn	Set home search acceleration	■	■	□
xx **OH** nn	Set home search high velocity	■	■	□
xx **OL** nn	Set home search low velocity	■	■	□
xx **PB** nn	Set start position of generation of pulses of synchronisation	■	■	□
xx **PE** nn	Set end position of generation of pulses of synchronisation	■	■	□
xx **PI** nn	Set step of generation of pulses of synchronisation	■	■	□
xx **PS** pp	Allow generation of pulses on motion	■	■	□
xx **PT** nn	Calculate necessary time for axis displacement	■	■	□
xx **SU** nn	Set encoder resolution	■	■	□
xx **SV** nn	Set stepper motor resolution	■	■	□
xx **SY** nn	Axis synchronization	■	■	□
xx **VU** nn	Set maximum velocity	■	■	□
xx **VW** nn	Set scaling velocity	■	■	□
xx **XH**	Read home preset position	■	■	■
	Trace mode			
xx **AQ** nn	Axis positions acquisition	■	■	■
GQ nn	Set global trace mode	■	■	■
NQ	Read global acquisition nr.	■	■	■
SP [nn]	Set trace sample rate	■	■	■
SQ [nn]	Set global sample rate	■	■	■
xx **TM** nn	Set trace mode	■	■	■
[xx] **TQ** [nn]	Read global trace data	■	■	□
[xx] **TT**	Read trace data	■	■	□
XN	Read number of acquisitions	■	■	■
XQ	Read global sample rate	■	■	■
XS	Read trace sample rate	■	■	■

Commande	Description	IMM	PGM	MIP
	Digital filter parameters			
xx **FE** nn	Set maximum following error	■	■	□
xx **KD** nn	Set derivative gain	■	■	□
xx **KI** nn	Set integral gain	■	■	□
xx **KP** nn	Set proportional gain	■	■	□
xx **KS** nn	Set saturation level of integral factor in position loop PID corrector	■	■	□
[xx] **PW**	Save parameters	■	■	□
xx **TF**	Read filter parameters	■	■	□
[xx] **UF**	Update servo filter	■	■	■
xx **XD**	Read derivative gain factor	■	■	■
xx **XF**	Read maximum following error	■	■	■
xx **XI**	Read integral gain factor	■	■	■
xx **XP**	Read proportional gain factor	■	■	■
	Motion device parameters			
xx **BA** [nn]	Set backlash compensation	■	■	□
xx **SC** [nn]	Set control loop type	■	■	□
xx **SF** name	Set axis mechanical motion device	■	■	□
xx **SL** nn	Set left travel limit	■	■	□
xx **SN** name	Set axis displacement units	■	■	□
xx **SR** nn	Set right travel limit	■	■	■
xx **TA**	Read motion device	■	■	■
xx **TC**	Read control loop type	■	■	■
xx **TL**	Read left travel limit	■	■	■
xx **TN**	Read displacement units	■	■	■
xx **TR**	Read right travel limit	■	■	■
xx **TU**	Read encoder resolution	■	■	■
xx **XB**	Read backlash compensation	■	■	■
xx **ZH** nn	Set and save home preset position	■	■	□
xx **ZL** nn	Set and save left travel limit	■	■	□
xx **ZR** nn	Set and save right travel limit	■	■	□
[xx] **ZT** [nn]	Read Axis/General parameters configuration	■	□	■
	I/O functions			
xx **AM** nn	Set analog input mode	■	■	■
[xx] **CB** [nn]	Clear I/O outputs bits	■	■	■
FT nn	Set output frequency	■	■	■
[xx] **RA**	Read analog input	■	■	■
[xx] **RB**	Read I/O input	■	■	■
[xx] **RO**	Read I/O output	■	■	■
[xx] **SB** [nn]	Set I/O output bits	■	■	■
SO [nn]	Set I/O output byte	■	■	■
[xx] **TG** [nn]	Toggle I/O output bits	■	■	■
xx **YO** nn	Send a value to an user analog port	■	■	□
xx **YR** nn	Read a value from an user analog port and affect variable	■	■	■
	Programming			
AP	Abort program	■	■	■
xx **CP**	Compile program	■	□	□
xx **EO** nn	Automatical execution on power on	■	■	□
xx **EP** nn	Edition of program	■	□	□
xx **EX** [nn]	Execute a program	■	□	□
xx **LP**	List program	■	□	□
MP	Download EEPROM to RAM	■	□	□
QP	Quit program mode	■	□	□
SM	Save program	■	□	□
xx **XL** nn	Delete one line of program	■	□	□
XM	Read available memory	■	■	■
[xx] **XX**	Erase program	■	□	□

Command	Description	IMM	PGM	MIP
	Flow control and sequencing			
CT [nn]	Read or set the controller time	■	■	■
xx DL	Define label	□	■	□
[xx] IE nn	If I/O input is equal	■	■	□
xx JL	Jump to label	□	■	□
KC	Abort command line	■	□	■
[xx] OE nn	Test I/O output	■	■	□
RP [nn]	Repeat command line	■	■	□
RQ nn	Generate service request (SRQ)	■	■	□
[xx] UH	Wait for I/O high	□	■	□
[xx] UL	Wait for I/O low	□	■	□
WA [nn]	Wait	■	■	□
WE	End While loop	□	■	□
xx WF	Wait for function key	□	■	□
xx WG [nn]	While variable is greater	□	■	□
xx WH [nn]	While I/O input is equal	□	■	□
WK [aa]	Wait for key	□	■	□
xx WL [nn]	While variable is less	□	■	□
xx WP nn	Wait for position	■	■	□
[xx] WS [nn]	Wait for motion stop	■	■	□
WT [nn]	Wait	■	■	□
xx WY [nn]	While variable is different	□	■	□
xx YE [nn]	If variable is equal	□	■	□
xx YG [nn]	If variable is greater	■	■	□
xx YL [nn]	If variable is less	■	■	□
xx YN [nn]	If variable is different	■	■	□
xx YW	Wait and read key	□	■	□
	Variable manipulation			
xx AS nn	Affect string	■	■	□
xx CS nn	Concatenate two strings	■	■	□
xx TY	Read a variable	■	■	■
xx YA [nn]	Add to variable	□	■	□
xx YB	Negate variable	□	■	□
xx YC nn	Add variables	■	■	□
xx YD nn	Divide variables	■	■	□
xx YF nn	Scale variable	■	■	□
xx YK	Read key to variable	■	■	□
xx YM nn	Multiply variables	■	■	□
xx YP nn	Set theoretical position in variable	■	■	□
xx YQ nn	Set current position in variable	■	■	□
xx YS [nn]	Initialize variable	■	■	□
xx YV	Read value from keyboard in a variable	■	■	□
xx YY nn	Copy variable	■	■	□
	Display functions			
xx DS [nn]	Display strings on screen	□	■	□
xx DY nn	Display a variable	□	■	□
xx FB [aa]	Label function key	□	■	□
FC	Clear function key line	□	■	□
FD	Display function keys	□	■	□
xx NP nn	Set decimal digits number of position display	■	■	□
RD	Disable display refresh	■	■	■
RE	Enable display refresh	■	■	■
	Status functions			
ED nn	Display program error	■	■	□
[xx] MS	Read motor status	■	■	■
TB [aa]	Read error message	■	■	■
TD	Read error line of program	■	■	□
TE	Read error code	■	■	■
TS	Read controller status	■	■	■
TX	Read controller activity	■	■	■
TX1	Read controller extended status	■	■	■
VE	Read controller version	■	■	■

Commande	Description	IMM	PGM	MIP
	Commands to define a trajectory			
AD nn	Define the maximum allowed angle of discontinuity	■	■	□
xx AX	Assign a physical axis as X geometric axis	■	■	□
xx AY	Assign a physical axis as Y geometric axis	■	■	□
CA nn	Define sweep angle and build an arc of circle = ƒ (CR, CA)	■	■	□
CR nn	Define radius for an arc of circle = ƒ (CR, CA)	■	■	□
CX nn	Define X position to reach with an arc of circle = ƒ (CX, CY)	■	■	□
CY nn	Define Y position to reach and build an arc of circle = ƒ (CX, CY)	■	■	□
EL	Erase the last element of trajectory	■	■	□
FA nn	Define the tangent angle for the first point	■	■	□
LX nn	Define X position and build a line segment = ƒ (LX, tangent)	■	■	□
LY nn	Define Y position and build a line segment = ƒ (LY, tangent)	■	■	□
MX nn	Define X position for a line segment = ƒ (MX, MY)	■	■	□
MY nn	Define Y position and build a line segment = ƒ (MX, MY)	■	■	□
NT	Start definition of a new trajectory	■	■	□
	Commands to execute a trajectory			
ET	Execution of trajectory	■	■	□
VS nn	Define the vector acceleration on trajectory (trajectory acceleration)	■	■	■
VV nn	Define the vector velocity on trajectory (trajectory velocity)	■	■	■
WI nn	Wait for a trajectory (curvi-linear) length	□	■	□
WN nn	Wait for a element of trajectory	□	■	□
	Commands to help geometric definition of a trajectory			
AT	Tell the element number under execution	■	■	■
xx LT	Extended list of the trajectory	■	■	□
XA	Tell the current maximum allowed angle of discontinuity	■	■	□
XE	Tell the last element	■	■	□
XT	Tell number of elements in the trajectory	■	■	□
XU nn	Tell the vector acceleration on trajectory (trajectory acceleration)	■	■	■
XV nn	Tell the vector velocity on trajectory (trajectory velocity)	■	■	■
	Master-slave mode definition			
xx FF nn	Set maximum master-slave following error	■	■	□
xx GR nn	Set master-slave reduction ratio	■	■	□
xx SS np	Set master-slave mode	■	■	□
	Trace mode on trajectory			
NB nn	Set trajectory element where the generation of pulses starts	■	■	□
NE nn	Set trajectory element where the generation of pulses ends	■	■	□
NI nn	Set step (curvi-linear distance) between synchronisation pulses	■	■	□
NN nn	Set number of synchronisation pulses to generate	■	■	□
NS	Allow generation of pulses on interpolation	■	■	□

AB — Abort motion

Usage	■ IMM ☐ PGM ■ MIP
Syntax	**AB**
Parameters	None.
Description	This command is an emergency stop. On reception of this command, the controller stops motion on all axes with a fast deceleration and then turns motor power OFF. It should be used only as an immediate command, not in a program.
Returns	None.
Errors	None.
Rel. Commands	**AP** — Abort program. **KC** — Abort command line. **MF** — Motor OFF. **MO** — Motor ON. **ST** — Stop motion.
Example	**AB** | *Used as an immediate command to stop motion.*

Usage	■ IMM	■ PGM	■ MIP
Syntax	**xxACnn**		

Parameters

Description	**xx** [int]	—	Axis number.
	nn [float]	—	Acceleration value.
Range	**xx**	—	**1** to **8.**
	nn	—	**10^{-6}** to **the programmed value in SETUP mode**.
Units	**xx**	—	None.
	nn	—	preset units in SETUP mode/sec^2.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command sets the acceleration/deceleration value for an axis. Its execution is immediate, meaning that the acceleration is changed when the command is processed, even while a motion is in progress. All subsequent accelerations and decelerations will be executed with the new value.

NOTE

The user-set acceleration is not saved in the nonvolatile memory. After power-on, the controller will use the default value (the maximum allowed acceleration).

NOTE

Avoid changing the acceleration during the acceleration or deceleration periods. For more predictable results, change acceleration only when the axis is not moving or when it is moving at a constant speed.

Returns None.

Errors	A	—	Unknown message code.
	B	—	Incorrect axis number.
	C	—	Parameter out of limits.
Rel. Commands	**AU**	—	Set maximum acceleration.
	DA	—	Read desired acceleration.
	PA	—	Move to absolute position.
	PR	—	Move to relative position.
	VA	—	Set velocity.

Example

2DA	*Read desired acceleration of axis #2.*
2DA10	*Controller returns an acceleration value of 10 units/sec^2.*
2PA15	*Move to absolute position 15 units.*
WT500	*Wait for 500 ms.*
2AC4	*Set axis #2 acceleration to 4 units/sec^2.*
2DA	*Read acceleration of axis #2.*
2DA4	*Controller returns an acceleration value of 4 units/sec^2.*

AD — Define the maximum allowed angle of discontinuity

Usage ■ IMM ■ PGM □ MIP

Syntax **ADnn** or **AD?**

Parameters

Description	**nn** [double]	—	Maximum allowed discontinuity angle value.
or	**nn** [?]	—	Read the actual maximum allowed.
Range	**nn**	—	**0.001** to **10.0.**
Units	**nn**	—	Degrees.
Defaults	**nn**	Missing:	0.001.
		< Min. value:	0.001.
		> Max. value:	Error W.

Description This command defines to the controller what will be the maximum allowed angle of discontinuity between two element of trajectory. This value will be used only for the elements of trajectory that will be defined after this command.

NOTE

On power up, the controller assumes that the maximum allowed discontinuity angle is equal to 0.001 degree.

NOTE

This value is necessary only when an a line segment element of type (MY, MY) followed by any other kind of elements.

NOTE

Before changing this value it is important to check if it is reasonable to change it. It is very complex to determine what is a suitable value for a given application because a lot of parameters that act on this value (Load condition of stages, type of stages, vector velocity, acceleration,...). When changing this value the precision on the trajectory will change.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the actual maximum allowed discontinuity angle value.

Errors

S — Communication time-out.
W — Trajectory: too big discontinuity angle.

Rel. Commands **XA** — Tell the current maximum allowed angle of discontinuity.

Example

AD0.1	*Define 0.1 degree as maximum discontinuity angle.*
AD?	*Request the value of the maximum allowed angle of discontinuity.*
AD0.1	*Controller returns a value of 0.1 degree.*

AM — Set analog input mode

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxAMnn** or **[xx]AM?**

Parameters

Description
- **xx** [int] — Analog input port number.
- **nn** [int]] — Analog input mode.

Range
- **xx** — **0** to **4.**
- **nn** — **0** to **3**.

Units
- **xx** — None.
- **nn** — None.

Defaults
- **xx** Missing: 0.
  Out of range: Error E.
  Floating point: Error E.
- **nn** Missing: 0.
  Out of range: Error C.

Description The MM4006 controller possess four analog inputs that user can program each input tension level with the AM command.

- If **nn** = 0 or missing: ±10 V tension input range.
- If **nn** = 1: ±5 V tension input range.
- If **nn** = 2: 0 to 10 V tension input range.
- If **nn** = 3: 0 to 5 V tension input range.

Returns If the **"?"** sign takes place of **nn** parameter and **xx** is missing, the controller returns the actual analog input mode.

Errors
- C — Parameters out of limits.
- E — Incorrect I/O channel number.

Rel. Commands
- **YR** — Read a value from an user analog port and affect variable.
- **YO** — Send a value to an user analog port.

Example

1AM2	*Set 0 to 10 V analog range at the input port #1.*
1AM?	*Request the actual analog input port #1 mode.*
1AM2	*Controller returns the actual analog input port #1 mode.*
AM	*Initializes all analog input ports to default mode (±10 V).*
AM?	*Request all actual analog input modes.*
AM, 1AM0, 2AM0, 3AM0, 4AM0	*Controller returns all actual analog input modes.*

AP — Abort program

Usage	■ IMM ■ PGM ■ MIP
Syntax	**AP**
Parameters	None.
Description	This command interrupts a motion program in execution. It will not stop a motion in progress. It will only stop the program after the current command line is finished executing. It can be used as an immediate command or inside a program. Inside a program it is useful in conjunction with program flow control commands. It could, for instance, terminate a program on the occurrence of a certain external event, monitored by an I/O bit.
Returns	None.
Errors	None.
Rel. Commands	**EX** — Execute a program.

Example

3EX	*Execute program #3.*
…	
…	
…	
AP	*Stop program execution.*

AQ — Axis positions acquisition

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxAQnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [int]	—	Integer value.
Range	**xx**	—	**1** to **8.**
	nn	—	**0** or **1**.
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	0.
		Out of range:	Error C.

Description This command records the actual position of:

- If **xx** = 0: all axis and analog inputs in the global trace buffer at the buffer actual pointer position.
- If **xx** = 1 to 8: the actual position of axis #**xx** in the axis trace buffer at the buffer actual pointer position.

The buffer actual pointer position is incremented of 1. To set the global trace buffer actual pointer position to zero, use GQ0. To set the axis trace buffer actual pointer position to zero, use TM0.

If **nn** = 1: this command generates one pulse at output on pin 12 (if **xx** = 0), or pin 11 (if **xx** = 1 to 8), of the 25-pin auxiliary connector, at the moment of command execution.

NOTE

xx = 0: **If the global acquisition mode is active (GQnn command with nn ≠ 0), this command will desactive this mode.**

xx ≠ 0: **If the axis acquisition mode is active (TMnn command with nn ≠ 0), this command will desactive this mode.**

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands **GQ** — Set global trace mode.

Example

GQ0	*Initialisation of global trace buffer.*
NT, FA90	*Initialisation of trajectory.*
CR10, CA 5	*Element 1.*
CA350	*Element 2.*
CA5	*Element 3.*
VV5	*Set trajectory velocity to 5 units/sec.*
ET	*Displacement with generation of pulses.*
WN2, **AQ**	*At the beginning of element 2, axis positions are enregistred without synchronization pulse.*
WN3, **AQ1**	*At the beginning of element 3, axis position are enregistred with a synchronization pulse.*

AS — Affect string

Usage	■ IMM	■ PGM	□ MIP

Syntax **xxASaa** or **xxAS?**

Parameters

Description	**xx** [int]	—	String variable number.
	aa [str]	—	String to be affected.
Range	**xx**	—	**1** to **8.**
	aa	—	**0** or **32** characters.
Units	**xx**	—	None.
	aa	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error C.
		Floating point:	Error A.
	nn	Missing:	Null string; clears string.
		Out of range:	Only first 32 characters are used.

Description This command affects a string in a string variable.
If **xx** is missing, this command erases all string variable (from 1 to 8).

Returns If the sign **"?"** takes place of **aa** and **xx** is different of zero, this command reportes actual **xx** string buffer content.

Errors	A	—	Unknown message code.
	C	—	Parameter out of limits.
	D	—	Unauthorized execution.
Rel. Commands	**CS**	—	Concatenates two strings.
	DS	—	Display strings on screen.

Example

1AS"This "	*Affects "This " in variable S1 (S1 = "This ").*
2AS"is"	*Affects "is" in variable S2 (S2 = "is").*
3AS" "	*Affects " " in variable S3 (S3 = " ").*
1CS$S2	*Concatenate S2 to S1 (S1 = "This is").*
1CS$S3	*Concatenate S3 to S1 (S1 = "This is ").*
1CS"a string"	*Concatenate "a string" to S1.*
DS$S1	*Contents of variable S1.*
This is a string	*Displayed on the controller's screen.*

AT — Tell the element number under execution

Usage	■ IMM ■ PGM ■ MIP
Syntax	**AT**
Parameters	None.
Description	This command retrieves from the controller the element number of the trajectory that is currently being executed.
Returns	**ATnn** **nn** — Element number.
Errors	S — Communication time-out.
Rel. Commands	**XT** — Tell number of elements in the trajectory. **LT** — Extended list of the trajectory.
Example	ET | *Execute trajectory.* **AT** | *Read current element number.* *AT1* | *Controller returns 1.*

AU — Set maximum acceleration

Usage	■ IMM	■ PGM	□ MIP

Syntax **xxAUnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [float]	—	Maximum acceleration value.
Range	**xx**	—	**1** to **8.**
	nn	—	>10^{-6}.
Units	**xx**	—	None.
	nn	—	preset units in SETUP mode/sec^2.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command sets the maximum acceleration value for an axis (see "Acceleration" in the Section "Local Mode" of this user's manual). This is the maximum value that the AC command can not exceed.

Returns If the sign **"?"** takes place of **nn**, this command reports the actual maximum acceleration of axis **#xx**.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands **AC** — Set acceleration.

Example **2AU80** | *Set axis # 2 maximum acceleration to 80 units/sec^2.*

AX — Assign a physical axis as X geometric axis

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxAX** or **AX?**

Parameters

Description	**xx** [int]	—	Physical axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.

Description This command tells to the controller which physical axis will be the X geometric axis for the next trajectory that will be loaded. ET command will verify the correct assignation in execution.

Returns If AX? takes place of the xxAX, this command reportes the actual number of assigned as X geometric axis.

Errors

B	—	Incorrect axis number.
S	—	Communication time-out.

Rel. Commands

AY	—	Assign a physical axis as Y geometric axis.
LT	—	Extended list of the trajectory.

Example

1AX	*Assign physical axis 1 as X geometric axis.*
AX?	
AX1	*Controller return value 1.*

AY — Assign a physical axis as Y geometric axis

Usage	■ IMM	■ PGM	□ MIP
Syntax	**xxAY** or **AY?**		
Parameters			
Description	**xx** [int]	—	Physical axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.

Description This command tells to the controller which physical axis will be the Y geometric axis for the next trajectory that will be loaded. ET command will verify the correct assignation in execution.

Returns If AY? takes place of the xxAY, this command reportes the actual number of assigned as Y geometric axis.

Errors

B — Incorrect axis number.
S — Communication time-out.

Rel. Commands

AX — Assign a physical axis as X geometric axis.
LT — Extended list of the trajectory.

Example

2AY	*Assign physical axis 2 as Y geometric axis.*
AY?	
AY2	*Controller return value 2.*

BA — Set backlash compensation

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxBAnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [float] — Backlash compensation value.
Range	**xx** — **1** to **8.** **nn** — **0** to **distance equivalent to 10000 encoder counts**.
Units	**xx** — None. **nn** — Defined motion units.
Defaults	**xx** Missing: Error B. Out of range: Error B. Floating point: Error A. **nn** Missing: 0. Out of range: Error C.

Description

This command initiates a backlash compensation algorithm when motion direction is reversed. The controller keeps track of the motion sequence and for each direction change it adds the specified **nn** correction. Setting **nn** to zero disables the backlash compensation.

NOTE

The command is active only after a home search or home set (OR or DH) is performed on the specified axis.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands **XB** — Read backlash compensation.

Example

OR	*Perform a home search on all installed axes.*
1BA0.0012	*Set backlash compensation of axis #1 to 0.0012 units.*
2BA0.0008	*Set backlash compensation of axis #2 to 0.0008 units.*

CA — Define sweep angle and build an arc of circle = *f* (CR, CA)

Usage ■ IMM ■ PGM □ MIP

Syntax **CAnn**

Parameters

Description	**nn** [double]	—	Sweep angle for an arc of circle.
Range	**nn**	—	-1.7×10^{304} to -1.0×10^{-12}.
		and	10^{-12} to 1.7×10^{304}.
Units	**nn**	—	Defined motion units.
Defaults	**nn**	Missing:	Error C.

Description This command defines to the controller the sweep angle and tells to the controller to build an element of trajectory of the type:
Arc of circle = *f*(CR, CA).

NOTE

The convention for the sweep angle is the following:

- **Sweep angle > 0 then it is used as Counter Clock Wise.**
- **Sweep angle < 0 then it is used as Clock Wise.**

Returns None.

Errors

C	—	Parameter out of limits
H	—	Calculation overflow.
V	—	Too long trajectory.
\	—	Type error (arc expected).
]	—	Trajectory: Arc (r, θ) radius is too small.
^	—	Trajectory: Arc (r, θ) radius is too big.
_	—	Trajectory: Arc (r, θ) sweep angle is too small.

Rel. Commands

CR	—	Define radius for an arc of circle = *f* (CR, CA).
XE	—	Tell the last element.

Example

NT	*Clear trajectory.*
CR10	*Define radius of an arc of circle = f (r, θ).*
CA90	*Build an arc of circle = f (r = 10 units, θ = 90°).*
CA480	*Build an arc of circle = f (r = 10 units, θ = 480°).*
XE	*Tell last element.*
XE, Arc (r, θ), 10, 10, 90	*Controller tells the built element.*

CB — Clear I/O outputs bits

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxCBnn**

Parameters

Description
xx [int] — I/O bit number.
nn [float] — I/O bit mask.

Range
xx — **0** to **8.**
nn — **0** to **255.**

Units
xx — None.
nn — None.

Defaults
xx Missing: 0.
Out of range: Error E.
Floating point: Error A.
nn Missing: 255.
Out of range: Error C.
Floating point: Decimal part truncated.

Description

This command clears one to all output bits of the I/O port. If **xx** is specified between 1 and 8, the **nn** mask must be missing and then the selected bit will be cleared.

If **xx** is missing or set to 0 and **nn** is between 1 and 255, the controller will clear all bits corresponding to the mask. For example, if **nn** is 140, the equivalent binary mask is 10001100 which means that I/O output bits number 3, 4 and 8 will be cleared (remember that I/O bits are numbered from 1 to 8).

If **xx** is missing or set to 0 and **nn** is not specified, the controller clears all 8 bits. This is equivalent to setting **xx** to 0 and **nn** to 255.

NOTE

Remember that having an open collector configuration, a clear bit means a non-conductive transistor. Using a pull-up resistor, a clear output bit will measure a logic high, thus making the output port be the reverse logic type.

NOTE

For the hardware definition of the I/O port, please see Appendix B, Connector Pinouts, GPIO Connector.

Returns None.

Errors
A — Unknown message code.
E — Incorrect I/O channel number.

Rel. Commands
RO — Read I/O output.
SB — Set I/O output bits.
SO — Set I/O output port byte.
TG — Toggle I/O output bits.

Example
CB224 | *Set I/O output port bits number 6, 7 and 8 low.*

CD — Set cycle value and activate periodic display mode

Usage ■ IMM ■ PGM □ MIP

Syntax **xxCDnn** or **xxCD?**

Parameters

Description
xx [int] — Axis number.
nn [double] — New value of cycle value.

Range
xx — **0** to **4.**
nn — **0** to **Distance equivalent to 1932735283 encoder counts** (0.9 MAXLONG)**.**
and **-(Distance equivalent to 1932735283 encoder counts)** to **0**.

Units
xx — None.
nn — Actual displacement unit (mm, µm, In…).

Defaults
xx Missing: Error B.
Out of range: Error B.
Floating point: Error A.
nn Missing or 0: Stop periodic display mode.
Out of range: Error C.

Description

This command sets new value of cycle in the periodic display mode and activate this mode.

During axe movement, in each cycle the displayed values of positions change between 0 and **nn**, as followings:

- If **nn** > 0: Start 0, end **nn** if positive displacement.
  Start **nn**, end 0 if negative displacement.
- If **nn** < 0: Start **nn**, end 0 if positive displacement.
  Start 0, end **nn** if negative displacement.

This command has effect not only on infinite movements (MV+, MV-), but also on other types of displacements (PA, PR, manual, joystick).

To set off this mode of display, use xxCD or xxCD0.

NOTE

If this command is used in conjunction with the SS command and GR command, the slave axis cycle value must be equal to the master axis cycle value multiplied by the master-slave reduction ratio.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the value of display cycle if the periodic display mode has been activated.

Errors
A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands
MV — Infinite movement.
PA — Move to absolute position.
PR — Move to relative position.

Example

2SFurm80app	*Set mechanical driver to URM80APP.*
2CD360	*Set cycle value to 360° and activate the mode.*
2MV+	*Infinite displacement with periodic display.*

CM — Change communication mode

Usage ■ IMM ■ PGM □ MIP

Syntax **CM[Mxx][Txx][Axx][Qxx][Bxx][Pxx][Lxx][Sxx][Oxx][Rxx][Xxx]** or **CM?**

Parameters

Description

M	—	Communication mode.
xx = 1		IEEE-488.
xx ≠ 1 or **missing**		RS-232-C.
T	—	Terminator character.
xx = 1		CR.
xx = 2		LFCR.
xx = 3		CRLF.
xx ≠ 1, 2, 3 or **missing**		LF
A	—	IEEE address.
xx		**0** to **31**
R	—	Reserved.
xx = 0		
Q	—	IEEE-488 SRQ mode.
xx = 1		YES.
xx ≠ 1 or **missing**		NO SRQ
B	—	Serial transmission speed.
xx		**1200**, **2400**, **4800**, **9600**, **38400**, **57600** or **115200**.
xx different or **missing**		**9600**
P	—	Parity
xx = 1		Odd parity.
xx = 2		Even parity.
xx ≠ 1, 2 or **missing**		NO parity
L	—	Data length.
xx		**7**
xx ≠ 7 or **missing**		**8**
S	—	Stop bit number
xx		**2**
xx ≠ 2 or **missing**		**1**
O	—	Communication time out
xx		**0.5** to **999** sec.

Defaults **xx** Missing: CMM0T0B9600P0L8S1O1R0

X — XON/XOFF mode.

Defaults

xx = 0 XON/XOFF mode enable.
xx = 1 XON/XOFF mode disable.

Description This command changes the communication mode between the controller and the PC.

Returns In the case of **CM?** command, it reports the actual communication parameters of the controller.

Errors

A — Unknown message code.
Q — Unauthorized command.

Rel. Commands None.

Example

CM | *Initialization of all parameters.*

CP — Compile program

Usage ■ IMM ☐ PGM ☐ MIP

Syntax **xxCP**

Parameters

Description	**xx** [int]	—	Program number.
Range	**xx**	—	**1** to **127.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error F.
		Out of range:	Error F.
		Floating point:	Error A.

Description This command compiles a motion program loaded in the controller's memory. It verifies the syntax of the program, the validity of commands in the program context and the correctness of the jump and while loops.

If an error is found, the compilation is interrupted and the error type is reported. In this case, correct the problem and recompile to verify the rest of the program. Repeat this operation until the controller reports back a full compilation without error.

If the program editing is done on a remote computer, do not forget to erase the old version of the program with XX command. Otherwise, the new version of the program will be appended to the old one.

A program can be executed without being first compiled with CP. This command is helpful only in catching typing or structural program errors, but it does not guarantee that the program is fail-safe.

Returns **xxCPaa**

xx — Program number.
aa — ASCII code of the error type. If no error is detected, **aa** is character @.

Errors

A — Unknown message code.
F — Program number incorrect.
G — Program does not exist.
See Appendix A for additional list of programming errors.

Rel. Commands

EP — Edition of program.
QP — Quit program mode.

Example

3XX	*Clear program 3 from memory, if any.*
3EP	*Activate program mode and enter following commands as program 3.*
…	
…	
…	
QP	*End entering program number 3 and quit program mode.*
3CP	*Compile program number 3.*
3CP@	*Controller confirms compilation of program number 3 without any error.*

CR — Define radius for an arc of circle = *f* (CR, CA)

Usage ■ IMM ■ PGM □ MIP

Syntax **CRnn**

Parameters

Description	**nn** [double]	—	Radius for an arc of circle.
Range	**nn**	—	**10^{-12}** to **10^{100}**.
Units	**nn**	—	Defined motion units.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command defines to the controller the radius for an element of trajectory of the type: arc of circle = *f* (CR, CA). Unless the case of successively builded with the same radius (r, θ) arcs, we have specified the radius by this command every time before the CA command.

Returns None.

Errors

C	—	Parameter out of limits.
V	—	Too long trajectory.
]	—	Trajectory: Arc (r, θ) radius is too small.
^	—	Trajectory: Arc (r, θ) radius is too big.
e	—	Trajectory: Units not translationnal or not identical.

Rel. Commands

CA	—	Define sweep angle and build an arc of circle = *f* (CR, CA).
XE	—	Tell the last element.

Example

NT	*Clear trajectory.*
CR10	*Define radius of an arc of circle = f (r, θ).*
CA90	*Define sweep angle an build an arc of circle = f (r, θ).*
XE	*Tell last element.*
XE, Arc (r, θ), 10, 10, 90	*Controller tells the built element.*

CS — Concatenate two strings

Usage	■ IMM	■ PGM	□ MIP

Syntax **xxCSaa** or **xxCS?**

Parameters

Description	**xx** [int]	—	String variable number.
	aa [str]	—	String to be to be concatenated.
Range	**xx**	—	**1** to **8.**
	aa	—	**0** or **32** characters.
Units	**xx**	—	None.
	aa	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error C.
		Floating point:	Error A.
	aa	Missing:	Null string.
		Out of range:	Only first 32 characters are used.

Description This command concatenates two strings, the **aa** string or the #**nn** string, or the #**pp** value converted to ASCII (when the parameter **aa** is a variable $Ypp or $Snn), is concatenated in the end of the **xx** string

Returns If the sign **"?"** takes place of **aa** and **xx** is different of zero, this command reportes actual string stocked in the **xx** numbered string buffer.

Errors

- A — Unknown message code.
- C — Parameter out of limits.
- D — Unauthorized execution.

Rel. Commands

- **AS** — Affect string.
- **DS** — Display strings on screen.

Example

1AS"This "	*Affects "This " in variable S1 (S1 = "This ").*
2AS"is"	*Affects "is" in variable S2 (S2 = "is").*
3AS" "	*Affects " " in variable S3 (S3 = " ").*
1CS$S2	*Concatenate S2 to S1 (S1 = "This is").*
1CS$S3	*Concatenate S3 to S1 (S1 = "This is ").*
1CS"a string"	*Concatenate "a string" to S1.*
DS$S1	*Contents of variable S1.*
This is a string	*Displayed on the controller's screen.*

CT — Read or set the controller time

Usage ■ IMM ■ PGM ■ MIP

Syntax **CTnn**

Parameters

Description	**nn** [float]	—	Controller time.
Range	**nn**	—	**Non-negative floating value.**
Units	**nn**	—	Second.
Defaults	**nn**	Missing or ?:	Return the controller time.

Description This command permits to read or set the controller time at the moment of the command execution. The controller time indicate the time (in seconds) evolved from an initial value. This value is set to zero at the moment of motors on (button MOTOR ON pressed or MO command executed), but it can be modified later by the CT command at any time. The controller time evolves only if the controller stays MOTOR ON, but stops immediately whenever the controller motors are put off (button MOTOR OFF pressed or MF command executed).

Returns If **nn** is missing or **nn** = **"?"**, this command reports the controller time at the moment of the command execution.

Errors

A — Unknown message code.
C — Parameter out of limits.

Rel. Commands — None.

Example

CT	*Read the controller time.*
…	…
CT0	*Set the controller time to zero.*

CX — Define X position to reach with an arc of circle = *f* (CX, CY)

Usage ■ IMM ■ PGM □ MIP

Syntax **CXnn**

Parameters

Description	**nn** [double]	—	X coordinate to reach with an arc of circle.
Range	**nn**	—	**-1.0 E^{12}** to **1.0 E^{12}.**
Units	**nn**	—	Defined motion units.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command defines to the controller X position to rich with an element of trajectory of the type: arc of circle = *f* (CX, CY).

Returns None.

Errors
C — Parameter out of limits.
V — Too long trajectory.
e — Trajectory: Units not translationnal or not identical.

Rel. Commands
CY — Define Y position to reach and build an arc of circle = *f* (CX, CY).
XE — Tell the last element.
EL — Erase the last element of trajectory.

Example

NT	*Clear trajectory.*
CX10	*Define X position of an arc of circle = f (x, y).*
CY10	*Define Y position an build an arc of circle = f (x, y).*
XE	*Tell last element.*
XE, Arc (x, y), 10, 10, 90	*Controller tells the built element.*

CY — Define Y position to reach and build an arc of circle = *f* (CX, CY)

Usage	■ IMM	■ PGM	□ MIP	
Syntax	**CYnn**			
Parameters				
Description	**nn**	[double]	—	Y coordinate to reach with an arc of circle.
Range	**nn**		—	**-1.0 E**^{12} to **1.0 E**^{12}**.**
Units	**nn**		—	Defined motion units.
Defaults	**nn**		Missing:	Error C.
			Out of range:	Error C.

Description This command defines to the controller Y position to reach and tells to the controller to build an element of trajectory of the type:

Arc of circle = *f*(CX, CY).

Returns None.

Errors

C	—	Parameter out of limits.
H	—	Calculation overflow.
V	—	Too long trajectory.
\	—	Type error (arc expected).
]	—	Trajectory: Arc (r, θ) radius is too small.
^	—	Trajectory: Arc (r, θ) radius is too big.
`	—	Arc (x, y) circle too small.
e	—	Trajectory: Units not translationnal or not identical.

Rel. Commands

CX	—	Define X position to reach with an arc of circle = *f* (CX, CY).
XE	—	Tell the last element.
EL	—	Erase the last element of trajectory.

Example

NT	*Clear trajectory.*
CX10	*Define X position of an arc of circle = f (x, y).*
CY10	*Define Y position an build an arc of circle = f (x, y).*
XE	*Tell last element.*
XE, Arc (x, y), 10, 10, 90	*Controller tells the built element.*

DA — Read desired acceleration

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxAC?

Syntax **xxDApp**

Parameters

Description	**xx** [int]	—	Axis number.
	pp [int]	—	Auxiliary parameter.
Range	**xx**	—	**1** to **8.**
	pp	—	**0** or **1**.
Units	**xx**	—	None.
	pp	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	pp	Missing:	0.
		≥1:	1.

Description

This command reads the motion acceleration assigned in the axis setup or redefined through the AC command. This is the acceleration of a trapezoidal type motion profile used in point-to-point moves.

On Power-Up, the acceleration defaults to the value preset in the front panel SETUP menu.

If **pp** is 0 or missing, the actual value of acceleration is reported. If **pp** ≥1, the maximum allowed value of acceleration is reported.

Returns **xxDAnn**

xx — Axis number.
nn — Acceleration value, in pre-defined units.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **AC** — Set acceleration.

Example

2DA	*Read acceleration of axis #2.*
2DA10	*Controller returns an acceleration value of 10.*
2PA15	*Move to absolute position 15.*
WT500	*Wait for 500 ms.*
2AC4	*Set axis #2 acceleration to 4.*
2DA	*Read acceleration of axis #2.*
2DA4	*Controller returns an acceleration value of 4.*
2DA1	*Read maximum acceleration of axis #2.*
2DA80	*Controller returns a value of 80.*

DF — Read following error

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxDF**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description

This command reads the following error on an axis. The following error is defined as the instantaneous difference between the real position, reported by the encoder, and the theoretical position, calculated by the controller according to the desired trajectory. Reading the following error for an axis is important in determining its performance and tuning the servo loop.

If the axis parameter **xx** is 0 or missing, the controller reads the following error for all axes simultaneously and returns all four values.

If the command is used inside a program, make sure a host computer is ready to receive and store the returned data.

Returns

xxDFnn or **xx_1DFnn$_1$, xx_2DFnn$_2$, xx_3DFnn$_3$, xx_4DFnn$_4$**

xx, xx_1, xx_2, xx_3, xx_4
— Axis number.

nn, nn_1, nn_2, nn_3, nn_4
— Following error, in pre-defined units.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
S	—	Communication time-out.

Rel. Commands

FE	—	Set maximum following error.
TF	—	Read filter parameters.
XF	—	Read maximum following error.

Example

2PA15	*Move axis #2 to absolute position 15.*
2WP10	*Wait for axis #2 to reach position 10.*
2DF	*Read following error of axis #2.*
2DF0.003	*Controller returns a following error for axis #2 of 0.003.*
2WS	*Wait for motion to stop on axis #2.*
WT200	*Wait 200ms for motion to settle.*
2DF	*Read following error at stop on axis #2.*
2DF0.001	*Controller returns a following error for axis #2 of 0.001.*

Usage	■ IMM	■ PGM	□ MIP
Syntax	**xxDH**		

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**0** to **4.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command defines current position, HOME position. This means that the current position will be reset to the value preset by ZH or by the front panel SETUP utility. If the home preset value is 0, this command is equivalent to ZP.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
D — Unauthorized execution.

Rel. Commands **OR** — Search for home.

Example

3OR	*Perform a home search on axis #3.*
...	
...	
...	
3DH	*Define current position on axis #3 HOME.*

DL — Define label

Usage	☐ IMM ■ PGM ☐ MIP	
Syntax	**xxDL**	
Parameters		
Description	**xx** [int]	— Label number.
Range	**xx**	— **1** to **100.**
Units	**xx**	— None.
Defaults	**xx** Missing:	Error N.
	Out of range:	Error N.
	Floating point:	Error A.

Description

This command defines a label inside a program. In combination with JL (jump to label) command, they provide program flow control.

The operation of the DL/JL command pair is similar to commands in other computer languages that allow conditional jumps (or GOTOs) to predefined labels in a program.

NOTE

This command does not generate an error when not used inside a program. Since it can not do any harm, it is only ignored.

Returns None.

Errors

A — Unknown message code.
L — Command not at the beginning of a line.
N — Incorrect label number.

Rel. Commands **JL** — Jump to label.

Example

3DL	*Define label number 3.*
…	
…	
…	
2YL20, 3JL	*If variable 2 is less than 20, jump to label 3.*

DM — Read manual velocity

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command: xxMH?

Syntax **xxDM**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the manual jog high velocity (from front panel or joystick). This is the high speed manual jog mode, when the center key is pressed simultaneously with a direction key. The manual jog low speed is 1/10 of the high speed.

The manual jog high speed is assigned in the axis setup or redefined with the MH command.

Returns **xxDMnn.**

xx — Axis number.
nn — Manual jog high velocity value, in pre-defined units/sec.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **MH** — Set manual velocity.

Example

2DM | *Read manual jog high velocity on axis #2.*
2DM2.5 | *Controller returns for axis #2 a manual jog.*

DO — Read home search velocity

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command: xxOH?

Syntax	**xxDO**		
Parameters			
Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the velocity to be used in the home search cycle. This is the high velocity of the algorithm, the other ones being scaled down from it.

The home search high velocity is set by the OH command or from the front panel SETUP menu.

Returns **xxDOnn**

xx — Axis number.
nn — Home velocity value, in pre-defined units/sec.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **OH** — Set home search high velocity.

Example

2DO	*Read home search high velocity on axis #2.*
2DO2.5	*Controller returns for axis #2 a home search velocity of 2.5.*

DP — Read desired position

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxDP**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**0** to **4.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the desired position, the destination of a certain motion component. If the axis specifier **xx** is missing or set to 0, the controller returns the desired position for all axes.

The command could be sent at any time but is most often invoked while a motion is in progress.

Returns **xxDPnn** or **xx_1DPnn$_1$, xx_2DPnn$_2$, xx_3DPnn$_3$, xx_4DPnn$_4$**

xx, xx_1, xx_2, xx_3, xx_4

— Axis number.

nn, nn_1, nn_2, nn_3, nn_4

— Desired position, in pre-defined units.

Errors

A — Unknown message code.
B — Incorrect axis number.
D — Unauthorized execution.
S — Communication time-out.

Rel. Commands

PA — Move to absolute position.
PR — Move to relative position.

Example

3TP	*Read position on axis #3.*
3TP5.32	*Controller returns position 5.32 for axis #3.*
3PR2.2	*Start a relative motion of 2.2 on axis #3.*
3DP	*Read desired position on axis #3.*
3DP7.52	*Controller returns desired position 7.52 for axis #3.*

DS — Display strings on screen

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **xxDSaa**

Parameters

Description
xx [int] — Field number.
aa [chaîne] — Strings to be displayed.

Range
xx — **1** to **2.**
aa — **0** to **N** characters, framed or not framed by two quotation marks "≤".
xx = **1** or **2** : **N** = **14**.
xx = **0** : **N** = **28**.

Units
xx — None.
aa — None.

Defaults
xx Missing: 0.
Out of range: Error C.
Floating point: Error A.
aa Missing: Null string; clears the line or field.
Out of range: Only first N characters are used.

Description This command prints a string on line #5 of the front panel display. If **xx** = 0 or default, the line #5 is entirely used (28 characters max.). If **xx** = 1 or 2, the line #5 is split in two fields, each 14 characters long. The first field on the left is defined as number 1 and the one on the right as number 2.

Writing to field number 1 (**xx** = 1) erases the previous text from field number 1. Writing to field number 2 (**xx** = 2) erases the previous text from field number 2. If **xx** = 0 or default, the controller erases the entire line #5 before writing its new text.

If somes strings (separated by spaces) are to be printed, these strings must be framed by two quotation marks (≤). The printable number of characters is always N, but the quotation marks are not part of characters of these strings.

This command is useful for monitoring the status and evolution of a complex program.

Returns None.

Errors
A — Unknown message code.
C — Parameter out of limits.
J — Command authorized only in programming mode.

Rel. Commands
AS — Affect string.
CS — Concatenates two strings.

Example

1AS "This "	*Define string #1.*
2AS "is "	*Define string #2.*
101YS10.5	*Define variable #101.*
DS$S1$S2"a value: "$Y101	*Display on screen.*
This is a value: 10.5	*Response.*
WT4000	*Wait for 4 sec.*

In this example, line five of the front panel will display "This is a value: 10.5" during 4 seconds.

DV — Read desired velocity

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxVA?

Syntax **xxDVpp**

Parameters

Description
xx [int] — Axis number.
pp — Auxiliary parameter.

Range
xx — **1** to **8.**
pp — **0** or **1.**

Units
xx — None.

Defaults
xx Missing: Error B.
Out of range: Error B.
Floating point: Error A.
pp Missing: 0.
≥1: 1.

Description
This command reads the motion velocity assigned in the axis setup or redefined through the VA command. This is the maximum velocity of a trapezoidal type motion profile used in point-to-point moves.

Since the command is reporting the programmed velocity, it can be used during motion or at stop.

If **pp** is 0 or missing, the actual value of velocity is reported. If **pp** > or = 1, the maximum allowed value of velocity is reported.

Returns **xxDVnn**
xx — Axis number.
nn — Velocity value, in pre-defined units/sec.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **VA** — Set velocity.

Example

2VA2.5	*Define velocity to 2.5 units/sec.*
2DV	*Read desired velocity on axis #2.*
2DV2.5	*Controller returns a value of 2.5.*
2DV1	*Read maximum allowed velocity on axis #2.*
2DV20	*Controller returns a value of 20.*

DY — Display a variable

Usage	☐ IMM ■ PGM ☐ MIP
Syntax	**xxDYnn**

Parameters

Description	**xx** [int]	—	Field number.
	nn [int]	—	Variable number.
Range	**xx**	—	**1** to **2.**
	nn	—	**1** to **120.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error C.
		Out of range:	Error C.
		Floating point:	Error A.
	nn	Missing:	Error O.
		Out of range:	Error O.

Description

This command prints a variable's value on line five of the front panel display. For this purpose, line five is split in two fields, each 13 characters long. The first field on the left is defined as number 1 and the one on the right as number 2. Parameter **xx** selects which field the variable will be displayed on.

Writing to field number 1 erases the previous text on the entire line. Writing to field number 2 erases the previous text only from field number 2.

This command is useful in monitoring the status and evolution of a complex program.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
J	—	Command authorized only in programming mode.
O	—	Variable number out of range.

Rel. Commands

DS	—	Display strings on screen.
YS	—	Initialize variable.
EX	—	Execute a program.

Example

1DS LOOP #	*Print on the first part of line 5 the string "LOOP #".*
3YS0	*Set variable #3 to 0.*
4DL	*Define label #4.*
…	
…	
…	
3YA1	*Increment variable #3 by 1.*
2DY3	*Display variable #3 on the second field of line 5.*
3YL50, 4JL	*If variable #3 is less than 50, jump to label #4.*

In this example, line five of the front panel will display "LOOP #N", where N is the loop count.

ED — Display program error

Usage	■ IMM	■ PGM	□ MIP
Syntax	**EDnn**		

Parameters

Description	**nn** [int]	—	Enable/disable code.
Range	**nn**	—	**0** to **1.**
Units	**nn**	—	None.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.
		Floating point:	Error C.

Description

This command activates the program execution error display utility. If **nn** is set to 1, the program execution will stop every time an error is encountered and the following message will be displayed on line 5 of the display: "Program aborted by error: * " where "*" represents the ASCII error code. On line 6, the last function key will be defined as QUIT and the program will resume execution after pressing this key.

Setting **nn** to zero disables the program execution error display utility. This is the default mode of operation. Any error encountered will stop and terminate the program. To determine the error causing the problem use the TB or TE commands.

Returns None.

Errors C — Parameter out of limits.

Rel. Commands

TB — Read error message.
TE — Read error code.

Example

ED1	*Activate program execution error display utility.*
3EX	*Start executing program #3.*
…	
…	
…	
ED0	*Disable program execution error display utility.*

EL — Erase the last element of trajectory

Usage	■ IMM ■ PGM □ MIP
Syntax	**EL**
Parameters	None.
Description	This command erases the last entered from actual elements of a trajectory.
Returns	None.
Errors	D — Unauthorized execution. S — Communication time-out. b — Trajectory is empty.
Rel. Commands	**LT** — Extended list of the trajectory. **NT** — Start definition of a new trajectory. **XE** — Tell the last element.
Example	NT | *Start a new trajectory.* LX10 | *Insert an element.* **EL** | *Erase this element.* XT | *Read number of trajectory elements.* *XT0* | *Controller returns 0.*

EO — Automatical execution on power on

Usage	■ IMM ■ PGM ☐ MIP
Syntax	**xxEOnn** or **EO?**
Parameters	
Description	**xx** [int] — Program number to execute on power on. **nn** [int] — Number of times on execution. **?** — Read program number to execute and number of times of execution.
Range	**xx** — **1** to **127.** **nn** — **1** to **2147385345**.
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: 0 (no program execution on power on). Out of range: Error F. Floating point: Error A. **nn** Missing: 1 (one time of execution). Out of range: Error C.
Description	This command sets the program number that is automatically executed on power on. If **xx** is zero or missing, no program is executed. If **nn** is missing, the **xx** numbered program is executed one time.
Returns	If the sign **"?"** takes place of the **nn** value and **xx** missing, this command reportes the number of the program that is executed on power on and number of times of execution.
Errors	A — Unknown message code. C — Parameter out of limits. F — Program number incorrect.
Rel. Commands	**EX** — Execute a program.
Example	**2EO** | *Set program #2 to be executed one time on power on.* **EO?** | *Read executed on power on program number.* *EO, 2, 1* | *Controller tells the program #2 to be executed one time on power on.* **EO** | *Reset automatical execution - no program is executed on power on.*

EP — Edition of program

Usage ■ IMM ☐ PGM ☐ MIP

Syntax **xxEPnn**

Parameters

Description
xx [int] — Program number
nn [int] — Program line number.

Range
xx — **1** to **127.**
nn — **1** to **32767**.

Units
xx — None.
nn — None.

Defaults
xx Missing: Error F.
Out of range: Error F.
Floating point: Error A.
nn Missing: Append to the end of program #**xx**.
Out of range: Error C.
Floating point: Error A.

Description

This command sets the controller in programming mode. All the commands following this one will not be executed immediately but stored in memory as part of program number **xx**. Programs can be entered in any order. To exit program entry mode and return to immediate mode, use the QP command.

If a program already exists, the new commands entered will be inserted to the line #**nn** of program if **nn** is valid, or added to the end of program if **nn** is missing. So to replace a program, it must be first deleted using XX command.

Returns None.

Errors
A — Unknown message code.
C — Parameter out of limits.
D — Unauthorized execution.
F — Program number incorrect.
I — Unauthorized command in programming mode.
M — Program is too long.

Rel. Commands
CP — Compile program.
EX — Execute a program.
QP — Quit program mode.
XL — Delete one line of program.
XX — Erase program.

Example

3XX	*Clear program 3 from memory, if existing*
3EP	*Edition of program 3*
1PA10	*Enter a line*
1WS	*Enter a line*
3QP	*Quit Edition of program 3*
3LP	*Liste program 3*
1PA10	
1WS	*The program is listed*
…	
…	
…	

3EP	*Edition of program 3.*
2PA10	*Enter a line.*
2WS	*Enter a line.*
3QP	*Quit edition of program 3.*
3LP	*Liste program 3.*
1PA10	
1WS	
2PA10	
2WS	*The program is now listed.*
…	
…	
…	
3EP3	*Edition of program 3, insert at line #3.*
DS "*** WAIT ***"	*Enter a line.*
1SB	*Enter a line.*
WT5000	*Enter a line.*
1CB	*Enter a line.*
3QP	*Quit Edition of program 3.*
3LP	*Liste program 3.*
1PA10	
1WS	
*DS*** WAIT ****	
1SB	
WT2000	
1CB	
2PA10	
2WS	*The program is now listed.*
3EX	*Execute the program 3.*

ET — Execution of trajectory

Usage ■ IMM ■ PGM □ MIP

Syntax **ET**, **ETx** or **ETy**

Parameters None.

Description

ET: Execute trajectory on two axis (X and Y).
ETx: Execute trajectory on axis X alone
ETy: Execute trajectory on axis Y alone.

This command first verifies all parameters and entered elements of the trajectory then direct the controller to start the execution of the trajectory. If an error occurs or the necessary conditions to the execution are not complete the trajectory execution is not started and the command returns a code of error.

NOTE

To avoid errors, the desired order of commands is:

- **Preparation: NT, FA.**
- **Edition of trajectory: LX, LY, MX, MY, CR, CA, CX, CY, etc.**
- **Edition of generation of synchronisation pulses: NB, NE, NI or NN (option).**
- **Set trajectory velocity and acceleration: VV, VS (option).**
- **Allow generation of pulses on interpolation: NS (option).**
- **Execution of trajectory: ET.**
- **Synchronisation software: WI or WN (option).**

Returns None.

Errors

B — Incorrect axis number.
D — Unauthorized execution.
S — Communication time-out.
b — Trajectory is empty.
e — Trajectory: Units not translationnal or not identical.
f — Synchronization pulses generation impossible.
h — Trajectory: execution exceeds physical or logical limits.

Rel. Commands

VS — Define the vector acceleration on trajectory (trajectory acceleration).
VV — Define the vector velocity on trajectory (trajectory velocity).

Example

NT	*Start new trajectory.*
LX10	*Element 1.*
CR20, CA90	*Element 2.*
LY40	*Element 3.*
NB2	*Set starting point of synchronisation pulses (beginning of element 2).*
NE2	*Set ending point of synchronisation pulses (end of element 2).*
NI0.1	*Set step: generate pulses for every (curvi-linear) trajectory step of 0.1 unit.*
VV5	*Set trajectory velocity of 5 units/sec.*
VS40	*Set trajectory acceleration of 40 units/sec^2.*
NS	*Allow generation of pulses on interpolation.*
ET	*Execute trajectory.*
WN2	*Wait for beginning of element 2.*
1SB	*Set I/O ouput port number 1.*
WN3	*Wait for beginning of element 3.*
1CB	*Clear I/O ouput port number 1.*

EX — Execute a program

Usage	■ IMM	☐ PGM	☐ MIP

Syntax **xxEXnn**

Parameters

Description	**xx** [int]	—	Program number.
	nn [int]	—	Number of times to execute the program.
Range	**xx**	—	**1** to **127.**
	nn	—	**1** to **2147483648.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error F.
		Out of range:	Error F.
		Floating point:	Error A.
	nn	Missing:	1.
		Out of range:	Error C.
		Floating point:	Decimal part truncated.

Description

This command starts executing a program. When the command is received, the controller first compiles the program, checking for errors similar to the CP command. If an error is encountered, the compilation stops and the controller reports the type of error found. If no error is found, the controller executes the program line by line or according to the flow control instructions.

During program execution, only commands that ask for information or that stop the motion are allowed. Any of the following commands will terminate a program: AB, AP, MF and ST. The easiest way to stop program execution is by using the AP command, the other ones have wider effects.

EX command is capable to execute subroutines (program without EX command inside, integrated in the main program)

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
F	—	Program number incorrect.
G	—	Program does not exist.
I	—	Unauthorized command in programming mode.

See Appendix A for additional list of programming errors.

Rel. Commands

AP	—	Abort program.
CP	—	Compile program.

Example

1EP	*Program 1 edition (main program).*
1PA10, 2PA10	*Displacement of two axis.*
2EX	*Execute the program 2 .*
3EX	*Execute the program 3.*
4EX	*Execute the program 4.*
OR	*Origine search for all of axis.*
QP	*Quit main program.*
2EP	*Program 2.*
SB	*Set bits.*
1AS"This "	*Define string #1.*
2AS"is "	*Define string #2.*
1CS$S2	*Concatenate string #1 and string #2.*
DS$S1"a string"	*Display on screen.*

WT3000	*Wait for 3 seconds.*
QP	*Quit the program 2.*
3EP	*Program 3.*
3AS"a value: "	*Define string #3.*
101YS99.99	*Define value #101.*
3CS$Y101	*Concatenate string #3 and value #101.*
DS$S1$S3 " !"	*Display on screen.*
WT3000	*Wait for 3 seconds.*
QP	*Quit the program 3.*
4EP	*program 4.*
1PR-20, WS	*Displacement of axe 1.*
2PR-20, WS	*Displacement of axe 2.*
CB	*Clear bits.*
WT1000	*Wait for 1 second.*
QP	*Quit the program 4.*
This is a string	*Displayed on the controller's screen.*
This is a value: 99.99 !	*Displayed on the controller's screen.*

FA — Define the tangent angle for the first point

Usage	■ IMM ■ PGM □ MIP
Syntax	**FAnn** or **FA?**

Parameters

Description	**nn** [double]	—	Tangent angle value at first point of trajectory.
Range	**nn**	—	≥**0.0** to < **360.0.**
Units	**nn**	—	Degrees.
Defaults	**nn**	Missing:	0.
		Out of range:	Error X.

Description This command defines to the controller what will be angle of the tangent at the first point. This value is necessary if you want to start a new trajectory with the start tangent angle other than 0. So this command is enabled only when the number of elements of trajectory is zero.

NOTE

On NT command, the controller assumes that the angle of the first tangent is 0.

Returns None.

Errors

D	—	Unauthorized execution.
S	—	Communication time-out.
X	—	Trajectory: first angle definition error.

Rel. Commands **NT** — Start definition of a new trajectory.

Example

NT	*Start a new trajectory.*
FA45.0	*Define first tangent angle is 45°.*

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **xxFBaa**

Parameters

Description

xx	[int]	—	Function key number.
aa	[chaîne]	—	Label to be displayed.

Range

xx	—	**1** to **8.**
aa	—	**0** to **6** characters.

Units

xx	—	None.
aa	—	None.

Defaults

xx	Missing:	Error O.
	Out of range:	Error O.
	Floating point:	Error A.
aa	Missing:	Clears the selected function key label.
	Out of range:	Truncates label to the first 6 characters.

Description This command allows the user to define a label for a function key. Using the FD or WF command will display the use-defined function keys.

For the **xx** value, the four function keys are numbered from 1 to 8, from left to right.

NOTE

The command is valid only in programming mode, where the function keys are not used by the normal operation of the controller.

Returns None.

Errors

A	—	Unknown message code.
J	—	Command authorized only in programming mode.
O	—	Variable number out of range.

Rel. Commands

FC	—	Clear function key line.
FD	—	Display function keys.
WF	—	Wait for function key.

Example

3XX	*Clear program #3 from memory, if any.*
3EP	*Activate program mode and enter following commands as program #3.*
4FBSTOP	*Define custom label for function key #4 as STOP.*
…	
…	
…	
7WF	*Display the custom function key label(s) (STOP), wait for a valid function key to be pressed and put its ASCII code in variable #7.*
FC	*Clear function key display line.*
…	
…	
…	

FC — Clear function key line

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **FC**

Parameters None.

Description This command clears the function key line displayed by the FD or WF commands. It is intended to be used in conjunction with the FB, FD and WF commands to build front panel interactive programs.

NOTE

The command is valid only in programming mode, applying only to the custom-defined function keys, not the ones used by the normal operation of the controller.

Returns None.

Errors
A — Unknown message code.
J — Command authorized only in programming mode.

Rel. Commands
FB — Label function key.
FD — Display function keys.
WF — Wait for function key.

Example

3XX	*Clear program #3 from memory, if any.*
3EP	*Activate program mode and enter following commands as program #3.*
4FBSTOP	*Define custom label for function key #4 as STOP.*
…	
…	
…	
7WF	*Display the custom function key label(s) (STOP), wait for a valid function key to be pressed and put its ASCII code in variable #7.*
FC	*Clear function key display line.*
…	
…	
…	

FD — Display function keys

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **FD**

Parameters None.

Description This command displays the function keys defined with the FB command. It is intended to be used in conjunction with the FB, FC and WF commands to allow the user to build front panel interactive programs.

NOTE

The command is valid only in programming mode, applying only to the custom defined function keys, not the ones used by the normal operation of the controller.

Returns None.

Errors
A — Unknown message code.
J — Command authorized only in programming mode.

Rel. Commands
FB — Label function key.
FC — Clear function key line.
WF — Wait for function key.

Example

3XX	*Clear program #3 from memory, if any.*
3EP	*Activate program mode and enter following commands as program #3.*
4FBSTOP	*Define custom label for function key #4 as STOP.*
…	
…	
…	
FD	*Display the custom function key label(s) (STOP).*
…	
…	
…	
FC	*Clear function key display line.*

FE — Set maximum following error

Usage ■ IMM ■ PGM □ MIP

Syntax **xxFEnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [float]	—	Maximum allowed following error.
Range	**xx**	—	**1** to **8.**
	nn	—	**2 x encoder resolution to maximum device travel.**
Units	**xx**	—	None.
	nn	—	Preset units in SETUP mode.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command sets the maximum allowed following error for an axis. This error is defined as the difference between the real position and the theoretical position of a motion device. The real position is the one reported by the position sensing device (encoder, scale, etc.) and the theoretical position is calculated by the controller each servo cycle. If, for any axes and any servo cycle, the following error exceeds the preset maximum allowed following error, the controller stops motion on all axes and turns power off to all motors.

The command can be sent at any time but it has no effect until the UF (update filter) is received.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands

TF — Read filter parameters.
UF — Update servo filter.
XF — Read maximum following error.

Example

3FE0.1	*Set maximum following error for axis #3 to 0.1.*
…	
…	
…	
3UF	*Update PID filter; only now the FE command takes effect.*

FF — Set maximum master-slave following error

Usage ■ IMM ■ PGM □ MIP

Syntax **xxFFnn** or **xxFF?**

Parameters

Description

xx	[int]	—	Axis number.
nn	[float]	—	New value of maximum allowed master-slave following error.
?		—	Read the actual maximum allowed master-slave following error.

Range

xx	—	**1** to **8.**
nn	—	**2 x axis encoder resolution** to **maximum axis travel.**

Units

xx	—	None.
nn	—	Preset units in SETUP mode.

Defaults

xx	Missing:	Error B.
	Out of range:	Error B.
	Floating point:	Error A.
nn	Missing:	Error C.
	Out of range:	Error C.

Description This command sets the maximum allowed master-slave following error for a slave axis. This error is defined as the difference between the position error of master axis and is divided by master-slave reduction ratio position error of the slave axis. The position error of an axis is defined as the difference between the theoretical position and the real position of this axis. If, for any axes and any servo cycle, the master-slave following error exceeds the preset maximum allowed master-slave following error, the controller stops motion on all axes and turns power off to all motors.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the actual value of the maximum allowed master-slave following error.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.

Rel. Commands **FE** — Set maximum following error.

Example

3FF0.1	*Set maximum allowed master-slave following error for axis #3 to 0.1 units.*
…	
…	
…	
3FF?	*Read maximum master-slave following error of axis #3.*
3FF0.1	*Controller tells the value of this parameter.*

Newport

FT — Set output frequency

Usage ■ IMM ■ PGM ■ MIP

Syntax **FTnn**

Parameters

Description **nn** [float] — Desired frequency.

Range **nn** — **0** and **0.01** to **5000.**

Units **nn** — Hz.

Defaults **nn** Missing: 0.
Out of range: Error D.
Non-increment: Rounded to nearest frequency increment (see table).

Description This command controls the output signal on pin 24 of the 25-pin auxiliary connector. The **nn** parameter represents the frequency of the output signal. Setting **nn** to 0 disables the frequency generator. The output has an open collector configuration and a frequency range and resolution shown in the following table:

F (Hz) (Output frequency)	ΔF (Hz) (Frequency resolution)
0.01 - 20	0.001
20 - 250	0.010
250 - 500	0.020
500 - 1000	0.100
1000 - 2500	0.500
2500 - 5000	1.000

NOTE

For the hardware definition of the frequency generator port, please see Appendix B, Connector Pinouts, Auxiliary Connector.

Returns None.

Errors D — Unauthorized execution.

Rel. Commands None.

Example

FT218.24 | *Set an output frequency of 218.240 Hz on pin 24 of the auxiliary connector.*

.

GQ — Set global trace mode

Usage ■ IMM ■ PGM ■ MIP

Syntax **GQnn** or **GQ?**

Parameters

Description	**nn** [int]	—	Number of samples.
Range	**nn**	—	**0** to **NMax (1500** to **4000).**
	?	—	Reading of the NMax Value.
Units	**nn**	—	None.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.
		Floating point:	Decimal part truncated.

Description This command sets the global trace mode. If the command is sent with the **nn** set between 1 and 4000, the controller starts immediately recording in memory the theoretical position, the actual position of all axes and the 4 analog inputs. The number of samples stored is the one specified by **nn** and the sample interval is the one set by the SQ command. To read the recorded trace data use the TQ command. To disable the global trace mode set **nn** to 0.

NOTE

Since it starts executing immediately, the best use of this command is in the same line of program with the displacement for better control of execution delays.

Returns If the sign **"?"** takes place of **nn**, this command reports the number of possible max. points in global trace mode.

Errors C — Parameter out of limits.

Rel. Commands

NQ — Read global acquisition nr.
SQ — Set global sample rate.
TQ — Read global trace data.
XQ — Read global sample rate.

Example

SQ0.002	*Set global trace sample period to 2 msec.*
GQ500, 2PR5, 2WS	*Set global trace mode for 500 data points, perform a motion of 5 units on axis #2 and wait for stop.*
TQ	*Read global trace data.*

GR — Set master-slave reduction ratio

Usage ■ IMM ■ PGM □ MIP

Syntax **xxGRnn** or **xxGR?**

Parameters

Description

xx	[int]	—	Axis number.
nn	[float]	—	New value of maximum allowed master-slave following error.
?		—	Read the actual maximum allowed master-slave following error.

Range

xx	—	**1** to **8.**
nn	—	**0.0001** to **10000.**

Units

xx	—	None.
nn	—	None.

Defaults

xx	Missing:	Error B.
	Out of range:	Error B.
	Floating point:	Error A.
nn	Missing:	1.0.
	Out of range:	Error C.

Description This command sets the master-slave reduction ratio for a slave axis. The displacement of the slave axis is the one of the master axis multipled by this coefficient.

NOTE

Use the GR command carefully. The slave axis will also have its speed and acceleration in the same ratio than the position.

Be careful that the ratio used for the slave axis doesn't cause overflow of this axis parameters (speed, acceleration), especially with ratios greater than 1.

NOTE

If the CD command is used in conjunction with the SS command and GR command, the slave axis cycle value must be equal to the master axis cycle value multiplied by the master-slave reduction ratio.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the actual value of the master-slave reduction ratio.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.

Rel. Commands **SS** — Set master-slave mode.

Example

2GR100	*Set master-slave ratio for axis #2 to 100.*
…	
…	
…	
2GR?	*Read master-slave reduction of axis #2.*
2GR100	*Controller tells the value of this parameter.*

HT — Set home type

Usage	■ IMM	■ PGM	□ MIP

Syntax **xxHTnn** or **xxHT?**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [double]	—	New value.
Range	**xx**	—	**1** to **8.**
	nn	—	**0** to **1**. **0** use home switch and encoder index pulse. **1** use floating home. **2** use home switch only.
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command sets the new type of home search executed by the controller when receiving the OR command. Real home means that the controller will move the stage to find the home position, floating means that the controller will assume the current position to be the new home.

Returns If the **"?"** sign takes place of **nn** parameter, the controller returns the current home type.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands **OR** — Search for home.

Example

1HT?	*Request the current home type for axis #1.*
1HT0	*Controller returns the current home type of this axis.*
1HT1	*Set floating home on axis #1.*
1HT?	*Request the current home type for axis #1.*
1HT1	*Controller returns the current home type of this axis.*

IE — If I/O input is equal

Usage ■ IMM ■ PGM □ MIP

Syntax **xxIEnn**

Parameters

Description	**xx** [int]	—	I/O input bit number.
	nn [int]	—	I/O input bit or byte state.
Range	**xx**	—	**0** to **8.**
	nn	—	**0** to **1** or **0** to **255.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.
		Floating point:	Error C.

Description This command is one of the flow control instructions, enabling a conditional execution of a command line depending on the state of an I/O input bit. It must be placed at the beginning of the command line of which execution it controls. If the selected bit **xx** has the specified state **nn**, all following commands on that line are executed. If **xx** is set to 0 or missing, the test is performed on the entire I/O input byte and then **nn** could have a value from 0 to 255, representing the byte value to compare it with.

As described in the Command Syntax paragraph, a line is defined as all commands between two line terminators.

Even though the command can be used on a line in immediate mode, its real value is inside a program.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
E	—	Incorrect I/O channel number.
L	—	Command not at the beginning of a line.

Rel. Commands **OE** — Test I/O output.

Example

3IE0, 1PA2.34 | *If I/O input bit #3 is low, move axis #1 to position 2.34.*

Usage	☐ IMM	■ PGM	☐ MIP

Syntax **xxJL**

Parameters

Description	**xx** [int]	—	Label number.
Range	**xx**	—	**1** to **100.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error N.
		Out of range:	Error N.
		Floating point:	Error A.

Description This command changes the flow of the program execution by jumping to a predefined label. This is a flow control command that alters the normal sequential flow of a program. It must be used in conjunction with a DL command which defines a label.

Returns None.

Errors

A	—	Unknown message code.
J	—	Command authorized only in programming mode.
K	—	Undefined label.
N	—	Incorrect label number.

Rel. Commands **DL** — Define label.

Example

3DL	*Define label number 3.*
…	
…	
…	
2YL20, **3JL**	*If variable 2 is less than 20, jump to label 3.*

Newport

KC — Abort command line

Usage	■ IMM ☐ PGM ■ MIP
Syntax	**KC**
Parameters	None.
Description	This command stops a program or a command line in execution. On reception of this command, the controller will finish executing the command in progress, abort execution of the remaining ones and return to the immediate mode.
Returns	None.
Errors	None.
Rel. Commands	**AB** — Abort motion. **ST** — Stop motion. **MF** — Motor OFF. **MO** — Motor ON.
Example	**KC** | *Finish executing command in progress and abort the remaining commands.*

KD — Set derivative gain

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxKDnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	Derivative gain factor Kd.
Range	**xx**		—	**1** to **8.**
	nn		—	**0** to **1.**
Units	**xx**		—	None.
	nn		—	None.
Defaults	**xx**		Missing:	Error B.
			Out of range:	Error B.
			Floating point:	Error A.
	nn		Missing:	Error C.
			Out of range:	Error C.

Description

This command sets the derivative gain factor Kd of the PID closed loop. It is active for any motion device that has been selected to operate in closed loop, including those driven by stepper motors.

The command can be sent at any time but it has no effect until the UF (update filter) is received.

See the "Servo Tuning" section on how to adjust the PID filter parameters.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands

KI — Set integral gain.
KP — Set proportional gain.
UF — Update servo filter.
XD — Read derivative gain factor.

Example

3KD0.01	*Set derivative gain factor for axis #3 to 0.01.*
…	
…	
…	
3UF	*Update PID filter; only now the KD command takes effect.*

KI — Set integral gain

Usage ■ IMM ■ PGM □ MIP

Syntax **xxKInn**

Parameters

Description **xx** [int] — Axis number.
nn [float] — integral gain factor Ki.

Range **xx** — **1** to **8.**
nn — **0** to **1.**

Units **xx** — None.
nn — None.

Defaults **xx** Missing: Error B.
Out of range: Error B.
Floating point: Error A.
nn Missing: Error C.
Out of range: Error C.

Description This command sets the integral gain factor Ki of the PID closed loop. It is active for any motion device that has been selected to operate in closed loop, including those driven by stepper motors.

The command can be sent at any time but it has no effect until the UF (update filter) is received.

See the "Servo Tuning" section on how to adjust the PID filter parameters.

Returns None.

Errors
A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands
KD — Set derivative gain.
KP — Set proportional gain.
UF — Update servo filter.
XI — Read integral gain factor.

Example

3KI0.01	*set integral gain factor for axis #3 to 0.01.*
…	
…	
…	
3UF	*Update PID filter; only now the KI command takes effect.*

Usage	■ IMM ■ PGM ☐ MIP
Syntax	**xxKPnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [float]	—	Proportional gain factor Kp.
Range	**xx**	—	**1** to **8.**
	nn	—	**0** to **1.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description

This command sets the proportional gain factor Kp of the PID closed loop. It is active for any motion device that has been selected to operate in closed loop, including those driven by stepper motors.

The command can be sent at any time but it has no effect until the UF (update filter) is received.

See the “Servo Tuning” section on how to adjust the PID filter parameters.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands

KD — Set derivative gain.
KI — Set integral gain.
UF — Update servo filter.
XP — Read proportional gain factor.

Example

3KP0.01	*Set proportional gain factor for axis #3 to 0.01.*
…	
…	
…	
3UF	*Update PID filter; only now the KP command takes effect.*

KS — Set saturation level of integral factor in position loop PID corrector

Usage ■ IMM ■ PGM □ MIP

Syntax **xxKSnn** or **xxKS?**

Parameters

Description

xx	[int]	—	Axis number.
nn	[float]	—	New saturation level of integral factor.
?		—	Read the actual saturation level.

Range

xx	—	**1** to **8.**
nn	—	**0** to **1.**

Units

xx	—	None.
nn	—	None.

Defaults

xx	Missing:	Error B.
	Out of range:	Error B.
	Floating point:	Error A.
nn	Missing:	Error C.
	Out of range:	Error C.

Description This command sets the saturation level of integral factor of the PID position closed loop. This is evaluated in **nn** (0 to 1) times of the maximum possible level of ouput signal.

The command can be sent at any time but it has no effect until the UF (update filter) is received.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the actual saturation level (0 to 1 times) of integral factor of the PID position closed loop.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.

Rel. Commands

KD	—	Set derivative gain.
KI	—	Set integral gain.
KP	—	Set proportional gain.
UF	—	Update servo filter.

Example

3KS0.5	*Set integral saturation factor for axis #3 to 0.5.*
…	
…	
…	
3UF	*Update PID filter; only now the KS command takes effect.*
3KS?	*Display actual value of KS.*
3KS0.5	*Controller tells the value.*

LP — List program

Usage	■ IMM ☐ PGM ☐ MIP
Syntax	**xxLP**
Parameters	
Description	**xx** [int] — Program number.
Range	**xx** — **1** to **127.**
Units	**xx** — None.
Defaults	**xx** Missing: Error F. Out of range: Error F. Floating point: Error A.
Description	This command reads a specified program from RAM and sends it to the selected communication port (RS232C or IEEE488). During the transmission no other command should be sent to the controller. To read a program from the non-volatile memory, first use the MP command to download its content in RAM.
Returns	Program listing.
Errors	A — Unknown message code. F — Program number incorrect. G — Program does not exist. I — Unauthorized command in programming mode.
Rel. Commands	**MP** — Download EEPROM to RAM.

Example

MP	*Copy programs from EEPROM to RAM.*
3LP	*List program number 3.*
1PA0	*Program listing.*
…	
…	
…	

LT — Extended list of the trajectory

Usage ■ IMM ■ PGM □ MIP

Syntax xxLT

Parameters

Description	**xx** [int]	—	Element number.
Range	**xx**	—	**1** to **100.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error C.

Description This command retrieves from the controller the extended form of an element of trajectory.

NOTE

When the element number is 0 or absent, all elements of the trajectory will be listed.

Returns The returned value is dependent of the element type and is as follow, when element is:

- *f* (LX): **xxLTaa, X=xx, Y=yy, A=tt**
- *f* (LY): **xxLTbb, X=xx, Y=yy, A=tt**
- *f* (MX, MY): **xxLTcc, X=xx, Y=yy, A=tt**
- *f* (CX, CY): **xxLTdd, X=xx, Y=yy, A=tt, R=rr, B=ss, S=ww**
- *f* (CR, CA): **xxLTee, X=xx, Y=yy, A=tt, R=rr, B=ss, S=ww**

where:

aa — Line (x, θ).
bb — Line (y, θ).
cc — Line (x, y).
dd — Arc (x, y).
ee — Arc (r, θ).
xx — X end position of the element.
yy — Y end position of the element.
tt — Angle of the tangent at the end position.
rr — Radius of the circle.
ss — Start angle for a circle.
ww — Sweep angle for a circle.

Errors
C — Parameter out of limits.
S — Communication time-out.

Rel. Commands
XT — Tell number of elements in the trajectory.
XE — Tell the last element.

Example

NT | *Clear trajectory.*
CR10 | *Define radius of an arc of circle = f (r, θ).*
CA90 | *Define sweep angle an build an arc of circle = f (r, θ).*
1LT | *Extended list of a trajectory.*
1LT, Arc (r, θ), X=10, Y=10, A=90, R=10, B=270, S=90
| *Controller tells the built element.*

LX — Define X position and build a line segment = *f* (LX, tangent)

Usage ■ IMM ■ PGM □ MIP

Syntax **LXnn**

Parameters

Description	**nn** [double]	—	X coordinate to reach with a line segment.
Range	**nn**	—	**-1.0 E^{12}** to **1.0 E^{12}.**
Units	**nn**	—	Preset units in SETUP mode.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command defines to the controller to build an element of trajectory of the type: line segment = *f* (LX, tangent).

Returns None.

Errors

C	—	Parameter out of limits.
H	—	Calculation overflow.
S	—	Communication time-out.
V	—	Too long trajectory.
Y	—	Trajectory: Line (x, y). Line expected.
[	—	Trajectory: Line (x, θ) or Line (y, θ) impossible.
e	—	Trajectory: Units not translationnal or not identical.

Rel. Commands

LY	—	Define Y position and build a line segment = *f* (LY, tangent).
XE	—	Tell the last element.

Example

NT	*Clear trajectory.*
FA45.0	*Define input tangent = 45°.*
LX10	*Define and build line segment = f (10.0, 45.0°).*
XE	*Tell last element.*
XE, Line (x, θ), 10, 10, 45	*Controller tells the built element.*

LY — Define Y position and build a line segment = *f* (LY, tangent)

Usage	■ IMM ■ PGM □ MIP
Syntax	**LYnn**

Parameters

Description	**nn** [double]	—	Y coordinate to reach with a line segment.
Range	**nn**	—	**-1.0 E^{12}** to **1.0 E^{12}.**
Units	**nn**	—	Preset units in SETUP mode.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command defines to the controller to build an element of trajectory of the type: line segment = *f* (LY, tangent).

Returns None.

Errors

C	—	Parameter out of limits.
H	—	Calculation overflow.
S	—	Communication time-out.
V	—	Too long trajectory.
Y	—	Trajectory: Line (x, y). Line expected.
e	—	Trajectory: Units not translationnal or not identical.
[	—	Trajectory: Line (x, θ) or Line (y, θ) impossible.

Rel. Commands

LX	—	Define X position and build a line segment = *f* (LX, tangent).
XE	—	Tell the last element.

Example

NT	*Clear trajectory.*
FA45.0	*Define input tangent = 45°.*
LY10	*Define and build line segment = f (10.0, 45.0°).*
XE	*Tell last element.*
XE, Line (y, θ), 10, 10, 45	*Controller tells the built element.*

Usage	■ IMM ■ PGM □ MIP
Syntax	**MC**
Parameters	None.

Description

This command activates the manual jog mode. In this mode, axes can be manually moved by pressing the appropriate low or high speed jog buttons on the front panel numerical keypad.

To exit the manual jog mode, press the QUIT function key. The manual jog mode can be terminated remotely by using the ST or AB commands. Turning the motor power off from the front panel or using the MF command also exits the manual jog mode.

NOTE

If the display was disabled by using the RD command, it will be re-enabled as long as the manual mode is active.

ATTENTION

If the motor power is off when the command is issued, it will turn it on and then enter the manual jog mode.

Returns None.

Errors D — Unauthorized execution.

Rel. Commands

MF — Motor OFF.
ML — Set local mode.
MR — Set remote mode.
ST — Stop motion.

Example

MC | *Enter manual jog mode.*

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxMF**
Parameters	
Description	**xx** [int] — Axis number.
Description	This command should be used as an emergency stop. On reception of this command, the controller stops motion on the indicated axis with a fast deceleration and then turns motor power OFF. If **xx** is missing, the controller stops motion on all axes The command can be also used to turn the motors off when a manual adjustment of the stage is desired.
Returns	None.
Errors	None.
Rel. Commands	**AB** — Abort motion. **MO** — Motor ON. **ST** — Stop motion.
Example	**MF** | *Stop all motion and turn motor off.*

MH — Set manual velocity

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxMHnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [float] — Manual jog high velocity value.
Range	**xx** — **1** to **8.** **nn** — **1E^{-6}** to **the programmed velocity value in SETUP mode**.
Units	**xx** — None. **nn** — Preset units in SETUP mode/second.
Defaults	**xx** Missing: Error B. Out of range: Error B. Floating point: Error A. **nn** Missing: Error C. Out of range: Error C.
Description	This command sets the manual jog high velocity value of an axis (from front panel or joystick). This is the high speed manual jog mode, activated by simultaneously pressing the center key with a direction key. The manual jog low speed is 1/10 of the high speed. The manual jog high speed can also be changed from the front panel SETUP menu.
Returns	None.
Errors	A — Unknown message code. B — Incorrect axis number. C — Parameter out of limits.
Rel. Commands	**DM** — Read manual velocity.

Example

2MH4.5	*Set axis #2 manual jog high velocity to 4.5.*
2DM	*Read manual jog high velocity of axis #2.*
2DM4.5	*Controller returns a manual velocity value of 4.5 units/sec.*

ML — Set local mode

Usage	■ IMM	■ PGM	□ MIP

Syntax **ML**

Parameters None.

Description This command activates the local mode. In this mode, the control is passed to the front panel and all its functionality becomes available.
To return to remote mode use MR command.

NOTE

If the ML command is issued while a program or a motion is in progress, the controller will first abort the program and stop all axes, similarly to a ST command, before switching to local mode.

Returns None.

Errors D — Unauthorized execution.

Rel. Commands **MC** — Set manual mode.
MR — Set remote mode.

Example

ML	*Set local mode.*

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxMO**

Parameters

Description **xx** [int] — Axis number.

Description This command turns motor power on for the indicated axis. If xx is missing, all axes are turned power on. It is equivalent to the front panel MOTOR ON button.

CAUTION

If the motor power is turned off by the controller detecting a fault condition, before turning the power back on, make sure that the cause of the fault is corrected.

Returns None.

Errors None.

Rel. Commands
AB — Abort motion.
MF — Motor OFF.
ST — Stop motion.

Example **MO** | *Turn power on to all motors.*

MP — Download EEPROM to RAM

Usage	■ IMM ☐ PGM ☐ MIP
Syntax	**MP**
Parameters	None.
Description	This command copies the programs stored in non-volatile memory to RAM. When a program is called for execution or editing, it is automatically copied to RAM. When it is erased from RAM with XX command, it can be restored from nonvolatile memory with this command.
Returns	None.
Errors	I — Unauthorized command in programming mode.
Rel. Commands	**LP** — List program. **XX** — Erase program. **SM** — Save program.
Example	**MP** | *Copy programs from non-volatile memory to RAM.*

MR — Set remote mode

Usage	■ IMM ■ PGM ☐ MIP
Syntax	**MR**
Parameters	None.
Description	This command activates the remote mode. In this mode all function keys and the keypad on the front panel are disabled. The front panel still displays motion and status information but only the power and motor power buttons remain active. Not recommended for use in programming mode.

NOTE

If the MR command is issued while a program or a motion is in progress, the controller will first abort the program and stop all axes, similarly to a ST command, before switching to remote mode.

Returns	None.
Errors	D — Unauthorized execution.
Rel. Commands	**MC** — Set manual mode. **ML** — Set local mode.
Example	**MR** | *Set controller in remote mode.*

MS — Read motor status

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxMS**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**0** to **4.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the motor status byte of the specified axis. If the axis number (**xx**) is missing or set to 0, the controller returns the motor status bytes for all four axes, separated by a comma.

Each bit of the status byte represents a particular axis parameter, as described in the following table:

Bit #	Function	Meaning for 0	Meaning for 1
0	Axis in Motion	NO	YES
1	Motor power	ON	OFF
2	Motion direction	Negative	Positive
3	Right (+) travel limit	Not tripped	Tripped
4	Left (-) travel limit	Not tripped	Tripped
5	Mechanical zero signal	Low	High
6	Not used	—	Default
7	Not used	Default	—

The byte returned is in the form of an ASCII character. Converting the ASCII code to binary will give us the status bits values.

NOTE

For a complete ASCII to binary conversion table, see Appendix F, ASCII Table.

Returns **xxMSaa** or **xx_1MSaa$_1$, xx_2MSaa$_2$, xx_3MSaa$_3$, xx_4MSaa$_4$**

xx, xx_1, xx_2, xx_3, xx_4
— Axis number.

aa, aa_1, aa_2, aa_3, aa_4
— ASCII character representing the status byte.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands
TS — Read controller status.
TX — Read controller activity.

Example

2MS	*Read motor status byte for axis #2.*
2MSe	*Controller returns character e, or ASCII character 101; converting 101 to binary we get 01100101 which has the following meaning: axis in motion, motor power ON, motion direction positive, no limits tripped and mechanical zero high.*

MT — Move to travel limit switch

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxMTnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** — Type of limit.
Range	**xx** — **1** to **8.** **nn** — + or **–.**
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: Error B. Out of range: Error B. **nn** Missing: Error C. Out of range: Error C.

Description

This command directs the MM4006 to move until it senses the physical travel limit. The parameter + or – sets the direction of motion.

Normally, when a travel limit switch is encountered during motion, the MM4006 stops all motion and generates an error message and turns the motor's power off. However, with this command, reaching the travel limit is the desired function so other motions will not be stopped and an error message will not be generated.

NOTE

It is recommanded to set the velocity of the stage to not more than 10% of its maximum velocity when using this command to avoid mechanical damage.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands **MV** — Infinite movement.

Example

1MT+ | *Move axis #1 to positive limit.*
3MT- | *Move axis #3 to negative limit.*

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxMV+** or **xxMV-**

Parameters

Description

xx	[int]	—	Axis number.
+		—	Displacement in positive direction.
-		—	Displacement in negative direction.

Range **xx** — **1** to **8.**

Units **xx** — None.

Defaults **xx**

Missing:	Error B.
Out of range:	Error B.
Floating point:	Error A.

Description This command starts an infinite movement with the velocity predefined by VA command. To stop movement, use ST command.

NOTE

While a motion is in progress, care should be taken not to reverse direction of motion. When this command is received, the controller verifies if it will produce a change of direction. If so, it will refuse the execution and set error code D.

NOTE

If the axis was previously defined as a synchronized axis, MV command do not generate a motion. For synchronized axes use SE command to execute a motion.NOTE

NOTE

Using of the this command is possible only after setting of a periodic cycle (CD command) and only for rotary stages.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
D — Unauthorized execution.

Rel. Commands

AC — Set acceleration.
MT — Move to travel limit switch.
PA — Move to absolute position.
PR — Move to relative position.
ST — Stop motion.
VA — Set velocity.

Example

2VA8	*Set velocity of axis #2 to 8 units/sec.*
2CD360	*Set cycle value to 360° and activate the mode.*
2MV+	*Move axis #2 with velocity of 8 units/sec.*
ST	*Stop motion.*

MX — Define X position for a line segment = *f* (MX, MY)

Usage	■ IMM ■ PGM □ MIP
Syntax	**MXnn**
Parameters	
Description	**nn** [double] — X coordinate to reach with a line segment.
Range	**nn** — **-1.0 E^{12}** to **1.0 E^{12}.**
Units	**nn** — Preset units in SETUP mode.
Defaults	**nn** Missing: Error C. Out of range: Error C.

Description This command defines to the controller X position to reach with an element of trajectory of the type:

Line segment = *f* (MX, MY).

Returns None.

Errors

C	—	Parameter out of limits.
S	—	Communication time-out.
V	—	Too long trajectory.
Y	—	Trajectory: Line (x, y). Line expected.
Z	—	Trajectory: Line (x, y). Too big discontinuity.
e	—	Trajectory: Units not translationnal or not identical.

Rel. Commands

MY	—	Define Y position and build a line segment = *f* (MX, MY).
XE	—	Tell the last element.

Example

NT	*Clear trajectory.*
MX10	*Define X position of a line segment = f (x, y).*
MY10	*Define Y position an build a line segment = f (x, y).*
XE	*Tell last element.*
XE, Line (x, y), 10, 10, 45	*Controller tells the built element.*

MY — Define Y position and build a line segment = f (MX, MY)

Usage ■ IMM ■ PGM □ MIP

Syntax **MYnn**

Parameters

Description	**nn** [double]	—	Y coordinate to reach with a line segment.
Range	**nn**	—	**-1.0 E^{12}** to **1.0 E^{12}.**
Units	**nn**	—	Preset units in SETUP mode.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command defines to the controller the Y position to reach and tells to the controller to build an element of trajectory of the type:

Line segment = f (MX, MY).

Returns None.

Errors

- C — Parameter out of limits.
- S — Communication time-out.
- V — Too long trajectory.
- Y — Trajectory: Line (x, y). Line expected.
- Z — Trajectory: Line (x, y). Too big discontinuity.
- e — Trajectory: Units not translationnal or not identical.

Rel. Commands

- **MX** — Define X position for a line segment = f (MX, MY).
- **XE** — Tell the last element.

Example

NT	*Clear trajectory.*
MX10	*Define X position of a line segment = f (x, y).*
MY10	*Define Y position an build a line segment = f (x, y).*
XE	*Tell last element.*
XE, Line (x, y), 10, 10, 45	*Controller tells the built element.*

NB — Set trajectory element where the generation of pulses starts

Usage	■ IMM	■ PGM	□ MIP

Syntax **NBnn** or **NB?**

Parameters

Description	**nn** [int]	—	Number of trajectory element where the pulses generation commences.
	?	—	Read the number of trajectory element where the pulses generation starts.
Range	**nn**	—	**1** to **100.**
Units	**nn**	—	None.
Defaults	**nn**	Missing:	1.
		Out of range:	Error C.

Description This command sets number of trajectory element where the generation of pulses commences. The generation of pulses is started immediately in the beginning of this element.

NOTE

As the total element number of a trajectory may be inferior than 100 and the value of NB must be ≤ the value of NE ≤ the total element number, this value of NB will be reexamined in NS and ET commands.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the number of trajectory element where the generation of pulses commences.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands

NE	—	Set trajectory element where the generation of pulses ends.
NI	—	Set step (curvi-linear distance) between synchronisation pulses.
NN	—	Set number of synchronisation pulses to generate.

Example

NB3	*Generation of pulses starts at the beginning of the element #3.*
NB?	*Read element number where the generation of pulses starts.*
NB3	*Controller tells the value.*

NE — Set trajectory element where the generation of pulses ends

Usage ■ IMM ■ PGM □ MIP

Syntax **NEnn** or **NE?**

Parameters

Description

nn	[int]	—	Number of trajectory element where the pulses generation ends.
?		—	Read the number of trajectory element where the pulses generation ends.

Range **nn** — **1** to **100.**

Units **nn** — None.

Defaults **nn** Missing: 1.
Out of range: Error C.

Description This command sets number of trajectory element where the generation of pulses ends. The generation of pulses is ended immediately in the end of this element.

NOTE

As the total element number of a trajectory may be inferior than 100 and the value of NB must be ≤ the value of NE ≤ the total element number, this value of NE will be reexamined in NS and ET commands.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the number of trajectory element where the generation of pulses ends.

Errors
- A — Unknown message code.
- B — Incorrect axis number.
- C — Parameter out of limits.
- D — Unauthorized execution.

Rel. Commands
- **NB** — Set trajectory element where the generation of pulses starts.
- **NI** — Set step (curvi-linear distance) between synchronisation pulses.
- **NN** — Set number of synchronisation pulses to generate.

Example

NE4	*Generation of pulses ends at the end of the element #4.*
NE?	*Read element number where the generation of pulses ends.*
NE4	*Controller tells the value.*

NI — Set step (curvi-linear distance) between synchronisation pulses

Usage	■ IMM	■ PGM	□ MIP

Syntax **NInn** or **NI?**

Parameters

Description	**nn** [long]	—	New value of step between pulses.
	?	—	Read step.
Range	**nn**	—	**≥2 * Max {X and Y axis encoder resolution}.**
Units	**nn**	—	Current unit.
Defaults	**nn**	0 or missing:	NO pulse is generated.
		Out of range:	Error C.

Description This command sets the value of step between pulses to generate between the elements defined by NB and NE. If **nn** is default or zero, the generation of pulses of synchronisation is disabled.

This value of NI will be reexamined in ET command.

NOTE

Because NI and NN are complement commands, the last entered NI command value replaces all of precedently entered NI or NN commands' one.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the step between synchronisation pulses to generate. 0 means that no pulse is generated.

Errors

- A — Unknown message code.
- B — Incorrect axis number.
- C — Parameter out of limits.
- D — Unauthorized execution.

Rel. Commands

- **NB** — Set trajectory element where the generation of pulses starts.
- **NE** — Set trajectory element where the generation of pulses ends.
- **NN** — Set number of synchronisation pulses to generate.

Example

NI0.5	*Set step between pulses to 0.5 unit.*
NI?	*Read number of pulses to generate.*
NI0.5	*Controller tells the value.*

NN — Set number of synchronisation pulses to generate

Usage	■ IMM	■ PGM	□ MIP
Syntax	**NNnn** or **NN?**		
Parameters			
Description	**nn** [long]	—	New value of number of pulses to generate.
	?	—	Read the defined number of pulses to generate.
Range	**nn**	—	**2** to **2147385345.**
Units	**nn**	—	None.
Defaults	**nn**	0 or missing:	NO pulse is generated.
		Out of range:	Error C.

Description This command sets the number of synchronisation pulses to generate between the elements defined by NB and NE. If **nn** is default or zero, the generation of pulses of synchronisation is disabled.

The possible maximum value of NN is MPN (Maximum Pulse Number) that will be examined in ET command.

NOTE

Because NI and NN are complement commands, the last entered NN command value replaces all of precedently entered NI or NN commands' one.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the number of synchronisation pulses to generate between the elements defined by NB and NE. 0 means that no pulse is generated.

Errors

- A — Unknown message code.
- B — Incorrect axis number.
- C — Parameter out of limits.
- D — Unauthorized execution.

Rel. Commands

- **NB** — Set trajectory element where the generation of pulses starts.
- **NE** — Set trajectory element where the generation of pulses ends.
- **NI** — Set step (curvi-linear distance) between synchronisation pulses.

Example

NN11	*Set number of pulses to 11.*
NN?	*Read number of pulses to generate.*
NN11	*Controller tells the value.*

NP — Set decimal digits number of position display

Usage ■ IMM ■ PGM □ MIP

Syntax **xxNPnn** or **xxNP?**

Parameters

Description
xx [int] — Axis number.
nn [int] — New value of displayed resolution.

Range
xx — **1** to **8.**
nn — **1** to **MDR** (Maximum Display Resolution).

Units
xx — None.
nn — None.

Defaults
xx Missing: Error B.
Out of range: Error B.
Floating point: Error A.
nn Missing: Default value of actual unit.
Out of range: Error C.

Description This command sets new value of number of digits after the decimal point of on screen displayed position values. The MDR value, dependant on the actual unit, is described below:

Unit	mm	µm	In	mIn	µIn	Dg	Gr	Rad	mRd	µRd	Inc
MDR	6	3	7	4	1	6	6	6	3	1	0

To restore the default value of the actual unit, use xxNP.

NOTE
This command returns an error code if the actual unit is Inc.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the number of decimal digits after the decimal point of on screen displayed position values.

Errors
A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands
SF — Set axis mechanical motion device.
SN — Set axis displacement units.

Example

1SFUTM100CC0.5HA	*Set mechanical driver to UTM100CC0.5HA.*
1SNmI.	*Set unit to mInch.*
1NP4	*Set position displayed precision to 4.*

NQ — Read global acquisition nr.

Usage ■ IMM ■ PGM ■ MIP

Syntax **NQ**

Parameters None.

Description This command reads the current number of global trace acquisitions. During a global trace mode initiated by the GQ command, the number of stored samples can be read to monitor the progress of the acquisition process.

Returns **NQnn**

nn — Number of acquired samples.

Errors S — Communication time-out.

Rel. Commands

GQ — Set global trace mode.
SQ — Set global sample rate.
TQ — Read global trace data.
XQ — Read global sample rate.

Example

SQ0.005	*Set global trace sample period to 5ms.*
GQ1000	*Enable trace mode for axis #2 and acquire 1000 samples.*
2PR0.2, 3PR1	*Start a relative motion on axis #2 and axis #3.*
NQ	*Read the number of samples acquired.*
NQ157	*Controller reports 157 global trace samples acquired.*
NQ	*Read the number of samples acquired.*
NQ342	*Controller reports 342 global trace samples acquired.*
WS, **NQ**	*Wait for stop and read the number of samples acquired.*
NQ1000	*Controller reports 1000 global trace samples acquired.*

NS — Allow generation of pulses on interpolation

Usage ■ IMM ■ PGM □ MIP

Syntax **NS**

Parameters None.

Description This command, together with ET command, verifies the correctness of data enterred by NB, NE and NI, NN before allow generation of pulses along the trajectory. If a condition is not satisfied, the generation of pulses is disabled and the command returns an error. If not, pulses are generated and at every moment where a pulse is generated, the X and Y axis positions are stocked in the global trace buffer and can be reread by TQ command.

The pulses are generated on pin 12 of the 25-pin auxiliary connector with a pulse width of about 5 µsec.

NOTE

This command, if successful, erases effect of the global trace mode precedently defined by any GQ command.

NOTE

This command, if used, must precede immediately ET command.

Returns None.

Errors

D — Unauthorized execution.

f — Synchronization pulses generation impossible.

Rel. Commands

NB — Set trajectory element where the generation of pulses starts.

NE — Set trajectory element where the generation of pulses ends.

NI — Set step (curvi-linear distance) between synchronisation pulses.

NN — Set number of synchronisation pulses to generate.

Example

NT, FA90	*Initialisation.*
CR10, CA 5	*Element 1.*
CA350	*Element 2.*
CA5	*Element 3.*
NB2	*Set pulses start to element 2.*
NE2	*Set pulses end to element 2.*
NN21	*21 pulses will be generated within element 2.*
VV5	*Set trajectory velocity to 5 units/sec.*
NS	*Allow generation of pulses on interpolation.*
ET	*Execute trajectory with generation of pulses.*
TQ	*Read stocked data.*

NT — Start definition of a new trajectory

Usage	■ IMM ■ PGM □ MIP
Syntax	**NT**
Parameters	None.
Description	This command tells to the controller to reset the trajectory buffer and to get ready to load a new trajectory for execution. NT sets the initial position (X, Y) and first tangent angle to 0.0.
Returns	None.
Errors	D — Unauthorized execution. S — Communication time-out.
Rel. Commands	**EL** — Erase the last element of trajectory. **LT** — Extended list of the trajectory. **XE** — Tell the last element.
Example	**NT** | *Reset current trajectory.*

OA — Set home search acceleration

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxOAnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [int] — Acceleration value.
Range	**xx** — **1** to **8.** **nn** — **10^{-6}** to **the programmed value in SETUP mode.**
Units	**xx** — None. **nn** — Preset units in SETUP mode/sec^2.
Defaults	**xx** Missing: Error B. Out of range: Error B. Floating point: Error A. **nn** Missing: Error C. Out of range: Error C.
Description	This command sets the acceleration and deceleration portion of the velocity profile generator for home search. All subsequent home search accelerations and decelerations will be executed with the new value.
Returns	None.
Errors	A — Unknown message code. B — Incorrect axis number. C — Parameter out of limits.
Rel. Commands	**OH** — Set home search high velocity. **OL** — Set home search low velocity. **OR** — Search for home.

Example

3OA50 | *Set home search acceleration to 50 units/sec^2 for axis #3.*

OE — Test I/O output

Usage	■ IMM	■ PGM	□ MIP

Syntax **xxOEnn**

Parameters

Description	**xx** [int]	—	I/O output bit number.
	nn [int]	—	I/O output bit or byte state.
Range	**xx**	—	**0** to **8.**
	nn	—	**0** to **1** or **0** to **255.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.
		Floating point:	Decimal part truncated.

Description

This command is one of the flow control instructions, enabling a conditional execution of a command line depending on the state of an I/O output bit. It must be placed at the beginning of the command line of which execution it controls. If the selected output bit **xx** has the specified state **nn**, all following commands on that line are executed. If **xx** is set to 0 or missing, the test is performed on the entire I/O output byte and then **nn** could have a value from 0 to 255, representing the byte value to compare it with.

As described in the "Command Syntax" paragraph, a line is defined as all commands between two line terminators.

Even though the command can be used on a line in immediate mode, its primary use is inside a program.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
E	—	Incorrect I/O channel number.
L	—	Command not at the beginning of a line.

Rel. Commands **IE** — If I/O input is equal.

Example

3OE0, 1PA2.34 | *If I/O output bit #3 is low, move axis #1 to position 2.34.*

OH — Set home search high velocity

Usage	■ IMM ■ PGM ☐ MIP
Syntax	**xxOHnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [float] — Home search velocity.
Range	**xx** — **1** to **8.** **nn** — **0.000001** to **Maximum motion speed defined in SETUP**.
Units	**xx** — None. **nn** — Units/sec.
Defaults	**xx** Missing: Error B. Out of range: Error B. Floating point: Error A. **nn** Missing: Error C. Out of range: Error C.
Description	This command sets the high velocity of the HOME search algorithm of the selected axis. For a detailed description of the home search routine see the Home Search paragraph in the Motion Control Tutorial section.
Returns	None.
Errors	A — Unknown message code. B — Incorrect axis number. C — Parameter out of limits.
Rel. Commands	**DO** — Read home search velocity. **OR** — Search for home.
Example 3OH10	*Set home search high velocity of axis #2 to 10 units/sec.*

OL — Set home search low velocity

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxOLnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	Low velocity value.
Range	**xx**		—	**1** to **8.**
	nn		—	**10^{-6}** to **Maximum motion speed defined in SETUP mode**.
Units	**xx**		—	None.
	nn		—	Preset units in SETUP mode/sec.
Defaults	**xx**	Missing:		Error B.
		Out of range:		Error B.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.

Description

This command sets the desired value for low absolute velocity used during home search.

For a detailed description of the home search routine see the home search paragraph in the Motion Control Tutorial section.

Returns None.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.

Rel. Commands

OA	—	Set home search acceleration.
OH	—	Set home search high velocity.
OR	—	Search for home.

Example

3OL5	*Set home search low velocity to 5 units/sec to axis #3.*

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxORnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [int] — Home search position option.
Range	**xx** — **0** to **4.** **nn** — **0** to **2.** **nn** = **0**: Move to zero position instead of origin search. **nn** = **1**: Search mechanical zero and encoders top zero. **nn** = **2**: Search mechanical zero, but do not search encoders top zero.
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: 0. Out of range: Error B. Floating point: Error A. **nn** Missing: 1. Out of range: Error C.

Description

This command executes a home search routine on the axis specified by **xx**. If **xx** is missing or set to 0, a home search is initiated sequentially on all installed axes, in the order specified in the General SETUP utility on the front panel.

For details on how to set the home search axes sequence see the System Setup paragraph of the Introduction section.

For a detailed description of the home search routine see the Home Search Motion Profile Section in the Motion Control Tutorial section.

NOTE

There is a maximum allowed time for this command to execute, defined in the front panel General SETUP menu. If the motion device does not find the home position in the specified time, the controller stops the search and turns motor power off.

NOTE

This command should be executed once every time the power is turned on. There is no need to issue this command in any other case since the controller always keeps track of position, even when the motor power is off.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Paramètre hors limites.

Rel. Commands

DH — Define home.
HT — Set home type.
OH — Set home search high velocity.
ZH — Set and save hard home preset position.

Example

3OR1 | *Perform a search of mechanical zero and encoders top zero, on axis #3.*

PA — Move to absolute position

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxPAnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	Absolute position destination.
Range	**xx**		—	**1** to **8.**
	nn		—	Any position within the software travel limits.
Units	**xx**		—	None.
	nn		—	Defined motion units.
Defaults	**xx**	Missing:		Error B.
		Out of range:		Error B.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.

Description This command initiates an absolute motion. When received, the selected axis **xx** will move, with the predefined acceleration and velocity to the absolute position specified by **nn**.

NOTE

If the motor power is turned off, MO command which turns motor power on is executed before PA commmand, except if the controller has detected a fault condition.

NOTE

Even though the command is accepted while a motion is in progress, care should be taken not to reverse direction of motion. When this command is received, the controller verifies if it will produce a change of direction. If so, it will refuse the execution and set error code D.

NOTE

If the axis was previously defined as a synchronized axis, PA command will only set the destination but not generate a motion. For synchronized axes use SE command to execute a motion.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands

AC — Set acceleration.
PR — Move to relative position.
ST — Stop motion.
VA — Set velocity.

Example

3VA8 | *Set velocity of axis #2 to 8 units/sec.*
3PA12.34 | *Move axis #3 to absolute position 12.34.*

PB — Set start position of generation of pulses of synchronisation

Usage ■ IMM ■ PGM □ MIP

Syntax **xxPBnn** or **xxPB?**

Parameters

Description
- **xx** [int] — Axis number.
- **nn** [float] — New value of start position of generation of pulses.
- **?** — Pulses of synchronisation for the axis #**xx**.

Range
- **xx** — **1** to **8.**
- **nn** — **Min. logical allowed position** to **Max. logical allowed position.**

Units
- **xx** — None.
- **nn** — Preset units in SETUP mode.

Defaults
- **xx** Missing: Error B.
  Out of range: Error B.
  Floating point: Error A.
- **nn** Missing: 0.
  Out of range: Error C.

Description This command sets start position of generation of pulses of synchronisation for an axis.

The command can be sent at any time but it has no effect until the PS command is received.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the start position of generation of pulses of synchronisation for **xx** numbered axis.

Errors
- A — Unknown message code.
- B — Incorrect axis number.
- C — Parameter out of limits.

Rel. Commands
- **PE** — Set end position of generation of pulses of synchronisation.
- **PI** — Set step of generation of pulses of synchronisation.

Example

2PB-10	*Set start position for axis #2 to -10 units.*
…	
…	
…	
2PB?	*Actual value of PB ?*
2PB-20	*Controller tells the actual value.*
…	
…	
…	
2PS	*Update PB, PE, PI and allow pulses.*
…	
…	
…	
2PB?	*Actual value of PB ?*
2PB-10	*Controller tells the actual value.*

PE — Set end position of generation of pulses of synchronisation

Usage ■ IMM ■ PGM □ MIP

Syntax **xxPEnn** or **xxPE?**

Parameters

Description

xx	[int]	—	Axis number.
nn	[float]	—	New value of end position of generation of pulses.
?		—	Read the actual end position of generation of pulses of synchronisation for the axis #**xx**.

Range

xx	—	**1** to **8.**
nn	—	**Min. logical allowed position** to **Max. logical allowed position.**

Units

xx	—	None.
nn	—	Preset units in SETUP mode.

Defaults

xx	Missing:	Error B.
	Out of range:	Error B.
	Floating point:	Error A.
nn	Missing:	0.
	Out of range:	Error C.

Description This command sets end position of generation of pulses of synchronisation for an axis.

The command can be sent at any time but it has no effect until the PS command is received.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the end position of generation of pulses of synchronisation for **xx** numbered axis.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.

Rel. Commands

PB	—	Set start position of generation of pulses of synchronisation.
PI	—	Set step of generation of pulses of synchronisation.

Example

2PE10	*Set end position for axis #2 to 10 units.*
...	
...	
...	
2PE?	*Actual value of PE ?*
2PE20	*Controller tells the actual value.*
...	
...	
...	
2PS	*Update PB, PE, PI and allow pulses.*
...	
...	
...	
2PE?	*Actual value of PE ?*
2PE10	*Controller tells the value.*

PI — Set step of generation of pulses of synchronisation

Usage	■ IMM		■ PGM	□ MIP
Syntax	**xxPInn** or **xxPI?**			
Parameters				
Description	**xx** [int]	—	Axis number.	
	nn [float]	—	New value of step of generation of pulses.	
	?	—	Read the actual step of generation of pulses of synchronisation for the axis #**xx**.	
Range	**xx**	—	**1** to **8.**	
	nn	—	**2 x Coder precision** to **maximum allowed travel**.	
Units	**xx**	—	None.	
	nn	—	Preset units in SETUP mode.	
Defaults	**xx**	Missing:	Error B.	
		Out of range:	Error B.	
		Floating point:	Error A.	
	nn	0 or missing:	Stop generation of pulses.	
		Out of range:	Error C.	

Description This command sets step of generation of pulses of synchronisation for an axis. If **nn** is default or zero, the generation of pulses of synchronisation is disabled.

The command can be sent at any time but it has no effect until the PS command is received.

The minimum value of step: the displacement of stage within T_{base} will be verified in PS command.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the step of generation of pulses of synchronisation for **xx** numbered axis. 0 means that no pulse is generated.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.

Rel. Commands

PB	—	Set start position of generation of pulses of synchronisation.
PE	—	Set end position of generation of pulses of synchronisation.

Example

2PI0.1	*Set step of pulses for axis #2 to 0.1 unit.*
…	
…	
…	
2PI?	*Actual value of PI ?*
2PI0.5	*Controller tells the actual value.*
…	
…	
…	
2PS	*Update PB, PE, PI and allow pulses.*
…	
…	
…	
2PI?	*Actual value of PI ?*
2PI0.1	*Controller tells the value.*

PR — Move to relative position

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxPRnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [float] — Relative motion increment.
Range	**xx** — **1** to **8.** **nn** — Any value that will not cause exceeding the software limits.
Units	**xx** — None. **nn** — Defined motion units.
Defaults	**xx** Missing: Error B. Out of range: Error B. Floating point: Error A. **nn** Missing: Error C. Out of range: Error C.

Description

This command initiates a relative motion. When received, the selected axis **xx** will move, with the predefined acceleration and velocity, to a relative position **nn** units away from the current position.

NOTE

If the motor power is turned off, MO command which turns motor power on is executed before PR commmand, except if the controller has detected a fault condition.

NOTE

Even though the command is accepted while a motion is in progress, care should be taken not to reverse direction of motion. When this command is received, the controller verifies if it will produce a change of direction. If so, it will refuse the execution and set error code D.

NOTE

If the axis was previously defined as a synchronized axis, PR command will only set the destination but not generate a motion. For synchronized axes use SE command to execute a motion.

Returns	None.
Errors	A — Unknown message code. B — Incorrect axis number. C — Parameter out of limits. D — Unauthorized execution.
Rel. Commands	**AC** — Set acceleration. **PA** — Move to absolute position. **ST** — Stop motion. **VA** — Set velocity.
Example	3VA8 | *Set velocity of axis #3 to 8 units/sec.* **3PR2.34** | *Move axis #3 2.34 units away from the current position.*

PS — Allow generation of pulses on motion

Usage ■ IMM ■ PGM □ MIP

Syntax **xxPSpp**

Parameters

Description **xx** [int] — Axis number.
pp [int] — Auxiliary parameter.

Range **xx** — **0** to **4.**
pp — **0** to **3.**

Units **xx** — None.
pp — None.

Defaults **xx** Missing: Error B.
Out of range: Error B.
Floating point: Error A.
pp Missing: 0.

Description

This command verifies the correctness of data enterred by PB, PE and PI before allow generation of pulses for an axis. If a condition is not satisfied, the generation of pulses is disabled and the command returns an error. If nof, pulses are generated in the course of axis displacement.

At every moment where a pulse is generated:

- If **pp** = 0 or missing: actual position of **xx** axis is stocked in the axis trace buffer and can be reread by TT command.
- If **pp** = 1: actual positions of all axis are stocked in the global trace buffer and can be reread by TQ command.
- If **pp** = 2: actual positions are not stocked.
- If **pp** = 3 this command is used on-line (axis in mouvement) to update PB, PE or PI commands that are newly entered.

The pulses are generated on pin 11 of the 25-pin auxiliary connector with a pulse width of about 5 µsec.

NOTE

This command, if successful, erases the effect of trace mode precedently defined by TM command if pp = 0 or missing, or the effect of trace mode precedently defined by GQ command if pp = 1.

NOTE

This command, if used with pp ≠ 3, must precede immediately PA, PR or SE command.

NOTE

The starting and ending axis motion must be out of the interval defined by PB and PE commands.

Returns None.

Errors
A — Unknown message code.
B — Incorrect axis number.
D — Unauthorized execution.
f — Synchronization pulses generation impossible.

Rel. Commands

PB — Set start position of generation of pulses of synchronisation.
PE — Set end position of generation of pulses of synchronisation.
PI — Set step of generation of pulses.

Example

1PB-20	*Set start position for axis #1 to -20 units.*
1PE0	*Set end position for axis #1 to 0 units.*
1PI2	*Set step to 2 units.*
1PA-50	*Displacement without generation of pulses.*
1PS	*Allow generation of pulses.*
1PA50	*Displacement with generation of pulses.*
1PB10	*New PB.*
1PE30	*New PE.*
1PI1	*New PI.*
1WP5	*Wait the axis #1 for 5 units.*
1PS3	*Update PB, PE, PI.*
TT	*Read data.*

PT — Calculate necessary time for axis displacement

Usage ■ IMM ■ PGM □ MIP

Syntax **xxPTnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn	—	Distance of displacement.
Range	**xx**	—	**0** to **4.**
	nn	—	Float.
Units	**xx**	—	None.
	nn	—	Actual unit.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command calculates the necessary time for the displacement of axis #**xx** of distance **nn**.

Returns The necessary time (seconds) for displacement of axis #**xx** of distance **nn**.

Errors

- A — Unknown message code.
- B — Incorrect axis number.
- C — Parameter out of limits.
- D — Unauthorized execution.

Rel. Commands None.

Example

1PT20	*Calculate the time for axis #1 displacement of 20 units.*
1PT1.25	*Controller returns value in seconds.*

PW — Save parameters

Usage	■ IMM	■ PGM	□ MIP	
Syntax	**xxPW**			
Parameters				
Description	**xx** [int]	—	Axis number.	
Range	**xx**	—	**0** to **4.**	
Units	**xx**	—	None.	
Defaults	**xx**	Missing:	0.	
		Out of range:	Error B.	
		Floating point:	Error A.	

Description

This command saves in non-volatile memory all parameters of the selected **xx** axis that have been changed through on-line commands or from within a program. If the axis specifier **xx** is not present or set to 0, parameters of all axes will be saved.

For a list and description of the axis parameters that are stored in non-volatile memory see the System Setup paragraph of the Introduction section.

NOTE

Since this command is equivalent to making changes in SETUP mode, it is valid only when motor power is turned off. If the command is issued when the motor power is on, the controller will ignore it and report error D.

NOTE

Before saving, make sure that the new set of parameters is correct and safe to use. Also, it is good practice to verify that the parameter saving procedure worked properly by issuing the TB or TE command afterwards.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
D — Unauthorized execution.
U — Failure while accessing the EEPROM.

Rel. Commands All device, motion and filter parameter setting commands.

Example

2KD0.02, 2UF	*Set Kd parameter of axis #2 to 0.02 and update filter.*
…	
…	*Verify the new parameter is working fine.*
…	
2PW	*Save parameters of axis #2 to non-volatile memory.*
TE	*Read error register.*
TE@	*Controller returns a no error code.*

QP — Quit program mode

Usage ■ IMM ☐ PGM ☐ MIP

Syntax **QP**

Parameters None.

Description This command terminates the program entry mode and sets the controller back to immediate mode. All the commands following this one will be executed immediately.

Returns None.

Errors I — Unauthorized command in programming mode.

Rel. Commands
EP — Edition of program.
XX — Erase program.

Example

3XX	*Clear program #3 from memory.*
3EP	*Activate program mode and enter following commands as rogram 3.*
…	
…	
…	
3QP	*End entering program number 3 and quit programming mode.*
3CP	*Compile program number 3.*
3CP@	*Controller confirms compilation of program number 3 without any errors.*

QW — Save general parameters

Usage ■ IMM ■ PGM □ MIP

Syntax **QW**

Parameters None.

Description This command saves in non-volatile memory all general parameters that have been changed through on-line commands or from within a program.

For a list and description of the general parameters that are stored in non-volatile memory see the General Setup paragraph of the Local Mode section.

NOTE

During the execution of this command, the communication (IEEE / RS232) is broken off.

NOTE

Since this command is equivalent to making changes in SETUP mode, it is valid only when motor power is turned off. If the command is issued when the motor power is on, the controller will ignore it and report error D.

NOTE

Before saving, make sure that the new set of parameters is correct and safe to use. Also, it is good practice to verify that the parameter saving procedure worked properly by issuing the TB or TE command afterwards.

Returns None.

Errors

A — Unknown message code.

D — Unauthorized execution.

Rel. Commands **PW** — Save parameters.

Example

CMM0B19200	*Set RS-232-C liaison with 19200 Baud.*
…	
…	*Verify the new parameter is working fine.*
…	
QW	*Save general parameters non-volatile memory.*
TE	*Read error register.*
TE@	*Controller returns a no error code.*

RA — Read analog input

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxRA**

Parameters

Description	**xx** [int]	—	Analog port number.
Range	**xx**	—	**0** to **4.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.

Description

This command reads one analog input port. The analog ports are four 12 bit A/D converters that accept signals in the predefined voltage range (±10 V, ±5 V, 0 to +10 V, 0 to +5 V). The read value, included between -10 and +10, is reported to the PC in floating format and is the direct tension.

If **xx** is missing or set to 0, the controller returns the values found in all four A/D converters, successively.

NOTE

For the hardware definition of the analog input port, please see Appendix B, Connector Pinouts, Remote Control Connector.

Returns

xxRAnn or **xx_1RAnn_1, xx_2RAnn_2**

xx, xx1, xx2

— Analog port number.

nn, nn_1, nn_2

— Analog port value, in ASCII format.

NOTE

The value sent for each not connected analog port is not significative.

Errors

A	—	Unknown message code.
E	—	Incorrect I/O channel number.
S	—	Communication time-out.

Rel. Commands

RB — Read I/O input.

Example

1RA	*Read value of analog port #1.*
1RA4.500	*Controller returns a value of 4.5 V, read for analog port #1.*

RB — Read I/O input

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxRB**

Parameters

Description	**xx** [int]	—	I/O bit number.
Range	**xx**	—	**0** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.

Description This command reads the I/O input port. If **xx** is specified between 1 and 8, the return is either 0 or 1, depending on the state of the selected I/O bit.

If the bit specifier **xx** is missing or set to 0, the controller returns the values for all 8 bits. The return is a decimal number in ASCII format representing the I/O byte. To find the values of each bit, the number must be converted to binary.

NOTE

For the hardware definition of the I/O port, please see Appendix B, Connector Pinouts, GPIO Connector.

Returns **xxRBnn, 0RBnn$_1$,** or **RBnn$_2$**

xx — I/O bit number.

nn — I/O bit value, 0 or 1.

nn$_1$, nn$_2$

— I/O byte value, 0 to 255 in ASCII format.

Errors

A — Unknown message code.

E — Incorrect I/O channel number.

S — Communication time-out.

Rel. Commands **RA** — Read analog input.

Example

0RB | *Read all 8 bits of the I/O input port.*

RB209 | *Controller returns a value of 209, which converted to binary gives us the following I/O input port status:*

bit 8	*bit 7*	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*
1	*1*	*0*	*1*	*0*	*0*	*0*	*1*

RD — Disable display refresh

Usage ■ IMM ■ PGM ■ MIP

Syntax **RD**

Parameters None.

Description This command disables the front panel display. It is used primarily to save the CPU overhead time during time consuming or time-sensitive operations. For instance, better accuracy can be obtained for WP command when used at high velocities and a higher communication throughput can be achieved while downloading the trace data.

While this command is active, the display shows only the following message: "Position display disabled".

To exit this mode and re-enable the display refresh, use the RE command. The display is also re-activated while waiting for a key with WK command, at the end of a program, when the local mode is selected with the ML command or while the manual jog mode is active. When the controller exits the manual jog mode, the display returns to its previous state, enabled or disabled.

NOTE

The command is not allowed in local mode or manual jog mode.

Returns None.

Errors D — Unauthorized execution.

Rel. Commands

MC — Set manual mode.
ML — Set local mode.
MR — Set remote mode.
RE — Enable display refresh.

Example

SP0.002	*Set trace period to 2 ms.*
2TM2000	*Set trace mode for axis #2 and 2000 data points.*
2PR0.1, WS	*Perform a motion of 0.1 units on axis #2 and wait for stop.*
RD	*Disable display refresh for faster communication throughput.*
9TT	*Read trace sample #9.*
…	
…	*Controller returns trace data.*
…	
RE	*Enable front panel display refresh.*

RE — Enable display refresh

Usage ■ IMM ■ PGM ■ MIP

Syntax **RE**

Parameters None.

Description This command enables the front panel display. It is used after the front panel display refresh is disabled using the RD command.

Returns None.

Errors None.

Rel. Commands

MC — Set manual mode.
ML — Set local mode.
MR — Set remote mode.
RD — Disable display refresh.

Example

SP0.002	*Set trace period to 2 ms.*
2TM2000	*Set trace mode for axis #2 and 2000 data points.*
2PR0.1, WS	*Perform a motion of 0.1 units on axis #2 and wait for stop.*
RD	*Disable display refresh for faster communication throughput.*
9TT	*Read trace sample #9.*
…	
…	*Controller returns trace data.*
…	
RE	*Enable front panel display refresh.*

RO — Read I/O output

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxRO**

Parameters

Description	**xx** [int]	—	I/O bit number.
Range	**xx**	—	**0** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.

Description This command reads the I/O output port. If **xx** is specified between 1 and 8, the return is ether 0 or 1, depending on the state of the selected I/O bit.

If the bit specifier **xx** is missing or set to 0, the controller returns the values for all 8 bits. The return is a decimal number in ASCII format representing the I/O output byte. To find the values of each bit, the number must be converted to binary.

NOTE

For the hardware definition of the I/O port, please see Appendix B, Connector Pinouts, GPIO Connector.

Returns **xxROnn, 0ROnn$_1$,** or **ROnn$_2$**

xx — I/O output bit number.

nn — I/O output bit value, 0 or 1.

nn$_1$, nn$_2$

— I/O output byte value, 0 to 255 in ASCII format.

Errors

A — Unknown message code.

E — Incorrect I/O channel number.

S — Communication time-out.

Rel. Commands

CB — Clear I/O outputs bits.

SB — Set I/O output bits.

SO — Set I/O output byte.

TG — Toggle I/O output bits.

Example

RO | *Read the I/O output port.*

RO209 | *Controller returns a value of 209, which converted to binary gives us the following I/O output port status:*

bit 8	*bit 7*	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*
1	*1*	*0*	*1*	*0*	*0*	*0*	*1*

Newport

RP — Repeat command line

Usage	■ IMM ■ PGM □ MIP
Syntax	**RPnn**
Parameters	
Description	**nn** [int] — Number of times to repeat command line.
Range	**nn** — **1** to **2147385345.**
Units	**nn** — None.
Defaults	**nn** Missing: 1. Out of range: 1 or 2147385345 (forced in range) Floating point: Decimal value truncated.

Description

This command is a flow control instruction that repeats the execution of a command line **nn** number of times. It must be placed at the end of a command line that has to be repeated. The line must have at least one more command on it, separated by a command separator.

If the **nn** parameter is missing or set to a value less than 1, the command line is executed one time, similar to a **nn** value of 1.

NOTE

Any command placed on a line after RP is ignored, without issuing an error.

NOTE

Be careful when using flow control commands, specially nested ones. Avoid mixing different type of flow control commands on the same line. As in other programming languages, improper loops and loop mixings could generate undesirable results.

Returns None.

Errors R — Command cannot be at the beginning of a line.

Rel. Commands

WG — While variable is greater.
WH — While input is equal.
WL — While variable is less.
WY — While variable is different.

Example

3PA0	*Move axis #3 to position 0.*
2PR.1, WS, 3PA10, WS, 3PA0, WS, **RP20**	*Make a relative move of 0.1 units on axis #2, wait for all motion to stop, move axis #3 to position 10, wait for all motion to stop, move axis #3 back to 0, wait to stop; repeat the entire cycle 20 times.*

RQ — Generate service request (SRQ)

Usage ■ IMM ■ PGM □ MIP

Syntax **RQnn**

Parameters

Description	**nn** [int]	—	Interrupt number.
Range	**nn**	—	**0** to **31.**
Units	**nn**	—	None.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command generates an interrupt service request to the host computer. The parameter **nn** is used to identify the RQ command which generated the interrupt. Upon receiving the interrupt, the host computer interrupt service routine should perform an IEEE-488 serial poll or send the TS command and read the response. If the interrupt was a result of the RQ command, then bit 7 of the response is 1 and the lower five bits equal the parameter **nn**.

This command can be used to notify the host computer of the progress or flow of command execution in the MM4006.

Returns None.

Errors C — Parameter out of limits.

Rel. Commands **TS** — Tell status.

Example

2PR200, WS, 1PR100, WS, **RQ** | *Generate interrupt when RQ command is encountered.*

SB — Set I/O output bits

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxSBnn**

Parameters

Description	**xx** [int]	—	I/O bit number.
	nn [int]	—	I/O bit mask.
Valeurs	**xx**	—	**0** to **8.**
	nn	—	**0** to **255.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.
	nn	Missing:	255.
		Out of range:	Error C.
		Floating point:	Decimal part truncated.

Description This command sets one to all output bits of the I/O port. If **xx** is specified between 1 and 8, the **nn** mask must be missing and then the selected bit will be set.

If **xx** is missing or set to 0 and **nn** is between 1 and 255, the controller will set all bits corresponding to the mask. For example, if **nn** is 140, the equivalent binary mask is 10001100 which means that I/O output bits number 3, 4 and 8 will be set (remember that I/O bits are numbered from 1 to 8).

If **xx** is missing or set to 0 and **nn** is not specified, the controller sets all 8 bits. This is equivalent to setting **xx** to 0 and **nn** to 255.

NOTE

Remember that having an open collector configuration, a set bit means a conducting transistor. Using a pull-up resistor, a set output bit will measure a logic low, thus making the output port be the reverse logic type.

NOTE

For the hardware definition of the I/O port, please see Appendix B, Connector Pinouts, GPIO Connector.

Returns None.

Errors

A — Unknown message code.
E — Incorrect I/O channel number.

Rel. Commands

CB — Clear I/O outputs bits.
RO — Read I/O output.
SO — Set I/O output byte.
TG — Toggle I/O output bits.

Example **SB224** | *Set I/O output port bits number 6, 7 and 8 high.*

SC — Set control loop type

Usage	■ IMM	■ PGM	□ MIP

Syntax **xxSCnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [int]	—	Loop type.
Range	**xx**	—	**1** to **8.**
	nn	—	**0** or **1.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command defines the type of motion control loop of an axis. If parameter **nn** is set to 0, the selected axis **xx** is set to operate in open loop. If **nn** is set to 1, the axis will operate in closed loop.

NOTE

Because this is a setup instruction, do not use it when motor power is on. If sent during a motion or when motors are on, the controller will refuse the execution and set error code D.

NOTE

Avoid using this comand in normal operation. It was intended to be used only for factory testing or very specialized applications.

Returns None.

Errors	A	—	Unknown message code.
	B	—	Incorrect axis number.
	C	—	Parameter out of limits.
	D	—	Unauthorized execution.
Rel. Commands	**TC**	—	Read control loop type.
Example	MF	|	*Turn power off to the motors.*
	3SC0	|	*Set axis #3 to operate in open loop.*

SD — Speed scaling

Usage	■ IMM ■ PGM □ MIP
Syntax	**SDnn**

Parameters

Description	**nn** [float]	—	Percentage of velocity.
Range	**nn**	—	**0.001** to **100.**
Units	**nn**	—	None (percentage).
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.
		Non-increment:	Aounded to nearest increment.

Description

This command reduces the velocity on all axes by a specified factor. The **nn** parameter represents the percentage of the nominal value all velocities will be reduced to. The command is identical to the Speed Scaling parameter in the General Setup menu. Using the SD command will actually modify the Speed Scaling percentage value.

NOTE

The motions affected are the ones initiated by PA and PR, issued in immediate mode or inside a program.

NOTE

This command is useful to reduce the speed of execution of a complex motion program for the purpose of observing and troubleshooting it.

Returns None.

Errors C — Parameter out of limits.

Rel. Commands **VA** — Set velocity.

Example

SD25	|	*Execute all following motions at 1/4 of the programmed velocity.*
2EX1	|	*Execute program #2 at the reduced velocity.*
SD100	|	*Restore velocities to nominal values.*

SE — Start synchronized motion

Usage ■ IMM ■ PGM □ MIP

Syntax **SE**

Parameters None.

Description This command starts execution of a synchronized motion. When some axes are defined as synchronized by the use of SY command, they do not execute any motion commands until SE is issued.

Use this command to execute coordinated (synchronized) motions on multiple axis, also defined as linear-interpolated motions. These simultaneous multi-axes motions generate a straight line in the defined coordinate system.

NOTE

When the axes synchronization feature is no longer needed, terminate it by using the SY command and returning the axes to the default non-synchronized mode.

Returns None.

Errors D — Unauthorized execution.

Rel. Commands **SY** — Axis synchronization.

Example

2SY1	*Define axis #2 as synchronized.*
4SY1	*Define axis #4 as synchronized.*
2PA12	*Set axis #2 destination.*
4PA7.3	*Set axis #4 destination.*
SE	*Start synchronized motion on the two axes.*
2SY0	*Define axis #2 as non-synchronized.*
4SY0	*Define axis #4 as non-synchronized.*

SF — Set axis mechanical motion device

Usage ■ IMM ■ PGM □ MIP

Syntax **xxSFname/p, xxSFnn** or **xxSF?**

Parameters

Description

xx [int] — Axis number.
nn [int] — Axis number.
name/p — Name of mecanical device to set, with:
p = 0 or **missing**: origin at center (center of the travel)
p = -1: Home position on motor side (–End-of-Travel)
p = 1: Home position on opposite motor side (+End-of-Travel)
? — Read the actual mecanical device name.

NOTE

/p takes effect only with motion devices with switchable home position (Mechanical Zero), such as families: MTM, UTM, EM, DEFAULT.

Range **xx, nn** — **1** to **8.**

Units **xx, nn** — None.

Defaults **xx** Missing: Error B.
Out of range: Error B.
Floating point: Error A.

Description This command set a new unit to an axis. All controller concerned parameters will be recalculed to adapt for the new mecanical motion device.
If **xxSFnn** (**nn** takes place of **name**), this command copies all configuration properties (device name, device units, parameters,...) of the axis **#nn** to the axis **#xx**.

NOTE

The SF command must be used carefully. All axis parameters are replaced by the new specified stage parameters.

NOTE

This command must be used when motor power is off, to avoid a displacement at the time of the stage modification.

NOTE

After use of the SF command, it is necessary to execute a home search routine on the axis with new parameters.

Returns If the sign **"?"** takes place of name, this command reportes the name of the actual mecanical motion device installed in the controller.

Errors
A — Unknown message code.
B — Incorrect axis number.
g — Mechanical familly name incorrect.

Rel. Commands **TA** — Read motion device.

Examples

2SF3	*Copy parameters from axis #3 to axis #2.*
2SFUTM100CC1HL or **2SFUTM100CC1HL/0**	*Set UTM100CC1HL mechanical device parameters of axis #2 with centered home position.*
2SF?	*Read mechanical device name of axis #2.*
2SFUTM100CC1HL	*Controller returns the name.*
2SFUTM100CC1HL/-1	*Set UTM100CC1HL mechanical device parameters of axis #2 with home position on opposite motor side.*

SL — Set left travel limit

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxSLnn**

Parameters

Description

xx [int] — Axis number.
nn [float] — Left (negative) software travel limit.

Range

xx — **1 to 8.**
nn — **-2147483647 x encoder resolution** to **min (home value set by ZH or current position or destination (if in motion))**.

Units

xx — None.
nn — Defined motion units.

Defaults

xx Missing: Error B.
Out of range: Error B.
Floating point: Error A.
nn Missing: Error C.
Out of range: Error C.

Description

This command defines the value for the negative (left) software travel limit. It should be used to restrict travel in the negative direction to protect the motion device or its load. For instance, if traveling full range, a stage could push its load into an obstacle. To prevent this, the user can reduce the allowed travel by changing the software travel limit.

Since a motion device must be allowed to find its home position, the home switch and/or sensor must be inside the travel limits. This means that both positive and negative travel limits cannot be set on the same side of the home position. A more obvious restriction is that the negative limit cannot be greater than the positive limit. If any of these restrictions is not met, the controller will return error C.

NOTE

If the command is issued for an axis in motion, the new limit should not be set inside the current travel. If the motion in progress could reach the new desired software limit, the command is not accepted and the controller returns error D.

NOTE

Be careful when using this command. The controller does not know the real hardware limits of the motion device or application. Always set the software limits inside the hardware limits (limit switches). In normal operation, a motion device should never hit a limit switch.

If you want to change the software limits, note that the values selected in remote mode can't exceed the values selected in local mode (already available as a standard parameter of the stage).

If you want to increase these limits:

❶ Do care about the hardware limits.

❷ Use the local mode, from the front panel.

Returns

If the sign **"?"** takes place of the **nn** value, the controller returns the value of the negative (left) software travel limit for **#xx** axis.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands

OR — Search for home.
SR — Set right travel limit.

Example

1SL-41.4	*Set negative software travel limit of axis #1 to -41.4 units.*
1SL?	*Reading of the negative software travel limit of axis #1.*
1SL-41.4	*The controller returns the value of the negative software travel limit.*

NOTE

Always, the stage position must be inside the interval set by the software limits

---

SM — Save program

Usage	■ IMM ☐ PGM ☐ MIP
Syntax	**SM**
Parameters	None.
Description	This command saves all programs from RAM in non-volatile memory. It should be used after creating or editing a program to assure that the program will not be lost when the controller is powered off.
Returns	None.
Errors	I — Unauthorized command in programming mode.
Rel. Commands	**CP** — Compile program. **EP** — Edition of program. **MP** — Download EEPROM to RAM. **QP** — Quit program mode.

Example

3XX	*Clear program #3 from memory.*
3EP	*Activate program mode and enter following commands as program #3.*
…	
…	
…	
QP	*End entering program and quit programming modee.*
3CP	*Compile program #3.*
3CP@	*Controller confirms compilation of program #3 with no errors.*
SM	*Save all program from RAM in non-volatile memory.*

SN — Set axis displacement units

Usage	■ IMM ■ PGM ☐ MIP
Syntax	**xxSNname** or **xxSN?**
Parameters	
Description	**xx** [int] — Axis number. **name** — Name of displacement unit to set. **?** — Read the actual displacement unit.
Range	**xx** — **1** to **8.**
Units	**xx** — None.
Defaults	**xx** Missing: Error B. Out of range: Error B. Floating point: Error A.

Description

This command set a new unit to an axis. All controller concerning parameters will be recalculed to adapt for the new unit.

The standard names of units are following:

- Translation groupe: mm, µm, In., mIn, µIn and Inc.
- Rotation groupe: Dg., Gr., Rad, mRd, µRd and Inc.

Returns

If the sign **"?"** takes place of name, this command reportes the name of the actual unit used in the controller.

Errors

A — Unknown message code.
B — Incorrect axis number.
c — Unit not translational or incorrect.
d — Unit not rotationnal or incorrect.

Rel. Commands

TN — Read displacement units.

Example

2SNum	*Set unit of axis #2 to µm.*
2SN?	*Read unit of axis #2.*
2SNum	*Controller returns axis #2 unit.*
2TN	*Read unit of axis #2.*
2TNum	*Controller returns axis #2 unit.*

SO — Set I/O output byte

Usage	■ IMM ■ PGM ■ MIP
Syntax	**SOnn**

Parameters

Description	**nn** [int]	—	I/O bit mask.
Range	**nn**	—	**0** to **255.**
Units	**nn**	—	None.
Defaults	**nn**	Missing:	0.
		Out of range:	Error C.
		Floating point:	Decimal part truncated.

Description This command sets all output bits of the I/O port. The **nn** parameter is the mask to be used in setting the I/O output port. For example, if **nn** is 140, the equivalent binary mask is 10001100 which means that I/O output bits number 3, 4 and 8 will be set and output bits number 1, 2, 5, 6 and 7 will be cleared (remember that I/O bits are numbered from 1 to 8).

NOTE

Remember that having an open collector configuration, a set bit means a conducting transistor. Using a pull-up resistor, a set output bit will measure a logic low, thus making the output port be the reverse logic type.

NOTE

For the hardware definition of the I/O port, please see Appendix, Connector Pinouts, GPIO Connector.

Returns None.

Errors C — Parameter out of limits.

Rel. Commands

CB — Clear I/O outputs bits.
RO — Read I/O output.
SB — Set I/O output bits.
TG — Toggle I/O output bits.

Example **SO224** | *Set I/O output port bits number 6, 7 and 8 and clear bits 1, 2, 3, 4 and 5.*

SP — Set trace sample rate

Usage ■ IMM ■ PGM ■ MIP

Syntax **SPnn**

Parameters

Description	**nn** [float]	—	Trace sample period.
Range	**nn**	—	**0.0003** to **9.**
Units	**nn**	—	Seconds.
Defaults	**nn**	Missing:	0.0003.
		Out of range:	Nearest range limit.
		Non-increment:	Rounded to nearest increment.

Description This command sets the sample period for the trace function. Refer to the trace command TM for the description on how to use the trace mode.

NOTE

The sampling is done in increments of the servo loop cycle. Since the servo cycle is not exactly 0.0003 sec, use the XS command to read the actual trace sample interval used.

Returns None.

Errors C — Parameter out of limits.

Rel. Commands **TM** — Set trace mode.
XS — Read trace sample rate.

Example

SP0.002	*Set trace period to 2 ms.*
XS	*Read actual trace period.*
XS0.002001374478	*Controller returns actual trace period.*
2TM500	*Set trace mode for axis #2 and 500 data points.*
2PR0.1, WS	*Perform a motion of 0.1 units on axis #2 and wait for stop.*
TT	*Read trace data.*

SQ — Set global sample rate

Usage	■ IMM ■ PGM ■ MIP
Syntax	**SQnn**
Parameters	
Description	**nn** [float] — Trace sample period.
Range	**nn** — **0.0003** to **9**.
Units	**nn** — Seconds.
Defaults	**nn** Missing: 0.0003. Out of range: Nearest range limit. Non-increment: Rounded to nearest increment.

Description This command sets the sample period for the global trace function. The global trace mode is similar to the trace mode but it samples all four axes in the same time. Refer to the global trace command GQ for the description on how to use the global trace mode.

NOTE

The sampling is done in increments of the servo loop cycle. Since the servo cycle is not exactly 0.0003 sec, use the XQ command to read the actual global trace sample interval used.

Returns None.

Errors C — Parameter out of limits.

Rel. Commands
GQ — Set global trace mode.
XQ — Read global sample rate.

Example

SQ0.002	*Set global trace period to 2 ms.*
XQ	*Read actual global trace period.*
XQ0.002001374478	*Controller returns actual global trace period.*
2GQ500	*Set global trace mode for axis #2 and 500 data points.*
2PR0.1, 3PR0.1, WS	*Perform a motion of 0.1 units on axis #2 and #3 and wait for stop.*
TQ	*Read global trace data.*

SR — Set right travel limit

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxSRnn**

Parameters

Description

xx	[int]	—	Axis number.
nn	[float]	—	Right (postive) software travel limit.

Range

xx	—	**1** to **8.**
nn	—	**Max (home value set by ZH or current position or destination (if in motion))** to **2147483647 x encoder resolution**.

Units

xx	—	None.
nn	—	Defined motion units.

Defaults

xx	Missing:	Error B.
	Out of range:	Error B.
	Floating point:	Error A.
nn	Missing:	Error C.
	Out of range:	Error C.

Description This command defines the value for the positive (right) software travel limit. It should be used to restrict travel in the positive direction to protect the motion device or its load. For instance, if traveling full range, a stage could push its load into an obstacle. To prevent this, the user can reduce the allowed travel by changing the software travel limit.

Since a motion device must be allowed to find its home position, the home switch and/or sensor must be inside the travel limits. This means that both positive and negative travel limits cannot be set on the same side of the home position. A more obvious restriction is that the negative limit cannot be greater than the positive limit. If any of these restrictions is not met, the controller will return error C.

NOTE

If the command is issued for an axis in motion, the new limit should not be set inside the current travel. If the motion in progress could reach the new desired software limit, the command is not accepted and the controller returns error D.

NOTE

Be careful when using this command. The controller does not know the real hardware limits of the motion device or application. Always set the software limits inside the hardware limits (limit switches). In normal operation, a motion device should never hit a limit switch.

If you want to change the software limits, note that the values selected in remote mode can't exceed the values selected in local mode (already available as a standard parameter of the stage).

If you want to increase these limits:

❶ Do care about the hardware limits.

❷ Use the local mode, from the front panel.

Returns If the sign **"?"** takes place of the **nn** value, the controller returns the value of the positive (right) software travel limit for **#xx** axis.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands

OR	—	Search for home.
SL	—	Set left travel limit.

Example

1SR41.4	*Set positive software travel limit of axis #1 to 81.4 units.*
1SR?	*Reading of the positive software travel limit of axis #1.*
1SR41.4	*The controller returns the value of the positive software travel limit.*

NOTE

Always, the stage position must be inside the interval set by the software limits

---

SS — Set master-slave mode

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxSSnp** or **xxSS?**

Parameters

Description

xx	[int]	—	Axis number to define.
n	[int]	—	Axis number of the master axis.
p	[int]	—	Following mode: theoretical/real position.
?		—	Read number of the master axis that this axis slaved to.

Range

xx	—	**1** to **8.**
n	—	**1** to **8.**
p	—	**0** or **1.**

Units

xx	—	None.
n	—	None.
p	—	None.

Defaults

xx	Missing:	Error B.
	Out of range:	Error B.
	Floating point:	Error A.
n	Missing:	0 (defined as master).
	Out of range:	Error C.
p	Missing:	0.
	Out of range:	Error C.

Description This command sets the master-slave mode. This defines **nn** numbered of the master axis that **xx** numbered axis belongs to. If n is zero or default, the **xx** axis is defined as master.
If **p** is zero or default, the **xx** axis must follow the theoretical position of its master axis. If **p** = 1, it follows the real position of its master.

NOTE
If the CD command is used in conjunction with the SS command and GR command, the slave axis cycle value must be equal to the master axis cycle value multiplied by the master-slave reduction ratio.

NOTE
The slave axis motor power may be turned on (MO command) or turned off (MF command) only if p = 1.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the number of the master axis that **xx** numbered axis slaved to (if 0: the axis is master).

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands

GR	—	Set master-slave reduction ratio.
FF	—	Set maximum master-slave following error.

Example

2SS1	*Set axis #2 to be slave of axis #1.*
…	
…	
…	
2SS?	*Read master axis of axis #2.*
2SS1	*Controller tells the master of this axis.*

ST — Stop motion

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxST**
Parameters	
Description	**xx** [int] — Axis number.
Valeurs	**xx** — **0** to **4.**
Units	**xx** — None.
Defaults	**xx** Missing: 0. Hors de limite: Error B. Floating point: Error A.

Description

This command stops a motion in progress on one or all axes. If parameter **xx** is set to 0 or missing, motion on all axes will be stopped. If **xx** is a valid axis number, only motion on that axis will be stopped.

A motion interrupted with this command will stop using the programmed acceleration/deceleration for each axis. This is the preferred motion termination method.

NOTE

This command does not terminate a program. It only stops the motion in progress and permits execution of the rest of the command line or program.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.

Rel. Commands

AB — Abort motion.
MF — Motor OFF.

Example

2PA40	*Move axis #2 to absolute position 40.*
2ST	*Stop motion on axis #2.*

SU — Set encoder resolution

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxSUnn** or **xxSU?**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	New value.
Range	**xx**		—	**1** to **8.**
	nn		—	**Positive value**.
Units	**xx**		—	None.
	nn		—	Axis preset unit.
Defaults	**xx**		Missing:	Error B.
			Out of range:	Error B.
	nn		Missing:	Error C.
			Out of range:	Error C.

Description This command sets a new value of axis encoder resolution in the following cases:

- The controller must be in state MOTOR OFF.
- The encoder type must be REAL (with the fictive encoder for stepper motors (encoder loop mode) encoder resolution value is set all together with motor resolution value (SV command)).
- If the motor is of type STEPPER, the entered value must not be smaller than the actual motor resolution value.
- In all other cases, the command is not executed and the controller returns error code D.

Returns If the **"?"** sign takes place of **nn** parameter, the controller returns the actual value of encoder resolution.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands

SV	—	Set stepper motor resolution.
TU	—	Read encoder resolution.

Example

MF	*Set motors OFF.*
1SU0.0012	*Set a new value for axis #1.*
1SU?	*Request the encoder resolution of axis #1.*
1SU0.0012	*Controller returns value of axis #1.*

SV — Set stepper motor resolution

Usage ■ IMM ■ PGM □ MIP

Syntax **xxSVnn** or **xxSV?**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [float]	—	New value.
Range	**xx**	—	**1** to **8.**
	nn	—	**Positive value**.
Units	**xx**	—	None.
	nn	—	Axis preset unit.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command sets a new value of motor resolution in the following cases:

- The controller must be in state MOTOR OFF.
- The motor type must be STEPPER.
- If the encoder does not exist (encoder loop mode), this command sets encoder resolution value and motor resolution value with the same value.
- If the encoder really exists (real encoder mode), the entered value must not be bigger than the actual encoder resolution value.
- In all other cases, the command is not executed and the controller returns error code D.

Returns If the **"?"** sign takes place of **nn** parameter, the controller returns the actual value of motor resolution.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands

SU	—	Set encoder resolution.
TU	—	Read encoder resolution.

Example

MF	*Set motors OFF.*
1SV?	*Request the motor resolution of axis #1.*
1SV0.001	*Controller returns value of axis #1.*
1SV0.0005	*Set a new value for axis #1.*
1SV?	*Request the motor resolution of axis #1.*
1SV0.0005	*Controller returns new value of axis #1.*

SY — Axis synchronization

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxSYnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[int]	—	Synchronization code.
Valeurs	**xx**		—	**1** to **8.**
	nn		—	**0** or **1.**
Units	**xx**		—	None.
	nn		—	None.
Defaults	**xx**	Missing:		Error B.
		Out of range:		Error B.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.
		Floating point:		Error C.

Description

This command defines if an axis should perform all subsequent motions as independent or synchronized moves. When two or more axes perform a synchronized motion, the load travels on a straight line in the defined coordinate system. This type of motion is also referred to as linear interpolation.

If the **nn** parameter is set to 0, the specified **xx** axis is defined as independent, non-synchronized. If **nn** is set to 1, the axis is defined as synchronized and all motion commands (using PA and PR) will not be executed until the SE command is received.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands **SE** — Start synchronized motion.

Example

2SY1	*Define axis #2 as synchronized.*
4SY1	*Define axis #4 as synchronized.*
2PA12	*Set axis #2 destination.*
4PA7.3	*Set axis #4 destination.*
SE	*Start synchronized motion on the two axes.*
2SY0	*Define axis #2 as non-synchronized.*
4SY0	*Define axis #4 as non-synchronized.*

TA — Read motion device

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxSF?

Syntax **xxTA**

Parameters

Description	**xx**	[int]	—	Axis number.
Valeurs	**xx**		—	**1** to **8.**
Units	**xx**		—	None.
Defaults	**xx**	Missing:		Error B.
		Hors limite:		Error B.
		Floating point:		Error A.

Description This command reads the type of motion device installed on the specified axis. The name of the device is the one found in the Newport catalog.

NOTE

The type of motion device installed on each axis can be changed only through the front panel SETUP menu.

Returns Name of installed motion device.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands None.

Example

1TA	*ad the name of motion device installed on axis #1.*
1TAUTM50PP0.1	*Controller returns the name UTM50PP0.1.*

TB — Read error message

Usage	■ IMM ■ PGM ■ MIP
Syntax	**TBaa**
Parameters	
Description	**aa** [char] — Error code character, in ASCII format.
Range	**aa** — **@** to **U**.
Units	**aa** — None.
Defaults	**aa** Missing: Reads current error. Out of range: Controller returns message Unknown error code.

Description

This command reads the error code and the associated message. If the **aa** parameter is missing, the controller reports the existing error. If **aa** is a valid error code, the controller returns the error message associated with that code. The error code is one ASCII character and the message is the description of the error associated with it.

NOTE

When an error is read using TB or TE, the error buffer is cleared. This means that an error can be read only once, with either command. If TB is used only for translating an error code by supplying the aa parameter, the existing error in the buffer is not cleared.

NOTE

The controller returns only the last error that has occurred. If more than one error has occurred since the last reading, only the last one is reported and the rest are lost.

Returns	**TBaabb** **aa** — Error code character. **bb** — Error description.
Errors	S — Communication time-out.
Rel. Commands	**TE** — Read error code.

Example

TB	*Read error message.*
TB@ No error	*Controller returns no error.*
5PA12.3	*Move axis #5 to position 12.3.*
TB	*Read error message.*
TBB Axis Number not Correct	*Controller returns error code and description.*

TC — Read control loop type

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command: xxSC?

Syntax xxTC

Parameters

Description **xx** [int] — Axis number.

Range **xx** — **1** to **8.**

Units **xx** — None.

Defaults **xx** Missing: Error B.
Out of range: Error B.
Floating point: Error A.

Description This command reads the type of control loop used on a specified axis. The two possibilities are open loop (0) or closed loop (1).

NOTE

The controller can operate both stepper and DC motors in closed or open loop. While the stepper motors operate fine in both modes, for normal operation, DC motors should not be used in open loop.

Returns **xxTC nn**

xx — Axis number.
nn — Loop type:
0 = Open loop.
1 = Closed loop.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **SC** — Set control loop type.

Example
1TC | *Read the type of control loop used on axis #1.*
1TC 1 | *Controller reports closed loop for axis #1.*

TD — Read error line of program

Usage	■ IMM ■ PGM □ MIP
Syntax	**TD**
Parameters	None.
Description	This command reads the line of a program where the error is if an error occurred in execution. The error line buffer will be reset after this operation.
Returns	**TDaaa** **aaa** — Program error line.
Errors	S — Communication time-out.
Rel. Commands	**TB** — Read error message. **TE** — Read error code.

Example

1EP	*Program #1.*
…	
…	
…	
5OR	*An error generating line.*
…	
…	
…	
QP	*End of program.*
1EX	*Execute program #1, an error will occur.*
TD	*Read program error line.*
TD5OR	*Controller returns error line.*
TB	*Read error message buffer.*
TBB Axis number missing or not correct	*Error message.*

TE — Read error code

Usage ■ IMM ■ PGM ■ MIP

Syntax **TE**

Parameters None.

Description This command reads the error code of the controller. The error code is one ASCII character, stored in the error register.

NOTE

When an error is read using TB or TE, the error buffer is reset. This means that an error can be read only once, with either command.

NOTE

The error reported is the last one that has occurred. If more than one error has occurred since the last reading, only the last one is reported and the rest are lost.

NOTE

For a complete listing and description of all error codes see Appendix A, Error Messages.

Returns **TEaa**

aa — Error code character.

Errors S — Communication time-out.

Rel. Commands **TB** — Read error message.

Example

TE	|	*Read error message.*
TE@	|	*Controller returns no error.*
5PA12.3	|	*Move axis #5 to position 12.3.*
TE	|	*Read error message.*
TEB	|	*Controller returns error code B meaning incorrect axis number.*

TF — Read filter parameters

Usage ■ IMM ■ PGM □ MIP

Syntax **xxTF**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the PID parameters and the maximum acceptable following error of an axis. It is equivalent to sending XP, XI, XD and XF, with the exception that the return comes on a single line.

NOTE

The command reads the value actually used in the servo loop. If the PID parameters are modified but the digital filter has not been updated by sending an UF, the command will still read the old values.

Returns **xxTF, xxXPnn_1, xxXInn_2, xxXDnn_3, xxXFnn_4**

xx — Axis number.
nn_1 — Proportional gain factor.
nn_2 — Integral gain factor.
nn_3 — Derivative gain factor.
nn_4 — Maximum acceptable following error.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands

XF — Read maximum following error.
XD — Read derivative gain factor.
XI — Read integral gain factor.
XP — Read proportional gain factor.

Example

2TF	*Read filter parameters of axis #2.*
2TF, 2XP0.07, 2XI0.001, 2XD.05, 2XF0.12	*controller returns the following digital filter parameters for axis #2:*
	Kp = 0.07 Ki = 0.001 Kd = 0.05 Fe = 0.12 units.

TG — Toggle I/O output bits

Usage	■ IMM	■ PGM	■ MIP
Syntax	**xxTGnn**		

Parameters

Description	**xx** [int]	—	I/O bit number.
	nn [int]	—	I/O bit mask.
Range	**xx**	—	**0** to **8.**
	nn	—	**0** to **255.**
Units	**xx**	—	None.
	xx	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.
	nn	Missing:	255.
		Out of range:	Error C.
		Floating point:	Decimal part truncated.

Description

This command toggles one to all output bits of the I/O port. If **xx** is specified between 1 and 8, the **nn** mask must be missing and then the selected bit will be inverted.

If **xx** is missing or set to 0 and **nn** is between 1 and 255, the controller will toggle all bits corresponding to the mask. For example, if **nn** is 140, the equivalent binary mask is 10001100 which means that I/O output bits number 3, 4 and 8 will be inverted (remember that I/O bits are numbered from 1 to 8).

If **xx** is missing or set to 0 and **nn** is not specified, the controller toggles all 8 bits. This is equivalent to setting **xx** to 0 and **nn** to 255.

Returns None.

Errors

A	—	Unknown message code.
E	—	Incorrect I/O channel number.

Rel. Commands

CB	—	Clear I/O outputs bits.
RO	—	Read I/O output.
SB	—	Set I/O output bits.
SO	—	Set I/O output byte.

Example

0RO	*Read all 8 bits of the I/O output port.*
0RO209	*Controller returns 209, which converted to binary means:*

bit 8	*bit 7*	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*
1	*1*	*0*	*1*	*0*	*0*	*0*	*1*

TG224	*Toggle bits number 6, 7 and 8 of the I/O output port.*
0RO	*Read all 8 bits of the I/O output port.*
0RO49	*Controller returns 49, which converted to binary means:*

bit 8	*bit 7*	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*
0	*0*	*1*	*1*	*0*	*0*	*0*	*1*

TH — Read theoretical position

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxTH**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**0** to **4.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the theoretical position and the instantaneous real position. If the axis specifier **xx** is missing or set to 0, the controller returns the desired position for all axes.

The command could be sent at any time but its primary use is while a motion is in progress.

The command is useful in determining the following error of a motion device by comparing the theoretical position to its real position.

Returns **xxTHnn** or **xx_1THnn$_1$, xx_2THnn$_2$, xx_3THnn$_3$, xx_4THnn$_4$**

xx, xx_1, xx_2, xx_3, xx_4
— Axis number.

nn, nn_1, nn_2, nn_3, nn_4
— Theoretical position, in pre-defined units.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **TP** — Read actual position.

Example

3TP,**3TH**	*Read real and theoretical position on axis #3.*
3TP 5.322	*Controller returns real position 5.322 for axis #3.*
3TH 5.323	*Controller returns theoretical position 5.323 for axis #3.*

TL — Read left travel limit

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxSL?

Syntax **xxTL**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the left software travel limit, the motion soft limit in the negative direction. This is the value set by the SL command or in the front panel setup menu.

NOTE

The software travel limit values are automatically changed when a home position is forced at a new location, or if the home preset value is modified.

This is done to maintain the travel limits fixed relative to the physical location of the limit switches and mechanical stops. By doing so, the motion device is protected from damage due to a hardware stop.

Returns **xxTLnn**

xx — Axis number.
nn — Negative software travel limit.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **SL** — Set left travel limit.

Example

3TL	|	*Read left software travel limit on axis #3.*
3TL14.5	|	*Controller returns left software travel limit 14.5 units.*

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxTMnn** or **xxTM?**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[int]	—	Number of samples.
Range	**xx**		—	**1** to **8.**
	nn		—	**0** to **NMax.**
	?		—	Reading of the NMax value.
Units	**xx**		—	None.
	nn		—	None.
Defaults	**xx**	Missing:		Error B.
		Out of range:		Error B.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.
		Floating point:		Decimal part truncated.

Description

This command sets the trace mode for an axis. If the trace mode is activated by setting **nn** between 1 and 4000, the controller will start recording in memory the theoretical and the actual position of the specified axis, starting with the execution of every PA or PR motion command. The number of samples stored is the one specified by **nn** and the sample interval is the one set by the SP command. To read the recorded trace data use the TT command. To disable the trace mode issue the TM command with **nn** set to 0.

NOTE

Once the trace mode is enabled, the controller will record data every time a PA or PR command is sent for the specified axis. When TT is issued, only the last set of data is returned.

To avoid unnecessary CPU overhead, after the desired measurement is completed, disable the trace mode by issuing the command with a 0 for the nn parameter.

Returns

If the sign **"?"** takes place of **nn**, this command turns the possible max. number of points in trace mode TM.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands

SP — Set trace sample rate.
TT — Read trace data.
XN — Read number of acquisitions.

Example

SP0, 002	*Set trace sample period to 2 ms.*
2TM500	*Set trace mode for axis #2 and 500 data points.*
2PR0.1, WS	*Perform a motion of 0.1 units on axis #2 and wait for stop.*
TT	*Read trace data.*

TN — Read displacement units

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxSN?

Syntax **xxTN**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the type of displacement units used on a specified axis. The units are defined in the SETUP menu of the front panel.

Returns **xxTNaa**

xx — Axis number.
aa — Displacement units; two or three ASCII characters.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands None.

Example

1TN	*Read the type of displacement units used on axis #1.*
1TNmm	*Controller reports mm for axis #1.*

TP — Read actual position

Usage ■ IMM ■ PGM ■ MIP

Syntax xxTP

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the actual position, the instantaneous real position of one or all motion devices. If the axis specifier **xx** is missing or set to 0, the controller returns the actual position of all axes. If **xx** is a number between 1 and 4, the controller returns the actual position of that axis.

Returns **xxTPnn or xx_1TPnn_1, xx_2TPnn_2, xx_3TPnn_3, xx_4TPnn_4**

xx, xx_1, xx_2, xx_3, xx_4
— Axis number.

nn, nn_1, nn_2, nn_3, nn_4
— Actual position, in pre-defined.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **TH** — Read theoretical position.

Example

3TP, 3TH	*Read real and theoretical position on axis #3.*
3TP 5.322	*Controller returns real position 5.322 for axis #3.*
3TH 5.323	*Controller returns theoretical position 5.323 for axis #3.*

TQ — Read global trace data

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxTQnn**
Parameters	
Description	**xx** [int] — Number of samples to read. **nn** [int] — 0 (or missing) or 1.
Range	**xx** — **0** to **number of samples** set by GQ command. **nn** — 0 or 1.
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: 0. Out of range: Error C. Floating point: Error A. **nn** Missing: 0. Out of range: Error C.

Description

This command reads the global trace data stored in global trace mode. The global trace mode is enabled by GQ command and defined by SQ and GQ commands.

If **xx** is a number different than 0 and in range, the controller returns the values for that sample number. If **xx** is 0, the controller returns all samples stored in the global trace buffer.

If **nn** = 0 or missing, the controller returns the values of theoretical and real positions stored in the global trace buffer. If **nn** = 1, the controller returns the values of theoretical and real positions, and in addition, the values of analog inputs stored in the global trace buffer at the moment of theoretical and real positions acquisition.

Returns

xxTQ — **xxTQ, 1THnn$_1$, 1TPnn$_2$, 2THnn$_3$, 2TPnn$_4$, 3THnn$_5$, 3TPnn$_6$, 4THnn$_7$, 4TPnn$_8$**

xxTQ1 — **xxTQ, 1THnn$_1$, 1TPnn$_2$, 2THnn$_3$, 2TPnn$_4$, 3THnn$_5$, 3TPnn$_6$, 4THnn$_7$, 4TPnn$_8$, 1RAnn$_9$, 2RAnn$_{10}$, 3RAnn$_{11}$, 4RAnn$_{12}$**

xx — Sample number.

nn$_1$, nn$_3$, nn$_5$, nn$_7$
— Theoretical position of axes 1, 2, 3 and 4 respectively.

nn$_2$, nn$_4$, nn$_6$, nn$_8$
— Actual position of axes 1, 2, 3 and 4 respectively.

nn$_9$, nn$_{10}$, nn$_{11}$, nn$_{12}$
— Analog values of inputs 1, 2, 3 and 4 respectively.

NOTE

If xx is set to 0 in the TQ command, all samples are returned (starting with number 1), each one on a separate line.

Errors

A — Unknown message code.
C — Parameter out of limits.
D — Unauthorized execution.
S — Communication time-out.

Rel. Commands

GQ — Set global trace mode.
NQ — Read global aquisition nr.
SQ — Set global sample rate.

Example

SQ0, 002	*Set global trace sample period to 2 ms.*
GQ500	*Set global trace mode for 500 data points.*
2PR0.1, WS	*Perform a motion of 0.1 units on axis #2 and wait for stop.*
9TQ	*Read global trace sample #9.*

9TQ, 1TH1.002, 1TP1.001, 2TH1.034, 2TP1.033, 3TH5.002, 3TP5.001, 4TH1.402, 4TP1.401

	Controller returns global trace data for sample #9.
9TQ1	*Read global trace sample #9 with analog inputs reported.*

9TQ, 1TH1.002, 1TP1.001, 2TH1.034, 2TP1.033, 3TH5.002, 3TP5.001, 4TH1.402, 4TP1.401, 1RA0.1, 2RA1, 3RA0, 4RA0

	Controller returns global trace data for sample #9.

TR — Read right travel limit

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command: xxSR?

Syntax xxTR

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the right software travel limit, the motion soft limit in the positive direction. This is the value set by the SR command or in the front panel setup menu.

NOTE

The software travel limit values are automatically changed when a home position is forced at a new location, or if the home preset value is modified.

This is done to maintain the travel limits fixed relative to the physical location of the limit switches and mechanical stops. By doing so, the motion device is protected from damage due to a hardware stop.

Returns **xxTRnn**

xx — Axis number.
nn — Positive software travel limit.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **SR** — Set right travel limit.

Example

3TR	*Read left travel limit on axis #3.*
3TR14.5	*Controller returns right software travel limit 14.5 units.*

Usage ■ IMM ■ PGM ■ MIP

Syntax **TS**

Parameters None.

Description This command reads the controller status byte. Each bit of the status byte represents a particular controller parameter, as described in the following table:

Bit #	Function	Meaning for	
		Bit LOW	Bit HIGH
0	Axis #1 motor state	Stationary	In motion
1	Axis #2 motor state	Stationary	In motion
2	Axis #3 motor state	Stationary	In motion
3	Axis #4 motor state	Stationary	In motion
4	Motor power	ON	OFF
5	Not used	Default	—
6	Not used	—	Default
7	IEEE SRQ Interruption Status (Sent by RQ command)	NO	YES

NOTE
If bit #7 is high after sending TS command, it toggles low automatically.

The byte returned is in the form of an ASCII character. Converting the ASCII code to binary will give the status bits values.

NOTE
For a complete ASCII to binary conversion table see Appendix F, ASCII Table.

Returns **TSaa**

aa — ASCII character representing the status byte.

Errors A — Unknown message code.
S — Communication time-out.

Rel. Commands **MS** — Read motor status.
TX — Read controller activity.

Example

TS | *Read controller status byte.*

TSF | *Controller returns character F, or ASCII character 70; converting 70 to binary we get 01000110 which has the following meaning: axis #1 not moving, axis #2 in motion, axis #3 in motion, axis #4 not moving and motor power is on.*

TT — Read trace data

Usage ■ IMM ■ PGM ☐ MIP

Syntax xxTT

Parameters

Description	**xx** [int]	—	Sample number to read.
Range	**xx**	—	**0** to **number of samples** set by TM command.
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error C.
		Floating point:	Error A.

Description This command reads the trace data stored in trace mode. The trace mode is enabled by TM command and defined by SP and TM commands.

If **xx** is a number different than 0 and in range, the controller returns the values for that sample number. If **xx** is 0, the controller returns all samples stored in memory.

Returns **xxTT, THnn$_1$, TPnn$_2$**

xx — Sample number.
nn$_1$ — Theoretical position.
nn$_2$ — Actual position.

NOTE

If xx was set to 0 in the TT command, all samples are returned (starting with number 1), each one on a separate line.

Errors

A — Unknown message code.
C — Parameter out of limits.
D — Unauthorized execution.
S — Communication time-out.

Rel. Commands

SP — Set trace sample rate.
TM — Set trace mode.
XN — Read number of acquisitions.

Example

SP0.002	*Set trace period to 2 ms.*
2TM500	*Set trace mode for axis #2 and 500 data points.*
2PR0.1, WS	*Perform a motion of 0.1 units on axis #2 and wait for stop.*
9TT	*Read trace sample #9.*
9TT, TH1.002, TP1.001	*Controller returns for trace sample #9 theoretical position 1.002 and real position 1.001 units.*

TU — Read encoder resolution

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command: xxSU?

Syntax **xxTU**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the encoder resolution defined for an axis. This is an important parameter which determines the overall accuracy of the motion device.

The encoder resolution is defined in the SETUP menu on the front panel. The units are the pre-defined displacement units in the SETUP menu.

Returns **xxTUnn**

xx — Axis number.
nn — Encoder resolution, in predefined displacement units.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands

SU — Set encoder resolution.
SV — Set stepper motor resolution.

Example

3TU	*Read encoder resolution for axis #3.*
3TU0.0001	*Controller returns an encoder resolution of 0.0001 units.*

TX — Read controller activity

Usage ■ IMM ■ PGM ■ MIP

Syntax **TX**

Parameters None.

Description This command reads the controller activity register. Each bit of the status byte represents a particular parameter, as described in the following table:

Bit #	Function	Meaning for	
		Bit LOW	Bit HIGH
0	Program is running	NO	YES
1	Command line is executing	NO	YES
2	Manual jog mode active	NO	YES
3	Remote mode active	NO	YES
4	Trajectory is executing	NO	YES
5	Not used	Default	—
6	Not used	—	Default
7	Not used	Default	—

The byte returned is in the form of an ASCII character. Converting the ASCII code to binary gives the status bit values.

NOTE

For a complete ASCII to binary conversion table see Appendix F, ASCII Table.

Returns **TXaa.**

aa — ASCII character representing the status byte.

Errors A — Unknown message code.

S — Communication time-out.

Rel. Commands **MS** — Read motor status.

TS — Read controller status.

Example

TX	*Read controller activity register.*
TXJ	*Controller returns character J, or ASCII character 74; converting 74 to binary we get 01001010 which has the following meaning: controller is in remote mode and is executing a command line.*

TX1 — Read controller extended status

Usage ■ IMM ■ PGM ■ MIP

Syntax **TX1**

Parameters None.

Description This command reports controller's dynamic status. As this controller can perform concurrently a lot of tasks, it is usefull to have one command that gives all the information on what the controller is doing. This reduce the traffic on the communication (otherwise you can use several commands TS, MS to get the same information) and simplifies the development of the user's software.

Returns **TX1**$w_1w_2a_1a_2b_1b_2c_1c_2d_1d_2$

w_1w_2 — Controller's task status (2 characters).
a_1a_2 — Axis #1 (2 characters).
b_1b_2 — Axis #2 (2 characters).
c_1c_2 — Axis #3 (2 characters).
d_1d_2 — Axis #4 (2 characters).

Bit #	Function for w_1	Meaning for Bit LOW	Meaning for Bit HIGH
0	Controller Power	ON	OFF
1	Executing a command line	NO	YES
2	Executing a program	NO	YES
3	Executing a X-Y trajectory	NO	YES
4	Not used	Default	——
5	Reduced communication	NO	YES
6	Not used	—	Default
7	Not used	Default	—

Bit #	Function for w_2	Meaning for Bit LOW	Meaning for Bit HIGH
0	Manual jog	NO	YES
1	Manual jog with joystick	NO	YES
2	Joystick is present	NO	YES
3	Searching for HOME	NO	YES
4	Display Refresh	Enable	Disable
5	Local / Remote mode	Local	Remote
6	Not used	—	Default
7	Not used	Default	—

REDUCED COMMUNICATION

This indicates that the controller is doing some tasks (e.g.: jog from local mode) that allows only reporting or safety external commands to be executed.

Bit #	Function for a_1 b_1 c_1 d_1	Meaning for Bit LOW	Meaning for Bit HIGH
0	Axis is connected	YES	NO
1	Axis Motor Power	ON	OFF
2	Axis Idle	YES	NO
3	Axis is moving	NO	YES
4	Axis in permanent motion	NO	YES
5	Following error	NO	YES
6	Not used	—	Default
7	Not used	Default	—

Bit #	Function for a_2 b_2 c_2 d_2	Meaning for Bit LOW	Meaning for Bit HIGH
0	Axis is referenced to HOME	YES	NO
1	Limit swich - is actived	NO	YES
2	Limit swich + is actived	NO	YES
3	Constant speed phase	NO	YES
4	Axis is synchronized	NO	YES
5	Not used	Default	—
6	Not used	—	Default
7	Not used	Default	—

Errors
A — Unknown message code.
S — Communication time-out.

Rel. Commands
TS — Read controller status.
TX — Read controller activity.

Example

TX1	*Read controller extended status.*
TX1@@@@@@@@@@	*Controller returns value.*

Newport

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command: xxYS?

Syntax **xxTY**

Parameters

Description	**xx** [int]	—	Variable number.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.

Description This command reads the value of a variable. If the variable was not previously defined with the YS command, the controller returns value 0.

NOTE

The controller allows 100 variables. They are not identified by a name but by a number as a parameter for different commands.

Returns **xxTYnn**

xx — Variable number.
nn — Variable value.

Errors

A — Unknown message code.
O — Variable number out of range.
S — Communication time-out.

Rel. Commands **YS** — Initialize variable.
All variable manipulation commands.

Example

17TY | *Read variable #17.*
17TY28 | *Controller returns value 28 for variable #17.*

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxUF**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**0** to **4.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command makes active the latest PID parameters entered. Any new value for Kp, Ki, Kd and the maximum following error are not being used in the PID loop calculation until the UF command is received. This assures that the parameters are loaded simultaneously and without problems.

If the axis specifier **xx** is missing or set to 0, the controller updates the filters for all axes. If **xx** is a number between 1 and 4, the controller updates only the filter for the specified axis.

Returns None.

Errors
- A — Unknown message code.
- B — Incorrect axis number.

Rel. Commands
- **FE** — Set maximum following error.
- **KD** — Set derivative gain.
- **KI** — Set integral gain factor.
- **KP** — Set proportional gain.

Example

3KP0.05	*Set proportional gain factor of axis #3 to 0.05.*
3KD0.07	*Set derivative gain factor of axis #3 to 0.07.*
3UF	*Update servo loop of axis #3 with the new parameters.*

UH — Wait for I/O high

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **xxUH**

Parameters

Description	**xx** [int]	—	I/O bit number.
Range	**xx**	—	**0** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.

Description This command causes a program to wait until a selected I/O input bit becomes high. It is level, not edge sensitive, which means that at the time of evaluation, if the specified I/O bit **xx** is high already, the program will continue executing.

If the bit specifier **xx** is missing or set to 0, the program will wait for all bits to be high.

NOTE

The command can be placed on a line by itself or with other commands. If placed on a line with other commands, they will be executed with a minimal delay after the I/O bit goes high.

Returns None.

Errors
- A — Unknown message code.
- E — Incorrect I/O channel number.
- J — Command authorized only in programming mode.
- L — Command not at the beginning of a line.

Rel. Commands **UL** — Wait for I/O low.

Example

7UH, 3PA12.3 | *Wait while bit #7 of the I/O input port becomes high and then move axis #3 to position 12.3 units and continue the rest of the program.*

UL — Wait for I/O low

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **xxUL**

Parameters

Description	**xx** [int]	—	I/O bit number.
Range	**xx**	—	**0** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.

Description This command causes a program to wait until a selected I/O input bit becomes low. It is level, not edge sensitive which means that at the time of evaluation, if the specified I/O bit **xx** is low already, the program will continue executing.

If the bit specifier **xx** is missing or set to 0, the program will wait for all bits to be low.

NOTE

The command can be placed on a line by itself or with other commands. If placed on a line with other commands, the advantage is that they will be executed with a minimal delay after the I/O bit goes low.

Returns None.

Errors

A — Unknown message code.
E — Incorrect I/O channel number.
J — Command authorized only in programming mode.
L — Command not at the beginning of a line.

Rel. Commands **UH** — Wait for I/O high.

Example

7UL, 3PA12.3 | *Wait while bit #7 of the I/O input port becomes low and then move axis #3 to position 12.3 units and continue the rest of the program.*

VA — Set velocity

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxVAnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	Velocity value.
Range	**xx**		—	**1** to **8.**
	nn		—	**10^{-6}** to **the programmed value in SETUP mode**.
Units	**xx**		—	None.
	nn		—	Preset units in SETUP mode/second.
Defaults	**xx**	Missing:		Error B.
		Out of range:		Error B.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.

Description This command sets the velocity value for an axis. Its execution is immediate, meaning that the velocity is changed when the command is processed, even while a motion is in progress.

NOTE

Avoid changing the velocity during the acceleration or deceleration periods. For better predictable results, change velocity only when the axis is not moving or when it is moving with a constant speed.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands

AC — Set acceleration.
PA — Move to absolute position.
PR — Move to relative position.

Example

2DV	*Read desired velocity of axis #2.*
2DV10	*Controller returns a velocity value of 10 units/sec.*
2PA15	*Move to absolute position 15.*
WT500	*Wait for 500 ms.*
2VA4	*Set axis #2 velocity to 4 units/sec.*
2DV	*Read velocity of axis #2.*
2DV4	*Controller returns a velocity value of 4 units/sec.*

VB — Set base velocity (Stepper motor only)

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxVBnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [float] — Velocity value.
Range	**xx** — **1** to **8.** **nn** — **0** to **Maximum motion speed defined by the VA command.**
Units	**xx** — None. **nn** — Preset units in SETUP mode/second.
Defaults	**xx** Missing: Error B. Out of range: Error B. Floating point: Error A. **nn** Missing: Error C. Out of range: Error C.

Description This command sets the start/stop velocity for stepping motors only. The allowed start/stop velocity must be less than or equal to the velocity set with the VA command.

NOTE
This command is available only for stepper motors.

Returns None.

Errors
A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands
PA — Move to absolute position.
PR — Move to relative position.
VA — Set velocity.

Example **2VB10** | *Set start/stop velocity for axis#2 to 10 units/sec.*

VE — Read controller version

Usage ■ IMM ■ PGM ■ MIP

Syntax **VE**

Parameters None.

Description This command reads the controller model and version.

NOTE

When asking for technical support with the motion control system or when reporting a problem, having the controller type and version enables us to help you fix the problem fast. Use this command to determine the controller type and in particular, the firmware version.

Returns **VE MM4006 Controller Version xx.yy**

xx.yy
— Version and release number.

Errors S — Communication time-out.

Rel. Commands None.

Example

VE	*Read controller model and version.*
VE MM4006 Controller Version 1.52	*Controller returns model MM4006 and version 1.52.*

VS — Define the vector acceleration on trajectory (trajectory acceleration)

Usage ■ IMM ■ PGM ■ MIP

Syntax **VSnn** or **VS?**

Parameters

Description	**nn** [double]	—	Desired trajectory acceleration.
Range	**nn**	—	>**0** to **Max Trajectory Acceleration** (MTA).
Units	**nn**	—	Units/sec^2.
Defaults	**nn**	Missing:	MTA.
		Out of range:	Error C.

Description This command defines the vector acceleration on trajectory that the controller uses to start and stop execution of the trajectory. In association with the trajectory velocity this will define the necessary time to reach the trajectory velocity.

NOTE

The controller calculates automatically MTA for the trajectory to execute (set of trajectory elements entered before this command) one time this command is entered and limits the vector acceleration to MTA if the parameter entered is greater than MTA. It is then pratical to read MTA just before this command with help of the command XU1 and to read assigned trajectory acceleration after this command with help of the command VS? or XU.

In fact, MTA is defined as the minimum value of maximum allowed X assigned axis and Y assigned axis accelerations.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the actual trajectory acceleration value.

Errors

C — Parameter out of limits.
S — Communication time-out.

Rel. Commands

VV — Define the vector velocity on trajectory (trajectory velocity).
XU — Tell the vector acceleration on trajectory (trajectory acceleration).

Example

XU1	*Read MTA.*
XU10.0	*Controller tells MTA.*
VS8	*Define 8 units/sec^2 as trajectory acceleration.*
XU	*Read trajectory acceleration.*
XU8.0	*Controller tells assigned trajectory acceleration.*

VU — Set maximum velocity

Usage	■ IMM ■ PGM ☐ MIP
Syntax	**xxVAnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	Maximum velocity value.
Range	**xx**		—	**1** to **8.**
	nn		—	**10^{-6}** to **the scaling speed value in SETUP mode**.
Units	**xx**		—	None.
	nn		—	Preset units in SETUP mode/second.
Defaults	**xx**	Missing:		Error B.
		Out of range:		Error B.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.

Description This command sets the maximum velocity value for an axis (see the “Maximum Speed” in the Section “Local Mode” of this user’s manual). This is the maximum value that the VA command can not exceed.

Returns If the sign **“?”** takes place of **nn**, this command reports the actual maximum velocity of axis **#xx**.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.
C	—	Parameter out of limits.
D	—	Unauthorized execution.

Rel. Commands

VA	—	Set velocity.
VW	—	Set scaling velocity.

Example **2VU20** | *Set axis #2 maximum velocity to 20 units/sec.*

VV — Define the vector velocity on trajectory (trajectory velocity)

Usage ■ IMM ■ PGM ■ MIP

Syntax **VVnn** or **VV?**

Parameters

Description	**nn** [double]	—	Desired trajectory velocity.
Range	**nn**	—	>**0** to **Max Trajectory Velocity** (MTV).
Units	**nn**	—	Units/sec.
Defaults	**nn**	Missing:	MTV.
		Out of range:	Error C.

Description This command defines the vector velocity on trajectory that the controller uses to start and stop execution of the trajectory. In association with the trajectory acceleration this will define the necessary time to reach the trajectory velocity.

NOTE

The controller calculates automatically MTV for the trajectory to execute (set of trajectory elements entered before this command) one time this command is entered and limits the trajectory velocity to MTV if the parameter entered is greater than MTV. It is then pratical to read MTV just before this command with help of the command XV1 and to read assigned trajectory velocity after this command with help of the command VV? or XV.

In fact, MTV is defined as the minimum value of minimum value of maximum allowed X assigned axis and Y assigned axis velocities and of minimum value of all trajectory arc elements maximum allowed contouring velocities, that are calculated as square of product of maximum allowed trajectory acceleration (MTA) with arc element contouring radius ($\sqrt{}$(MTA * Radius)).

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the actual trajectory velocity value.

Errors
C — Parameter out of limits.
S — Communication time-out.

Rel. Commands
VS — Define the vector acceleration on trajectory (trajectory acceleration).
XV — Tell the vector velocity on trajectory (trajectory velocity).

Example

XV1	|	*Read MTV.*
XV20	|	*Controller tells MTV.*
VV5	|	*Define 5 units/sec as trajectory velocity.*
XV	|	*Read trajectory velocity.*
XV5.0	|	*Controller tells trajectory velocity.*

VW — Set scaling velocity

Usage ■ IMM ■ PGM □ MIP

Syntax **xxVWnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	Scaling velocity value.
Range	**xx**		—	**1** to **8.**
	nn		—	>$\mathbf{10^{-6}}$.
Units	**xx**		—	None.
	nn		—	Preset units in SETUP mode/second.
Defaults	**xx**	Missing:		Error B.
		Out of range:		Error B.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.

Description This command sets the scaling velocity value for an axis (see the "Scaling Speed" in the Section "Local Mode" of this user's manual). That is the maximum value that the VU command can not exceed.

Returns If the sign **"?"** takes place of **nn**, this command reports the actual scaling velocity of axis **#xx**.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands

VA — Set velocity.
VU — Set maximum velocity.

Example **2VW24** | *Set axis #2 scaling velocity to 24 units/sec.*

Usage ■ IMM ■ PGM □ MIP

Syntax **WAnn**

Parameters

Description	**nn** [int]	—	Wait time (delay).
Range	**nn**	—	**0** to **1073741824.**
Units	**nn**	—	Milliseconds.
Defaults	**xx**	Missing:	0.
		Out of range:	0.
		Floating point:	Decimal part truncated.

Description This command causes the controller to pause for a specified amount of time. This means that the controller will wait **nn** milliseconds before executing the next command.

NOTE
Even though this command can be executed in immediate mode, its real value is as a flow control instruction inside programs.

NOTE
This command is identical to WT. Both exist only for program compatibility reasons with other controllers.

Returns None.

Errors None.

Rel. Commands **WT —** Wait.

Example

6UL, **WA400**, 2PA2.3 | *Wait for I/O input bit #6 to go low, wait an additional 400 ms and then move axis 2 to position 2.3 units.*

WE — End While loop

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **WE**

Parameters None.

Description This command terminates a WHILE loop initiated by any of the WG, WH, WL or WY commands.

Up to 100 While loops can be nested, but they must follow the general rule of multiple loops: last one opened is the first one closed.

NOTE

To be accepted, WE must be placed on a different line than WG, WH, WL or WY. To improve program clarity, it is recommended to place the WE on a separate line.

NOTE

All While loop commands (WG, WH, WL and WY) must be terminated with a WE command.

Returns None.

Errors J — Command authorized only in programming mode.

Rel. Commands

WG — While variable greater than value.
WH — While I/O input is equal.
WL — While variable is less.
WY — While variable is different.

Example

2YS0	*Initialize variable #2 to 0.*
2WY10, 2YA1	*Open first while loop: while variable #2 is different than 10, add 1 to variable #2.*
5WH1, 3PR1.2, WS	*Open second while loop: while I/O input bit #5 is high, move axis #3 incremental 1.2 units and wait for stop.*
WE	*End second while loop.*
WE	*End first while loop.*

WF — Wait for function key

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **xxWF**

Parameters

Description	**xx** [int]	—	Variable number.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.

Description This command interrupts the execution of a program and waits for user input. When the command is executed, the controller displays the function key labels assigned with FB and waits for a valid function key to be pressed. A valid function key is one that has been labeled previously. When a valid function key is pressed, the controller beeps shortly to acknowledge the entry, places the ASCII code of the key in the specified **xx** variable and continues program execution.

Key pressed	ASCII code	Variable value
1st (Left)	A	65
2nd	B	66
3rd	C	67
4th (Right)	D	68

Returns None.

Errors
A — Unknown message code.
J — Command authorized only in programming mode.
O — Variable number out of range.

Rel. Commands
FB — Label function key.
FC — Clear function key line.
FD — Display function keys.

Example

3XX	*Clear program #3 from memory, if any.*
3EP	*Activate program mode and enter following commands as program #3.*
4FBSTOP	*Define custom label for function key #4 as STOP.*
…	
…	
…	
7WF	*Display the custom function key label(s) (STOP), wait for a valid function key to be pressed and put its ASCII code in variable #7.*
FC	*Clear function key display line.*
…	
…	
…	

WG — While variable is greater

Usage	☐ IMM	■ PGM	☐ MIP
Syntax	**xxWGnn**		

Parameters

Description	**xx** [int]	—	Variable number.
	nn [int]	—	Comparison value.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**-32767** to **32767.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	0.
		Out of range:	Error C.

Description This command starts a WHILE loop based on a variable's value. While the selected variable **xx** is greater than the **nn** value, all following commands up to the corresponding WE are executed. The loop is repeated until the test becomes false. At that point, program execution continues with the line immediately following the WE command.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
J	—	Command authorized only in programming mode.
L	—	Command not at the beginning of a line.
O	—	Variable number out of range.

Rel. Commands

WE	—	End While loop.
WH	—	While I/O input is equal.
WL	—	While variable is less.
WY	—	While variable is different.

Example

5YS30	*Initialize variable #5 to 30.*
5WG18	*While variable #5 is greater than 18 repeat next commands.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA-1	*Subtract 1 from variable #5.*
WE	*End while loop.*

WH — While I/O input is equal

Usage ☐ IMM ■ PGM ☐ MIP

Syntax xxWHnn

Parameters

Description	**xx** [int]	—	I/O input bit number.
	nn [int]	—	I/O input bit or byte state.
Range	**xx**	—	**0** to **8.**
	nn	—	**0** to **1** or **0** to **255.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error E.
		Out of range:	Error E.
		Floating point:	Error A.
	nn	Missing:	0.
		Out of range:	1 for a bit, error C for a byte.

Description This command starts a WHILE loop based on the state of an I/O input bit or byte. While the state of the selected I/O input bit **xx** is equal to **nn**, all following commands up to the corresponding WE are executed. The loop is repeated until the test becomes false. At that point, the program executed continues with the line immediately following the WE command.

If **xx** is set to 0 or missing, the test is performed on the entire I/O input byte and then **nn** could have a value from 0 to 255, representing the byte value to compare it with.

NOTE

If the command is set to look for a bit by specifying xx between 1 and 8, a non-zero value for the nn parameter will be considered as a 1 and the while loop will execute until the I/O bit becomes Low.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
E	—	Incorrect I/O channel number.
J	—	Command authorized only in programming mode.
L	—	Command not at the beginning of a line.

Rel. Commands

WE	—	End While loop.
WG	—	While variable greater than value.
WL	—	While variable is less.
WY	—	While variable is different.

Example

5WH1	*While I/O input bit #5 is high, repeat next commands.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
WE	*End while loop.*

WI — Wait for a trajectory (curvi-linear) length

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **WInn**

Parameters

Description	**nn** [double]	—	Trajectory length to wait for.
Range	**nn**	—	**0** to **Trajectory total length**.
Units	**nn**	—	X axis actual unit.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command stops the execution of the program up to when the defined by WI trajectory length is reached.

NOTE

This is a command used in phase of execution and its place is always after ET command. If the value defined by WI is superior than the trajectory total length or by error after trajectory stop the trajectory execution do not reach the desired length, the command execution breaks and returns an error.

NOTE

This command must be used in a program.

Returns None.

Errors

C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands None.

Example

NT	*Initialisation.*
LX10	*Element 1.*
LX30	*Element 2.*
LX40	*Element 3.*
VV5	*Set trajectory velocity to 5 units/sec.*
ET	*Trajectory execution with generation of pulses.*
WI10, *5SB*	*At the realized length of 10 units, set I/O ouput bit #5.*
WI30, *5CB*	*At the realized length of 30 units, reset I/O ouput bit #5.*

WK — Wait for key

Usage	☐ IMM	■ PGM	☐ MIP	
Syntax	**WKaa**			
Parameters				
Description	**aa** [str]		—	String to be displayed, in ASCII format.
Range	**aa**		—	**1** to **14** characters.
Units	**aa**		—	None.
Defaults	**aa**		Missing:	Null string; clears the line.
			Out of range:	Only first 14 characters are used.

Description This command stops the execution of a program. When it is executed, the specified message **aa** is displayed in the middle of line 5 and the menus on line 6 offer two choices: QUIT and EXEC. If the function key corresponding to QUIT is pressed, program execution is terminated. If EXEC is pressed, the program continues execution.

Returns None.

Errors J — Command authorized only in programming mode.

Rel. Commands
WA — Wait.
WP — Wait for position.
WS — Wait for motion stop.
WT — Wait.

Example

WKContinue ?	top program and display on line #5 the string “Continue ?”.
3PA1.2	*M*ove axis #3 to position 1.2 units.

In this EXAMPLE, line five of the front panel will display “ Continue ? ” until QUIT or EXEC is pressed.

WL — While variable is less

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **xxWLnn**

Parameters

Description
xx [int] — Variable number.
nn [int] — Comparison value.

Range
xx — **1** to **100** (integers) and **101** to **120** (floats).
nn — **-32767** to **32767.**

Units
xx — None.
nn — None.

Defaults
xx Missing: Error O.
Out of range: Error O.
Floating point: Error A.
nn Missing: 0.
Out of range: Error C.
Floating point: Decimal part truncated.

Description This command starts a WHILE loop based on a variable's value. While the selected variable **xx** is less than the **nn** value, all following commands up to the corresponding WE are executed. The loop is repeated until the test becomes false. At that point, the program executed continues with the line immediately following the WE command.

Returns None.

Errors
A — Unknown message code.
C — Parameter out of limits.
J — Command authorized only in programming mode.
L — Command not at the beginning of a line.
O — Variable number out of range.

Rel. Commands
WE — End While loop.
WG — While variable is greater than value.
WH — While I/O input is equal.
WY — While variable is different.

Example

5YS0	*Initialize variable #5 to 0.*
5WL18	*While variable #5 is less than 18 repeat next commands.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

WN — Wait for a element of trajectory

Usage ☐ IMM ■ PGM ☐ MIP

Syntax **WNnn**

Parameters

Description	**nn** [int]	—	Number of trajectory element to wait for.
Range	**nn**	—	**0** to **Element total number of the trajectory**.
Units	**nn**	—	None.
Defaults	**nn**	Missing:	Error C.
		Out of range:	Error C.

Description This command stops the execution of the program up to the beginning of the execution of the defined by WN numberred element of the trajectory.

NOTE

This is a command used in phase of execution and its place is always after ET command. If the value defined by WN is superior than the total number of element of the trajectory or by error after trajectory stop the trajectory execution do not reach the desired element, the command execution breaks and returns an error.

NOTE

This command must be used in a program.

Returns None.

Errors
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands None.

Example

NT, FA90	*Initialisation.*
CR10, CA5	*Element 1.*
CA350	*Element 2.*
CA5	*Element 3.*
VV5	*Set trajectory velocity to 5 units/sec.*
ET	*Trajectory execution with generation of pulses.*
WN2, 5SB	*At the beginning of element 2, set I/O ouput bit #5.*
WN3, 5CB	*At the beginning of element 3, reset I/O ouput bit.*

Newport

WP — Wait for position

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxWPnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [float]	—	Position to wait for.
Range	**xx**	—	**0** to **4.**
	nn	—	Starting position to destination of axis number **xx**.
Units	**xx**	—	None.
	nn	—	Preset units in SETUP mode.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error D.

Description This command stops the program execution until a position is reached. The program continues executing the immediate following commands only after axis **xx** reaches position **nn**.

NOTE

Make sure that position nn is inside the travel of axis xx. The controller cannot always detect if a value is outside the travel range of an axis to flag the error, especially in a complex motion program.

Returns None.

Errors

- A — Unknown message code.
- B — Incorrect axis number.
- C — Parameter out of limits.
- D — Unauthorized execution.

Rel. Commands

- **WA** — Wait.
- **WK** — Wait for key.
- **WS** — Wait for motion stop.
- **WT** — Wait.

Example

2PA-10, WS	*Move axis #2 to position -10 units and wait for stop.*
2PA10, **2WP0**, 3PA5	*Move axis #2 to position 10 units, wait for axis #2 to reach position 0 and then move axis #3 to position 5 units.*

WS — Wait for motion stop

Usage	■ IMM	■ PGM	□ MIP
Syntax	**xxWSnn**		

Parameters

Description	**xx** [int]	—	Axis number.
	nn [int]	—	Delay after motion is complete.
Range	**xx**	—	**0** to **4.**
	nn	—	**0** to **1073741824.**
Units	**xx**	—	None.
	nn	—	Milliseconds.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.
	nn	Missing:	0.
		Out of range:	0.
		Floating point:	Decimal part truncated

Description

This command stops the program execution until a motion is completed. The program is continued only after axis **xx** reaches its destination. If **xx** is not specified, the controller waits for all motion in progress to end. If **nn** is specified different than 0, the controller waits an additional **nn** milliseconds after the motion is complete and then executes the next commands.

NOTE

There are a few milliseconds of delay between execution of different command lines. If you need precise delays, place the critical commands on the same line immediately following WS.

NOTE

To terminate an excessively large delay, turn the motor power off and on.

Returns None.

Errors

A	—	Unknown message code.
B	—	Incorrect axis number.

Rel. Commands

WA	—	Wait.
WK	—	Wait for key.
WP	—	Wait for position.
WT	—	Wait.

Example

2PA10, **2WS500**, 3PA5 | *Move axis #2 to position 10 units, wait for axis #2 to reach destination, wait an additional 500ms and then move axis #3 to position 5 units.*

Usage ■ IMM ■ PGM □ MIP

Syntax **WTnn**

Parameters

Description	**nn** [int]	—	Wait time (delay).
Range	**nn**	—	**0** to **1073741824.**
Units	**nn**	—	Milliseconds.
Defaults	**nn**	Missing:	0.
		Out of range:	Nearest range limit.
		Floating point:	Decimal part truncated.
		Non-increment:	Rounded to nearest increment.

Description This command causes the controller to pause for a specified amount of time. This means that the controller will wait **nn** milliseconds before executing the next command.

NOTE

Even though this command can be executed in immediate mode, its real value is as a flow control instruction inside programs.

NOTE

This command is identical to WA. Both exist only for program compatibility reasons with other controllers.

Returns None.

Errors None.

COMMANDES ASSOCIES **WA —** Wait.

Example

6UL, **WT400**, 2PA2.3 | *Wait for I/O input bit #6 to go low, wait an additional 400 ms and then move axis 2 to position 2.3 units.*

WY — While variable is different

Usage	☐ IMM ■ PGM ☐ MIP
Syntax	**xxWYnn**
Parameters	
Description	**xx** [int] — Variable number. **nn** [int] — Comparison value.
Range	**xx** — **1** to **100** (integers) and **101** to **120** (floats). **nn** — **-32767** to **32767.**
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: Error O. Out of range: Error O. Floating point: Error A. **nn** Missing: 0. Out of range: Error C.

Description This command starts a WHILE loop based on a variable's value. While the selected variable **xx** is different than the **nn** value, all following commands up to the corresponding WE are executed. The loop is repeated until the test becomes false. At that point, the program executed continues with the line immediately following the WE command.

Returns None.

Errors
- A — Unknown message code.
- C — Parameter out of limits.
- J — Command authorized only in programming mode.
- L — Command not at the beginning of a line.
- O — Variable number out of range.

Rel. Commands
- **WE** — End While loop.
- **WG** — While variable is greater than value.
- **WH** — While I/O input is equal.
- **WL** — While variable is less.

Example

5YS0	*Initialize variable #5 to 0.*
5WY18	*While variable #5 is different than 18 repeat following commands up to the next WE command.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

XA — Tell the current maximum allowed angle of discontinuity

Usage	■ IMM ■ PGM □ MIP
Syntax	**XA**
Parameters	None.
Description	This command retrieves from the controller the current maximum allowed discontinuity angle.
Returns	**XAnn**
	nn — Maximum discontinuity angle.
Errors	S — Communication time-out.
Rel. Commands	**AD** — Define the maximum allowed angle of discontinuity.

Example **XA**	|	*Tell maximum discontinuity angle.*
XA0.001	|	*Controller returns 0.001°.*

XB — Read backlash compensation

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxBA?

Syntax **xxXB**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the backlash compensation set for an axis. The controller returns the value last set with the BA command.

Returns **xxXBnn**

xx — Axis number.
nn — Backlash compensation in use.

Errors

A — Unknown message code.
B — Incorrect axis number.

Rel. Commands **BA** — Set backlash compensation.

Example

OR	*Perform a home search on all installed axes.*
1BA0.0012	*Set backlash compensation of axis #1 to 0.0012 units.*
2BA0.0008	*Set backlash compensation of axis #2 to 0.0008 units.*
…	
…	
…	
1XB	*Read backlash compensation of axis #1.*
1XB0.0012	*Controller returns axis #1 backlash compensation of 0.0012 unit.*
2XB	*Read backlash compensation of axis #2.*
2XB0.0008	*Controller returns axis #2 backlash compensation of 0.0008 units.*

XD — Read derivative gain factor

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxKD?

Syntax **xxXD**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the derivative gain factor of an axis.

NOTE

The command reads the value actually used in the servo loop. If the PID parameters are modified using the KD command but the digital filter has not been updated by sending an UF, the command will still read the old value.

Returns **xxXDnn**

xx — Axis number.
nn — Derivative gain factor.

Errors

A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands

KD — Set derivative gain.
TF — Read filter parameters.
UF — Update servo filter.

Example

2XD	*Rad derivative gain factor of axis #2.*
2XD0.05	*Controller returns a derivative gain factor of 0.05.*

XE — Tell the last element

Usage	■ IMM ■ PGM □ MIP
Syntax	**XE**
Parameters	None.
Description	This command retrieves from the controller the informations of the last defined element.
Returns	**XEaa, bb, cc, dd**

aa — Type of element:
Line (x, θ), or
Line (y, θ), or
Line (x, y), or
Arc (x, y), or
arc (r, θ).

bb — x end position of the element.

cc — y end position of the element.

dd — Angle of the tangent at the end position.

Errors S — Communication time-out.

Rel. Commands
XT — Tell number of elements in the trajectory.
LT — Extended list of the trajectory.

Example

NT	*Clear trajectory.*
FA45.0	*Define initial tangent angle = 45°.*
LX10	*Define and build line segment = f (10.0, 45.0°).*
XE	*Tell last element.*
XE, Line (x, θ), 10, 10, 45	*Controller tells the built element.*

XF — Read maximum following error

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxFE?

Syntax **xxXF**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8**.
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Hors limitest:	Error B.
		Virgule flottantet:	Error A.

Description This command reads the maximum following error allowed for an axis. If, at any time, the following error is greater than the acceptable value, the controller will stop all motion and turn motors off.

NOTE

The command reads the value actually used in the servo loop. If the PID parameters are modified using the FE command but the digital filter has not been updated by sending an UF, the command will still read the old value.

Returns **xxXFnn**.

xx — Axis number.
nn — Maximum allowed following error.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands
FE — Set maximum following error.
TF — Read filter parameters.
UF — Update servo filter.

Example

2XF	*Read maximum acceptable following error for axis #2.*
2XF0.2	*Controller returns a maximum following error of 0.2 units.*

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxDH?

Syntax **xxXH**

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the home preset position value. This value is the one loaded in the position counter after a home search is initiated and home is found.

NOTE

See the Motion Control Tutorial Section for a description of the Home search algorithm.

Returns **xxXHnn.**

xx — Axis number.
nn — Home preset position.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands
ZH — Set and save hard home preset position.
OR — Search for home.

Example

2XH	*Read home preset position for axis #2.*
2XH8.2	*Controller returns a home preset position of 8.2 units.*

XI — Read integral gain factor

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxKI?

Syntax xxXI

Parameters

Description xx [int] — Axis number.

Range xx — **1** to **8.**

Units xx — None.

Defaults xx Missing: Error B.
Out of range: Error B.
Floating point: Error A.

Description This command reads the integral gain factor of an axis

NOTE

The command reads the value actually used in the servo loop. If the PID parameters are modified using the KI command but the digital filter has not been updated by sending an UF, the command will still read the old value.

Returns **xxXInn**
xx — Axis number.
nn — Integral gain factor.

Errors A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands **KI** — Set integral gain factor.
TF — Read filter parameters.
UF — Update servo filter.

Example **2XI** | *Read integral gain factor of axis #2.*
2XI0.005 | *Controller returns an integral gain factor of 0.005.*

XL — Delete one line of program

Usage ■ IMM ☐ PGM ☐ MIP

Syntax xxXLnn

Parameters

Description	**xx** [int]	—	Program number.
	nn [int]	—	Line number.
Range	**xx**	—	**1** to **127.**
	nn	—	**1** to **32767**.
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error F.
		Out of range:	Error F.
	nn	Missing:	The last line.
		Out of range:	Error C.
		Floating point:	Error A.

Description This command deletes the line **#nn** of **xx** numberred program. If **nn** is default, the command deletes the last line of this program.

NOTE

This command is useful for modifying an existing program without delete it.

Returns None.

Errors

A — Unknown message code.
C — Parameter out of limits.
D — Unauthorized execution.
F — Program number incorrect.
I — Unauthorized command in programming mode.

Rel. Commands

EP — Edition of program.
QP — Quit program mode.

Example

1LP	*Liste program #1.*
1PA10	
WS	
1PR10	
WS	
OR	*Program #1 is listed.*
1XL2	*Delete the line #2 of program #1.*
1XL	*Delete the last line of program #1.*
1LP	*Liste program #1.*
1PA10	
1PR10	
WS	*Program #1 is now listed.*

XM — Read available memory

Usage	■ IMM ■ PGM ■ MIP
Syntax	**xxXM**

Parameters

Description	**xx** [int]	—	Program number.
Range	**xx**	—	**0** to **127.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error F.

Description

If **xx** = 0 or missing this command reads the amount of unused program memory. The controller has 30720 bytes of non-volatile memory available for permanently storing programs. This command reports the amount not used.

If **xx** ≥1 and **xx** ≤127 this command reports the length of the program number **xx**. If the returned value is 0, the program does not exist.

NOTE

The controller is saving programs in their original format, thus using one byte per character.

Returns **xxXMnn**

nn — Returned value, in bytes.

Errors S — Communication time-out.

Rel. Commands

LP — List program.
MP — Download EEPROM to RAM.
SM — Save program.
XX — Erase program.

Example

XM	*Read available program memory.*
XM29873	*Controller returns 29873 bytes of available program memory.*
1XM	*Read the length of the program number #1.*
1XM15	*The length of program #1 is 15 bytes.*
100XM	*Read the length of the program number #100.*
100XM0	*The program #100 does not exist.*

XN — Read number of acquisitions

Usage ■ IMM ■ PGM ■ MIP

Syntax **XN**

Parameters None.

Description This command reads the current number of trace acquisitions. During a trace mode initiated by the TM command, the number of stored samples can be read to monitor the progress of the acquisition process.

Returns **XNnn**

nn — Nnumber of acquired samples.

Errors S — Communication time-out.

Rel. Commands None.

Example

SP 0.005	*Set trace sample period to 5 ms.*
2TM1000	*Enable trace mode for axis #2 and acquire 1000 samples.*
2PR0.2	*Start a relative motion on axis #2 and the acquisition process.*
XN	*Read the number of samples acquired.*
XN157	*Controller reports 157 trace samples acquired.*
XN	*Read the number of samples acquired.*
XN342	*Controller reports 342 trace samples acquired.*
2WS, **XN**	*Wait for stop and read the number of samples acquired.*
XN1000	*Controller reports 1000 trace samples acquired.*

XP — Read proportional gain factor

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxKP?

Syntax xxXP

Parameters

Description	**xx** [int]	—	Axis number.
Range	**xx**	—	**1** to **8.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command reads the proportional gain factor of an axis.

NOTE

The command reads the value actually used in the servo loop. If the PID parameters are modified using the KP command but the digital filter has not been updated by sending an UF, the command will still read the old value.

Returns **xxXPnn**

xx — Axis number.
nn — Proportional gain factor.

Errors
A — Unknown message code.
B — Incorrect axis number.
S — Communication time-out.

Rel. Commands
KP — Set proportional gain.
TF — Read filter parameters.
UF — Update servo filter.

Example

2XP	*Read proportional gain factor of axis #2.*
2XP0.005	*Controller returns an proportional gain factor of 0.005.*

XQ — Read global sample rate

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxSQ?

Syntax **XQ**

Parameters None.

Description This command reads the global trace acquisition period. This is the period that will be used to sample the real and desired positions of all motion devices and store the values in memory.

NOTE
The returned value may differ slightly from the one preset with the SQ command because it reflects the real clock used for the timer. For that reason, the sample period is reported with at least 10 decimal points.

Returns **XQnn**
nn — Real global trace sample period, in seconds.

Errors S — Communication time-out.

Rel. Commands **GQ** — Set global trace mode.
SQ — Set global sample rate.

Example

SQ0.005	*Set global trace sample period to 5 ms.*
GQ1000	*Enable global trace mode and acquire 1000 samples.*
2PR0.2	*Start a relative motion on axis #2 and the acquisition process.*
XQ	*Read the global trace sample period.*
XQ0.005003436196	*Controller returns the real sample rate, in seconds.*

XS — Read trace sample rate

Usage ■ IMM ■ PGM ■ MIP

Recommended equivalent command:
xxSP?

Syntax **XS**

Parameters None.

Description This command reads the trace acquisition period. This is the period that will be used to sample the real and desired positions of a motion device and store the values in memory.

NOTE

The returned value usually differs from the one preset in the SP command because it reflects the real clock used for the timer. For that reason, the sample period is reported with at least 10 decimal points.

Returns **XSnn**

xx — Real trace sample period, in seconds.

.

Errors S — Communication time-out.

Rel. Commands
SP — Set trace sample rate.
TM — Set trace mode.

Example

SP0.005	*Set global trace sample period to 5 ms.*
GQ1000	*Enable global trace mode and acquire 1000 samples.*
2PR0.2	*Start a relative motion on axis #2 and the acquisition process.*
XS	*Read the global trace sample period.*
XS0.005003436196	*Controller returns the real sample rate, in seconds.*

XT — Tell number of elements in the trajectory

Usage ■ IMM ■ PGM □ MIP

Syntax **XT**

Parameters None.

Description This command retrieves from the controller the number of valid elements that have been loaded into the trajectory.

Returns **XTnn**

nn — Number of elements.

Errors S — Communication time-out.

Rel. Commands

AT — Tell the element number under execution.

LT — Extended list of the trajectory.

Example

NT	|	*Reset trajectory buffer.*
XT	|	*Read number of elements.*
XT0	|	*Controller returns 0.*

XU — Tell the vector acceleration on trajectory (trajectory acceleration)

Usage	■ IMM ■ PGM ■ MIP
Syntax	**XUnn**
Parameters	
Description	**nn** [int] — **0** or ≥**1**.
Defaults	**nn** Missing: 0.
Description	This command retrieves from the controller the current trajectory acceleration or Max Trajectory Acceleration (MTA).
Returns	**XUaa** **aa** — The current trajectory acceleration if **nn** missing or 0, **aa** = MTA if **nn** greater or equal 1.
Errors	S — Communication time-out.
Rel. Commands	**VS** — Define the vector acceleration on trajectory (trajectory acceleration). **XV** — Tell the vector velocity on trajectory (trajectory velocity).

Example

VS10	*Define 10 units/sec^2 as trajectory acceleration.*
XU1	*Read MTA.*
XU20.0	*Controller tells MTA.*
XU	*Read trajectory acceleration.*
XU10.0	*Controller tells trajectory acceleration.*

XV — Tell the vector velocity on trajectory (trajectory velocity)

Usage ■ IMM ■ PGM ■ MIP

Syntax **XVnn**

Parameters

Description **nn** [int] — **0** or ≥**1**.

Defaults **nn** Missing: 0.

Description This command retrieves from the controller the current trajectory velocity or Max Trajectory Velocity (MTV).

Returns **XVaa**

aa — The current trajectory velocity if **nn** missing or 0,
aa = MTV if **nn** greater or equal 1.

Errors S — Communication time-out.

Rel. Commands **VV** — Define the vector velocity on trajectory (trajectory velocity).
XU — Tell the vector acceleration on trajectory (trajectory acceleration).

Example

VV5	*Define 5 units/sec as trajectory velocity.*
XV1	*Read MTV.*
XV10.0	*Controller tells MTV.*
XV	*Read trajectory velocity.*
XV5.0	*Controller tells trajectory velocity.*

Newport

XX — Erase program

Usage ■ IMM ☐ PGM ☐ MIP

Syntax xxXX

Parameters

Description	**xx** [int]	—	Program number.
Range	**xx**	—	**0** to **127.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error F.
		Floating point:	Error A.

Description This command erases one or all motion programs loaded in the controller's RAM. It does not erase programs stored in the non-volatile memory. If **xx** is missing or set to 0, all programs in RAM will be erased.

NOTE

On power up, the controller automatically loads all programs stored in non-volatile memory into RAM. If a program is erased using the XX command, to run the same program number, a new one must be created or the old one downloaded from non-volatile memory using the MP command

Returns None.

Errors

A — Unknown message code.
F — Program number incorrect.
I — Unauthorized command in programming mode.

Rel. Commands

LP — List program.
MP — Download EEPROM to RAM.

Example

3XX	*Clear program #3 from memory.*
3EP	*Activate program mode and enter following commands as program #3.*
…	
…	
…	
3QP	*End entering program #3 and quit programming mode.*
3CP	*Compile program #3.*
3CP@	*Controller confirms compilation of program #3 without any errors.*

YA — Add to variable

Usage	☐ IMM ■ PGM ☐ MIP
Syntax	**xxYAnn**

Parameters

Description	**xx** [int]	—	Variable number.
	nn [int]	—	Value to add.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**-32767** to **32767.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	0.
		Out of range:	Error C.
		Floating point:	Decimal part truncated.

Description

This command adds a value to a variable. It is useful for creating loops in a program. The value may be positive or negative.

NOTE

If, at any time the operation will cause the variable value to go out of range, error H (unauthorized execution) is generated.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
H	—	Calculation overflow.
J	—	Command authorized only in programming mode.
O	—	Variable number out of range.

Rel. Commands

TY	—	Read a variable.
YS	—	Initialize variable.

Example

5YS30	*Initialize variable #5 to 30.*
5WG18	*While variable #5 is greater than 18 repeat next commands.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA-1	*Subtract 1 from variable #5.*
WE	*End while loop.*

YB — Negate variable

Usage	☐ IMM	■ PGM	☐ MIP

Syntax xxYB

Parameters

Description	**xx** [int]	—	Variable number.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.

Description This command negates the value of a variable. After executing this command, the value of variable **xx** takes the opposite sign.

Returns None.

Errors

A — Unknown message code.
O — Variable number out of range.

Rel. Commands

TY — Read a variable.
YS — Initialize variable.

Example

3XX	*Clear program #3 from memory, if any.*
3EP	*Activate program mode and enter following commands as program #3.*
7YS3	*Initialize variable #7 to 3.*
7YA2	*Add 2 to variable #7; the new value for the variable is 5.*
7YB	*Negate variable #7; the new value for the variable is -5.*
…	
…	
…	

YC — Add variables

Usage	■ IMM ■ PGM ☐ MIP	
Syntax	**xxYCnn**	
Parameters		
Description	**xx** [int]	— Variable number.
	nn [int]	— Variable number.
Range	**xx**	— **1** to **100** (integers) and **101** to **120** (floats)**.**
	nn	— **1** to **100** (integers) and **101** to **120** (floats)**.**
Units	**xx**	— None.
	nn	— None.
Defaults	**xx** **Missing**:	Error O.
	Out of range:	Error O.
	Floating point:	Error A.
	nn Missing:	Error O.
	Out of range:	Error O.
	Floating point:	Decimal part truncated.

Description This command adds the values of two variables. Variable **xx** is added to variable **nn** and the result placed in variable **xx**. If the result is outside the -32767 to 32767 variable range, the operation is not performed and error H is generated.

Returns None.

Errors
- A — Unknown message code.
- C — Parameter out of limits.
- H — Calculation overflow.
- O — Variable number out of range.

Rel. Commands
- **YA** — Add to variable.
- **YS** — Initialize variable.

Example

5YS0	*Initialize variable #5 to 0.*
2YS6	*Initialize variable #2 to 6.*
5WY18	*While variable #5 is different than 18 repeat next commands.*
5YE10, **5YC2**	*If variable #5 is equal to 10, add variable #2 to variable #5; the value of variable #5 becomes 16.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

YD — Divide variables

Usage	■ IMM	■ PGM	☐ MIP

Syntax **xxYDnn**

Parameters

Description	**xx** [int]	—	Variable number.
	nn [int]	—	Variable number.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**1** to **100** (integers) and **101** to **120** (floats).
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Decimal part truncated.

Description This command divides the values of two variables. Variable **xx** is divided by variable **nn** and the result placed in variable **xx**. If variable **nn** is zero, the operation is not performed and error H is generated. The decimal part of the division result is truncated.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
H	—	Calculation overflow.
O	—	Variable number out of range.

Rel. Commands

YA	—	Add to variable.
YC	—	Add variables.
YS	—	Initialize variable.

Example

5YS5	*Initialize variable #5 to 5.*
2YS9	*Initialize variable #2 to 9.*
1YR3	*Load analog port #3 value into variable #1.*
3YY1	*Copy variable #1 in variable #3.*
3YA-32	*Subtract 32 from variable #3.*
3YM5	*Multiply variable #3 with variable #5.*
3YD2	*Divide variable #3 by variable #2; if variable #1 represents a temperature measured in degrees Fahrenheit, variable #3 will be the equivalent temperature in degrees Celsius.*

YE — If variable is equal

Usage	☐ IMM ■ PGM ☐ MIP
Syntax	**xxYEnn**

Parameters

Description	**xx** [int]	—	Variable number.
	nn [int]	—	Comparaison value.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**-32767** to **32767.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command will allow execution of a command line based on a variable's value. If the selected variable **xx** is equal to the **nn** value, all following commands on that line are executed. The command must be at the beginning of a line and it applies only to that command line.

NOTE

Even though this command can be executed in immediate mode, its real value is as a flow control instruction inside programs.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
L	—	Command not at the beginning of a line.
O	—	Variable number out of range.

Rel. Commands

YA	—	Add to variable.
YG	—	If variable is greater.
YL	—	If variable is less.
YS	—	Initialize variable.

Example

5YS0	*Initialize variable #5 to 0.*
5WY18	*While variable #5 is different than 18 repeat next commands.*
5YE10, 2PR2.6, WS	*If variable #5 is equal to 10, move axis #2 incremental 2.6 units and wait for stop.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

YF — Scale variable

Usage	■ IMM	■ PGM	□ MIP
Syntax	**xxYFnn**		

Parameters

Description	**xx** [int]	—	Variable number.
	nn [int]	—	Scaling factor.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**-32767** to **32767.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command scales the values of a variable. The value of variable **xx** is multiplied by the constant **nn** and the result placed in variable **xx**. If the value of **nn** is zero, the operation is not performed and error H is generated. If the result of the multiplication is outside the -32767 to 32767 range, the operation is not performed and error H is generated. The decimal part of the multiplication result is truncated.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
H	—	Calculation overflow.
O	—	Variable number out of range.

Rel. Commands

YA	—	Add to variable.
YD	—	Divide variable.
YS	—	Initialize variable.

Example

1YR3	*Load analog port #3 value into variable #1.*
3YY1	*Copy variable #1 in variable #3.*
3YA-32	*Subtract 32 from variable #3.*
3YF0.5555556	*Multiply variable #3 by 0.5555556; if variable #1 represents a temperature measured in degrees Fahrenheit, variable #3 will be the equivalent temperature in degrees Celsius.*

YG — If variable is greater

Usage	■ IMM	■ PGM	□ MIP
Syntax	**xxYGnn**		
Parameters			
Description	**xx** [int]	—	Variable number.
	nn [int]	—	Comparison value.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**-32767** to **32767.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description

This command allows execution of a command line based on a variable's value. If the selected variable **xx** is greater than the **nn** value, all following commands on that line are executed. The command must be at the beginning of a line and it applies only to that command line.

NOTE

Even though this command can be executed in immediate mode, its real value is as a flow control instruction inside programs.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
L	—	Command not at the beginning of a line.
O	—	Variable number out of range.

Rel. Commands

YA	—	Add to variable.
YE	—	If variable is equal.
YL	—	If variable is less.
YS	—	Initialize variable.

Example

5YS0	*Initialize variable #5 to 0.*
5WY18	*While variable #5 is different than 18, repeat next commands.*
5YG10, 2PR2.6, WS	*If variable #5 is greater than 10, move axis #2 incremental 2.6 units and wait for stop.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

YK — Read key to variable

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxYK**

Parameters

Description	**xx** [int]	—	Variable number.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
Units	**xx**	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.

Description This command reads the front panel keys and if one is pressed, it places its ASCII code in variable **xx**. If no key is pressed at the time of testing, the variable is set to zero. The following table lists all possible values returned.

Key pressed	ASCII code	Variable value
None	None	0
0	0	48
1	1	49
2	2	50
3	3	51
4	4	52
5	5	53
6	6	54
7	7	55
8	8	56
9	9	57
–	-	45
.	.	46
1st (Left)	A	65
2nd	B	66
3rd	C	67
4th (Right)	D	68

Returns None.

Errors
A — Unknown message code.
O — Variable number out of range.

Rel. Commands **YW** — Wait and read key.

Example

5YS0	*Initialize variable #5 to 0.*
5WL1	*While variable #5 is less than 1, repeat next commands.*
4YK	*Read keys and place code variable #4.*
4YE49, 1PR-0.1	*If key "1" is pressed, move axis #1 -0.1 units incrementally.*
4YE51, 1PR0.1	*If key "3" is pressed, move axis #1 0.1 units incrementally.*
4YE48, 5YS1	*If key "0" is pressed, set variable #5 to 1 to end loop.*
WE	*End while loop.*

YL — If variable is less

Usage	■ IMM	■ PGM	□ MIP

Syntax **xxYLnn**

Parameters

Description	**xx** [int]	—	Variable number.
	nn [int]	—	Comparison value.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**-32767** to **32767.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command will allow execution of a command line based on a variable's value. If the selected variable **xx** is less than the **nn** value, all following commands on that line are executed. The command must be at the beginning of a line and it applies only to that command line.

NOTE

Even though this command can be executed in immediate mode, its real value is as a flow control instruction inside programs.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
L	—	Command not at the beginning of a line.
O	—	Variable number out of range.

Rel. Commands

YA	—	Add to variable.
YE	—	If variable is equal.
YG	—	If variable is greater.
YS	—	Initialize variable.

Example

5YS0	*Initialize variable #5 to 0.*
5WY18	*While variable #5 is different than 18 repeat next commands.*
5YL10, 2PR2.6, WS	*If variable #5 is less than 10, move axis #2 incremental 2.6 units and wait for stop.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

YM — Multiply variables

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxYMnn**
Parameters	
Description	**xx** [int] — Variable number. **nn** [int] — Variable number.
Range	**xx** — **1** to **100** (integers) and **101** to **120** (floats). **nn** — **1** to **100** (integers) and **101** to **120** (floats).
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: Error O. Out of range: Error O. Floating point: Error A. **nn** Missing: Error O. Out of range: Error O. Floating point: Decimal part truncated.

Description This command multiplies the values of two variables. Variable **xx** is multiplied by variable **nn** and the result placed in variable **xx**. If the result is out of the -32767 to 32767 range, the operation is not performed and error H is generated.

Returns None.

Errors
- A — Unknown message code.
- H — Calculation overflow.
- O — Variable number out of range.

Rel. Commands
- **YC** — Add variables.
- **YD** — Divide variables.
- **YF** — Scale variable.
- **YS** — Initialize variable.

Example

5YS5	*Initialize variable #5 to 5.*
2YS9	*Initialize variable #2 to 9.*
1YR3	*Load analog port #3 value into variable #1.*
3YY1	*Copy variable #1 in variable #3.*
3YA-32	*Subtract 32 from variable #3.*
3YM5	*Multiply variable #3 with variable #5.*
3YD2	*Divide variable #3 by variable #2; if variable #1 represents a temperature measured in degrees Fahrenheit, variable #3 will be the equivalent temperature in degrees Celsius.*

YN — If variable is different

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxYNnn**
Parameters	

Description	**xx** [int]	—	Variable number.
	nn [int]	—	Comparison value.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**-32767** to **32767.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	0.
		Out of range:	Error C.

Description

This command allows execution of a command line based on a variable's value. If the selected variable **xx** is different than the **nn** value, all following commands on that line are executed. The command must be at the beginning of a line and it applies only to that command line.

NOTE

Even though this command can be executed in immediate mode, its real value is as a flow control instruction inside programs.

Returns None.

Errors

A	—	Unknown message code.
C	—	Parameter out of limits.
L	—	Command not at the beginning of a line.
O	—	Variable number out of range.

Rel. Commands

YE	—	If variable is equal.
YG	—	If variable is greater.
YL	—	If variable is less.
YS	—	Initialize variable.

Example

5YS0	*Initialize variable #5 to 0.*
5WY4	*While variable #5 is different than 4 repeat next commands.*
5YN1, 2PR2.6, WS	*If variable #5 is different than 1, move axis #2 incremental 2.6 units and wait for stop.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

YO — Send a value to an user analog port

Usage ■ IMM ■ PGM □ MIP

Syntax xxYOnn

Parameters

Description	**xx**	[int]	—	Analog output port number.
	nn	[float]	—	Value to send out.
Range	**xx**		—	**1** to **8**.
	nn		—	**-10.0** to **10.0**.
Units	**xx**		—	None.
	nn		—	None.
Defaults	**xx**	Missing:		0.
		Out of range:		Error E.
		Floating point:		Error A.
	nn	Missing:		Error C.
		Out of range:		Error C.

Description This command writes value to an user analog port. The output value will be limited between -10.0 and 10.0. If **xx** missing, this command writes output value to all of analog output port (1 to 8). If **nn** absolute value is bigger than 10 (**nn** >10 or **nn** <-10), **nn** will be limited between -10 and 10.

NOTE

For the hardware definition of the analog input port, please see Appendix, Connector Pinouts, Remote Control Connector.

Returns None.

Errors

- A — Unknown message code.
- C — Parameter out of limits.
- E — Incorrect I/O channel number.
- Q — Unauthorized command.

Rel. Commands

- **AM** — Set analog input mode.
- **YR** — Read a value from an user analog port and affect variable.

Example

2YS0	*Initialize the variable #2 to zero.*
2WL10	*While the variable #2 is smaller than 10.*
1PR2, WS	*Displacement of 2 units, wait for stop.*
1YO$Y2	*Send the value of variable #2 to analog port number 1.*
2YA1	*Variable #2 is incremented.*
WE	*End of loop.*

YP — Set theoretical position in variable

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxYPnn**
Parameters	
Description	**xx** [int] — Axis number. **nn** [int] — Float variable.
Range	**xx** — **1** to **8.** **nn** — **101** to **120.**
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: Error B. Out of range: Error B. **nn** Missing: Error C. Out of range: Error C.
Description	This command sets the current theoretical position in the desired Y float variable.
Returns	None.
Errors	B — Incorrect axis number. C — Parameter out of limits.
Rel. Commands	**YQ** — Set current position in Y variable.
Example	
1YP101	*Set the theoretical position of the axis #1 in the float variable #101.*

YQ — Set current position in variable

Usage	■ IMM	■ PGM	☐ MIP

Syntax **xxYQnn**

Parameters

Description	**xx** [int]	—	Axis number.
	nn [int]	—	Float variable.
Range	**xx**	—	**1** to **8.**
	nn	—	**101** to **120.**
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error B.
		Out of range:	Error B.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command sets the current position in the desired Y float variable.

Returns None.

Errors
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands **YP** — Set theoretical position in Y variable.

Example

2YQ110 | *Set the current position of the axis #2 in the float variable #110.*

YR — Read a value from an user analog port and affect variable

Usage ■ IMM ■ PGM ■ MIP

Syntax **xxYRnn**

Parameters

Description	**xx** [int]	—	Analog port number.
	nn [int]	—	Variable number.
Range	**xx**	—	**1** to **8.**
	nn	—	**1** to **100** (integers) and **101** to **120** (floats).
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error E.
		Floating point:	Error A.
	nn	Missing:	Error C.
		Out of range:	Error C.

Description This command reads an user analog port and saves the value in a variable. The selected port **xx** is read and the value is loaded in variable **nn**.

NOTE

For the hardware definition of the analog input port, please see Appendix, Connector Pinouts, Remote Control Connector.

Returns None.

Errors

A — Unknown message code.
C — Parameter out of limits.
O — Variable number out of range.

Rel. Commands **RA** — Read analog input.

Example

5YS0	*Initialize variable #5 to 0.*
5WL18	*While variable #5 is less than 18 repeat next commands.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
1YR101	*Load analog port #1 value into float variable #101.*
5YA1	*#5 is incremeted of 1.*
WE	*End while loop.*

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxYSnn**
Parameters	
Description	**xx** [int] — Variable number. **nn** [int] — Initializing value.
Range	**xx** — **1** to **100** (integers) and **101** to **120** (floats). **nn** — **-32767** to **32767.**
Units	**xx** — None. **nn** — None.
Defaults	**xx** Missing: Error O. Out of range: Error O. Floating point: Error A. **nn** Missing: Error C. Out of range: Error C.
Description	This command initializes a variable. When this command is received, the specified variable **xx** is initialized to **nn** value.

NOTE

Always initialize a variable before using it. At power up or after running other programs, the value of a variable may be unknown.

Returns	None.
Errors	A — Unknown message code. C — Parameter out of limits. O — Variable number out of range.
Rel. Commands	**TY** — Read a variable.

Example

5YS0	*Initialize variable #5 to 0.*
5WY18	*While variable #5 is different than 18 repeat next commands.*
3PR1.2, WS	*Move axis #3 incremental 1.2 units and wait for stop.*
5YA1	*Add 1 to variable #5.*
WE	*End while loop.*

YV — Read value from keyboard in a variable

Usage	■ IMM ■ PGM □ MIP
Syntax	**xxYVmessage**

Paramètres

Description	**xx** [int]	—	Variable number.
	message	—	Prompt message.
Value	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	message	—	**1** to **15** characters.
Units	**xx**	—	None.
	message	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	message		
		Out of range:	15 characters.

Description

This command read a value from the keyboard and places it in the variable **xx**. If **xx** = from 1 to 100, the value is entered as an integer. Else if **xx** = from 100 to 120, the value is entered as a float.

If message exists, message is displayed in the value line, else the message **Y[xx]** = takes place. The length of message should not bigger 15 characters, otherwise message will be truncated.

Returns None.

Errors

A	—	Unknown message code.
O	—	Variable number out of range.

Rel. Commands

TY	—	Read a variable.
YK	—	Read key to variable.
YW	—	Wait and read key.

Example

5YVValue is:	*Enter a value in the variable #5.*
3EX	*Execute the program #3.*

YW — Wait and read key

Usage ☐ IMM ■ PGM ☐ MIP

Syntax xxYW

Parameters

Description **xx** [int] — Variable number.

Range **xx** — **1** to **100** (integers) and **101** to **120** (floats).

Units **xx** — None.

Defaults **xx** Missing: Error O.
Out of range: Error O.
Floating point: Error A.

Description This command waits for a front panel key to be pressed and places its ASCII code in variable **xx**. The following table lists all possible values returned.

Key pressed	ASCII code	Variable value
None	None	0
0	0	48
1	1	49
2	2	50
3	3	51
4	4	52
5	5	53
6	6	54
7	7	55
8	8	56
9	9	57
–	-	45
·	.	46
1st (Left)	A	65
2nd	B	66
3rd	C	67
4th (Right)	D	68

Returns None.

Errors A — Unknown message code.
J — Command authorized only in programming mode.
O — Variable number out of range.

Rel. Commands **YK** — Read key to variable.

Example

5YS0	*Initialize variable #5 to 0.*
5WL1	*While variable #5 is less than 1, repeat next commands.*
4YW	*Wait for any key and place its code in variable #4.*
4YE49, 1PR-0.1	*If key "1" is pressed, move axis #1 -0.1 units incrementally.*
4YE51, 1PR0.1	*If key "3" is pressed, move axis #1 0.1 units incrementally.*
4YE48, 5YS1	*If key "0" is pressed, set variable #5 to 1 to end loop.*
WS, WE	*Wait for all motion to stop; end while loop.*

YY — Copy variable

Usage	■ IMM	■ PGM	□ MIP
Syntax	**xxYYnn**		
Parameters			
Description	**xx** [int]	—	Variable number.
	nn [int]	—	Variable number.
Range	**xx**	—	**1** to **100** (integers) and **101** to **120** (floats).
	nn	—	**1** to **100** (integers) and **101** to **120** (floats).
Units	**xx**	—	None.
	nn	—	None.
Defaults	**xx**	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Error A.
	nn	Missing:	Error O.
		Out of range:	Error O.
		Floating point:	Decimal part truncated.

Description This command copies the values of one variable to another. The value of variable **nn** is copied to variable **xx**. After execution, both variables will have the same value.

Returns None.

Errors
A — Unknown message code.
O — Variable number out of range.

Rel. Commands
YC — Add variables.
YD — Divide variables.
YM — Multiply variables.
YS — Initialize variable.

Example

5YS5	*Initialize variable #5 to 5.*
2YS9	*Initialize variable #2 to 9.*
1YR3	*Load analog port #3 value into variable #1.*
3YY1	*Copy variable #1 in variable #3.*
3YA-32	*Ssubtract 32 from variable #3.*
3YM5	*Multiply variable #3 with variable #5.*
3YD2	*Divide variable #3 by variable #2; if variable #1 represents a temperature measured in degrees Fahrenheit, variable #3 will be the equivalent temperature in degrees Celsius.*

ZH — Set and save home preset position

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxZHnn**

Parameters

Description
xx [int] — Axis number.
nn [float] — Home preset position.

Range
xx — **1** to **8.**
nn — **Any value within the hard travel limits.**

Units
xx — None.
nn — Defined motion unit.

Defaults
xx Missing: Error B.
Out of range: Error B.
nn Missing: Error C.
Out of range: Error C.

Description This command defines the value that is loaded in the position counter when home is found. The factory default for this value for all motion devices is 0. This means that, unless a new value is defined with ZH or in the front panel SETUP mode, when a home search is initiated using the OR command or from the front panel, the home position will be set to 0.

NOTE

The change takes effect only when a subsequent home search routine is performed. To make the change permanent, use the PW command to save it in the non-volatile memory.

Returns If the sign **"?"** takes place of the **nn** value, this command reportes the value of the home preset position for the **xx** axis.

Errors
A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands
DH — Define home.
OR — Search for home.

Example

3TP	*Read position on axis #3.*
3TP0.000	*Controller returns position.*
3ZH11	*Set and save home position for axis #3 to 11.*
3TP	*Read position on axis #3.*
3TP11.000	*Controller returns the new position.*

ZL — Set and save left travel limit

Usage ■ IMM ■ PGM ☐ MIP

Syntax **xxZLnn**

Parameters

Description	**xx**	[int]	—	Axis number.
	nn	[float]	—	Left (negative) travel limit.
Range	**xx**		—	**1** to **8.**
	nn		—	**No limit.**
Units	**xx**		—	None.
	nn		—	Defined motion unit.
Defaults	**xx**		Missing:	Error B.
			Out of range:	Error B.
	nn		Missing:	Error C.

Description This command defines the value for the negative (left) travel limit. It is used to restrict travel in the negative direction to protect the motion device or its load.

Returns If the sign **"?"** takes place of the nn value, this command reportes the value of negative (left) travel limit for the **xx** axis.

Errors

A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands

OR — Search for home.
ZR — Set and save right travel limit.

Example **1ZL-50** | *Set and save negative travel limit of axis #1.*

ZP — Zero position

Usage ■ IMM ■ PGM □ MIP

Syntax **xxZP**

Parameters

Description	**xx** [int]	—	Variable number.
Range	**xx**	—	**0** to **4.**
Units	**xx**	—	None.
Defaults	**xx**	Missing:	0.
		Out of range:	Error B.
		Floating point:	Error A.

Description This command forces current position to zero. This means that the coordinate system of the specified **xx** axis will be moved so that the current position becomes zero. If **xx** is not specified, the zeroing operation will be performed on all axes.

NOTE

Because the mechanical system must be protected regardless of the zero position, the positive and negative software limits are recalculated to stay in the same place in space.

Returns None.

Errors

A — Unknown message code.
B — Incorrect axis number.
D — Unauthorized execution.

Rel. Commands

DH — Define home.
OR — Search for home.
ZH — Set and save hard home preset position.

Example

3PA1.23	*Move axis #3 to position 1.23 units.*
3TR	*Read positive software travel limit of axis #3.*
3TR50.000	*Controller returns positive travel limit 50 units for axis #3.*
3ZP	*Set current position of axis #3 to 0.*
3TP	*Read real position of axis #3.*
3TP0.000	*Controller returns real position 0 for axis #3.*
3TR	*Read positive software travel limit of axis #3.*
3TR48.770	*Controller returns positive travel limit 48.77 units for axis #3.*

ZR — Set and save right travel limit

Usage ■ IMM ■ PGM □ MIP

Syntax **xxZRnn**

Parameters

Description **xx** [int] — Axis number.
nn [float] — Right (positive) travel limit.

Range **xx** — **1** to **8.**
nn — **No limit.**

Units **xx** — None.
nn — Defined motion unit.

Defaults **xx** Missing: Error B.
Out of range: Error B.
nn Missing: Error C.

Description This command defines the value for the hard positive (right) travel limit. It is used to restrict travel in the negative direction to protect the motion device or its load.

Returns If the sign **"?"** takes place of the nn value, this command reportes the value of hard positive (right) travel limit for the **xx** axis.

Errors A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.
D — Unauthorized execution.

Rel. Commands **OR** — Search for home.
ZL — Set and save left travel limit.

Example **1ZR50** | *Set and save positive travel limit of axis #1.*

ZT — Read Axis/General parameters configuration

Usage ■ IMM ☐ PGM ■ MIP

Syntax xxZTnn

Parameters

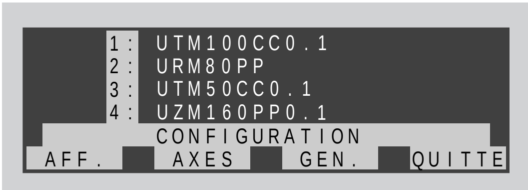
Description
xx [int] — Axis number.
nn — Type of report:
0: Axis configuration.
1: General configuration.

Range
xx — **0** to **8.**
nn — **0** or **1.**

Units
xx — None.

Defaults
xx Missing: 0.
Out of range: Error B.
nn Missing: 0.
Out of range: Error C.

Description This command reports the MM4006 axis/general parameters configuration that is found in the menu SETUP MENU AXES or SETUP MENU GEN. .



- If **xx** ≠ 0: Reports the parameters configuration of the axis #**xx**.
- If **xx** = 0 (missing) and **nn** = 0 (missing): Reports the parameters configuration of all of axes.
- If **xx** ≠ 0 and **nn** = 1: reports the general parameters configuration.

NOTE

It is recommanded to save the controller axes/general parameters configuration in a computer file to avoid the parameters loss when an electrical accident occurs, or when the firmware is upgraded.

Returns Controller axis/general parameters configuration.

Errors
A — Unknown message code.
B — Incorrect axis number.
C — Parameter out of limits.

Rel. Commands None.

Example
1ZT | *Read axis #1 parameters configuration.*
ZT | *Read all of axes parameters configuration.*
ZT1 | *Read general parameters configuration.*