**Quin Systems Limited**

**Programmable Transmission System**

**Registration Controller Reference Manual**

*Issue 14*

*July 1996*

*(MAN432)*

## Copyright Notice



## Software Version

This manual reflects the following software version.

- Registration Controller firmware version 27.9 or higher.

## Important Notice

Quin Systems reserves the right to make changes without notice in the products described in this document in order to improve design or performance and for further product development. Examples given are for illustration only, and no responsibility is assumed for their suitability in particular applications.

Although every attempt has been made to ensure the accuracy of the information in this document, Quin Systems assumes no liability for inadvertent errors.

Suggestions for improvements in either the products or the documentation are welcome.

# Contents



 

# Figures and Tables



 

# 1. Introduction

This document describes the Registration Controller, a member of the Quin Systems digital Programmable Transmission System (PTS) range.

The systems comprise both hardware and software to control a number of servo motors. They are controlled by sending commands via an RS-232 serial interface, either from a standard computer terminal, or from a host computer system. The software allows the user to fully control the servo system using simple high level commands.

**PLEASE READ THIS MANUAL THOROUGHLY !**

Digital control systems are not simple, but can be very useful when applied correctly. It is important to understand the basics of the operation of the system before it is installed on an expensive machine. The system is completely programmable in all aspects of its operation, and it is recommended that users experiment to familiarize themselves with the facilities available. This is best done on a machine which is not required for production !

 

# 2. General Description

This section gives a brief description of the facilities of the Registration Controller. The Registration Controller hardware consists of the TRC-1 two axis controller module.

The system is controlled by high level commands, received directly from an RS-232 computer terminal or serial link. Most commands are two letters, sometimes followed by a numerical parameter. The command set allows full control over every aspect of the servo system. Once a system is programmed, its complete setup, including complete sequences of operations, may be saved in a nonvolatile memory. The system operation can be made fully automatic, so that the programming terminal can be removed once the setup is complete.

A motor may be controlled using simple proportional control, where the demand signal depends on only the position error. The proportional gain constant is set by the user. It is also possible for the user to set gain constants for integral feedback, differential feedback, velocity feedback, and velocity feed-forward terms, providing very flexible control over the system transfer function.

When a move command is entered, the system moves the motor according to a trapezoidal velocity profile defined by the acceleration, velocity, and distance of the requested move. The system velocity and acceleration may be set by the user. The motor speed increases at the set acceleration until it reaches the set velocity. It continues at this velocity until it is near enough to the required position to begin decelerating. The system calculates the point at which it should start decelerating, to minimize any overshoot. The rate of deceleration at the end of the move is the same as the acceleration at the start. If the change in position is small, the motor may not reach the set velocity, and follows a triangular profile instead.

The motors may be controlled at a constant velocity instead of controlling the motor position. In velocity control mode, the system accelerates the motor until it reaches the specified system velocity, and then maintains that velocity. The motor may be stopped with the normal deceleration, or may be stopped abruptly in an emergency.

The system is intended for use with digital incremental position encoders which provide two signals in quadrature. This allows the system to measure both the distance and direction of motion of the motor, thus providing the closed-loop feedback information for the controller. The encoder input interface circuit multiplies the resolution of the encoder by four, such that each complete cycle of the encoder signals represents four counts. The standard systems include full isolation of the encoder input signals, and are designed for use with encoders having differential line driver outputs. This is to get best performance and noise rejection in an industrial environment.

 

The Registration Controller module has seven digital input and eight digital output lines. Normally the module is programmed with default functions for these inputs and outputs, as required by the default command sequences. These default input/output assignments may be changed if required using the standard commands. Inputs may be programmed to start either single commands such as a move or stop command, or to execute a string of commands or a stored sequence. Outputs may be explicitly set and cleared, and can be used to control external relays or valves, or just for status indication. They are often used to allow the system to be controlled from an industrial programmable logic controller (PLC). The inputs and outputs are fully isolated, and are compatible with 24 V logic signals.

The demand signal outputs to the motor drives are analogue signals, ranging from –10V to +10V. These are compatible with most motor drive systems.

Comprehensive sequence commands allow complex operations to be completely specified. Sequences may include any valid commands or command strings.

All facilities may be applied to point-to-point positioning applications, or to continuous process machinery where the motors operate continuously in one direction. The systems may be used with a wide range of motors and drives, depending on the requirements of the application.

# 3. Commands

## 3.1 General Notes

The command reference section gives full details of all the system commands and syntax. Numeric parameters are denoted by “nn” or ‘n’. Parameters entered as a binary string (‘0’s and ‘1’s) are denoted by “bb”. All input commands or command strings are terminated by a carriage return <CR>. The system responses are all followed by <CR><LF>. Note that the system echoes <CR> as <CR><LF>.

Numeric parameters are input and output in either decimal or hexadecimal. Commands are available to set the system to use one or the other. Decimal numbers are output by the system as signed seven digit numbers. Hexadecimal numbers are output in 24 bit two’s complement format as six hex digits with no sign. Decimal numbers are entered as signed or unsigned (assumed positive) numbers. Hex numbers are entered as signed 23 bit or unsigned 24 bit two’s complement numbers. Leading zeros may be omitted when entering values.

The normal character set consists of the letters “A-Z” and “a-z”, the numbers “0-9”, and the ‘+’, ‘–’, and space characters. Commands may be sent in either lower case or upper case. Lower case commands are echoed unchanged, but are converted to upper case before storing as command sequences or function input strings. Strings of multiple commands may be entered as one command line, with the individual commands separated by a ‘/’ delimiter character. The ‘/’(slash) character must be used as the command delimiter. The maximum input line length is 255 characters. The backspace or delete character is used to remove characters from the current input line. Most other non-printing characters are echoed as a dot, and have no effect. Spaces and tab characters are echoed, but are not inserted in any command strings or sequences. If a ‘#’ hash character is received, then it and all subsequent characters up to the end of the lne are ignored. This allows comments to be included in the command file. The escape character may be used to stop a list command, or to exit from the list of commands given by the help command.

The standard command set provides flexible and complete control of the system. The commands fall broadly into the following categories.

- Miscellaneous.
  Commands to change between channels, and to handle the stored setup data.

- Mode commands.
  These include commands to change between motor off and position control modes, and between privileged and normal modes.

- Move commands.
  These are the basic commands for moving and stopping the motors, using the normal trapezoidal move profile.

- Set parameter commands.
  These commands set up a wide range of system parameters, including the velocity and acceleration of the normal moves.

- Sequence commands.
  These commands allow the user to enter, list, and execute complex command sequences.

- Map commands.
  These commands allow the user to execute channel-to-channel position mappings (Software Gearbox).

- Wait commands.
  These commands are used in command sequences to wait until a condition is true before executing the next command in the sequence.

- Error handling.
  These commands set up the system error monitoring functions.

- Gain commands.
  These commands set up the gain constants used in the closed-loop control algorithm.

- Reference commands.
  These commands set up the continuous position correction facilities for use with position reference input signals.

- Digital input and output commands.
  These commands directly control the digital input and output lines.

- Configuration commands.
  These commands configure the digital input and output lines for various automatic functions.

- Phase advance commands
  These commands control a speed-dependent phase advance mechanism for slave channels in mapping, and for position trigger outputs.

- Display commands.
  These commands output parameter values and status information via the serial port.

The command reference section gives the allowable range and any default value of all the system parameters, and in most cases gives an example of the use of the command. Any lengths or length related units are defined in terms of position encoder counts.

The current value of any parameter may be found by entering the command to set the parameter, without entering a new value. The system then shows the current value on the display, followed by a ‘?’ prompt character. The user may then enter a new value, or just type return to keep the current value. The current definitions of all the input and output lines are listed with the LI command.

Many commands that affect the behaviour of the system are *restricted*, or *privileged*, and can be used only in privileged mode after entering a password. This allows the system to be programmed as required by the Control Engineer or Systems Engineer, while preventing access to the more fundamental setup parameters by the machine operator. When programming is complete, the programming terminal may be removed. The system can be programmed to start up automatically, or to operate from external digital signals.

The complete system setup, including all parameter values, input and output line definitions, sequences and profiles, may be stored in nonvolatile memory using the SP save parameters command. The setup data is saved together with a checksum value. This is used when the system is initially powered up to check the integrity of the stored data. If the data has changed at all, the checksum test fails, and the system gives an error message and resets the system to the factory default configuration.

 

## 3.2 Command Execution

Commands can be executed in a number of different ways. This section explains how the system deals with different methods of execution, and how to get the most out of your system. The main ways of executing commands are as follows.

- Command line.
  Commands can be entered singly or as a string at the RS-232 terminal and they are executed immediately when <CR> is typed.

- Sequences.
  Sequences containing one or more lines of commands can be defined using the ES command. The commands are executed by issuing the XS command. Sequences can themselves call other sequences.

- Input line function.
  The DI command can be used to define a command string to be executed when the input line is activated.

The simplest way to execute a command is to type it on its own at the command line. A command entered in this way can be executed at any time, provided the current state of the motor allows it, and it is not a restricted command being entered in normal (unprivileged) mode. If the motor is in an inappropriate state, a context error message will be displayed in the form “cannot execute <cmd> while <state>”. For example, if a VC+ command is entered while the motor is executing a move command the following error message is displayed.

```
Cannot execute VC+ while moving
```

The second way to execute commands is to type a string of several commands at the command line. The advantage is that the system waits for one command to finish before executing the next thus ensuring that the motor is in the correct state to execute each command and avoiding the type of conflict that can occur with single commands. For example, the following command string will safely move to the initial position, run at constant velocity until the input line is activated and then stop.

```
MA1500/VC+/WI3-/ST
```

Note that only one command string can be executing at any one time. For example if a second command string is entered to display the position and velocity while the above string is still being executed an error message is displayed and the second string is rejected.

```
DP/DV
Cannot execute command string while busy
```

A sequence consists of one or more lines of commands which can be executed by issuing the appropriate XS command. Unlike command strings, it is possible to execute one sequence while another is already in progress provided the motor state allows it. In this case the system effectively nests the second sequence inside the first. It suspends execution of the first sequence, executes the new sequence to completion, and then returns to finish the original sequence. The implication is that to execute several command strings “in parallel” it is necessary to first define them as sequences. If sequences 1 and 2 are defined as follows, sequence 2 can be executed successfully while sequence 1 is running.

```
1> ES1
S1: CH1/MA1500/VC+/WI3-/ST
S1:
1> ES2
S2: CH1/DP/DV
S2:
1> XS1
1M XS2
DP345
DV256
1M
```

Input line functions are executed in the same way as command strings and are subject to the same rules. An input line function can not be executed if the channel is already executing another command string or sequence. The exception to this rule is when the function consists of a single command which can be executed at any time provided the motor state allows it. If an input line function fails to execute because the channel is currently busy, the following error message is displayed and the command string is rejected.

```
Cannot execute command string on DIn while busy
```

In this case the function should be entered into a sequence and the input line redefined to execute the sequence. Execute sequence (XS) is a single command and may be executed at any time.

 

# 4. Command Reference

## 4.1 Miscellaneous Commands

**VN** **Print version number.**

This command prints information about the version of software fitted to the system. This version information should be noted for reference in any customer support questions.

**SP** **Save parameters (restricted).**

This command saves all the programmable parameters to nonvolatile memory. There may be a delay while the save operation takes place, depending on the amount of data to be saved. The saved parameters become the new defaults, used by the system on power-up. The SP command also saves any sequences. At the end of the save operation, the system calculates a checksum on the saved data by means of a cyclic redundancy check (CRC) algorithm. The checksum is then also saved in nonvolatile memory. This allows the saved data to be verified at any time by comparing the stored checksum with a newly calculated one. If the saved data has changed at all, the stored checksum will not be the same as the calculated checksum. If the save operation fails for any reason, then an error message such as “nvm write failed” is returned. In this case, please contact your sales office. The SP command is restricted, and is only available in privileged mode.

**CS** **Checksum test.**

This command is used to verify the data stored in the nonvolatile memory. The system calculates a new checksum value for the stored data, and displays it. It then compares the new value with the checksum value that was stored with the data when it was saved. If the values are different, a “checksum error” message is displayed. If it was not possible to calculate the checksum for the stored data, a “checksum failed” error message is displayed. If the checksum test fails, it indicates that the stored data has changed since it was saved. If this occurs, please contact your sales office.

**RD** **Reload stored data (restricted).**

This command reloads all the parameters, input and output line definitions, sequences and profiles from the stored setup in the nonvolatile memory. It also resets all output lines to their power-up state. Undefined outputs are set high. If the stored data checksum is not correct, then this command returns the “stored data invalid” error message, and the stored parameter values are not loaded.

 

**RS** **Reset to default setup (restricted).**

This command resets all the parameters, input and output line definitions, sequences and profiles to their default settings. It also resets all output lines to the default off state (set).

On power-up, the system recalculates the checksum on the saved data in the nonvolatile memory. If the calculated checksum does not match the stored checksum, then the RS function is executed automatically to reset the system to the default state.

**LA** **List all parameters.**

This command lists all the parameters, input and output line definitions, sequences and profiles to the screen in a suitable format for entering the parameters etc. at a later date. If the system is connected to an IBM PC running the PTSTerm program, or any similar communications program that allows logging data to disk, the parameters can be recorded on disk for backup purposes and downloaded into the Registration Controller at a later date if the parameter settings are lost for any reason.

The escape key may be used to stop the LA command early.

**LD** **List default sequences.**

This command lists all the default sequences (stored in the firmware eprom) to the screen, in the same format as the LA command. The LA command lists everything except the default sequences, to prevent them from being redefined by downloading a recorded LA file.

**FM** **Display free memory.**

This command displays information about the memory space available for sequences. It returns the amount of free space in the system memory, string memory and nvm areas as a number of bytes. For more details refer to the full description of the FM command on page 41.

 

## 4.2 Mode Commands

**PC** **Enter position control mode.**

This command puts the current motor channel back into the normal state with the motor position continuously controlled, after the MO motor off command has been executed or any motor error has occurred. The prompt character ‘>’ is returned in position control mode. The demand position is initialized to the current measured position when the PC command is executed.

In position control mode, an onboard relay on the servo controller is energized such that the motor command signal is available from the command signal output. The spare contacts of the relay are also switched over, for use as a drive enable signal if required.

**MO** **Motor off.**

Turns off the position control servo loop action. All other facilities still operate normally, including the input and output lines, and the encoder position is continuously monitored. When the system is returned to position control mode, the motor does not jump back to its last controlled position, but remains at its new position. The system returns a ‘:’ colon character as a prompt when in the motor off state.

In the motor off state, the motor command signal output is switched directly to 0V by the onboard relay. The spare relay contacts are also switched to their normal unenergized state. It is recommended that this relay is used to disable the motor drive completely. If the drive is not disabled in the motor off state, then it is likely that the motor position will drift, due to some offset in the drive circuits, since the motor position is not controlled in this state.

The MO command may also be used as a third stop command, to put the motor directly to the motor off state from any other state, instead of using the ST stop or AB abort commands.

 

**PM** **Enter privileged mode.**

The full set of commands available on the Registration Controller is very powerful and complete. However, in most applications, it is only necessary to make full use of the commands when the system is first programmed, and not during normal operation. Many of the commands control the basic setup of the system, such as the gain commands used to tune the system. Unauthorized access to these commands could result in a severe loss of performance or even damage to the machine. For this reason, the command set is divided into **normal**, and **restricted** or **privileged** commands.

The normal commands are always available. These include the basic move commands, and many of the simple set parameter commands such as those used to set the velocity or acceleration for the system. Restricted commands are only available in what is termed **privileged mode**. Entry to privileged mode is only permitted with a password, which itself is programmable.

If restricted parameters must be changed during normal operation, the relevant commands may be executed from a stored sequence. This bypasses the privileged mode check at runtime, but still prevents unauthorized access to the system programming since the ES enter sequence command is also restricted.

The PM command is used to enter privileged mode from normal mode and gain access to the complete command set. The system responds with “Enter password : ” to prompt the user to enter the password. The password is **not** echoed as it is entered. If the password is correct, the system responds with an “O.K.” message, and goes into privileged mode. If the password is incorrect, the system prints the error message “password incorrect” and stays in normal mode.

**Example :**

| System | User | Comments |
|---|---|---|
| > | PM<CR> | Privileged mode command |
| Enter password : | | The password is not echoed |
| O.K. | | Password accepted |
| > | | |

**NM** **Enter normal mode.**

This command is used to return to normal mode from privileged mode, if the user no longer needs access to the restricted commands. Note that the system powers up in normal mode.

**PW** **Set password (restricted).**

This command allows the user to set the privileged mode password. The system replies “Enter password : ”, and the user should then type in the new password. The new password is limited to a maximum of ten characters. The password is saved in nonvolatile memory with the other setup parameters when the SP command is executed. The PW command is itself restricted, and is only available in privileged mode.

**Example :**

| System | User | Comments |
|---|---|---|
| `>` | `PW<CR>` | Set password command |
| `Enter password` | `:xxxxx` | The new password is echoed |
| `>` | | |

**VMn** **Set virtual motor mode (restricted).**
**Range : 0 to 1**

This command defines whether the axis is in normal or virtual motor mode. In virtual motor mode, the axis can operate without a motor or encoder connected, since the actual position is calculated internally from the demand position. If the value passed with the VM command is zero, the channel is set to normal mode. If the value passed is 1, the channel is set to virtual motor mode. The motor enable relay is held in the off state when in virtual mode.

Virtual motor mode may be particularly useful in the following circumstances.

- For testing commands and sequences before the controller is connected to the machine.

- For providing a dummy master axis in position mapping.

**Safety Note :**
The control signal to the motor is switched to 0V and the enable relay is set to the motor off state when the axis is in virtual motor mode.

## 4.3 Move Commands

**MA±nn** **Move to absolute position ± nn.**
**Range : ± 4 000 000 (4.0E6) encoder counts.**

The motor moves to the absolute position given in the command. It follows a trapezoidal velocity profile (graph of velocity against time). The motor accelerates from rest at the system acceleration, set by the SA command, until it reaches the system velocity, set by the SV command. At the end of the move, the motor decelerates at the same rate to stop at the desired final position. The position is entered in encoder counts.

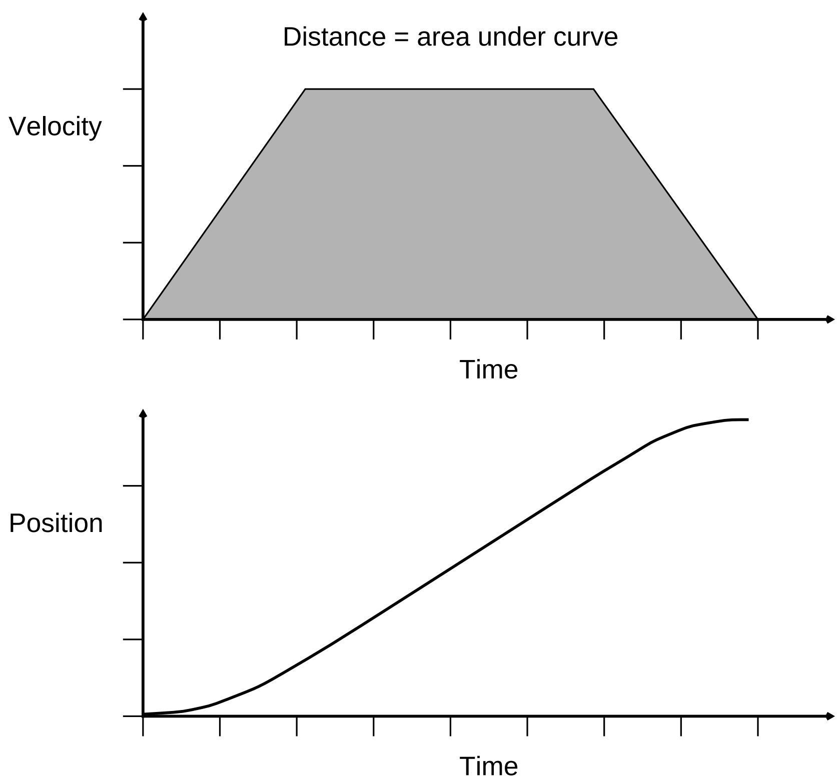


**Figure 1. Trapezoidal move profile.**

If the move distance is small, the velocity is high, or the acceleration is low, the motor may not reach the set velocity within the given move distance. In this case the motor follows a triangular velocity profile instead of a trapezoidal one.

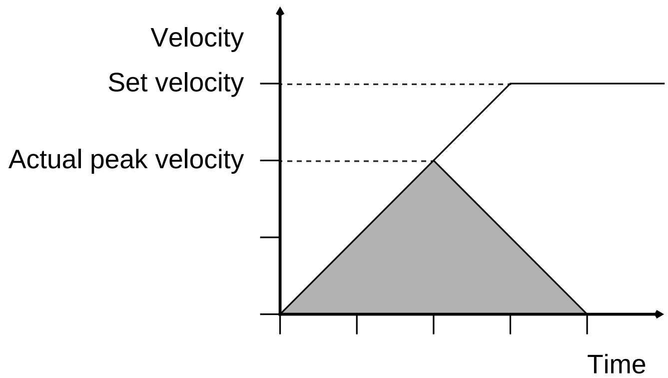


**Figure 2. Triangular move profile.**

When the system executes an absolute or relative move command, it gives the ‘M’ move prompt character. The move commands may only be used from the idle position control state. If the channel is not in position control mode when a move command is entered, the system returns the error message “cannot execute MA while ...”. If no parameter is given, the system gives the error message “invalid command”. If the position is outside the allowed range, it returns the error message “parameter out of range”.

On cyclic machines it may be useful to constrain absolute moves such that the motor always moves in one direction, or always moves the shortest distance to the required absolute position. These options are controlled by the move direction constraint bits (bits 1–3) in the MW parameter.

**Example : `MA+2000`**

The motor moves to absolute position +2000 counts.

**MR±nn** **Move ± nn units relative to current position.**
**Range : ± 8 000 000 (8.0E6) encoder counts.**

The system performs a move similar to the absolute move above, but the move distance is defined relative to the current demand position. The move distance is entered in encoder counts.

The direction of a move relative is not constrained by the direction constraint bits in the MW parameter. The direction of the move is given by the sign of the move relative parameter.

**Example : `MR-3000`**
The motor moves 3000 counts from its current position in the negative direction.

**ST** **Stop.**

The motor stops under controlled deceleration, set by the DC command. The stop command may be used during any motion to decelerate the motor to a stop. When the motor is stopping, the system gives the ‘S’ stopping prompt character. The ST command may also be used to exit prematurely from any wait condition.

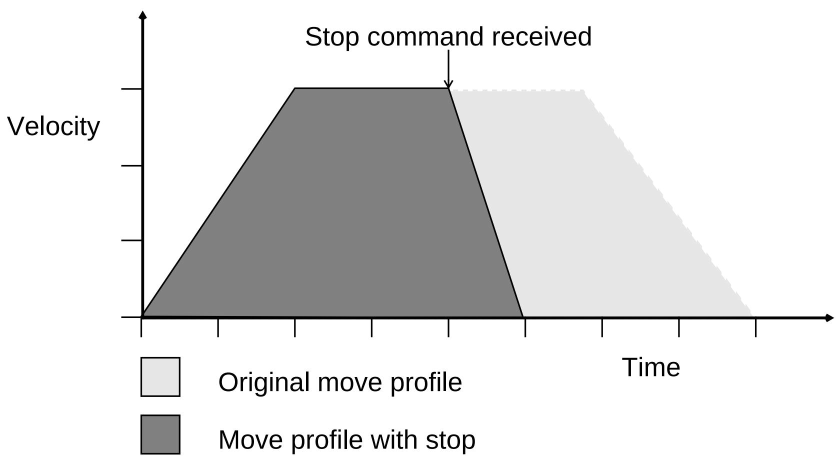


**Figure 3. Move with normal stop.**

The ST command does not break out of any local command sequences or repeat loops currently being executed. If it is required to stop command execution, the AX command must be used.

If the ST stop command is used to stop from mapping, then the deceleration starts at the current instantaneous speed.

**AB** **Abort, emergency stop.**

The motor stops immediately, ignoring the system deceleration value set by the DC command. This may be used instead of the ST command, where an immediate stop is required. It can be used at any time to stop the motor immediately. The AB command is also used to exit prematurely from any wait condition.

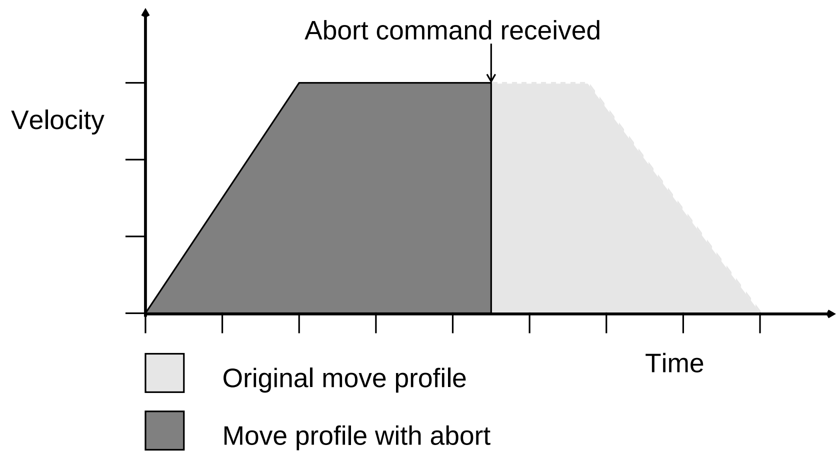


**Figure 4. Move with abort.**

The AB command does not break out of any local command sequences or repeat loops currently being executed. If it is required to abort command execution, the AX command must be used.

**VC**[±] **Move at constant velocity.**

This command is used to move the motor at a constant velocity in the direction specified, without any target position. If the direction is not specified, the motor moves in the direction given by the DN command. The system accelerates the motor at the defined acceleration until it reaches the velocity set by the SV command. It then controls the motor at constant velocity, until it is told to stop. While in constant velocity mode, the system gives the ‘V’ velocity control prompt character.

Velocity control mode can only be entered from position control mode, and not directly from the motor off state. If the channel is not in the idle position control state, then the VC command returns an error message.

**Example :** **`SA1000/SV2000/VC+`**

This command sequence sets the acceleration to 1000 units per second squared, the velocity to 2000 units per second, and then accelerates to the set velocity in the positive direction.

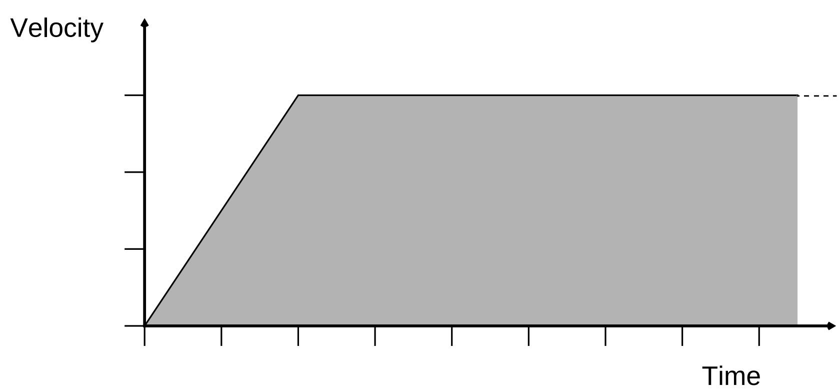


**Figure 5. Constant velocity move.**

 

**IN[±]** **Initialize position.**

The system performs the initialization sequence to find a zero position reference signal. The system gives the ‘I’ initialize prompt character while executing the initialization sequence. From the normal PC state, the motor accelerates to the system velocity in the direction specified. If no direction is specified, the motor moves in the direction given by the DN command. When the system detects any reference input signal, it decelerates to a stop and resets the position counters as required. The motor then (optionally) moves back to the new zero position.

This command may also be used in states other than PC. In this case the system simply waits for a reference signal and sets the zero position accordingly.

**NOTE :** The IN command works independently of the settings of all the other reference commands. This is so that whatever the reference setup for normal running, the IN command always works normally. The exceptions to this are bit 3 of the RW reference options word, which disables the move back to the new zero point after the reference input is detected, and bit 4 of RW which defines whether any reference input is valid, or only a combination of them. The RF reference offset value is also effective during the initialization sequence, such that the position at which the reference signal is detected is defined as the absolute position given by the value of RF, not necessarily zero. For more details please refer to section 4.10 later in this manual.

If no reference input or marker input is defined, then the IN command returns the error message “no reference input defined”, and the initialization sequence is not executed.

**Example : `IN+`**

The motor moves in the positive direction until a valid reference input is seen. It then stops, and moves to the newly defined zero position. In this example, the motor moves back to the position where the reference input signal was detected.

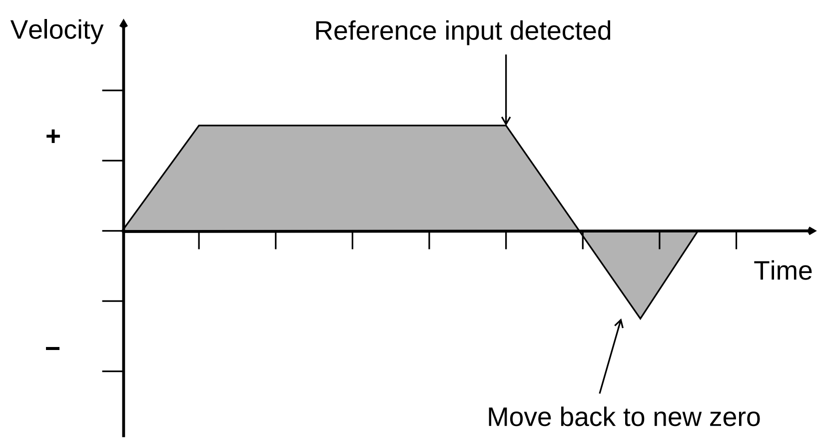


**Figure 6. Initialization to zero position.**

**IB[±]** **Initialize position and bounds.**

This command is similar to the IN command but also sets the position bounds in addition to finding the zero position. The system gives the 'I' initialize prompt character while executing the initialization sequence. From the normal PC state, the motor accelerates to the system velocity in the direction specified. If no direction is specified, the motor moves in the direction given by the DN command. When the system detects a reference input signal, it stores the position value immediately. The motor continues to move until a second reference input signal is detected and sets the position bounds to the distance moved since the first reference signal was received. The motor then decelerates to a stop, resets the position counters, and moves back to the new zero position.

This command may also be used in states other than PC when the system simply measures the distance between two successive reference signals and sets the zero position and bounds accordingly.

**NOTE :** The IB command works independently of the settings of all the other reference commands. This is so that whatever the reference setup for normal running, the IB command always works normally. The exceptions to this are bit 3 of the RW reference options word, which disables the move back to the new zero point after the reference input is detected, and bit 4 of RW which defines whether any reference input is valid, or only a combination of them. The RF reference offset value is also effective during the initialization sequence, such that the position at which the reference signal is detected is defined as the absolute position given by the value of RF, not necessarily zero. For more details please read the Reference Commands section later in this manual.

If no reference input or marker input is defined, then the IB command returns the error message "no reference input defined", and the initialization sequence is not executed.

**DN±** **Set motor direction.**
**Range :** **+ or –**
**Default :** **+**

This command is used to set the default motor direction for the VC, IN and IB commands. If the command is issued without a sign the current default direction is displayed.

**Example :** `DN-/VC`
This sets the default direction to negative and starts the motor at constant velocity backwards. It is equivalent to issuing a VC- command.

**ID** **Initialize demand signal offset.**

Under normal conditions, there may be some constant offset in the demand signal analogue output amplifiers which causes the motor to settle at a position slightly different to the required position. The ID command sets the system up to correct for this (assumed constant) offset in all subsequent position control operations. It must be used every time the system is powered on, when the system is in the position control mode, to set the actual position as close as possible to the required position. This is particularly necessary when the final position window as set by the SW command is small, otherwise the output offset may be such that the motor normally settles at a position outside the final position window, and at the end of a move command it returns the error message “failed to reach target position”. The ID command is only effective in normal position control mode, with the motor actually controlling the position, and it has no effect if the motor is not driving the system. Note that friction in the mechanical system can also cause a position offset after a move command is executed.

## 4.4 Set Parameters

**SVnn** **Set velocity (speed).**
**Range : 0 to 4 000 000 (4.0E6) counts per second**
**Default : 1024**

This command is used to set the system velocity, in encoder counts per second. It may be used at any time, including when the motor is already moving. The diagram below shows a typical velocity profile where the velocity is increased part way through a normal move.

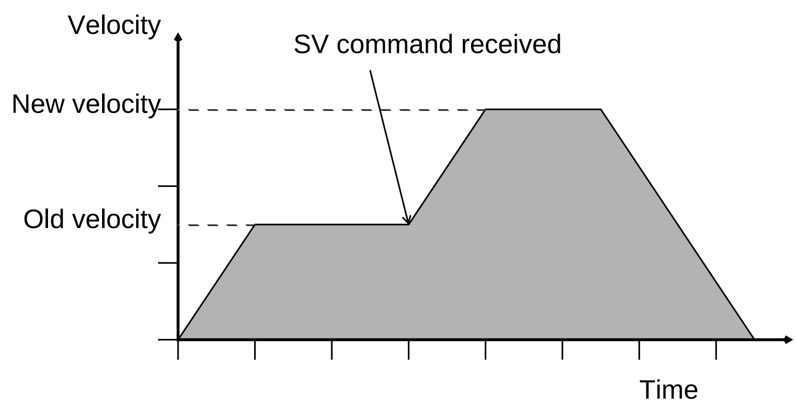


**Figure 7. Move with change of velocity.**

**Example : `SV5000`**
This sets the system velocity to 5000 counts per second.

 

**SAnn** **Set acceleration.**
**Range : 1 to 2 000 000 000 (2.0E9) counts per second squared**
**Default : 1024**

This command sets the normal system acceleration to the specified value, in encoder counts per second squared. The acceleration value is used in the normal trapezoidal move functions for both the acceleration and deceleration ramps. It may be changed at any time. If a new acceleration value is given when the system is executing a move command, then the new value is accepted but is not used until the start of the next move. Note that the actual acceleration is rounded to the nearest multiple of 256 counts/second$^2$, and the minimum acceleration value is 256 counts/second$^2$.

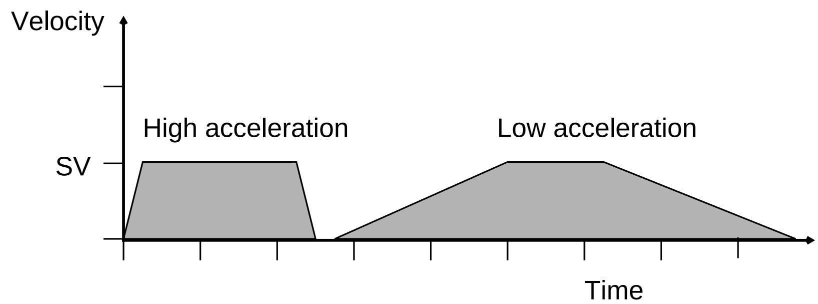


**Figure 8. Set acceleration.**

**Example : `SA10000`**

This sets the system acceleration to about 10000 counts/second$^2$. The actual acceleration value used in this case is 9984 counts/second$^2$ because of the rounding to the nearest multiple of 256.

**DCnn** **Set deceleration for ST command.**
**Range : 1 to 2 000 000 000 (2.0E9) counts per second squared**
**Default : 1024**

This command sets the deceleration to be used when stopping as a result of the ST command, and in normal initialization when the reference input is detected. The deceleration at the end of a normal move is set by the SA command. The command specifies the deceleration value in encoder counts per second squared. Note that the actual deceleration is rounded to the nearest multiple of 256 counts/second$^2$, and the minimum value is 256 counts/second$^2$.

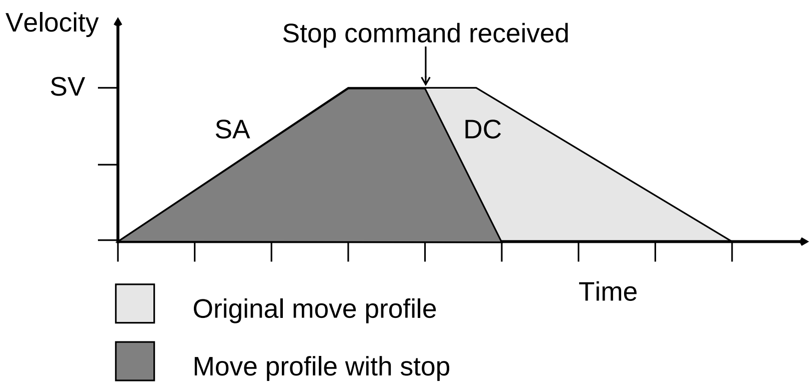


**Figure 9. Effect of DC command.**

The figure above shows the effect of setting the DC value higher than the SA value. This allows a gentle acceleration and deceleration for normal move profiles but gives a sharper deceleration if the move has to be terminated early by a ST command.

**SSnn** **Set slow speed.**
**Range : 0 to 4 000 000 (4.0E6) counts per second**
**Default : 32**

This command allows the user to set the slow speed to be used in slow velocity mode when VJ is set to 1. It is specified in the same units as the system velocity.

**Example : `SS100`**
This sets the slow speed to 100 counts per second.

 

**VJn** **Set slow velocity mode.**
**Range : 0 to 1**
**Default : 0**

Setting VJ to 1 enables slow velocity mode. In this mode all moves are made at slow velocity as set by the SS command. Setting VJ to 0 puts the axis into normal velocity mode where moves are made at normal velocity as set by the SV command.

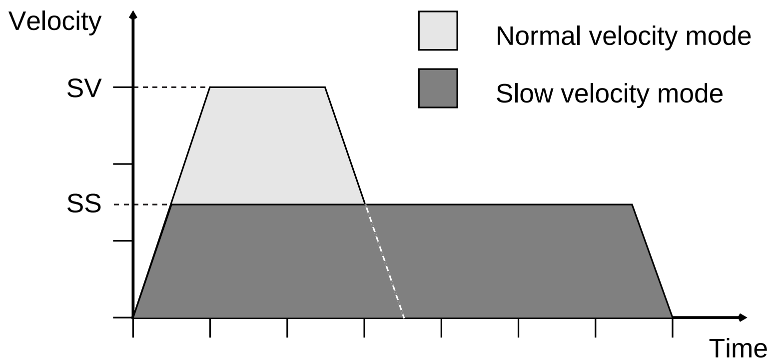


**Figure 10. Normal/slow velocity mode.**

**SWnn** **Set window (restricted).**
**Range : 0 to 65535 encoder counts**
**Default : 10**

This command sets a window or tolerance around the required final position of a move. The system defines the endpoint of a move as being when the demand position has reached the target position and the measured position is inside the window. It returns from the move state to the position control state only when the motor is within this window. Note that when using a narrow window, it is important that the demand signal offset has been initialized with the ID command. If not, the offset may be large enough to put the motor outside the window when it is stopped, and the system will return the error message “failed to reach target position”. Note that the window is specified in encoder counts, and is not scaled by the user scale factor. This command is restricted, and may only be used in privileged mode.

**Example : `SW25`**

This command sets the window to 25 counts. Thus the system returns the normal prompt at the end of a move only when the motor is within 25 counts of the required position.

**ISn** **Set increment select code (restricted).**
**Range : 0 to 8**
**Default : 0**

This command selects the parameter which is incremented by the IP command. Each channel has a separate IS value. The parameter selected is defined by the code as follows.

| Code | Parameter | Limits |
|---|---|---|
| 0 | None | |
| 1 | Current running speed | See below |
| 2 | Reserved | |
| 3 | Map base (MB) | ±4000000 |
| 4 | Map offset (MF) | ±4000000 |
| 5 | Setup/Ratio number (SN) | 1 to 8 |
| 6 | Set velocity (SV or SS) | See below |
| 7 | Scaled map multiplier (SM) | 0 to 65535 |
| 8 | Set bounds (SB) | 1 to 4000000 |

**IPnn** **Increment selected parameter.**
**Range : dependent on select code**

This command adds the value given to the parameter selected by IS. This allows a selected parameter, such as motor speed, to be increased or decreased in steps by repeating a single command. The increment value may be positive or negative and may be of any size. If the incremented value would exceed its allowed range, it is set to its maximum or minimum value as appropriate. When a parameter value is changed with the IP command, then the new value is retained until the unit is turned off, or the previously saved values are restored. The changed values are not automatically saved, but may be saved with the SP command if required.

The lower limit for incrementing the running speed (IS1) is zero. If the axis is running in normal velocity mode, at the speed set by the SV command, the upper limit is equal to twice SV. If the axis is running in slow velocity mode at the speed set by the SS command, the upper limit is equal to SV. The increment affects only the current running speed and not the SV or SS parameters, and is effective only until the motor stops. Subsequent moves start with the original speed of either SV or SS, as set by the VJ parameter.

Changes to map base (IS3) or offset (IS4) affect the MB/MF parameter values. If map base is changed, the change is subject to AV and the map base value wraps round at the master axis bound. If map offset is changed, the change is subject to AV and the map offset value wraps round at the slave axis bound.

 

Changes to the setup number (IS5) affect the SN parameter. SN is incremented or decremented by the value of the IP argument, and wrapped around to keep the result within the range 1 to NS. NS is the number of setups available, set by the NS command. If SN is incremented above NS, it is set to 1, and if it is decremented below 1, it is set to NS. Note that SN may be set to other values manually if required.

The set velocity increment (IS6) is applied to the current set speed parameter, depending on the value of the VJ parameter. If the axis is in normal velocity mode (VJ0), then the speed increment is applied to the SV parameter, and the upper limit is the same as that for the SV command. If the axis is in slow velocity mode(VJ1), then the speed increment is applied to the SS parameter, and the upper limit is equal to SV. The lower limit for the set velocity in both cases is zero.

**Example :**

| System | User | Comments |
|---|---|---|
| `1>` | `SV1000` | Move velocity is 1000 |
| `1>` | `IS1` | Select running speed for increment |
| `1>` | `VC+` | Start motor running |
| `1V` | `IP100/DV` | Increment running speed |
| `1 DV1100` | | Running speed is now 1100 |
| `1V` | `IP100/DV` | Increment speed again |
| `1 DV1200` | | Speed is now 1200 |
| `1V` | `SV` | Display set speed |
| `1 SV1000` | | Set speed has not changed |

**MWbb** **Set move/map options word (restricted).**
**Range : 8 bit binary value.**
**Default : 0**

This command allows the user to modify the operation of absolute moves as described previously. It also controls various aspects of the position mapping functions; these are described later in section 4.6. The value is entered as a binary number. Leading zeros may be omitted. The bit functions for MW relevant to the move absolute command are described here. These options are useful on cyclic machines, where a given absolute position repeats once every machine cycle, as defined by the SB value.

Bit 0 Used in mapping. Please refer to section 4.6, Map Commands.

Bit 1 When set to 0, the target position for an absolute move is taken from the command argument. The move distance may be larger than the set bound value SB if required.
When set to 1, the target position for an absolute move is set to the **nearest** correct cyclic position within SB. If MW bit 2 is set to 0, then the move distance is limited to ±SB/2. If MW bit 2 is set to 1, then the direction for absolute moves is set by MW bit 3, and the maximum move distance is SB, in the set direction only.

Bit 2 This bit is used if absolute moves must be constrained to only one direction.
When set to 0 it has no effect, and absolute moves may be in either direction, depending on the current position and the target position of the move.
When set to 1, absolute moves are always in one direction, as set by bit 3 of MW (see below).

Bit 3 This bit sets the direction for all absolute moves, if they are constrained by setting bit 2 of MW to 1.
When set to 0, absolute moves are in the positive direction.
When set to 1, absolute moves are in the negative direction.

Bit 4 Used in mapping.

Bit 5 Used in mapping.

Bit 6 Used in mapping.

Bit 7 Used in mapping.

 

**CWbb** **Set control word (restricted).**
**Range : 8 bit binary value.**
**Default : 0100 0000**

This command allows the user to write a value into the control word for the current channel. Note that leading zeros may be omitted. The control word allows the state in which the axis powers up to be defined, and allows the sense of the encoder input and of the command signal output to be reversed. The control word bit functions are described below.

**NOTE :** The encoder and command signal sense should only be changed while the module is in the motor off state, as the system may be made completely unstable by reversing either of these. This facility is intended to be used only when initially connecting the module to the motor system, to avoid having to rewire the system if the encoder connections are reversed. It also allows the logical positive and negative directions to be reversed under software control, by toggling both the encoder and output reversal bits in the control word.

Bit 0 Reserved.

Bit 1 Reserved.

Bit 2 Reserved.

Bit 3 This bit enables operation with a unipolar analogue output signal. The command signal is limited to the range 0–10V, and the direction is given by a digital output line, set by the DU command. This option is used with inverters or other similar drives which only accept speed input signals between 0 and 10V.

Bit 4 This bit defines the sense of the main analogue output for the motor command signal.
When set to 0, the command signal sense is normal; if the encoder is moved in the positive direction, a negative output voltage is produced at the command output.
When set to 1, the sense of the command signal output is reversed; if the encoder is moved in the positive direction, the command signal goes positive.

Bit 5 This bit defines the logical sense of the encoder input.
When set to 0, the encoder direction sense is normal; if encoder signal track A leads track B the motion is positive.
When set to 1, the encoder direction is reversed; if track A leads track B the motion is negative.

Bit 6 This bit defines the initial state of the system at power-up.
When set to 1, the axis powers up in the motor off state with the servo loop disabled. This is the default case for standard systems.
When set to 0, the axis powers up in position control mode with the servo loop active.

Bit 7 This bit modifies the integral control action to help avoid the problem of wind-up during a move.
When set to 0, the integral term is active continuously. This is the normal setting.
When set to 1, the operation of the integral action is modified such that the position error is only added to the current integral total when the motor is static, in the idle position control state. The integral value is maintained constant throughout any moves to sustain any offset correction accumulated while static.

The default control word value of 01000000 makes the channel power up in the motor off state.

**Example : `CW01110000`**

This reverses the sense of both the encoder feedback and the analogue output to the drive amplifier.

 

## 4.5 Sequence Commands

This section describes the sequence commands. They provide comprehensive facilities for defining, reviewing and executing complex command sequences. Sequence definitions may be entered up to the memory capacity of the system. If the system runs out of memory, it returns a “memory full” error message.

**ESnn** **Enter sequence (restricted).**
**Range : 1 to 255**

The sequence commands allow the user easily to build up complex sequences of machine operations and store them in the Registration Controller. A stored sequence may be called up and executed with a single command.

The ES command is used to enter stored command sequences into the system. The system responds with a “Snn:” prompt for the sequence entries. Each entry in the sequence can be any valid command line. Command strings on one command line are accepted as one sequence entry. Sequence entries may also include commands to execute other sequences and profiles, to allow sequences to be nested. To end the sequence, make a blank entry by just typing a carriage return, and the system then returns to normal operation. The sequence is accessed by means of the sequence number assigned by the user when it is entered.

The enter sequence command is restricted. This is because sequences may themselves contain restricted commands. When a sequence is executed, any restricted commands within the sequence execute normally, even if the system is not in privileged mode. This allows the user to set up predefined sequences including restricted commands, which can normally only be used in privileged mode.

A previously defined sequence may be changed simply by entering a new sequence definition with the same number. A sequence is deleted by re-defining it as an empty sequence with no commands.

**Example :** Entering a sequence

| System | User | Comments |
|---|---|---|
| `>` | `ES1<CR>` | Enter sequence 1 |
| `S1:` | `ID/IN-<CR>` | Initialize position |
| `S1:` | `MA100/WT256/MA0/WT256/RP3<CR>` | Repeat this 3 times |
| `S1:` | `MA2000<CR>` | A single move |
| `S1:` | `<CR>` | End sequence |
| `>` | | Normal prompt |

**Example :** Deleting a sequence

| System | User | Comments |
|---|---|---|
| `>` | `ES1<CR>` | Enter sequence 1 |
| `S1:` | `<CR>` | End sequence |
| `>` | | Normal prompt |

**LS[nn]** **List sequence.**
**Range : 1 to 255, or no parameter**

This command allows the user to examine a sequence that has previously been entered into the system. The sequence is listed on the display or terminal, one command entry per line. If no sequence number is given in the command, the system lists the numbers of all sequences which are currently defined. The sequence may be listed continuously, or the system can print one line at a time and wait for the user to press the return key before printing the next line. This is useful when using the system with a membrane keyboard or a portable terminal which only has a small number of display lines. This list pause facility is controlled by one of the flag bits in the DW command.

The escape key may be used to stop the LS command early.

**Example : `LS1`**
This lists sequence 1 on the display or terminal. The output for the sequence given above would look like this.

| System | User | Comments |
|---|---|---|
| `>` | `LS1<CR>` | User input to list sequence. |

```
S1: ID/IN-
S1: MA100/WT256/MA0/WT256/RP3
S1: MA2000
>
```

**Example : `LS`**
This lists all sequences that are currently defined.

| System | User | Comments |
|---|---|---|
| `>` | `LS<CR>` | List all sequences. |
| `S1` | | Sequence 1 is defined... |
| `S4` | | ...and sequence 4... |
| `S5` | | ...and sequence 5. |
| `>` | | |

**XSnn** **Execute sequence.**
**Range : 1 to 255**

This command tells the system to execute sequence number nn. The normal status messages for each part of the sequence are printed on the display as they are executed. The sequence aborts automatically if any error occurs. A sequence may be aborted manually by using the AX command.

The XS command saves the current channel, and returns to this channel when the sequence is finished. This allows easier nesting of sequences.

**Example : `XS3`**
The system executes stored sequence no. 3.

**RPnn** **Repeat command line.**
**Range : 1 to 255, or no value**

This command tells the system to repeat the sequence of commands on the current command line, up to the RP command, nn times. If no repeat count is given, the command line is repeated indefinitely. A repeat command with a specified repeat count of zero is ignored. If the repeat command is the first command in a command string, it has nothing to repeat and returns the “no commands before RP” error message. Only one repeat command is allowed on any command line. By using repeats within sequences, it is simple to set up a complete cycle of operations which can be started with one execute sequence command.

Note that the AX command may be used to break out of any repeat loop prematurely, and the ER end repeat command may be used to break out of the loop at the end of the current loop.

**Example : `MA2000/MA0/RP5`**
This moves the motor to position 2000 and then back to position 0, and repeats it five more times, giving a total of six operations.

**ER** **End repeat.**

This command allows the user to exit from a repeat loop cleanly, at the end of the current loop. This is in contrast to the AX command, which stops command processing immediately, in the middle of whatever action is taking place. It may be used in repeat loops with a repeat count, or in endless repeat loops. In either case, the loop terminates normally at the end of the command line.

When the ER command is executed, any commands following the original RP command are not executed. Commands following the ER command are executed when the repeat loop terminates. This allows a command line beginning with the ER command to override the current operation and neatly replace it with a new operation at the end of the repeat loop.

**AX[nn]** **Abort command execution.**

This command aborts execution of any command strings or sequences running on the current channel. It leaves the motor in its current state.

If the AX command is issued while the channel is in the mapping or constant velocity states, the motor is left in mapping or constant velocity as appropriate. If AX is issued while the channel is executing a move or waiting, the move or wait is finished normally but following commands are aborted.

If a sequence number is specified in the AX command, then command execution stops only when that particular sequence is executed. If it was called from another sequence, then the parent sequence is also aborted.

**Example :**

| System | User | Comments |
|---|---|---|
| `1>` | `VC+` | Execute constant velocity move |
| `1V` | `DP/WT50/RP` | Display position repeatedly |
| `1 DP1000` | | Position display |
| `1 DP1500` | | |
| `1 DP2000 ...` | | |
| `1V` | `AX` | Terminates position display |
| `1V` | | Channel is still moving |

 

**BK[nn]** **Break out of sequence.**

This command causes the system to break out of the current sequence and to continue executing commands in the calling sequence (if any). This is useful for terminating a sequence early depending on the state of an input line.

If a sequence number is specified in the BK command, then the break applies to only that particular sequence.

**Example :**

```
1> LS10
S10: CH1/II1-/BK
S10: CH1/IN+/CO3
1> XS10/MA1000
1>
```

In the above example, the initialization of channel 1 will only take place if input line 1 is positive. If output line 3 is fed back to input line 1, sequence 10 can be executed at any time but only causes initialization once after power-up.

**ASnn** **Set autostart sequence (restricted).**
**Range : 0 to 255**
**Default : 0**

This command is used to set up a command sequence to execute automatically when the system starts up, after all the saved setup parameters and configuration details are loaded from the nonvolatile memory. If no sequence number is given, the system prints the current autostart sequence number.

To disable the autostart sequence facility, set it to zero. If the sequence specified in the AS command is not defined, then the system simply does nothing at start-up.

 

**SNnn** **Set setup number (restricted).**
**Range : 0 to 255**
**Default : 0**

This command is used to store a setup number. The SN value is used as the sequence number for the XN command. It may also be indicated on any expanded output lines using the ON command.

The Registration Controller implements a number of different setups. The number of setups available is set to between 8 and 32 by the NS command. Each setup comprises the SB set bound, RF reference offset, and RJ deferred adjust position values on both master and slave axes, the MB map base value, and the SM scaled map value. SN values 1 to NS are used to access these setups, normally selected by incrementing the value of SN using the digital input lines. All other values of SN (0 and NS+1 to 255) use an additional (default) set of values.

**NSnn** **Set number of setups available (restricted).**
**Range : 8 to 32**
**Default : 8**

This command defines how many different setups are available on the Registration Controller for use with the SN command.

**XN** **Execute sequence SN.**

This command tells the system to execute the sequence given by the value of the SN parameter.

**ON** **Output Number SN.**

This command outputs the value of SN as a binary number on the current expanded output lines, if defined. It is normally used to indicate which of the eight ratio setups has been selected.

**LD** **List default sequences.**

This command lists all the default sequences (stored in the firmware eprom) to the screen, in the same format as the LA command. The LA command lists everything except the default sequences, to prevent them from being redefined by downloading a recorded LA file.

 

**SQn** **Enable default sequences.**
**Range : 0 to 1**
**Default : 1**

This command enables and disables the default sequences stored in the firmware eprom. If SQ is set to 1, the default sequences are available and may be executed or listed as normal. If SQ is set to 0, then the default sequences are inhibited. This feature is provided for use in applications that do not require the default sequences, to prevent confusion between the application sequences and the default sequences.

**FM** **Display free memory.**

This command displays information about the memory space available for sequences. It returns the amount of free space in the system memory, string memory and nvm areas as a number of bytes. The system keeps track of the amount of spare memory each time a sequence is entered or deleted.

**Example :**

| System | User | Comments |
|---|---|---|
| > | FM | |

```
Free memory space 2785 bytes
Free string space 512 bytes
Free nvm space 220 bytes
>
```

**CHn** **Change channel.**
**Range : 1 to 2, M or S**

This command allows the user to switch between motor channels. It may be used at any time.

If the CH command is used in a command string or sequence, then the system executes the commands given for each channel in strict sequence. The commands given for the second channel are not executed until the commands for the first channel have been completed.

The channel may also be specified as the current master or slave axis, by using the commands CHM (change to master axis) or CHS (change to slave axis). If one channel is mapping, the master axis is the other channel. Otherwise, the master axis is defined by the MX command.

If the system is executing a command string or sequence, then the CH command should not be used. If it is used to change channels, then the system continues executing its commands on the new channel, which is unlikely to be correct. Setting bit 6 of EW enables a new error check which tests for this and prevents changing channels while commands are executing.

**Example :** `CH1/MA1000/CH2/MA2000`
This shows an example of sequential command execution on two channels. Channel 2 does not start to move until channel 1 has finished moving.

**MXn** **Set master axis.**
**Range : 1 to 2**
**Default : 2**

This command allows the user to define the master axis. It is used with the CHM and CHS commands to allow sequences to be written in terms of master and slave axes, without specifying which is channel 1 or channel 2.

**GS** **Global stop.**

This command sends a stop command ST to both axes.

**GA** **Global abort.**

This command sends an abort command AB to both axes.

**GF** **Global motor off.**

This command sends a motor off MO command to both axes.

 

**MEnn** **Set motor off error sequence (restricted).**

**Range : 0 to 255**

**Default : 0**

This command sets up a sequence to execute when any motor off error occurs on the current channel. If no sequence number is given, the system prints the current motor off error sequence number for the current channel. To disable the motor error sequence on this channel, set ME to zero. Note that the motor off error sequence may include commands for any channel(s).

## 4.6 Map Commands

This section describes the commands to use the position mapping or “Software Gearbox” facilities.

The position mapping commands provide a mechanism for linking the slave axis to the master axis at a user-defined ratio. The master channel may itself be controlling a motor, or it may simply be monitoring the position of, for example, a line shaft to which other motors must be synchronized. The system allows a wide range of linear gear ratios to be defined between the master and slave axes.

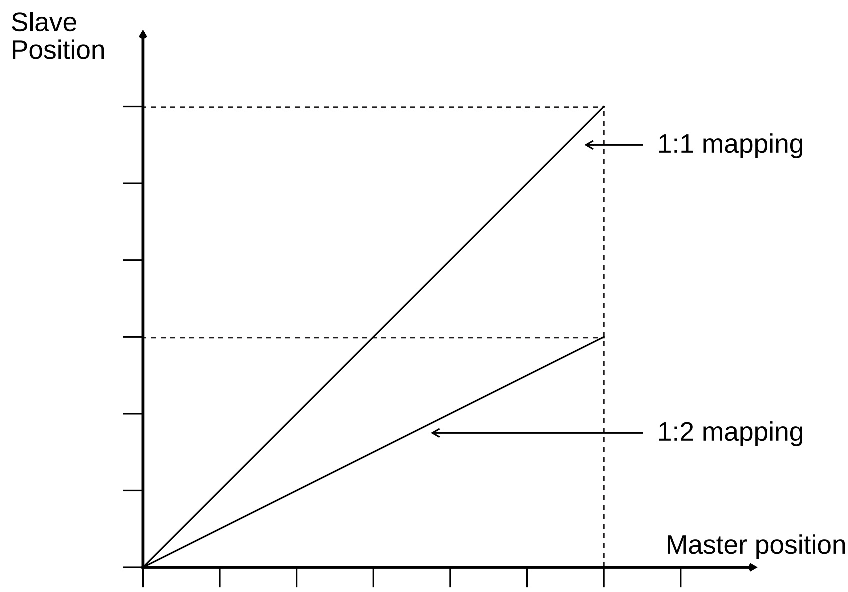


**Figure 11. Simple position maps.**

On a machine where either or both axes are cyclic, the position bounds on each axis are set to the cycle length for that axis.

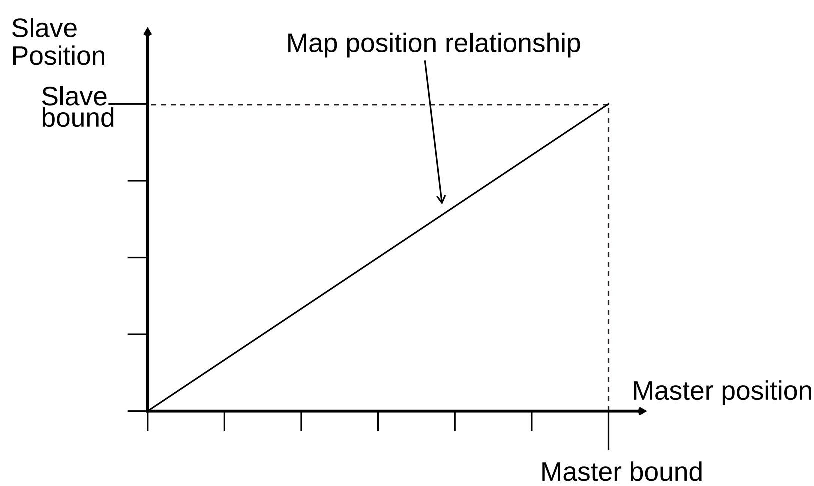


**Figure 12. Position map for a cyclic machine.**

 

The diagram above shows a system where the master and slave axes both complete one cycle in the same time, although they cover different distances, and the master and slave bound positions are coincident. It is not necessary for the master and slave bound positions to coincide, or for the slave axis bound position to repeat in the same interval as the master axis bound. In fact, in any linear ratio mapping other than the simple 1:1 case, the bound positions will not coincide and may not necessarily repeat at the same rate. This does not cause any problems in executing the map. It is also possible for the required slave position to go outside the slave bound value without any problems. In this case, the slave channel compensates automatically for its wraparound when it passes the bound position.

The position mapping between the master and slave channels may be modified by the map base and map offset commands. These appear to be similar, but have subtly different effects in practice. They shift the position map relationship along either the master position or the slave position axis. The map base value is subtracted from the master axis position before indexing into the map table. This has the effect of shifting the map curve to the right on the graph. The map offset value is added to the slave position value calculated in the mapping. This has the effect of moving the map curve up the graph. These two parameters allow any or all slave axes to shifted or rotated relative to the master axis, even while executing a mapping. An example of a similar situation is the ignition timing on a car engine; the timing adjustment involves rotating the distributor shaft relative to the crankshaft, so that the spark is generated earlier or later in the engine cycle.

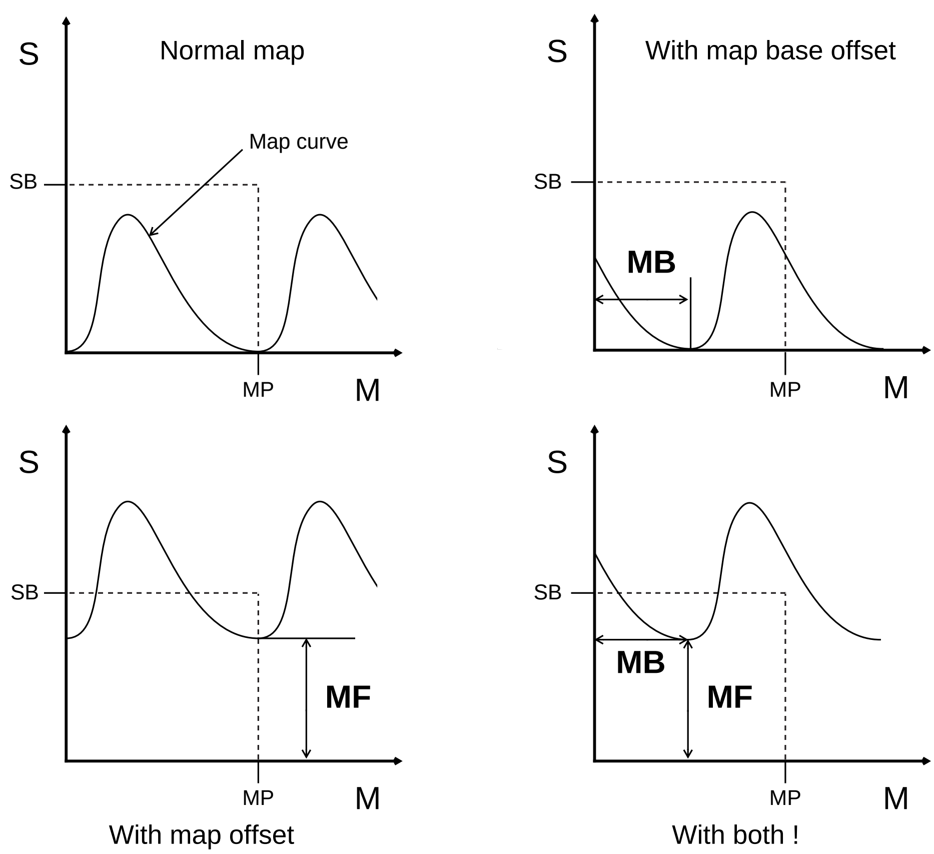


**Figure 13. Effects of map base and map offset.**

 

**XM** **Execute map.**

This command puts the currently selected axis into the position mapping mode, where its demand position is calculated according to the specified ratio from the other (master) channel's position. The ratio between master axis and slave axis positions is given by the SM parameter. While a channel is executing a position mapping, it gives the 'X' map mode prompt character. The stop or abort commands are used to exit from map mode.

When the slave axis executes a map, it has a choice of startup methods, as described here. If the master axis is stationary, then the slave axis can align itself with the required mapped position and then go into the mapped state (the alignment move). If the master axis is moving, then the slave axis can wait for the required mapped position to pass the current slave position, and then start following the mapping as it passes. This facility is called the Software Clutch, and is selected by bit 0 of MW, the map options word, which is described later. A third option is available, where the map offset value MF is adjusted automatically to maintain the position relationship between the master and slave axes; thus there is no alignment move and mapping starts immediately. This is selected by setting bits 0 and 2 of MW.

If the XM command is part of a string, the commands following XM are not executed until the slave axis has finished the alignment move or has synchronized with the master axis using the Software Clutch.

 

**MB±nn** **Set map base offset for master map positions.**
**Range : ± 4 000 000 (4.0E6)**
**Default : 0**

This command sets a map base value. This value is subtracted from the master axis position before using the position data on the slave axis. An alternative description is that it defines the base position of the mapped region on the master channel, such the slave channel is mapped onto the master channel position for the range (MB) to (MP+MB). Normally the mapping is defined over the range from zero to the master axis bound position MP. The MB parameter allows the slave channel to be advanced or retarded relative to the master axis.

**MF±nn** **Set slave map position offset.**
**Range : ± 4 000 000 (4.0E6)**
**Default : 0**

This command sets a map offset value. This value is added to the slave position obtained from the mapping. This allows the slave axis map profile to be rotated or shifted relative to the master axis.

Note that when speed mapping is enabled by setting MW bit 4, the MB and MF parameters have no effect on the initial relative positions of the master and slave axes. However, any changes in MB or MF while in speed mapping are applied to give the required change in position of the slave axis.

 

**SMn/n** **Scale mapping.**

| | | |
|---|---|---|
| **Range :** | **multiplier :** | **0 to 65535** |
| | **divisor :** | **1 to 65535** |
| | **overall ratio :** | **0/255 to 255/1** |
| **Default :** | **1/1** | |

This command is used to set a scale factor for mapping. The required absolute position on the slave channel, as defined by the mapping from the master position (or result of any software differential), is multiplied by the first parameter value and divided by the second value. This allows a wide range of scale factors to be realized, while keeping a simple integer ratio scale function. Note that changing the map scale factor while executing a map can give erratic results, because this may change the required slave axis demand position by a large amount. To avoid this problem AV can be used to smooth out the changes in SM.

The MB map base value scales in the same way, as it represents a shift on the master axis. The MF map offset value does not scale, as it represents a shift on the slave axis, measured in encoder counts.

**Example :** `SM1096/361`

This shows the map scale factor set to 1096 / 361. Thus the required slave position at any point is calculated by multiplying the result of the normal mapped position calculation (absolute slave demand position) by the factor 1096 / 361.

**BR** **Set map scale factor from bounds ratio.**

This command is used to automatically calculate the map scale factor (SM) on the slave axis from the ratio of the slave bounds to the master bounds. The scale factor calculated is equivalent to SB(slave)/SB(master). If one channel is mapping, the master axis is the other channel. Otherwise, the master axis is defined by the MX command.

**Example :**

| System | User | Comments |
|---|---|---|
| `1>` | `MX1` | Define master axis |
| `1>` | `SB4000` | Set master axis bounds |
| `1>` | `CH2` | Change to slave channel |
| `2>` | `SB5000` | Set slave bounds |
| `2>` | `BR` | Calculate scale factor |
| `2>` | `XM` | Execute map |

 

**AVn** **Set map base/offset/scale factor adjustment velocity**
**Range : 0 to 8**
**Default : 0**

This command sets the adjustment speed for any change in the map base, map offset or map scale factor values entered while the axis is executing a map. It is used to make large adjustments smoothly by spreading them over several time steps. If AV is set to zero, then any map base, offset or scale factor adjustment is performed immediately in one step.

If AV is not zero and the axis is executing a mapping, then the adjustment of **map base or offset** is limited to a set maximum speed, given by the sum of the adjustment velocity and the current (instantaneous) motor velocity. The adjustment velocity is a power of two fraction of the current motor speed, defined by the value of AV. This means that the adjustment speed scales automatically with the machine speed, such that the value of AV may be chosen for correct operation at full machine speed without causing unnecessarily quick adjustments at lower machine speeds.

At the maximum value of AV=8, the adjustment speed is equal to the current motor speed, and the adjustment is thus performed at twice the current motor speed. Each time the value of AV is reduced by one, the adjustment speed is halved, down to the minimum value of AV=1 when the adjustment speed is 1/128 times the current motor speed.

If AV is not zero and the axis is executing a mapping, then the change in **map scale factor** is applied at a rate defined by the value of AV. At the maximum value of AV=8, the adjustment is applied at a rate of $\Delta$SM per 4ms tick. Each time the value of AV is reduced by one, the rate of adjustment is halved, down to the minimum value of AV=1 when the adjustment is applied at a rate of $\Delta$SM/128 per 4ms tick.

If the module is not currently executing a map when a change is made to the map base, offset or scale factor, then the change takes place immediately. If the axis is in the middle of the alignment move at the start of execution of a map, then the adjustment is delayed until the alignment move is complete, so that there is no sudden change in required slave position when it reaches the end of the alignment move.

**MPnn** **Set master position bound, or map bound**
**Range : 1 to 4 000 000 (4.0E6)**
**Default : 4 000 000**

This command is used to set or display the master channel position bound value. If one channel is mapping, the master axis is the other channel. Otherwise, the master channel is defined by the MX command.

**NOTE :** It is essential for the slave channel to have this information so that the mapping is continuous at the point where the master position wraps around to zero, that is, the master bound position. On a rotary system, where both the master and slave axis positions wrap around at their bound positions, it is very important to make sure that the distance between bounds on the master corresponds with the distance between bounds on the slave axis. If this is not correct, then the slave axis will appear to drift relative to the master axis position, because of the mismatch on the bounds values on the master and slave axes. This is quite difficult to spot, as similar symptoms arise if there are problems with the reference input setup.

 

**MWbb** **Set map/move options word (restricted).**
**Range : 8 bit binary value.**
**Default : 0**

This command allows the user to modify the operation of absolute moves and position mapping in various ways. The value is entered as a binary number, with each bit controlling a different aspect of the position mapping. Leading zeros may be omitted. The bit functions for the map options word are described below.

Bit 0 This bit controls the behaviour of the system when a position mapping is executed.
When set to 0, the slave axis calculates its demand position as required by the mapping, and executes a normal trapezoidal move to align itself to that position before going into the mapped state.
When set to 1, the “Software Clutch” facility is enabled. This is used when it is required to lock a slave channel to a master axis which is already moving. In this case, the slave channel remains at its current position, until the calculated demand position from the mapping approaches the current slave position. The slave channel then ramps up to the required speed in such a way as to reach this speed at the correct mapped position. The time for this clutch acceleration ramp is defined by the CT clutch time command.
If bit 2 of MW is set as well as bit 0, then the XM command executes with an automatic offset adjustment. In this case, the map offset parameter MF is adjusted to maintain the current relative positions of the master and slave axes when the XM command is executed, and mapping starts immediately. No alignment move or clutch acceleration ramp is performed.

Bit 1 This bit is used when bit 0 of the map options word is zero, such that when a map is executed, it starts with an alignment move.
When set to 0, the target position for the alignment move is taken directly from the result of the mapping. The alignment move distance may be larger than the set bound value SB if required.
When set to 1, the target position for the alignment move is set to the **nearest** correct cyclic position within SB. If MW bit 2 is set to 0, then the move distance is limited to ±SB/2. If MW bit 2 is set to 1, then the direction for absolute moves is set by MW bit 3, and the maximum move distance is SB, in the set direction only.

 

Bit 2 If mapping starts with the alignment move, this bit specifies that it is constrained to move in only one direction.
When set to 0 it has no effect, and the map alignment move may be in either direction, depending on the current positions of both master and slave axes, and on the mapping.
When set to 1, the map alignment move is always in one direction, as set by bit 3 of the map options word (see below).
If bit 0 of MW is set as well as bit 2, it enables an automatic offset adjustment function. The MF value required to maintain the current position relationship between the master and slave axes is calculated at the start of the XM command, and there is no alignment move or clutch acceleration ramp.

Bit 3 This bit sets the direction of the map alignment move, if it is constrained by setting bit 2 of the map options word to 1.
When set to 0, the map alignment move is in the positive direction.
When set to 1, the map alignment move is in the negative direction.

Bit 4 This bit is used to execute a mapping as a speed mapping, where the slave speed is related to the master speed by the mapping, instead of relating the slave and master positions. This is useful in speed ratio control applications where the absolute position relationship between the master and slave channels is not important.
When set to 0, position mapping is executed.
When set to 1, speed mapping is executed.
If this bit is changed from a 1 to a 0 while mapping, the system changes from speed mapping to position mapping. The difference between the current demand position and that required by the position mapping is set into the MF parameter, and an MF increment is set up to return to the original MF value. This gives a smooth adjustment of the slave axis to the new mapped demand position, with the speed of the adjustment set by the AV parameter.

Bit 5 Reserved.

Bit 6 This bit controls an optional automatic setting of the bound value used on the slave axis while in mapping.
When set to 1, the value calculated on the slave axis as the mapped master position bound is copied temporarily to the slave axis position bound value **while in mapping**. This value is used for position wraparound, reference position comparisons, etc. in the same way as the normal position defined by the SB command. This facility is useful only when the slave axis cycle should correspond to the master axis cycle. It allows the system to automatically set up the slave axis bounds correctly for zero drift between axes, without any user interaction.
When set to 0, the bound value as set by SB is used as normally while in mapping.

Bit 7 Reserved.

**CTn** **Set clutch time.**
**Range :** **0 to 2560**
**Default :** **0**

This command sets the acceleration ramp time for the “Software Clutch” facility. The software clutch is enabled by bit 0 of MW, the map options word. It allows a slave channel to be mapped onto a master axis that is already moving.

When the software clutch is enabled, a slave axis does not go immediately into the mapped state on performing an execute map command. Instead, it goes into a holding state and waits for the projected master position to pass the projected slave position at the end of the slave acceleration ramp, as given by the clutch ramp time factor. This clutch time defines the number of 1/256 second (about 4 ms) time steps, or ticks, taken for the slave axis to accelerate from rest to the required mapped speed.

While in the map hold state, the slave channel calculates the projected master axis position at the end of the slave acceleration ramp, and from this calculates the required slave axis position **and** speed at the end of the ramp. It then compares the start position of the ramp with the current slave axis position, and starts the acceleration ramp when this start position passes the current slave axis position. In this way the slave channel accelerates from rest to reach the required mapped speed **at the correct mapped position** in the number of time steps set by the CT command.

This provides a very powerful and accurate method of bringing slave axes into and out of mapping with a master axis that is running continuously, while preserving the position relationship between master and slave axes. This is unlike a mechanical clutch, where the positional accuracy is dependent on the response time of the clutch mechanism, and the accuracy of the start/stop control signal to the clutch actuator.

**Example : `CT128`**

This example shows the clutch time factor set to 128 time steps. Thus the slave axis will ramp up to the required mapped speed in 128 ticks (0.5s), at the correct mapped absolute position.

 

**BAnn** **Set map base advance.**
**Range : 0 to 65535**
**Default : 0**

The map base advance is a mechanism for shifting the mapped position of a slave axis, relative to the master axis, by some amount dependent on the current master axis speed. The base advance is applied to the slave axis in the same way as the fixed MB map base offset value, and thus is defined as a shift along the master axis proportional to master axis speed. This command sets the scale factor between the measured master axis speed and the actual map base advance. The scaling of the base advance is given by the expression

$$\text{Map base advance} = (\text{master speed} / 256) \times (\text{BA} / 256)$$

where the master speed is measured in encoder counts per second. For example, with a measured master axis speed of 10,000 counts per second, a value for BA of 200 gives a map base advance of 30 encoder counts.

**BTn** **Set master speed averaging time constant.**
**Range : 0 to 8**
**Default : 0**

Master speed averaging is useful in speed mapping, and with the map base advance facility. When the master axis is also controlled by the system, the demand speed for the master axis is available to the slave, giving very smooth operation. When the master is not controlled, but is just an encoder fitted to some other part of the machine, the master speed can only be calculated from successive encoder positions. These positions are measured at 4ms intervals, and so the measured master speed is only accurate to 256 counts per second. In some applications, this may cause slight variations in the speed of the slave axis.

The BT command sets up an averaging mechanism on the slave axis, such that the number of samples of master speed doubles for each increment in the value of BT. At BT=0, no averaging is done, and the immediate measured master speed is used for the base advance calculations. At BT=8, $2^8$=256 samples are averaged over a period of 1 second, to give a speed measurement accurate to 1 count per second. The system keeps a running average of the master speed which is updated at each 4ms sample, so that the latest average speed is always available. This averaged master speed is used by the slave axis in speed mapping, allowing smoother operation in such applications.

Note that whenever the averaging time is changed, the current average value is reset to zero and the running average is restarted.

 

**GM** **Get mapped master axis bound position.**

This command allows the user to find the slave position which corresponds to the master axis bound position when executing a position mapping. This value is calculated internally by the slave channel when a map is executed, by feeding zero and the master position bound value into the mapping algorithm. The value is used by the slave channel when the master axis position wraps around to zero at the master axis bound position to keep the slave demand position continuous at this point. It is not normally used in programming the system, as it is used locally within the slave axis. However, it is sometimes useful to confirm that the mapping is behaving as expected by checking this value.

**GW** **Get wraparound offset value.**

This command allows the user to read the internal wrap offset value on a slave axis in mapping. This may be used as another confirmation that the mapping is executing correctly.

When the master axis position wraps around at the bound position, the position value passed to the slave channel changes by the master bound value. In order to keep the required slave demand position continuous either side of the master bound, an offset equal to the mapped master bound is then added or subtracted to the slave demand position. This value is called the wraparound offset. When the slave channel then wraps around at its bound position, the slave bound value is subtracted or added to the wraparound offset as required, again to keep the slave demand position continuous either side of the slave bound position.

The GW command reads back the current value of this internal wraparound offset value. In the simple case where the master and slave bound values are the same, it should be zero, +SB or –SB. If it has any values other than these, or if it has an upward or downward trend, then it is likely that there are problems with the map setup.

 

## 4.7 Wait Commands

The wait commands are most useful in command sequences. They allow the user to specify some condition that must be satisfied before the system executes the commands following the wait command. The system returns a ‘W’ status message to indicate that it is waiting.

**WTnn** **Wait for time.**
**Range : 0 to 65535**

This command tells the system to wait for the given time, before proceeding to the next command. The wait time is specified in units of 1/256 second (about 4ms).

**Example :** `MA2000/WT512/MA0`

This command sequence tells the system to move to position 2000, wait there for 2 seconds, and then move to position 0.

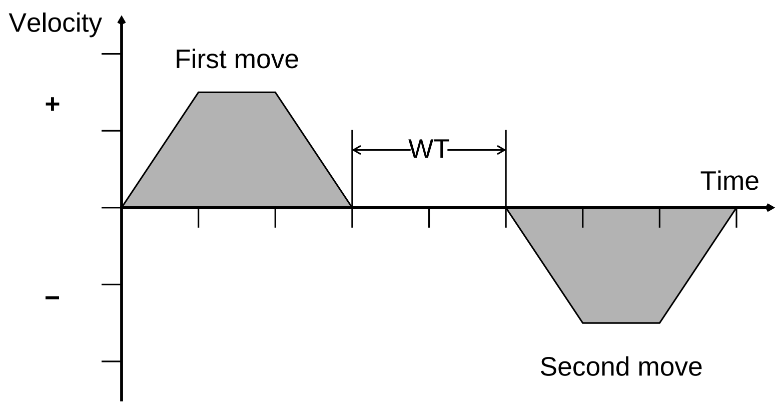


**Figure 14. Wait for time.**

 

**WIn±** **Wait for input line nn.**
**Range : 1 to 7**

This command tells the system to wait until the specified input line goes to the specified state. If the input is already in that state then the WI command terminates immediately. An input line that is defined for some other function may be used in a WI command.

**Example : `MA5000/WI2-/MA0`**

This sequence tells the system to move to position 5000 units, wait there until input line 2 goes to a logic low, and then move to position 0.

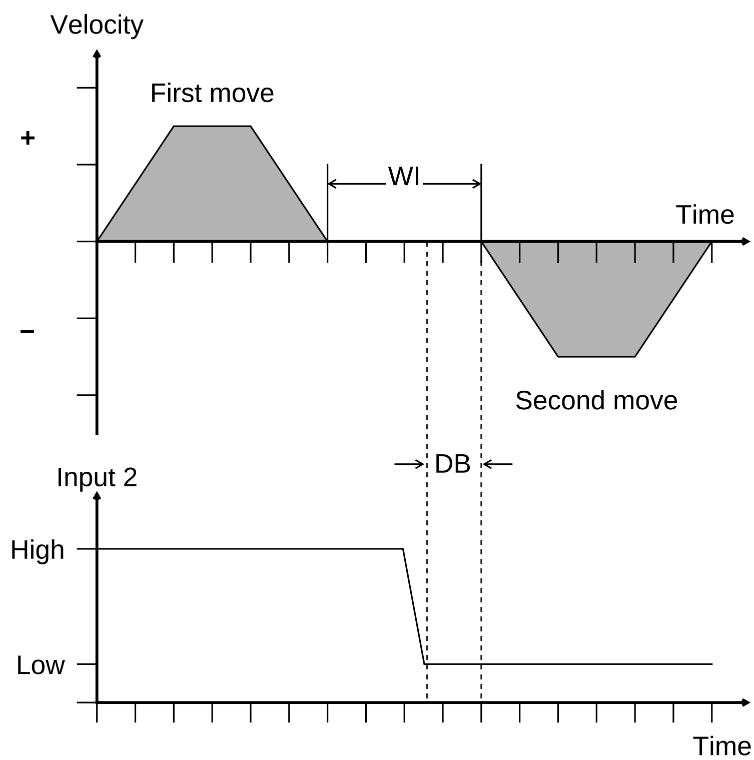


**Figure 15. Wait for input line.**

**WA±nn** **Wait for absolute position.**
**Range : ± 4 000 000 (4.0E6) encoder counts.**

This command tells the system to wait until it reaches the given absolute position before executing the next command. If the position specified in a wait for position command is outside the range of the previous move command, then the system gives a “parameter out of range” error message to indicate that the position was out of range.

**Example : `SV200/MA2000/WA1500/SV100`**

This sequence performs a move with a change of speed at a certain position. The velocity is initially set to 200 units per second. The motor begins a move to position 2000 at this velocity, and at position 1500 the velocity is changed to 100 units per second. The move is completed at the new velocity.

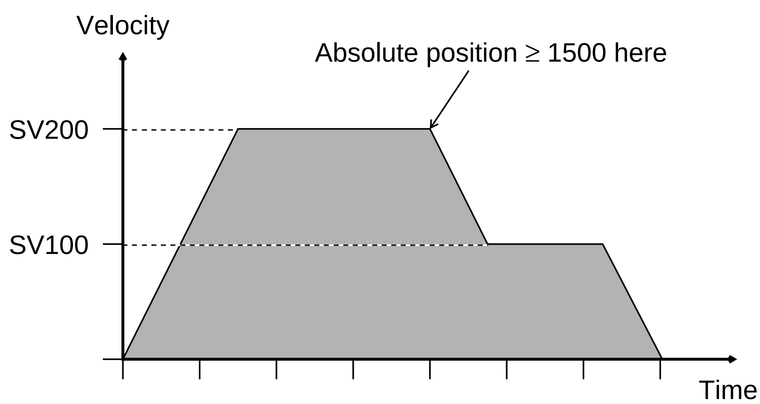


**Figure 16. Wait for absolute position.**

 

**WR±nn** **Wait for relative position.**
**Range : ± 8 000 000 (8.0E6) encoder counts.**

This command is similar to the WA command above. It tells the system to wait until it reaches the specified position relative to some previous position.

The wait relative position counter is reset to zero on the following events. If a WR command is executed, it will terminate at the given distance relative to whichever of these events ocurred last.

| | |
|---|---|
| Entering position control mode | (PC) |
| Setting the current position | (ZC) |
| Start of a move | (MA, MR or VC) |
| Start of mapping | (XM) |
| End of wait for reference | (WF) |
| End of wait for bound | (WB) |
| End of wait for bounds count | (WC) |
| End of wait for position | (WA, WR) |
| End of other wait commands | (WT, WI) |

**Example :** `VC+/WR5000/SV1000/WR2000/ST`

The system starts moving at constant velocity. It moves at the previously specified system velocity until it reaches 5000 units from the start position. At this point, the velocity is changed to 1000 units per second. This velocity is held for the next 2000 units, and then the motor decelerates to a stop.

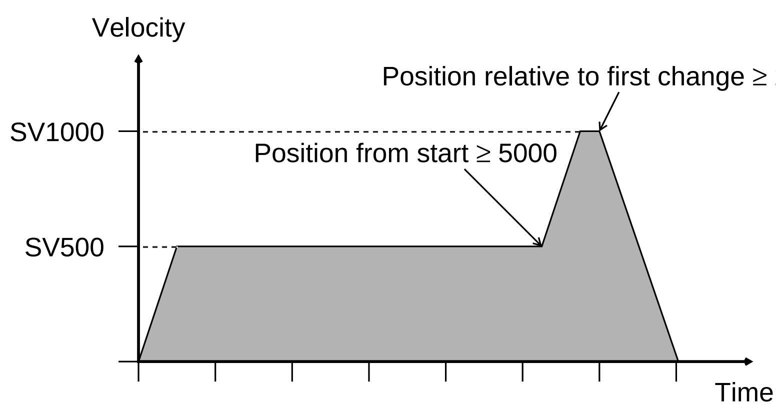


**Figure 17. Wait for relative position.**

**WF** **Wait for reference input.**

This command sets the system into the wait state, until a reference input is seen. It may be useful in sequences, to allow the reference action to be changed after detecting the first reference since the system was started.

If no reference input or marker input is defined, then the WF command returns the error message “no reference input defined”.

**Example :** **`RW1/SR0/RM1/WF/SR100/SO3`**
This command string sets the SR value to zero initially, such that the first reference is detected without limiting the reference error. It then waits for the first reference input to be detected, and changes SR to give a maximum reference error of 100 counts. Finally an output line is set to indicate that the unit has seen one reference and is ready.

**WB** **Wait for bound position.**

This command tells the system to wait until the motor passes the next bound position (positive or negative) before continuing with the command string.

**WC±nn** **Wait for bound overflow count.**
**Range : ± 2,000,000,000**

This command tells the system to wait until the bound overflow counter has changed by the specified count value before continuing with the command string. It may be used, for example, to wait for a given number of machine cycles to complete before stopping.

**WE** **End wait state.**

This command ends the current wait state as if it had completed normally. This allows the user to escape from a wait state early but to continue with commands following the wait command.

**Example :**

| System | User | Comments |
|---|---|---|
| `1>` | `DI1-/WE` | Escape from wait state on input line 1 going negative. |
| `1>` | `MA10000/WT1024/MA0` | |

In this example, channel 1 moves to position 10000, waits 4 seconds and then moves back to zero. However if input line 1 goes negative during the wait state, the WT command is terminated early and the motor moves back to zero immediately.

 

## 4.8 Error Trapping

The system continuously monitors various aspects of its performance, in order to detect a range of error conditions. Some errors are critical, in that they prevent the system from controlling the motor correctly, or they indicate some external failure such as an encoder wiring fault. Others may be more or less important depending on the application.

Critical errors are called “motor off” errors. If such an error is detected, the axis shuts down to the motor off state, with the motor enable relay switched off. This is the safest course of action for the system to take when these error conditions occur. An error message is output on the main serial port, to indicate which error has been detected. An error code is also shown on the hexadecimal LED display. The motor error conditions are as follows.

- Motor position error.
- Motor timeout.
- Limit switch input detected.
- Map position overflow.

The following error conditions may also be enabled as motor off errors, by setting bits in the error options word EW. When enabled, these also cause the axis to shut down to the motor off state. If not enabled as a motor off error, they are treated as user errors and just give an error message on the serial port. These optional motor off errors are as follows.

- Reference timeout.
- Reference outside limits.
- Reference correction overrun.

 

**SEnn** **Set maximum position error (restricted).**
**Range : 1 to 65535**
**Default : 800**

This command sets a maximum position error which is continuously monitored by the system. If the position error at any time exceeds this value, the system gives a “motor position error” message and enters the motor off state. The system must be returned to position control mode before any further motion commands are accepted by the system. See the mode commands section 4.2 for details of the MO motor off and PC position control commands. The value is defined in encoder counts.

**Example :** `SE500`
This sets the maximum position error to 500 counts.

**TOnn** **Set timeout (restricted).**
**Range : 1 to 65535**
**Default : 32**

This command sets a timeout value, in units of 1/256 seconds. When the system expects the motor to move and the encoder position does not change for a period that exceeds the timeout, then the system prints a “motor timeout” error message and goes to the motor off state. The system must be returned to position control mode before any further move commands are accepted.

**Example :** `TO512`
This sets the timeout to 2 seconds.

**RTnn** **Set reference timeout (restricted).**
**Range : 0 to 127**
**Default : 0**

This command sets up a timeout on the reference input. It is used when the system is set up for continuous monitoring of the reference input, to give a warning error message if the reference input is not detected. A counter is incremented each time the system passes a position half way between the expected reference positions, and cleared each time a valid reference input is detected. If the counter exceeds the RT value, the system gives the “reference timeout” error message. The reference timeout function is disabled by setting it to zero.

**MEnn** **Set motor off error sequence (restricted).**
**Range : 0 to 255**
**Default : 0**

This command sets up a sequence to execute when any motor off error occurs on the current channel. If no sequence number is given, the system prints the current motor off error sequence number for the current channel. To disable the motor error sequence on this channel, set ME to zero. Note that the motor off error sequence may include commands for any channel(s).

**EWbb** **Set error options word (restricted).**
**Range : 8 bit binary value.**
**Default : 0**

This command allows the user to write a value into the error options word for this channel. Note that the leading zeros may be omitted. The error word allows various user and motor error options to be turned on or off. The error word bit functions are described below.

Bit 0 When set to 1, the reference timeout error is treated as a motor error, and the system goes to the motor off state when a reference timeout occurs. When set to 0, the reference timeout error is treated as a user error, and the system simply prints an error message.

Bit 1 When set to 1, the reference limit error is treated as a motor error, and the system goes to the motor off state when it occurs.
When set to 0, the reference limit error is treated as a user error, and the system simply prints an error message.

Bit 2 When set to 1, the reference correction overrun error is treated as a motor error, and the system goes to the motor off state when it occurs.
When set to 0, the reference overrun error is treated as a user error, and the system simply prints an error message.

Bit 3 Reserved.

Bit 4 Reserved.

Bit 5 Reserved.

Bit 6 When set to 1, this bit enables a check on the CH channel change command. It prevents changing channels if there is a command string running on the current channel, and gives an error message “cannot change channels while busy”.
When set to 0, there is no check on the CH command.

Bit 7 When set to 1, suppress any reference error messages that are not defined as motor errors.
When set to 0, display all reference error messages.

 

**LE** **Display last error.**

This command redisplays the error message for the last error detected by the system. It is useful for finding an error message which has stopped the system when there is not normally a display connected to the machine, or to display the long error message for an error which has been reported with a short error message. This is done by setting bit 5 of DW before executing LE.

 

## 4.9 Gain Commands

The motor control system operates by sampling the position of the motor at regular intervals, and calculating a motor demand signal according to some control algorithm. The algorithm used is of the following form.

$$V_{out} = KPe_i + KI\sum e_i + KV(p_i - p_{i-1}) + KF(d_i - d_{i-1})$$

where KP = proportional gain constant
KI = integral gain constant
KV = velocity feedback gain constant
KF = velocity feed-forward gain constant
$e_i$ = position error (= demand position – measured position)
$d_i$ = demand position
$p_i$ = measured position

The dynamic behaviour of the system depends on these gain constants, and on the mechanical characteristics of the system being controlled. Tuning the control system to get best performance on a particular mechanical setup requires setting up these gain constants.

The actual scaling between position error and output voltage, for proportional gain only, is as follows:

$$V_{out} = Error \times \frac{KP}{256} \times \frac{10}{2048}$$

where KP is the proportional gain term, and Error is the position error, measured in encoder counts. The other control terms are similar.

The performance of either axis may be monitored by using the other channel’s analogue output as a monitor output. Commands are provided to output various signals on this channel for viewing on an oscilloscope or chart recorder. These are described at the end of this section. The scaling of the monitor output is similar to that of the main demand output, but uses the KM monitor output gain.

**KPnn** **Set proportional gain constant (restricted).**
**Range : 0 to 65535**
**Default : 256**

This command sets the proportional gain of the system. The proportional gain acts on the measured position error, which is calculated as the difference between the current demand position and the position measured by the encoder. High gain gives the system a faster response and tighter position control, but if the gain is too high the system may oscillate. For best results, the proportional gain should be set as high as possible without inducing severe overshoot or oscillation.

**KInn** **Set integral gain constant (restricted).**
**Range : 0 to 65535**
**Default : 0**

This command sets the gain for the integral term in the controller transfer function. When integral control is used, the system integrates the position error by adding the current error to a running total. Integral gain is useful to remove a constant position error, due to a steady load or friction, or in steady-state velocity control, but also tends to make the system overshoot the target position at the end of a move because of the error accumulated during the move. This problem is known as “wind-up”. The integral action may be set up to avoid this problem such that it is operative only when the system is static, by setting bit 7 of the control word to a 1.

**KVnn** **Set velocity feedback gain constant (restricted).**
**Range : 0 to 65535**
**Default : 0**

This command sets the velocity feedback gain constant. The system uses the measured position to calculate the motor velocity, and this velocity, scaled by KV, is used in the controller transfer function. Note that differential control uses the rate of change of error, while velocity feedback uses the rate of change of position. Adding velocity feedback is similar to the effect of a tachogenerator connected externally to the motor drive, in that it adds damping into the system. This allows higher values of proportional gain to be used without giving excessive overshoot or oscillation, thus improving the speed of response of the system.

**KFnn** **Set velocity feed-forward gain constant (restricted).**
**Range : 0 to 65535**
**Default : 0**

This command allows the user to set the gain for the velocity feed-forward term in the controller transfer function. It uses the demand velocity as opposed to the measured velocity, and is particularly useful when following a set position or velocity profile. If a system is using proportional gain only, then there will be a steady position error when running at constant velocity, known as velocity lag. The feed-forward gain has the effect of reducing the velocity lag by adding a component dependent on the demand velocity into the demand signal output. The velocity lag error may be easily reduced to zero or even made negative, by increasing the value of the feed-forward gain. Alternatively, velocity lag may be reduced to zero by the use of integral gain, but this has other effects as well.

**DK** **Display system constants.**

The system displays various parameter values, including the four main gain terms:

- Proportional gain constant KP
- Integral gain constant KI
- Velocity feedback gain constant KV
- Velocity feed-forward gain constant KF
- Velocity SV
- Acceleration SA
- Maximum position error SE

**ITn** **Set integration time constant (restricted).**
**Range : 0 to 2**
**Default : 1**

The position error is integrated with respect to time by adding the position error at each sample to a running total. This integral of error is then multiplied by the integral gain when required in the control algorithm. This command allows the time constant for the error integration to be set to three different values, as given in the table below. Note that the different time constants also give different scale factors on the integral gain; this means that the integral gain setting is only correct for one time constant setting.

| Code | Time constant | Scale factor |
|---|---|---|
| 0 | 1/256 | 256 |
| 1 | 1 | 1 |
| 2 | 256 | 1/256 |

The table indicates that with a short time constant, only small values of integral gain are usable without producing instability, because of the increased scale factor. Conversely, with a larger time constant, larger gain values may be used.

**OLnn** **Set analogue output limit (restricted).**
**Range : 0 to 2047**
**Default : 2047**

This command is used to set an upper limit on the absolute value of the demand signal output to the motor. Once set the limit is active at all times.

**Example : `OL1024`**
The analogue output is limited between ±5V.

 

**SFn** **Set monitor output function (restricted).**
**Range : 0 to 16**
**Default : 0**

This command selects a particular control value to output on the auxiliary analogue output channel. The possible monitor output functions and their associated commands where applicable are as follows:

| Code | Function | Associated command |
|---|---|---|
| 0 | No output function | |
| 1 | Demand velocity | KF |
| 2 | Measured velocity | KV |
| 3 | Position error | KP |
| 4 | Integral of error | KI |
| 5 | Velocity error | |
| 6 | Absolute demand position | DD |
| 7 | Absolute measured position | DP |
| 10 | Averaged measured speed | DV |
| 11 | Master speed | |
| 12 | Master averaged speed | |
| 14 | Reference error | DF |
| 15 | Snapshot position | DS |

The monitor signal may be viewed with a storage oscilloscope, or recorded on a chart or UV recorder. This allows the servo control loop to be easily monitored as an aid to tuning a system.

**KMnn** **Set monitor output gain (restricted).**
**Range : 0 to 65535**
**Default : 0**

This command sets the gain for the monitor output signal. The monitor output functions are scaled by the monitor gain, and not by the gains used in the control algorithm.

**OMnn** **Set monitor output offset (restricted).**
**Range : ± 32767**
**Default : 0**

This command allows the auxiliary monitor output to be offset by a fixed voltage.

**Example :** `SF2/KM100/OM25`
This selects the measured velocity function to be output on the monitor line, sets a gain of 100 and an offset of 25.

**AOn** **Set auxiliary output channel (restricted).**
**Range : 0 to 2**
**Default : 0**

This command allocates the monitor signal for the current channel to one of the two channel analogue outputs. The monitor function SF may be defined at any time. The auxiliary output channel may be set for a particular analogue output only when its channel is in the motor off or virtual motor modes. If the channel is in any other state then the analogue output is not available for use as a monitor output. Conversely, if the analogue output has been allocated to a channel as a monitor signal, then this channel cannot be taken out of motor off or virtual mode.

To return an analogue output channel to normal operation, use AO0 on the channel where the auxiliary output is defined. Note that it is also possible to have the auxiliary output signal allocated to the current channel when it is in virtual mode; the signal does not have to be defined on a different channel's output. This may be useful in open-loop control applications.

On the Registration Controller, only one channel at a time may have a defined auxiliary output for the monitor function. The SF, KM and OM commands may be used from either channel, and apply to the channel which has the auxiliary output defined.

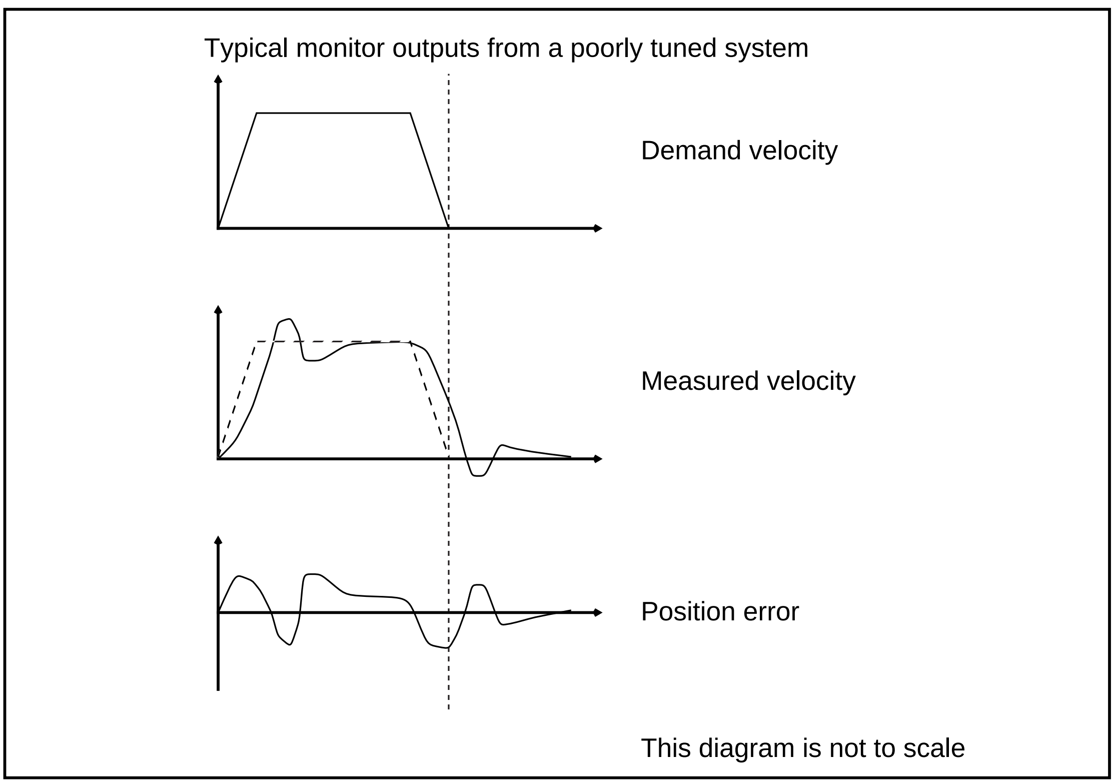


**Figure 18. Monitor output functions.**

 

## 4.10 Reference Commands

This section describes the commands available to make use of the position reference facilities. In particular, these commands allow the user to set up a repetitive position reference signal, and to use it to automatically adjust the absolute position of the system. The position of the encoder is immediately stored when the reference input signal is detected. This position is compared with an expected reference position, either the current zero position or the nearest bound position. The difference is defined as the reference error, and the absolute position may be corrected by this amount if required.

The system supports two types of reference input signal. The encoder marker signal is connected via the Z and /Z inputs to a dedicated fast reference input, called the **zero marker** input, which responds to signals down to a minimum pulse width of 60ns. This is fast enough to deal with the single marker pulse from a high resolution encoder running at high speed. This input is configured with the DZ command. In addition to this, reference inputs may be programmed on inputs 1 to 4, using the DR command. These additional reference inputs are intended for use with other devices such as proximity switches or microswitches, and only respond to signals having a minimum pulse width of at least 50µs, because of the internal operation of the particular hardware device used. Note that the fast reference or zero marker input is only used for the encoder marker signal, and cannot be programmed for any other uses, while inputs 1 to 4 may be programmed for any purpose.

**ZC[nn]** **Zero position counters or set position.**
**Range : ± 4 000 000 (4.0E6) encoder counts, or no value**

If a position value is given, the system sets the current demand position to the given (absolute) value. If no value is given, the current demand position is set to zero. The bounds counter (BC) is set to zero in either case. If a position value larger than the set bounds value (SB) is given, then the ZC position value is divided by SB. The demand position is set to the remainder, and BC is set to the quotient.

The ZC command may be used at any time, although the accuracy of the set position clearly depends on the actual speed of the motor.

**Example :** `MA–5000/ZC`
This moves the motor to absolute position –5000, and sets the position counters to zero at that position.

**Example :** `ZC1000`
This defines the current demand position as position 1000.

**SBnn** **Set position overflow bound (restricted).**
**Range : 1 to 4 000 000 (4.0E6) counts**
**Default : 4 000 000**

This command sets upper and lower bounds on the absolute position of the system. If the position of the motor exceeds the upper bound then the position bound value is subtracted from the current position. If the position goes below the lower bound, the bound value is added to the current position to keep the position within bounds. Note that this does **not** limit the range of any move commands, but only changes the value of the final position for moves outside the position bounds. This is illustrated by the example below. There is also a 32-bit position overflow counter which is incremented when the position passes the upper bound, and is decremented when the position passes the lower bound. This effectively provides a 32-bit high order extension to the absolute position. The overflow count may be displayed and reset to zero by the BC command.

The bound position defined by the SB command is normally used as the expected reference position when the system is set up to continuously monitor the reference inputs.

**Example :** `SB1000`

This sets the position bounds to ±1000 counts. A typical application of this is on a cyclic or rotary machine, where it is required to know the motor position to within one revolution of the motor only, but it is not necessary to distinguish between complete revolutions of the motor. If a move from zero to position 1500 is executed, the final displayed position value is 500. The motor has moved a total distance of 1500 counts as required, but the final position is the remainder when divided by the bound value. If a move from zero to position –1500 is executed, the final position value is –500. In this application, the position overflow counter represents the number of complete revolutions of the motor from the zero position to the current position, and the normal position value defines the position within one revolution.

 

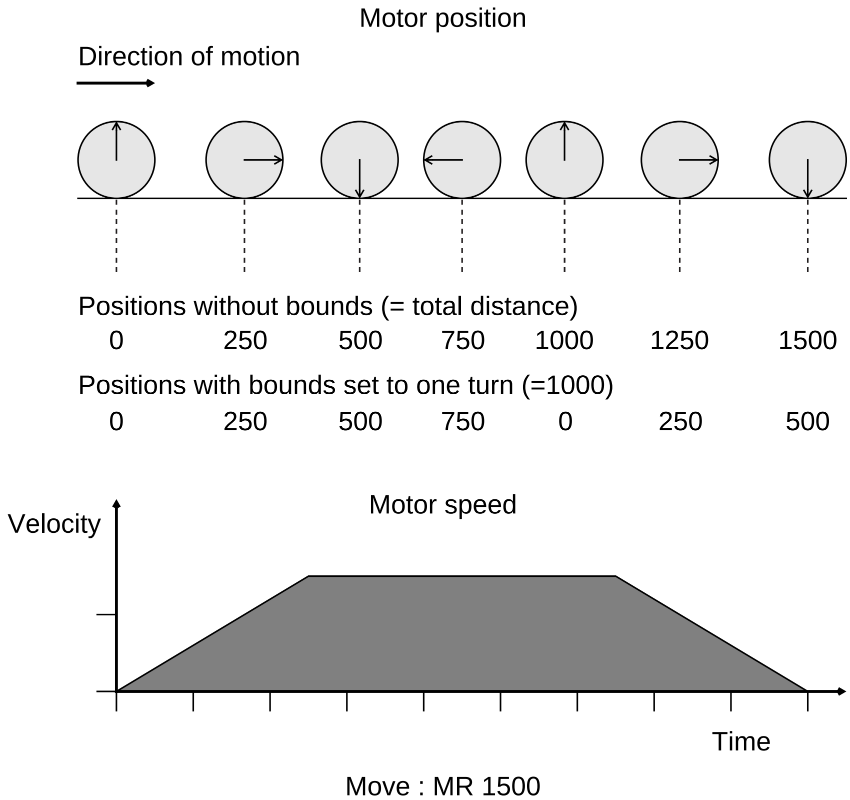


**Figure 19. Position bounds.**

**BCnn** **Set/display position overflow counter.**
**Range : ± 2147483647 (signed 32 bit)**

This command sets the position bound overflow counter to the specified value. The overflow counter is incremented when the position exceeds the upper bound, and is decremented when the position passes the lower bound. If no value is given, the current value of the bound overflow counter is displayed.

The BC value is also set when the ZC command is used. If no value is given with the ZC command, then BC is set to zero. If a position value is given, then it is divided by the set bounds value (SB) and BC is set to the result.

**DZn** **Define zero marker input on/off (restricted).**

**Range : 0 to 1**

**Default : 1**

This command defines whether the encoder zero marker input (the fast reference input) is on or off. The sense of this input is fixed and cannot be programmed. If the value passed with the DZ command is zero, the fast reference input is turned off. If the value is non-zero, the zero marker input is turned on. When the zero marker input is enabled, the system looks for a pulse on the zero marker input or a transition on any other reference inputs when the IN initialize position command is executed, and when the automatic reference functions are enabled by the RM and RW commands.

**DRn±** **Define reference input (restricted).**

**Range : 1 to 4**

This command defines the specified input line as a position reference input for the system. The sign defines which logic transition is used as the reference position. The system looks for the specified change in the reference input when the IN initialize position command is executed, and when the automatic reference functions are enabled by the RM and RW commands. Note that only inputs 1 to 4 may be defined as a reference input. This command is restricted, and is only available in privileged mode.

 

**RLnn** **Set reference repeat length (restricted).**
**Range : 0 to 4 000 000**
**Default : 0**

This command sets the reference repeat length. This is the position at which the channel expects to see the reference position signal. If RL is set to zero, then the system uses the bound position, set by SB, as the expected reference position. If RL is set to some value greater than zero, then it is used as the expected reference position instead of the bound value. When a reference signal is detected, the position is compared with the nearest multiple of the reference repeat length, instead of to the nearest zero or bound position. This allows the expected reference position to be set independently of the bound position. Note that if RW bit 7 is set to 1, then the RL value has no effect and the reference error is calculated relative to the zero position.

A typical example where this is useful is a leadscrew application, where the encoder is mounted on the motor and provides a marker signal every turn of the motor, while the bound value must be set larger than the total travel required by the motor. Using the RL command, the reference repeat length is set to the number of counts per turn of the motor, while the position bound is set as required by the linear motion. Each encoder marker signal detected then gives a useful reference error measurement which may be used for correction if required.

**RMn** **Set continuous reference mode on/off.**
**Range : 0 to 1**
**Default : 0**

This command enables and disables the fast reference input (if defined by the DZ command), and any reference inputs defined by the DR command. If RM is set to 1, all reference inputs are enabled. If it is set to zero, all reference inputs are disabled.

With the reference inputs enabled, the system measures the reference position error by comparing the position on seeing the reference input signal with zero or the nearest bound position. The system can then correct its position according to the reference error, when enabled by bit 0 of the reference options word RW.

 

**RWbb** **Set reference options word (restricted).**
**Range : 8 bit binary value.**
**Default : 0**

This command allows various reference functions to be enabled and disabled. The bit functions for the reference word are described below.

Bit 0 When set to 1, enables the position correction on detecting a reference signal, if the reference mode is enabled with the RM command.
When set to 0, disables the position correction but still allows the measurement of reference error.

Bit 1 This bit defines the action taken if the reference error is greater than the maximum value set by the SR command (when SR is not zero), provided that the FR parameter is zero. If FR is non-zero, then this bit has no effect.
When set to 0, a reference error greater than the maximum value is ignored completely and its value is discarded. This also inhibits the "reference out of limits" error message, and does not update the reference error value displayed by the DF command.
When set to 1, if the reference error is greater than the maximum value set by the SR command, the system corrects by this maximum value. The "reference out of limits" error is reported if LR is set to zero, and may set the channel to motor off if required.

Bit 2 When set to 0, the correction takes place immediately the reference signal is detected.
When set to 1, defers the position correction until the motor passes the adjustment position set by the RJ command.

Bit 3 When set to 0, the IN command finishes with a move back to the new zero position defined by the just detected reference input.
When set to 1, inhibits the move back to the new zero position in the IN command sequence.

Bit 4 When set to 0, any reference input is accepted as a valid zero position reference signal.
When set to 1, the system only accepts a valid reference on a combination of the defined reference inputs. The exact operation depends on the configuration of the fast reference or zero marker input. If the zero marker input is defined on with the DZ command, then any reference inputs defined on lines 1 to 4 are used to qualify the zero marker input. Thus a reference is only detected on the zero marker, and it is only accepted as a valid zero position signal if all other reference inputs are true at the same time. If the zero marker signal is defined off with the DZ command, then a valid reference is accepted when all reference inputs defined with the DR command are true, that is, the system performs a logical AND operation on the reference inputs to define a zero position signal.

 

Bit 5 When set to 1, the system only corrects the displayed position value, not the motor position.
When set to 0, it corrects the motor position as well as the displayed position.
**NOTE:** This bit has no effect on a slave channel in the execute map state; a mapped slave axis will always correct the position of the motor for any reference error when the correction is enabled. This is because the mapping relates the absolute position on the master axis to a required absolute position on the slave axis.

Bit 6 When set to 1, the reference holdoff function is enabled. This means that when a reference input is detected, no further reference signals are recognized until the debounce time has elapsed.
When set to 0, the reference holdoff function is disabled and all reference signals are processed in the normal way.

Bit 7 When set to 1, the system calculates the reference error with respect to the current zero position, and the RL command has no effect.
When set to 0, the system calculates the smallest possible reference error with respect to the reference repeat length defined by the RL parameter. If RL is set to zero, then the bound value SB is used. This is the normal setting for use on a cyclic machine where, for example, SB is set to the repeat distance between encoder marker positions.

**Example :** `RW1/SR20/RM1`

This enables the position correction on detection of a reference input, limits the allowed correction to a maximum of 20 encoder counts, and finally enables the reference inputs.

**Example :** `RW10100001`

This enables the reference input, and sets the system up to (a) only correct the displayed position, and (b) always set the position to zero on detecting a reference signal. In this setup the reference input has the same effect as a ZC command, but is effective on the fly at any time.

 

**SRnn** **Set maximum reference correction (restricted).**
**Range : 0 to 65535**
**Default : 0**

This command, when set to a non-zero value, limits the maximum allowed reference correction to the specified number of encoder counts. It may be used to eliminate false reference signals at positions far away from the expected reference position, or to allow the position reference facilities to be used even when the machine cycle length is not the same as the distance between reference marker signals.

When a reference signal is seen, the reference error is calculated as the difference between the zero position defined by the reference input, and the zero position or nearest bound position as measured by the normal system encoder counters. If enabled by bit 0 of RW and inside the limit defined by the SR command, the position is corrected by this reference error. If the reference error is greater than SR, then the action taken depends on bit 1 of RW and the FR parameter. If RW bit 1 is clear and FR is set to zero, then the position is not corrected, the out of limits reference error value is discarded, and the reference is ignored completely. If RW bit 1 is set, or FR is non-zero, the position is corrected by an amount equal to SR.

When the reference error is larger than SR, the LR parameter is zero, and correction up to the value of SR is enabled by RW bit 1 or FR non-zero, then a “reference out of limits” error message is given. If LR is non-zero, then this error is given for reference errors larger than LR. If required, the system may be programmed to generate a motor error when this error is detected. This facility depends on bit 1 of EW, the error options word. When EW bit 1 is set, a reference limit error sets the channel to motor off. Note that the reference error value and the out of limits error are only reported if the system has not already decided to ignore the reference error because it is outside the maximum limit.

**FRnn** **Set filter on reference error (restricted).**
**Range : 0 to 65535**
**Default : 0**

This command, when set to a non-zero value, defines a maximum value for the reference error in encoder counts. Any reference which gives a reference error value greater than FR is ignored completely. It is used to eliminate false reference signals at positions far away from the expected reference position. It is independent of the value of SR, which defines the maximum allowed reference correction.

**Example :** `FR500`

This sets the reference filter value to 500 counts. This means that any reference which gives a reference error greater than 500 counts is ignored.

 

**LRnn** **Set reference error limit (restricted).**
**Range : 0 to 65535**
**Default : 0**

This command, when set to a non-zero value, defines a limit value for the reference error in encoder counts. If a reference is detected and gives a reference error value greater than LR, then the “reference out of limits” error is reported. It is independent of the value of SR, which defines the maximum allowed reference correction, and of FR, the reference error filter value. When EW bit 1 is set, a reference limit error also sets the channel to motor off.

**Example : `LR800`**

This sets the reference filter value to 800 counts. This means that any reference which gives a reference error greater than 800 counts is reported as an error.

**RF±nn** **Set reference offset (restricted).**
**Range : ± 4 000 000**
**Default : 0**

This command sets the offset for the reference position. It defines the absolute position of the reference input signal.

**Example : `RF1000`**

This sets the reference offset to 1000 counts. This means that the position where the reference input signal is seen is defined as absolute position 1000, not zero.

 

**RVnn** **Set reference correction velocity (restricted).**
**Range : 0 to 8**
**Default : 0**

This command sets the correction speed for any reference error. It is used to make large reference error corrections less harsh by spreading the correction over several time steps. If RV is set to zero, then the reference error correction is performed immediately in one step. If RV is not zero, then the position correction is limited to a set maximum speed, given by the sum of the reference velocity and the current (instantaneous) motor velocity. The reference correction velocity is a power of two fraction of the current motor speed, defined by the value of RV. This means that the reference correction speed scales automatically with the machine speed, such that the value of RV may be chosen for correct operation at full machine speed without causing unnecessarily quick corrections at lower machine speeds.

At the maximum value of RV=8, the correction speed is equal to the current motor speed, and the correction is thus performed at twice the current motor speed. Each time RV is reduced by one, the correction speed is halved, down to the minimum value of RV=1 when the correction speed is 1/128 times the current motor speed.

If the reference correction velocity is set too small, or the reference error is too large, then it is possible for the next reference signal to arrive before the correction for the previous reference is complete. This condition is called reference correction overrun, and is indicated by the “reference correction overrun” error message. This error may be set to give either a user error or a motor error, by setting bit 2 of EW, the error word. If this error occurs, it indicates either that the machine is not performing correctly and is giving excessive reference errors, or that the value of RV is too small and should be increased.

 

**RJ±nn** **Set deferred reference adjustment position (restricted).**
**Range : 0 to 4 000 000 (4.0E6)**
**Default : 0**

This command allows the position correction on a reference input signal to be deferred until the motor passes a defined position. In some circumstances it may not be desirable to allow a sudden position correction to occur at the reference position, for example because of some mechanical interaction with other parts of a machine. In such a case, the RJ command defines a position which the motor must pass before the correction due to the reference signal is actioned. This function is enabled by bit 2 of RW. If this bit is set to zero, the reference correction takes place immediately.

This command may also be accessed with the SJ command, which is retained for compatibility with previous software versions.

**NOTE:** If the RJ position is set to a value which is greater than the bound (set by SB), or the reference repeat length if in use (set by RL), then the reference correction is deferred until after the next reference is detected and is not performed.

**RTnn** **Set reference timeout (restricted).**
**Range : 0 to 127**
**Default : 0**

This command sets up a timeout on the reference input. It is used when the system is set up for continuous monitoring of the reference input, to give a warning error message if the reference input is not detected. A counter is incremented each time the system passes a position half way between the expected reference positions, and cleared each time a valid reference input is detected. If the counter exceeds the RT value, the system gives the “reference timeout” error message. The reference timeout function is disabled by setting it to zero.

The RT reference timeout check is automatically disabled when an IB initialize bounds command is executed. This is to stop the system giving unnecessary reference timeout errors based on the previous set bound value, which may be smaller than the new bound length being measured.

**WF** **Wait for reference input.**

This command sets the system into the wait state, until a reference input is seen. It may be useful in sequences, to allow the reference action to be changed after detecting the first reference since the system was started.

If no reference input or marker input is defined, then the WF command returns the error message “no reference input defined”.

**DF** **Display reference error.**

Displays measured absolute position error relative to the last valid reference input, in encoder counts.

 

## 4.11 Digital Inputs and Outputs.

The Registration Controller has seven digital inputs and eight digital outputs. This section describes the commands available to read the inputs and control the outputs. They may be used immediately at the prompt, or in command strings and sequences.

**SO[n]** **Set output line n.**
**Range : 1 to 8, or no parameter**

This command sets the specified output line to a logic high. The output state is maintained until superseded by another command for the same output line. If the specified output is programmed for some defined function, then the “line already defined” error is reported.

If no output line number is specified, then all available outputs are set. No error is reported if all outputs are in use.

**Example : `SO3`**
This sets output line 3 to a logic high.

**CO[n]** **Clear output line n.**
**Range : 1 to 8, or no parameter**

This command clears the specified output line to a logic low. The output state is maintained until superseded by another command for the same output line. If the specified output is programmed for some defined function, then the “line already defined” error is reported.

If no output line number is specified, then all available outputs are cleared. No error is reported if all outputs are in use.

**Example : `CO7`**
This clears line 7 to a logic low.

**PUn±/tt** **Pulse output line n for tt ticks.**
**Range : line number 1 to 8,**
**pulse time 0 to 65535**

This command pulses the specified output line to the given state for the specified time (in ticks), then returns it to the opposite state. A time value of 0 has the same effect as a SO or CO command for that line.

The pulse output time is specified in units of 1/256 second (about 4ms).

**Example : `PU4-/256`**
This pulses output 4 low for 1 second, then sets it high.

**RI[n]** **Read input line(s).**
**Range : 1 to 7, or no parameter**

This command reads the current state of the specified input line and prints it as a ‘0’ or ‘1’ on the display. A ‘0’ represents a logic low, and a ‘1’ represents a logic high. If no line number is given, then the system prints the current state of all 7 input lines and a letter to show whether each line is masked (M), inhibited (B) or enabled (E).

**Example : `RI2`**
This reads the state of input line 2, and prints it on the display.

| System | User | Comments |
|---|---|---|
| `>` | `RI2` | Read a particular input |
| `0` | | Input line 2 is low |
| `>` | | Normal prompt |

**Example : `RI`**
This reads the states of all inputs, and prints them, together with their mask/enable status.

| System | User | Comments |
|---|---|---|
| `>` | `RI` | Read all inputs |
| `1 2 3 4 5 6 7` | | Line numbers |
| `0 1 1 0 0 1 0` | | Lines 2, 3, 6 are high |
| `E E M E M E E` | | Lines 3 & 5 are masked |
| `>` | | Normal prompt |

**RO[n]** **Read output line state(s).**
**Range : 1 to 8, or no parameter**

This command reads the current state of the specified output line and prints it as a ‘0’ or ‘1’ on the display. A ‘0’ represents a logic low, and a ‘1’ represents a logic high. If no line number is given, then the system prints the current state of all 8 output lines.

**Example : `RO`**
This reads the states of all outputs, and prints them.

| System | User | Comments |
|---|---|---|
| `>` | `RO` | Read all outputs |
| `1 2 3 4 5 6 7 8` | | Line numbers |
| `0 1 1 0 0 0 1 0` | | Lines 2, 3, 7 are high |
| `>` | | Normal prompt |

 

**OCnn** **Output code via expanded outputs.**
**Range : 0 to maximum value possible on expanded outputs.**

This command sets the expanded output lines (defined with the OX command) to the given code data value. It allows a number of output lines to be set or cleared at the same time, instead of using a string of separate SO and CO commands. If the expanded outputs were defined as active low, the data is inverted.

If this command is used when there are no expanded outputs defined, the “no output group defined” error message is returned. If the parameter value given cannot be represented as a binary number with the number of lines defined as expanded outputs, then the “parameter out of range” error message is returned.

**Example : `OC5`**
This sets the expanded output lines to the binary value 0101 (=$5_{10}$).

**ON** **Output Number SN.**

This command outputs the value of SN as a binary number on the current expanded output lines, if defined. If the SN value is larger than may be represented with the available output lines in the expanded outputs, then the value is ANDed with the OX line mask, to keep only the lower bits of the SN number. It is normally used to indicate which of the eight ratio setups has been selected.

**ODn±** **Output Direction.**
**Range : 0 to 8**

This command outputs the current direction, as set by the DN parameter, as a signal level on the specified output line. It allows the current direction to be indicated on an output line without knowing its current value. This is useful particularly in the autostart sequence, to indicate the saved DN value. The sign allows the output line to be used as either an active high or active low signal.

If the current value for DN is the same as the sign given with the OD command, then the specified output line is set, in the same way as the SO command. If the DN value is opposite to the sign given, then the specified output line is cleared, in the same way as the CO command.

**IIn±** **If input true do command line.**
**Range : 1 to 7**

This command allows the programmer to specify that a command or command line is conditional on the current state of an input line. If the input line specified in the II command is in the specified state (the condition is true) then the remainder of the command line is executed. If the input line is not in the specified state, the remainder of the command line is skipped, and execution proceeds to the next line of input, either the next line of a sequence, or new input commands.

**Example : `II2–/MR–1000`**

This example shows a conditional relative move. If input line 2 is low, then the motor moves relative –1000 counts.

**IOn±** **If output true do command line.**
**Range : 1 to 8**

This command allows the programmer to specify that a command or command line is conditional on the current state of an output line. If the output line specified in the IO command is in the specified state (the condition is true) then the remainder of the command line is executed. If the output line is not in the specified state, the remainder of the command line is skipped, and execution proceeds to the next line of input, either the next line of a sequence, or new input commands.

**Example : `IO3+/XS25`**

This example shows a conditional sequence. If output line 3 is high, then the system executes sequence 25.

 

**MI[n]** **Mask function input.**
**Range : 1 to 7, or no parameter**

This command is used to mask the action of defined function inputs or any expanded input lines. It allows several input lines to selectively mask out defined actions, depending on the current function activated. For example, a machine start sequence assigned to a function input may mask itself once the machine has started, until the stop sequence assigned to another input re-enables it. This prevents any subsequent signal on the start input from generating unnecessary start sequence commands, which may not be allowed when the machine is running.

Masked inputs are enabled again by the EI command. If a DI input line changes state while it is masked, then the function assigned to the change of state executes when the line is enabled. If the line changes state twice and returns to the same state as when it was masked, then nothing executes when it is enabled again.

If a line number is given as a parameter, then the specified line is masked. If no line number is given, then all function and expanded inputs are masked.

**BI[n]** **Inhibit function input.**
**Range : 1 to 7, or no parameter**

This command is used to disable the action of defined function inputs or any expanded input lines. It is similar to the MI mask input function, but with the difference that **all** input state changes are ignored on inhibited inputs. Inhibited inputs are enabled again by the EI command.

If a line number is given as a parameter, then the specified line is inhibited. If no line number is given, then all function and expanded inputs are inhibited.

**EI[n]** **Enable function input.**
**Range : 1 to 7, or no parameter**

This command is used to enable the action of defined function inputs or any expanded input lines, where they have been masked by the MI command or inhibited by the BI command. If a line number is given as a parameter, then the specified line is enabled. If no line number is given, then all function and expanded inputs are enabled.

If a DI input line changes state while it is masked, then the function assigned to the change of state executes when the line is enabled. If the line changes state twice while it is masked and returns to its original state, then nothing executes when it is enabled. If an input is inhibited with the BI command, then all changes of state are ignored while the input line is inhibited.

**NOTE:** The MI, BI and EI commands may be used on any input lines. The mask/inhibit/enable action applies **only** to inputs defined as function inputs with the DI command, or as expanded inputs with the DX command. It is not possible to mask or inhibit other types of defined inputs with the MI or BI commands, or to mask or inhibit the WI command.

**WIn±** **Wait for input line nn.**
**Range : 1 to 7**

This command tells the system to wait until the specified input line goes to the specified state. An input line that is defined for some other function may be used in a WI command.

## 4.12 Configuration Commands

**DZn** **Define zero marker input on/off (restricted).**
**Range : 0 to 1**
**Default : 1**

This command defines whether the encoder zero marker input (the fast reference input) is on or off. The sense of this input is fixed and cannot be programmed. If the value passed with the DZ command is zero, the fast reference input is turned off. If the value is non-zero, the zero marker input is turned on. When the zero marker input is enabled, the system looks for a pulse on the zero marker input or a transition on any other reference inputs when the IN initialize position command is executed, and when the automatic reference functions are enabled by the RM and RW commands. For more details of the operation of the reference inputs, please refer to section 4.10, Reference Commands, on page 71.

**DRn±** **Define reference input (restricted).**
**Range : 1 to 4**

This command defines the specified input line as a position reference input for the system. The sign defines which logic transition is used as the reference position. The system looks for the specified change in the reference input when the IN initialize position command is executed, and when the automatic reference functions are enabled by the RM and RW commands. See section 4.10, Reference Commands, on page 71 for more details on the reference facilities and commands. Note that only inputs 1 to 4 may be defined as reference inputs. This command is restricted, and is only available in privileged mode.

**Example : `DR1–`**
This specifies that the reference position is detected on a high-to-low transition on input line 1.

 

**DLn±** **Define limit switch input (restricted).**
**Range : 1 to 7**

This command defines the specified input line as a limit switch input. The sign defines which logic state represents the out-of-limit condition. When the line goes to the specified state, the system stops the motor immediately, prints a “limit switch detected” error message, and goes to the motor off state. A line which has been defined as a limit switch input may be returned to normal operation by entering this command without the sign. This command is restricted, and is only available in privileged mode.

**Example : `DL7–`**

This defines input line 7 as an active low limit switch input. The system detects a limit switch when line 7 goes to a logic low.

**Example : `DL5`**

This returns line 5, previously defined as a limit switch input, to normal operation.

**PSn±** **Define position snapshot input (restricted).**
**Range : 1 to 4**

This command defines the specified input line as a position snapshot input for the system. The sign defines which logic transition is used to detect the snapshot position. The system monitors the snapshot input and stores the absolute position value at that time. The snapshot position data may be read at any time by using the DS command. The snapshot function uses the same mechanism as the reference input function to get an accurate measurement of position on an input signal. Note that because of this, only inputs 1 to 4 may be defined as snapshot inputs. A position snapshot input line may be returned to normal operation by entering this command without the sign. This command is restricted, and is only available in privileged mode.

**Example : `PS4+`**

This specifies that the snapshot position is detected on a low-to-high transition on input line 4.

**DIn±/...** **Define function input (restricted).**
**Range : 1 to 7**

This command defines a specified input line to have the given function. The sign specifies the active state of the input, such that the system executes the function when the input changes to the specified state. The command function may be a single command, or may be a sequence of commands. The text of the command function is separated from the DI command by any delimiter character.

It is possible to define a different function on each state of the input line. For example, by connecting the input line to a push button and defining a move function on the high state and a stop function on the low state, the axis can be made to move when the button is pressed and to stop when the button is released.

Each axis controller reserves a fixed memory area for function input definitions. Functions may be defined until this memory area is full. If there is not enough memory for the new line definition, the system returns the "memory full" error message. A function input line may be returned to normal operation by entering this command without the sign and the function text. To delete a function defined on one state of the line but leave the other function intact, enter the command with a sign but without the function text. In this case the DI command must appear at the end of a line. This command is restricted, and is only available in privileged mode.

If an input line is already in the true state when it is defined as a function input, the system does not act on the input until it has gone false and become true again. If an input is currently masked by the MI command when it is defined, it does not become active until it is enabled by the EI command.

**Example : `DI2+/AB`**
This defines input line 2 as a single command, such that when line 2 goes to a logic high, the system executes the AB command.

**Example : `DI2–/IN–/WT256/MR–5000/ZC`**
This defines input line 2 as a command string, such that when it goes to a logic low, the system executes the given string of commands. It initializes the motor to the reference position, waits for 1 second, moves –5000 units, and zeros the position counters at this position.

**Example : `DI2–`**
This removes the function defined on input line 2 going low. Any command defined on this input line going high is unchanged.

 

**DXn±** **Define expanded input lines (restricted).**
**Range : 0 to 6**

This command sets up a block of input lines as an expanded input command facility. It operates in a similar way to the DI function input lines, but allows a larger range of different functions to be programmed into the system. When the DX function is used, input line 7 is used as a strobe or trigger input. The sense of the strobe input is given by the sign after the parameter. Lines 7 downwards are used for the expanded input function to avoid clashes with any requirement for position reference inputs, which are only available on inputs 1-4. To reset the expanded input definition, use “DX 0” or “DX n” without the sign.

On detecting the strobe input, the remaining input lines from line 6 (most significant bit) down to the line number specified in the DX command are read as a binary code. On the Registration Controller this value is used as the number of a sequence, which is executed immediately if it is defined. This allows a maximum of 63 possible different operations to be controlled by the 7 input lines. If the strobe input is active low, as specified by the sign in the command, then the data lines are also inverted when deriving the number of the sequence to be executed. This keeps the strobe input and the data inputs consistent.

The MI mask input command may be used to disable any of the expanded input lines. If the strobe input is masked, no DX functions are executed until it is enabled by the EI command. If any DX data input lines are disabled, those input bits are masked out when deriving the DX number for the signal or sequence.

**Example : `DX3–`**

This sets up an active low expanded input group on lines 3 upwards. When a strobe signal is detected on line 7, the unit looks for the sequence number derived from the other input lines in the group, and if it is defined then it is executed. The sequence number is derived by taking the input lines in the group as a binary number, and takes account of the active sense of the group as defined by the sign in the DX command. Thus if the input lines 6-3 are in the binary pattern 0111 when the strobe input is seen, then sequence 7 is executed ($0111_2 = 7_{10}$).

| Line | | State | Bit Value | Decimal Value |
|---|---|---|---|---|
| 7 | ● | | | |
| 6 | ● | High | 0 | 0 |
| 5 | ● | Low | 1 | 4 |
| 4 | ● | Low | 1 | 2 |
| 3 | ● | Low | 1 | 1 |
| 2 | ● | | | Total 7 |
| 1 | ● | | | |

**Figure 20. Expanded input group.**

**MGbb** **Define input mask group (restricted).**
**Range : 8 bit binary value.**
**Default : 0**

This command specifies a set of input lines, such that if one line in the set goes active, all the input lines in the set are immediately masked to prevent them acting. The lines remain masked until they are explicitly enabled. The set of lines is specified by a binary parameter where a bit set to 1 includes the corresponding input line in the group. Bit 0 of the parameter corresponds to input line 1.

The input mask group should only include inputs defined as function inputs with the DI command or as expanded group inputs with the DX command. The MG command has no effect on other types of input line.

**Example :**

| System | User | Comments |
|---|---|---|
| `1>` | `DI1-/ST/EI2` | Execute stop on input line 1 |
| `1>` | `DI2-/VC+/EI1` | Execute move on input line 2 |
| `1>` | `MG00000011` | Place input lines 1 & 2 in mask group |

In the above example, the input line mask group is defined to include input lines 1 and 2. When input line 1 goes low, inputs 1 and 2 are both masked, the axis stops and then input 2 is enabled. When input line 2 goes low, inputs 1 and 2 are both masked, the motor starts, and input 1 is enabled. This makes sure that only one of the start or stop functions is enabled at one time, and that when one function is triggered, it is masked and the other function is enabled.

**BGbb** **Define input inhibit group (restricted).**
**Range : 8 bit binary value.**
**Default : 0**

This command specifies a set of input lines, such that if one line in the set goes active, all the input lines in the set are immediately inhibited to prevent them acting. The lines remain inhibited until they are explicitly enabled. This is similar to the action of the MG parameter, but note that the inputs are inhibited instead of masked. Refer to the description of the MI, BI and EI commands for more details.

**DEn±** **Define error output (restricted).**
**Range : 1 to 8**

This command defines the specified output line as an error output. The line is set to the specified state when the system detects any motor off error condition, and is cleared to the opposite state when the axis is returned to the position control state with the PC command.

More details of the error conditions and commands to set them up are given in section 4.8. This command is restricted, and is only available in privileged mode.

**Example : `DE6+`**

This sets up an active high error output signal on output line 6. When an error condition is detected, the error output is set to a logic high.

**POn±** **Define position trigger output (restricted).**
**Range : line number 1 to 8**
**positions ±4 000 000**

This command defines the specified output line as a position trigger output. If the PO command is given with a sign, it **must** be followed by two position values. These define the range of positions, within which the output line goes to the state specified by the sign in the command. A line which has been defined as a position trigger output may be returned to normal operation by entering this command without the sign. This command is restricted, and may only be used in privileged mode.

The position range for the PO position trigger outputs may be subject to a phase advance proportional to average measured speed. This is set by the PA phase advance parameter. If the PA value is non-zero, then the position trigger outputs occur at an earlier position than that defined. The amount of position shift is given by the value of PA and the average measured speed. Refer to section 4.13 for more details. Note that for correct operation, speed averaging with the VT command may also be required.

**Example : `PO5–/1000/2000`**

This example defines output line 5 as a position trigger output, such that it goes low between positions 1000 and 2000.

The position range for the PO outputs is cyclic and repeats at the bound position defined by the SB command. To illustrate this, consider the above example again. Suppose the bound position is set to 2000. In this case, the active position range for the position trigger output repeats at positions 3000 to 4000 (one cycle later), at 5000 to 6000 (two cycles later), and so on. It also repeats in the negative direction, at –1000 to 0, at –3000 to –2000, etc.

 

**OXn±** **Define expanded output lines (restricted).**
**Range : 0 to 8**

This command sets up a block of output lines that may be set to a binary code with a single OC command, instead of using a string of individual SO and CO commands. It reserves from line number 1 up to the line number given for the expanded outputs, and the sign determines whether or not the output data should be inverted.

If any of the outputs required for the expanded output function are already defined as some other function, the error message “line already defined” is returned. To reset the expanded output definition, use “OX0”.

**Example : `OX3+`**

This example defines output lines 1-3 as active high expanded outputs. This allows output codes from 0 to 7 to be put on these three output lines with the OC command.

**BOn±** **Define bound overflow output (restricted).**
**Range : 1 to 8**

This command defines the specified output line as a position bound overflow output. Each time the system passes the position bound set by the SB command, a logic high or low pulse is output on the specified output line. The sense of the pulse is defined by the sign given in the command. The output pulse lasts for a minimum of 4ms. A line which has been defined as a bound overflow output may be returned to normal operation by entering this command without the sign. This command is restricted, and may only be used in privileged mode.

**Example : `BO7+`**

This example defines output line 7 as a position bound overflow output signal. Each time the system passes the position bound, a logic high pulse is output on line 7.

**RRn±** **Define reference reject output (restricted).**
**Range : 1 to 8**

This command defines the specified output line as a reference reject signal output. The RR output is set true if any reference error occurs, and is cleared when a valid reference signal is detected. The sense of the output is defined by the sign given in the command. The output state is held until the next valid reference is detected. A line which has been defined as a reference reject output may be returned to normal operation by entering this command without the sign. This command is restricted, and may only be used in privileged mode.

A typical application of the RR command is for a simple product reject facility. If the reference input is triggered by the leading edge of a product on a conveyor belt, detected by a photocell or proximity switch, then the RR output together with the SR command indicates when a product is out of position by more than the SR value. This signal may then be used to trigger a product reject actuator if required.

**Example : `RR5+`**

This example defines output line 5 as a reference reject output signal. If any reference error occurs, then output line 5 is set high. It stays high until the next valid reference is detected, when it is reset low.

 

**PJn±/dd/tt** **Define product reject output (restricted).**

| | | |
|---|---|---|
| **Range :** | **line number** | **1 to 8** |
| | **delay distance** | **0 to 4 000 000** |
| | **pulse time** | **0 to 65535** |

This command defines the specified output line as a product reject output. The PJ command requires three parameters: the output line number and sign, a delay distance and an output pulse time. A line which has been defined as a product reject output may be returned to normal operation by entering this command with the line number only. Only one product reject output line may be defined. This command is restricted, and may only be used in privileged mode.

The product reject facility operates as follows. A position range is defined using the PR command below. If a position snapshot input is detected within the PR position range, then the product reject output is triggered. The reject output pulse starts when the motor has moved by the delay distance beyond the detected snapshot position. The output then gives a pulse for the specified time in units of 1/256 second (about 4ms), with the output state given by the sign in the PJ command.

The delay distance allows for the difference between the measurement position and the reject actuator position on the machine. The delay distance is optionally subject to a phase advance effect, where it is reduced by some amount proportional to machine speed. This allows for any fixed delay time in the reject actuator mechanism. This reduction in delay distance with machine speed is set by the PA phase advance parameter. Note that for smooth operation, speed averaging with the VT command may also be required.

**PRll/hh** **Define position range for product reject (restricted).**
**Range :** **±4 000 000**

This command defines the product reject position range. A product reject output is triggered, if one is defined, when a snapshot input is detected inside this position range.

**Example :**
```
PS3-
PR1000/2000
PJ8–/400/10
```
This example defines input 3 as the position snapshot input, output 8 as the product reject output, and sets the reject position range to between 1000 and 2000 counts. If a snapshot input is detected within the range 1000 to 2000, the reject action is triggered. After the motor has moved through the delay distance of 400 counts, the reject output is pulsed low for about 20ms.

**DUn±** **Define unipolar direction control output (restricted).**
**Range : 1 to 8**

This command defines the specified output line as the direction output signal for use with the unipolar analogue output option, enabled by CW bit 3. The sense of the direction output is set in the DU command, but is reversed if CW bit 4 is set. The direction output will stay in any one state for a given minimum time, in order to comply with the requirements of most common inverter drives. This time is set by the UT command below.

**UTnn** **Set unipolar direction output delay time (restricted).**
**Range : 0 to 255**
**Default : 0**

This command sets up a delay time for the unipolar direction output signal. It is specified in units of 1/256 second (about 4ms). Most inverter drives specify that their direction control inputs must be held in one state for a certain minimum time before reversing again. The UT parameter should be set to give a delay time slightly longer than that required by the drive.

**DBnn** **Set input debounce time (restricted).**
**Range : 0 to 255**
**Default : 1**

This command sets up a debounce time for all the digital inputs. It is specified in units of 1/256 second (about 4ms). Before an input signal is recognized as valid, it must be stable for the number of samples given by the DB command. This facility may be used to reduce the effect of noise in a system by reducing the number of false triggers due to noise.

**NOTE :** The debounce value does not apply to reference or position snapshot inputs. These inputs are programmed so as to be detected immediately on a change of state, to get the most accurate position information possible.

**Example :** `DB2`
This sets the debounce time to about 8 ms (2 samples).

 

**LI** **List input and output line definitions.**

This command lists the current definitions of the input and output lines on the display. The list shows the input or output line number, followed by a sign (+ or –) and a letter representing its function. Lines not defined are left blank. Function inputs also have their command string listed. The definitions are listed on the display or terminal, one per display line. They may be listed continuously, or the system can print one line at a time and wait for the user to press a key before printing the next line. This is useful when using the system with the membrane terminal, which only has a two line display. This list pause facility is controlled by one of the flag bits in the DW command.

**Examples :**

| System | User | Comments |
|---|---|---|
| `1>` | `LI<CR>` | User command to list inputs |
| `Inputs :` | | |
| `1 – R1` | | Reference input (ch.1) |
| `2 + I AB` | | Function input definition |
| `3 – I IN–/WT256/ID/MR–5000/ZC` | | |
| `4 + S2` | | Snapshot input (ch.2) |
| `5` | | Normal input lines, undefined |
| `6` | | |
| `7 – L1,2` | | Active low limit switch input |
| `Outputs :` | | |
| `1 + X` | | Expanded output group |
| `2 + X` | | |
| `3 + X` | | |
| `4 – R1` | | Reference reject output (ch. 1) |
| `5 – P2 >1000 <2000` | | Position trigger output (ch. 2) |
| `6 + E` | | Error output |
| `7` | | |
| `8` | | |
| `1>` | | |

**FSnn** **Select feedback encoder type (restricted).**
**Range : 0 to 4**
**Default : 0**

This command sets up different encoder feedback options for the current channel. The encoder feedback type may be set differently on each channel if required. The different feedback options are listed in the table below.

| FS | Option |
|---|---|
| 0 | Quadrature $\times 4$ |
| 1 | Quadrature $\times 2$ |
| 2 | Quadrature $\times 1$ |
| 3 | Up/down count |
| 4 | Count and direction |

**Table 1: Encoder feedback options**

 

## 4.13 Phase Advance

**PAnn** **Set phase advance scale factor.**
**Range : 0 to 65535**
**Default : 0**

The phase advance feature is a mechanism for shifting all position trigger output signals and the product reject output signal programmed on the current axis by some amount dependent on the instantaneous measured speed. The phase advance is defined as a shift proportional to the current speed of the motor. This command sets the scale factor between the measured speed and the actual phase advance. The scaling of the phase advance is given by the expression

$$\text{Phase advance} = (\text{speed} / 256) \times (\text{PA} / 256) \text{ counts}$$

where the axis speed is measured in encoder counts per second. For example, with a measured speed of 20,000 counts per second, a value for PA of 500 gives a phase advance of 153 encoder counts.

**VTn** **Set velocity averaging time constant.**
**Range : 0 to 8**
**Default : 0**

When using the phase advance facility, the measured axis speed is used to calculate the required amount of phase advance. The measured speed is calculated from successive encoder positions at 4ms intervals, and so the measured speed is only accurate to 256 encoder counts per second. When used with the phase advance, any variation in the measured speed causes a varying phase advance term, giving erratic operation of any phase advanced output signals. In this case speed averaging is required to maintain correct operation.

The VT command sets up an averaging mechanism on the axis, such that the number of samples of speed doubles for each increment in the value of VT. At VT=0, no averaging is done, and the immediate measured speed is used for the phase advance calculations. At VT=8, $2^8$=256 samples are averaged over a period of 1 second, to give a speed measurement accurate to 1 count per second. The system keeps a running average of the speed which is updated at each 4ms sample, so that the latest average speed is always available.

Note that whenever the averaging time is changed, the current average value is reset to zero and the running average is restarted. The averaged speed value is returned by the DV display velocity command.

**BAnn** **Set map base advance.**
**Range : 0 to 65535**
**Default : 0**

The map base advance is a mechanism for shifting the mapped position of a slave axis, relative to the master axis, by some amount dependent on the current master axis speed. The base advance is applied to the slave axis in the same way as the fixed MB map base offset value, and thus is defined as a shift along the master axis proportional to master axis speed. This command sets the scale factor between the measured master axis speed and the actual map base advance. The scaling of the base advance is given by the expression

$$\text{Map base advance} = (\text{master speed} / 256) \times (\text{BA} / 256)$$

where the master speed is measured in encoder counts per second. For example, with a measured master axis speed of 10,000 counts per second, a value for BA of 200 gives a map base advance of 30 encoder counts.

**BTn** **Set base advance time constant.**
**Range : 0 to 8**
**Default : 0**

When using the map base advance facility, the master axis speed is used to calculate the required amount of advance. When the master axis is also controlled by the system, the demand speed for the master axis is available to the slave, giving very smooth operation. When the master is not controlled, but is just an encoder fitted to some other part of the machine, the master speed can only be calculated from successive encoder positions. These positions are measured at 4ms intervals, and so the measured master speed is only accurate to 256 counts per second. When used with the map base advance, any variation in the master speed causes a varying base advance term. In this case master speed averaging is required to maintain smooth operation of the slave axis.

The BT command sets up an averaging mechanism on the slave axis, such that the number of samples of master speed doubles for each increment in the value of BT. At BT=0, no averaging is done, and the immediate measured master speed is used for the base advance calculations. At BT=8, $2^8$=256 samples are averaged over a period of 1 second, to give a speed measurement accurate to 1 count per second. The system keeps a running average of the master speed which is updated at each 4ms sample, so that the latest average speed is always available.

Note that whenever the averaging time is changed, the current average value is reset to zero and the running average is restarted. When a slave axis is mapped to a master axis using speed mapping instead of position mapping , the slave axis uses the averaged master speed as its input.

 

## 4.14 Display Commands

**DP** **Display actual position.**

Displays current actual position, in encoder counts.

**DD** **Display demand position.**

Displays current demand position, in encoder counts.

**DV** **Display velocity.**

Displays the current measured velocity of the system, in encoder counts per second. Note that the velocity is normally calculated as the difference between two successive position samples, and is therefore a multiple of 256 counts per second. If speed averaging is enabled by the VT command, then the displayed velocity value is the average measured velocity, and has a correspondingly higher resolution.

**DF** **Display reference error.**

Displays measured absolute position error relative to the last valid reference input, in encoder counts. For more details see the reference commands in section 4.10, and the DR and DZ configuration commands in section 4.12.

**DS** **Display snapshot position data.**

Displays the last absolute position measured when a snapshot input signal was detected. For more details see the PS configuration command in section 4.12.

**DT** **Display time.**

Displays current time, in hh:mm:ss format.

**TShh:mm:ss** **Time set.**

This command allows the user to set the system time. It expects the time to be in the format hh[:mm[:ss]]. If no value is given, the system displays the current time and prompts for a new time, which should be entered in the same format.

**DM[nn]** **Continuous display mode.**
**Range : 1 to 65535, or no parameter**

If no parameter value is given, this comand turns on a continuous display of demand position, measured position, position error, and time. The data is displayed for the current selected channel, unless a particular channel is specified by the MC parameter. The system also displays the change in position since the previous DM display line (used by the PC front end software), and the measured position for the other axis, to allow a direct comparison of the positions of the two axes. If a reference input or snapshot input is detected while the DM display is enabled, then the reference error or snapshot position value is displayed to the right of the continuous data.

If a parameter value is given, the system prints that number of lines of the position data.

**DO** **Display mode off.**

Turns off the DM continuous display. This is the default state.

**MCn** **Set monitor channel.**
**Range : 0 to 2**
**Default : 0**

This command allows the user to define the channel to be monitored with the DM command. If MC is set to 0, then the DM command displays data for the current active channel. If MC is set to 1 or 2, then the DM command always displays data for the specified channel, independent of the current active channel. This facility is used with the Windows front end software for the tuning displays.

**DK** **Display system constants.**

The system displays various parameter values, in the following order:

- Proportional gain constant KP
- Integral gain constant KI
- Velocity feedback gain constant KV
- Velocity feed-forward gain constant KF
- Velocity SV
- Acceleration (to nearest multiple of 256) SA
- Maximum position error SE

 

**DWbb** **Display options word (restricted).**
**Range : 8 bit binary value.**
**Default : 0**

This command allows the user to set various display configuration options. Note that the leading zeros may be omitted when entering a new value. The display options word bit functions are described below.

Bit 0 When set to 0, the system does not restrict the length of any output display lines.
When set to 1, the system restricts output to 40 columns with the DK and DM commands.

Bit 1 This bit controls zero suppression for all commands (except DK) and the DM display output.
When set to 1, all leading zeros are suppressed, except for the parameter values in the formatted DK display.
When set to 0, leading zeros are printed.

Bit 2 This bit controls the list pause facility. The list pause is available on the LS, LP, LI and HE commands.
When set to 1, the list pause function is enabled. In the list commands, after each line is printed the system waits for a <CR> carriage return character before printing the next line.
When set to 0, the list pause function is disabled and the list commands print continuously without waiting.

Bit 3 When set to 0, the system prints <CR><LF> at the end of each display line.
When set to 1, the system does not print <CR><LF> at the end of each display line. Instead, the display line is padded at the end with spaces, such that for each line a multiple of 16 characters in total is printed. This has the effect of formatting the display into 16 character fields. This may be used to view large amounts of data such as a profile list without the screen scrolling after each data item. It also allows the use of a simple portable terminal with a 16 column display and only limited terminal emulation facilities, such as a Psion Organizer fitted with its RS-232 serial adaptor.

Bit 4 Reserved.

Bit 5 When set to 0, the normal one or two character error messages are returned by the system.
When set to 1, the system returns longer error messages. This is the default setting.

Bit 6 When set to 0, the system expects input values in decimal.
When set to 1, it expects input values in hexadecimal.
Note that this does not affect parameter values that are entered as binary numbers.

 

Bit 7 When set to 0, the system prints output values in decimal.
When set to 1, it prints output values in hexadecimal.
Note that this does not affect values that are printed as binary numbers.

The default value of zero is for a standard 80 column terminal, and uses decimal input and output. This command is restricted, and is only available in privileged mode.

**Example : `DW00100110`**

This example turns on the long error messages, list pause and zero suppression options.

**HE** **Print help display.**

This command prints a complete list of all commands available on the system, in alphabetical order, a screenful at a time. It pauses between each page until a carriage return is entered. The escape key may be used to exit from the help command early. Help on a single command is displayed if the command mnemonic followed by ‘?’ is entered. Single command help for commands such as SF or RW prints a list of the options for the command.

**Example :**

| System | User | Comments |
|---|---|---|
| `1>` | `DP?<CR>` | Request help on DP command |

```
Display measured position
1>
```

**LE** **Display last error.**

This command redisplays the error message for the last error detected by the system. It is useful for finding an error message which has stopped the system when there is not normally a display connected to the machine, or to display the long error message for an error which has been reported with a short error message. This is done by setting bit 5 of DW before executing LE.

 

# 5. Status and Error Messages

## 5.1 Status Messages

This section gives the system status messages in various circumstances.

| | |
|---|---|
| > | Normal prompt<br>This is the prompt character in position control mode. The system is ready for the next command. |
| : | Motor off prompt.<br>This is the prompt character in the motor off state. |
| ? | New value prompt.<br>This character is used to prompt for a new parameter value. |
| I | Initialising.<br>The system is executing the IN initialize command. |
| M | Moving.<br>The system is executing a normal trapezoidal move. |
| S | Stopping<br>The system is executing a normal controlled stop. |
| V | Velocity control mode.<br>The system is executing a constant velocity move. |
| W | Waiting.<br>The system is waiting for some condition before continuing. |
| X | Executing a position mapping.<br>This channel is linked to another master axis and is executing a position mapping. The ‘X’ prompt is used to indicate that the channel is cross-linked to another channel, since the ‘M’ prompt is used for normal moves. |

 

## 5.2 Error Messages

This section describes the various error messages produced by the system. The short error message is given first, followed by the corresponding long error message and a description of the error condition. The system defaults to using long error messages, as set by DW bit 5.

B Binary number required.
The system received a non-binary character when it expected a binary number as input.

CT Clutch timeout.
The software clutch was unable to start the mapping because the current slave position is outside the range of positions onto which the master position is mapped.

D Decimal number required.
The system received a non-decimal character when it expected a decimal number as input.

E Unknown command <cmd> – type HE for help.
The system received a command which was not recognized.

E Invalid command entry <cmd>.
A command was given with an invalid parameter.

E Cannot execute <cmd> while <state>.
The command given cannot be executed while the system is in its current state.

E Cannot change <cmd> while <state>.
The parameter given cannot be changed while the system is in its current state.

E <cmd>: Command not available.
This command is not available with the current configuration, or in the current state.

E Cannot execute <cmd> while monitor function defined.
It is not possible to use this channel while a monitor function is defined on its analogue output.

E Monitor function not available while output in use.the auxiliary output cannot be assigned to the specified channel’s analogue output
The monitor function (SF) cannot be defined for this channel while the other channel is using its analogue output.

E <cmd>: No reference input defined.
The command cannot be executed because there is no reference input defined.

E Cannot execute command string while busy.
It is possible to execute a single command while the system is busy, but it is not possible to execute a string of commands.

| | |
|---|---|
| E | Cannot execute command string on DIn while busy.<br>It is not possible to execute an input line function consisting of more than one command while the system is busy. |
| E | <cmd>: No output group defined.<br>The command given can only be used when there are expanded output lines defined. |
| E | No commands after <cmd>.<br>This command can not be used at the end of a command line. |
| E | No commands before <cmd>.<br>This command can not be used at the beginning of a command line. |
| E | Only one repeat allowed in any command line.<br>Only one repeat command may be given on one command line. |
| E | Command <cmd> not available.<br>The command given is not available on this version of the system. |
| F | Nvm write failed.<br>The parameters and data could not be saved to nonvolatile memory successfully. |
| F | Nvm verify failed.<br>The parameters and data saved to nonvolatile memory have not verified correctly. |
| F | Saved data overflows nvm space.<br>The save parameters command has more data to save than will fit into the available nonvolatile memory space. |
| F | Checksum error.<br>The calculated checksum value does not match the checksum saved with the data. |
| F | Stored data invalid.<br>The stored data is invalid. |
| FA | Sequence/profile data pointer error<br>An internal error has occurred on the axis controller module. |
| FA | <cmd>: String pointer error<br>An internal error has occurred on the axis controller module. |
| G | Motor position error.<br>The system measured an instantaneous position error greater than the maximum allowed position error set by the SE command. |
| H | <cmd>: Hexadecimal number required.<br>The system received a non-hexadecimal character when it expected a hexadecimal number as input. |
| Ln | Limit switch detected.<br>A limit switch input has been detected. |

MO Map position overflow.
Either the demand position calculated for the slave channel by the mapping, or the mapped master axis bound calculated at the start of mapping, was outside the maximum range for absolute positions.

N <cmd>: Memory full.
There is no more room in memory for sequences.

N <cmd>: String memory full.
There is no more room in memory for input function strings.

NI <cmd>: Input line too noisy.
An input line remained unstable for at least two debounce times and the II command could not make any decision.

O <cmd>: Parameter out of range.
The value entered was outside the allowed range for this command.

O <cmd>: Moving away from wait position.
The motor is moving away from the position specified in a wait for position command.

O <cmd>: Undefined sequence.
The specified sequence is not defined.

O <cmd>: Second position below first.
The two positions for this command must be in order, with the second position higher than the first.

R Restricted command <cmd>.
This command may only be used in privileged mode.

R Restricted parameter <cmd>.
This parameter may only be changed in privileged mode.

RL Reference error outside limits.
The latest reference error measured was outside the limit set by the SR or LR command.

RO Reference correction overrun.
The previous reference correction was not complete when the next reference input signal was detected.

RT Reference timeout.
The system has completed several reference length cycles without detecting any reference signals.

T Motor timeout.
The system has detected a timeout error. The encoder has not moved at all for the timeout period, although the system expected it to move.

T Failed to reach target position.
The motor did not reach the move target position to within the position window before the timeout expired.

 

U <cmd>: Line already defined.
It is not possible to manually set or clear an output line that has been defined for some output function, or to redefine an input or output line that is already defined for a different function.

WF Watchdog test failed
The watchdog test has failed.

WT Watchdog timeout.
The watchdog timeout period has expired.

# 6. Interfacing

## 6.1 Notes on Installation

The Registration Controller system is a sophisticated computer system, and care should be taken in all installations to protect the unit from high voltages and to minimize electrical noise on signal and power supply lines. **Quin Systems can accept no responsibility for problems arising from poor installation.** Please refer to the PTS Installation Manual for more information.

A digital servo controller relies on the position information from its incremental encoder, and any noise on the encoder signals can give rise to errors in the absolute position. Care must be taken in installation of the Registration Controller unit and the encoders to minimize any noise on the encoder signal lines. The standard systems have full optical isolation on all the encoder signals, and require encoders with complementary line driver outputs. The encoder input interface has a differential input stage for use with such encoders, providing high rejection of common-mode noise. In addition, spurious signals on one encoder track produce both an up and a down count, and thus cancel out. However, in particularly electrically noisy environments it is still possible to get position counting errors. Noise is reduced by using encoders with line driver outputs. Where the environment is electrically noisy, or where the system will be used continuously and reliability is important, it is possible to set up the system such that its position is continuously adjusted for any errors by using a repetitive reference signal to correct them. Without such facilities, such errors would otherwise be accumulated over long periods of continuous operation, unless the system was stopped at regular intervals to reinitialize the absolute position.

The digital input and output lines are also fully isolated from the machine or plant, both for protection and to allow 24V signals to be used. This provides greater noise immunity and allows direct interfacing to industrial control equipment such as a programmable logic controller (PLC). Isolation is also available as an option on the analogue output signals if required.

 

## 6.2 Safety

The Registration Controller system provides many safety facilities, and it is recommended that these are used in addition to external safety systems such as hardwired limit switches. **Quin Systems can accept no responsibility for problems due to incorrect use of the safety features provided.**

The safety features of the system are provided for very good reasons ! It is important to understand the operation of all these facilities, as it is possible to do vast amounts of damage to both machinery and people with high performance motors and drives. It is not sufficient to decide that these facilities are not relevant to a particular application; they are provided to monitor the correct operation of the whole system, and if the system gives an error then it is telling you something important. The relevant commands are listed here.

| | |
|---|---|
| SE | Set maximum position error |
| TO | Set timeout |
| DL | Define limit switch inputs |

Please read thoroughly the descriptions of these commands at least, if no others.

## 6.3 Position Encoder

The system is designed for use with digital incremental position encoders. These encoders provide two signals in quadrature (one is phase shifted by 90° relative to the other). The system can monitor these signals and determine both the direction and distance of any movement. The direction is defined by which signal leads the other. The normal definition is such that the track A encoder input leads the track B input for movement in the positive direction.

The system generates four counts for each complete cycle of the input signals, such that an encoder with 1000 counts per revolution is seen as generating 4000 counts per revolution. The maximum input frequency of the encoder signals is 1.2 MHz with the normal ×4 multiplication in the encoder counter, giving a maximum count rate of 4.8 MHz. On a 2500 line encoder giving 10000 counts per turn, this is equivalent to a maximum speed of 480 revolutions per second, or 28800 r.p.m.

The encoder input signals are all fully isolated. The standard systems are designed for use with encoders with complementary line driver outputs, to get maximum noise immunity. The position encoder feedback is fundamental to the correct operation of the system, and so all precautions against noise are justified. The input configuration is for 5V encoders.

 

## 6.4 Demand Output

The demand output signal to the high power motor drive is an analogue signal with a range of ±10V, at 12 bits resolution. This output is switched directly to 0V in the motor off state by a reed relay on each axis. The motor drive is normally connected such that a positive demand output signal causes the motor to move in the positive direction.

## 6.5 Relay Contacts

The axis controller uses an onboard reed relay to switch the demand output signal to 0V in the motor off state. The relay has a spare set of changeover contacts which are uncommitted and available for external use. These may be used to derive an inhibit/ enable signal to the motor drive, or for example to switch a joystick onto the drive input to allow manual control of the motor.

## 6.6 Digital Inputs and Outputs

The system has 7 digital inputs and 8 outputs. The inputs and outputs may be programmed for a wide range of predefined functions, or they may be controlled explicitly if required. All the digital inputs and outputs are optically isolated, providing protection for the control system and 24V interfacing capability.

## 6.7 Operation of Limit Switches

The limit switch inputs are programmable by means of the DL command. This allows the user to select any input lines as limit switch inputs, and to define the active state of each input. The inputs float to a logic high if left unconnected.

If a limit switch is operated, the system stops the motor immediately and goes into the “motor off” state. The system displays the “limit switch detected” error message. All limit switches should be wired such that operation of any switch gives an error signal to the system. The system may be programmed to execute an error sequence automatically on any motor off error condition, including detection of a limit switch, by using the ME command.

## 6.8 Reference Inputs

A reference input is required on each motor channel during the IN initialization sequence to define the zero reference position for the motor. Reference inputs may also be used to continuously update the absolute position of the motor from the external zero reference if required. They may be connected to a marker signal from the position encoder, or to a microswitch that senses the position of the motor. The initialization sequence is as follows.

- Accelerate to the system velocity in specified direction.
- When the reference switch is detected, set the absolute position counters to the reference offset value (set by RF) and decelerate the motor to stop.
- Move to the new zero position (if allowed by RW options).

Reference inputs are programmable on inputs 1-4 (not 5-7) by using the DR command. A dedicated encoder marker input is also available, enabled by the DZ command. See section 4.10 for more details on the full range of reference facilities.

## 6.9 Serial Communications

The serial link uses RS-232 signal levels as standard. The serial word format used is 8 data bits, 1 stop bit, and no parity. The baud rate is fixed at 9600 baud.

The serial interface is buffered in software and echoes back the characters as they are received. It uses xon/xoff software handshake, where it sends xoff to signal that its input buffer is becoming full, and sends xon when it is ready for more characters. Note that if the xoff is ignored, the buffer may overflow and characters will be lost. The system also responds to xon/xoff to control its output.

 

# 7. Summary

## 7.1 Commands

Note that some commands are restricted. These commands can be used only in privileged mode.

**Miscellaneous commands**

| | | |
|---|---|---|
| VN | display Version Number and revision date | |
| SP | Save Parameters to nonvolatile memory | (restricted) |
| CS | CheckSum test | |
| RD | Read Data from nonvolatile memory | (restricted) |
| RS | ReSet complete setup to defaults | (restricted) |
| LA | List All parameters | |
| LD | List Default sequences | |
| FM | display Free Memory | |

**Mode commands**

| | | |
|---|---|---|
| MO | set to Motor Off | |
| PC | set to Position Control mode | |
| NM | set to Normal Mode | |
| PM | set to Privileged Mode | |
| PW | set PassWord | (restricted) |
| VMn | set Virtual Motor mode | (restricted) |

**Move commands**

| | |
|---|---|
| MA±nn | Move to Absolute position |
| MR±nn | Move Relative to current position |
| ST | STop with normal deceleration |
| AB | ABort, emergency stop |
| VC[±] | set to Velocity Control mode |
| IN[±] | INitialize to reference position |
| IB[±] | Initialize position and Bounds |
| DN± | set default motor DirectioN |
| ID | Initialize Demand signal offset |

 

**Set parameter commands**

| | | |
|---|---|---|
| SVnn | Set Velocity | |
| SAnn | Set Acceleration | |
| DCnn | set DeCeleration for stop command | |
| SSnn | Set Slow jog/creep speed | |
| VJn | set Velocity mode to Jog or normal | |
| SWnn | Set Window on final position | (restricted) |
| ISn | set Increment Select code | (restricted) |
| IPnn | Increment selected Parameter | |
| MWbb | set Move/map options Word | (restricted) |
| CWbb | set Control Word | (restricted) |

**Sequence commands**

| | | |
|---|---|---|
| ESnn | Enter Sequence | (restricted) |
| LS[nn] | List Sequence | |
| XSnn | eXecute Sequence | |
| RP[nn] | RePeat command line | |
| ER | End Repeat | |
| AX[nn] | Abort command eXecution | |
| BK[nn] | BreaK out of sequence | |
| ASnn | set AutoStart sequence | (restricted) |
| SNnn | set Setup/Ratio Number | |
| NSnn | set Number of Setups available | (restricted) |
| XN | eXecute Number sequence | |
| ON | Output Number on expanded outputs | |
| LD | List Default sequences | |
| SQ | enable default SeQuences | (restricted) |
| FM | display Free Memory | |

**Multi-channel commands**

| | | |
|---|---|---|
| CHn | CHange channel | |
| MXn | set Master aXis | |
| GS | Global Stop | |
| GA | Global Abort | |
| GF | Global motor oFf | |
| MEnn | set Motor Error sequence | (restricted) |

**Map commands**

| | | |
|---|---|---|
| XM | eXecute Map | |
| MB±nn | set Map Base offset | |
| MF±nn | set Map oFfset | |
| SMnn/nn | Scale Map | |
| AVn | set map base/offset/scale Adjustment Velocity | |
| BR | Set map scale factor from Bounds Ratio | |
| MWbb | set Map/move options Word | (restricted) |
| CTn | set software Clutch Time | |
| BAnn | set map Base Advance | |
| BTn | set map Base advance master speed averaging Time | |
| GM | Get Mapped master bound position | |
| GW | Get Wraparound offset value | |

**Wait commands**

| | |
|---|---|
| WTnn | Wait for Time |
| WIn± | Wait for Input line |
| WA±nn | Wait for Absolute position |
| WR±nn | Wait for Relative position |
| WF | Wait for reFerence signal |
| WB | Wait for Bound position |
| WC±nn | Wait for bound overflow Count |
| WE | Wait End |

**Error handling**

| | | |
|---|---|---|
| SEnn | Set maximum position Error | (restricted) |
| TOnn | set TimeOut | (restricted) |
| RTnn | set Reference Timeout | (restricted) |
| MEnn | set Motor Error sequence | (restricted) |
| EWbb | set Error options Word | (restricted) |
| LE | display Last Error | |

**Gain commands**

| | | |
|---|---|---|
| KPnn | set Proportional gain constant | (restricted) |
| KInn | set Integral gain constant | (restricted) |
| KVnn | set Velocity feedback gain constant | (restricted) |
| KFnn | set velocity feed-Forward gain constant | (restricted) |
| DK | Display system constants | |
| ITn | set Integration Time constant | (restricted) |
| OLnn | set analogue Output Limit | (restricted) |
| SFn | Set monitor Function | (restricted) |
| KMnn | set Monitor output gain constant | (restricted) |
| OMnn | set Offset on Monitor output | (restricted) |
| AOn | set Auxiliary Output channel | (restricted) |

 

**Reference commands**

| | | |
|---|---|---|
| ZC[nn] | Zero position Counters or set position | |
| SBnn | Set Bound position | (restricted) |
| BCnn | set Bounds Counter | |
| DZn | Define Zero marker input | (restricted) |
| DRn± | Define Reference input | (restricted) |
| RLnn | set Reference repeat Length | (restricted) |
| RMn | set continuous Reference Mode on/off | |
| RWbb | set Reference options Word | (restricted) |
| SRnn | Set maximum Reference correction | (restricted) |
| FRnn | set Filter on Reference error | (restricted) |
| LRnn | set Limit on Reference error | (restricted) |
| RF±nn | set Reference oFfset | (restricted) |
| RVnn | set Reference correction Velocity | (restricted) |
| RJ±nn | set Reference adJustment position | (restricted) |
| RTnn | set Reference Timeout | (restricted) |
| WF | Wait for reFerence signal | |
| DF | Display reFerence error | |

**Input/output commands**

| | |
|---|---|
| SO[n] | Set Output line(s) |
| CO[n] | Clear Output line(s) |
| PUn±/time | PUlse output line |
| OCnn | Output Code value on expanded outputs |
| ON | Output Number on expanded outputs |
| ODn± | Output Direction |
| RI[n] | Read Input line(s) |
| RO[n] | Read Output line state(s) |
| IIn± | If Input true do command line |
| IOn± | If Output true do command line |
| MI[n] | Mask Input(s) |
| BI[n] | inhiBit Input(s) |
| EI[n] | Enable Input(s) |
| WIn± | Wait for Input line |

 

**Configuration commands**

| | | |
|---|---|---|
| DZn | Define Zero marker input | (restricted) |
| DRn± | Define Reference input | (restricted) |
| DLn± | Define Limit switch input | (restricted) |
| DIn± | Define function Input | (restricted) |
| DXn± | Define eXpanded input lines | (restricted) |
| PSn± | define Position Snapshot input | (restricted) |
| MGbb | define input Mask Group | (restricted) |
| BGbb | define input inhiBit Group | (restricted) |
| DEn± | Define Error output | (restricted) |
| POn± | define Position trigger Output | (restricted) |
| OXn± | define eXpanded Output lines | (restricted) |
| BOn± | define Bound Overflow output | (restricted) |
| RRn± | define Reference Reject output | (restricted) |
| PJn±/dd/tt | define Product reJect output | (restricted) |
| PRnn/nn | set Position Range for product reject | (restricted) |
| DUn± | Define Unipolar direction control output | (restricted) |
| UTnn | set Unipolar direction output delay Time | (restricted) |
| DBnn | set input DeBounce time | (restricted) |
| LI | List Input and output line definitions | |
| FSnn | Feedback Select encoder type | (restricted) |

**Phase advance commands**

| | |
|---|---|
| PAnn | set Phase Advance scale factor |
| VTn | set Velocity averaging Time |
| BAnn | set map Base Advance |
| BTn | set map Base advance master speed averaging Time |

**Display commands**

| | | |
|---|---|---|
| DP | Display current Position | |
| DD | Display Demand position | |
| DV | Display current Velocity | |
| DF | Display reFerence error | |
| DS | Display position Snapshot data | |
| DT | Display Time | |
| BC | Display bound overflow Count | |
| TShh:mm:ss | Time Set | |
| DM[nn] | set continuous Display Mode on | |
| DO | Display mode Off | |
| MCn | set Monitor Channel for display mode | |
| DK | Display system constants | |
| DWbb | set Display Word | (restricted) |
| HE | print HElp display | |
| LE | display Last Error | |

 

## 7.2 Prompts and Status Messages

| | |
|---|---|
| > | normal prompt in position control mode |
| : | motor off prompt |
| ? | parameter value prompt |
| I | Initialising to reference position |
| M | Moving to new position |
| S | Stopping under normal deceleration |
| V | Velocity control mode |
| W | Waiting |
| X | executing a position mapping |

 

## 7.3 Error Messages

| | |
|---|---|
| B | Binary number required |
| CT | Clutch Timeout |
| D | Decimal number required |
| E | Error - unrecognized command,<br>invalid parameter,<br>command not allowed at this time,<br>or several others |
| F | Failed parameter save or checksum test |
| FA | internal FAilure on axis controller module |
| G | position error Greater than maximum |
| H | Hexadecimal number required |
| Ln | Limit switch detected |
| MO | Map demand position Overflow |
| N | No room in memory |
| NI | Input line too Noisy |
| O | parameter Out of range |
| R | Restricted command/parameter |
| RL | Reference Limit error |
| RO | Reference Overrun error |
| RT | Reference Timeout error |
| T | motor Timeout |
| U | line in Use |
| WF | Watchdog test Failed |
| WT | Watchdog Timeout |

# A. LED Status Codes

## A.1 Introduction

The TRC-2 and TRC-1 issue E two axis controllers used on the Registration Controller are fitted with two 7 segment LED displays. These are used to indicate the state of each axis, and to identify that an error has occurred, without a terminal or host system connected to the serial port. Older TRC-1 controllers have two hexadecimal LED displays, which use slightly different display codes.

The following pages show the status and error code values used on the TRC-2 and TRC-1 controller modules.

 

## A.2 Status Codes

The status code for each axis is updated any time the axis state changes. This is particularly useful to show activity in command sequences. Note that a waiting status code overrides any other status code until the wait condition terminates. The left digit shows the status for channel 1, and the right digit for channel 2.

**7 segment displays**

| Display | Mode | |
|---|---|---|
| 0 | MO | Motor off |
| P | PC | Position control |
| 1 | VC | Velocity control mode |
| 2 | MA, MR | Move to absolute or relative position |
| 4 | XM | Executing a position mapping |
| 5 | ST | Stopping |
| 6 | IN, IB | Initialising |
| 8 | Reset | Initial display at power-up |
| 9 | WT, WI, WA, WR, WF, WB, WC | Waiting |

**Hexadecimal displays**

| Display | Mode | |
|---|---|---|
| 0 | PC | Position control |
| 1 | VC | Velocity control mode |
| 2 | MA, MR | Move to absolute or relative position |
| 4 | XM | Executing a position mapping |
| 5 | ST | Stopping |
| 6 | IN, IB | Initialising |
| 8 | MO | Motor off |
| 9 | WT, WI, WA, WR, WF, WB, WC | Waiting |

## A.3 Error Codes

An error code is displayed when a motor error occurs on an axis. The motor error code remains on the LED display until the next PC command is executed.

**7 segment displays**

| Error Code | |
|---|---|
| E0 | Nvm error |
| E1 | Position error |
| E2 | Timeout error |
| E5 | Reference timeout |
| E6 | Reference out of limits |
| E7 | Reference overrun |
| E8 | Watchdog timeout |
| EA | Map position overflow |
| ED | Analogue input high limit |
| EE | Analogue input low limit |
| | |
| L1-L7 | Limit switch input n detected |
| FA | System fault |

**Hexadecimal displays**

As above except for:

| | |
|---|---|
| F1-F7 | Limit switch input n detected |

# B. Default Sequences

## B.1 Introduction

The Registration Controller is normally supplied with the standard set of default sequences programmed into the firmware eprom. This appendix describes these sequences and their use of the digital input and output lines, as a guide for experienced users.

The details given here are for systems using the new TRC-2 hardware, which has active high digital inputs and outputs for use in DIN standard panel wiring. Older systems and systems using the TRC-1 issue E hardware have active low inputs and outputs. For these systems all input and output definitions and all SO/CO/II/IO commands in the default sequences are reversed.

## B.2 Input and Output Line Functions

This section gives the machine functions for the digital input and output lines.

| Registration Controller | Line number | Function |
|---|---|---|
| | | |
| Inputs | 1 on<br>off | CH1 registration input |
| | 2 on<br>off | CH2 registration input |
| | 3 on<br>off | Slave start<br>Slave stop |

**Table 2: Input and output line functions**

 

| Registration Controller | Line number | Function |
|---|---|---|
| | 4 on<br>off | Initialize (with inputs 6 or 7) |
| | 5 on<br>off | Error shutdown and BCD switch common |
| | 6 on<br>off | Increment phase, initialize or jog forwards |
| | 7 on<br>off | Decrement phase, initialize or jog backwards |
| Outputs | 1 | BCD switch 1’s bit (ratio number) |
| | 2 | BCD switch 2’s bit (ratio number) |
| | 3 | BCD switch 4’s bit (ratio number) |
| | 4 on<br>off | Master axis has been initialized<br>Master axis has not been initialized |
| | 5 on<br>off | Motor error |
| | 6 on<br>off | Running<br>Stopped |
| | 7 | |
| | 8 on<br>off | Ready<br>Busy |

**Table 2: Input and output line functions**

The increment and decrement inputs 6 and 7 perform several different actions, depending on the current machine state on input line 4 and output line 6. The tables below show the actions for these combinations.

<table>
<tr><th>Input 6<br>Up</th><th>Input 7<br>Down</th><th>Input 4<br>Initialize</th><th>Output 6<br>Running</th><th>Function</th></tr>
<tr><td rowspan="4">On</td><td rowspan="4">Off</td><td rowspan="2">Off</td><td>Off</td><td>Jog forwards</td></tr>
<tr><td>On</td><td>Inc phase</td></tr>
<tr><td rowspan="2">On</td><td>Off</td><td>Select ratio and<br>initialize slave bound</td></tr>
<tr><td>On</td><td>None</td></tr>
<tr><td rowspan="4">Off</td><td rowspan="4">On</td><td rowspan="2">Off</td><td>Off</td><td>Jog backwards</td></tr>
<tr><td>On</td><td>Dec phase</td></tr>
<tr><td rowspan="2">On</td><td>Off</td><td>Select ratio and<br>initialize master bound</td></tr>
<tr><td>On</td><td>None</td></tr>
<tr><td colspan="2" rowspan="4">Both together<br>(within 64ms)</td><td rowspan="2">Off</td><td>Off</td><td>Initialize slave position<br>(and master position if required)</td></tr>
<tr><td>On</td><td>None</td></tr>
<tr><td rowspan="2">On</td><td>Off</td><td>Select ratio and<br>initialize slave bound</td></tr>
<tr><td>On</td><td>None</td></tr>
</table>

**Table 3: Increment and decrement functions**

 

## B.3 Input and Output Line Commands

The table below shows the low level programming commands for the input and output line definitions. The Registration Controller is normally used in slave only mode, without controlling the master motor, and the default sequences assigned to the input lines are programmed accordingly.

| Registration Controller | Line number | Standard (slave only) |
|---|---|---|
| | | |
| Inputs | 1 on<br>off | CH1/DR1+ |
| | 2 on<br>off | CH2/DR2+ |
| | 3 on<br>off | DI3+/XS36<br>DI3–/XS37 |
| | 4 on<br>off | DI4+/XS45<br>DI4–/XS46 |
| | 5 on<br>off | CH1/DL5+/CH2/DL5+ |
| | 6 on<br>off | DI6+/XS65<br>DI6–/XS66 |
| | 7 on<br>off | DI7+/XS75<br>DI7–/XS76 |
| Outputs | 1 | |
| | 2 | |
| | 3 | |
| | 4 | |
| | 5 | DE5+ |
| | 6 | |
| | 7 | |
| | 8 | |

**Table 4: Input and output line commands**

 

## B.4 Command Sequences

This section lists the standard default sequences for the Registration Controller, with comments to give some explanation of their operation.

Note that inputs 6 and 7 are programmed as the input mask group with the MG command. This allows the sequences assigned to these inputs to distinguish whether one or both of these inputs are on and take appropriate action.

Sequences 1-8 are reserved for use as setup sequences with SN and XN commands.

Sequence 9 - start of action

```
S9: CO8                     Set busy output
```

Sequence 10 - end of action

```
S10: EI/SO8                 Enable inputs and set ready output
```

Input 3 on sequence - start slave

```
S36: IO6+/BK                Exit sequence if already running
S36: XS9/SO6/EI3            Clear ready, set motor on outputs
S36: CHS/PC/XM/RM1/XS10     Start slave in mapping
```

Input 3 off sequence - stop slave

```
S37: XS9/CO6
S37: CHS/ST/RM0/AX/CHM/ST/XS10
```

Input 4 on sequence - master running

```
S45: XS9/CHS/EI3/WT16
S45: II6+/XS101/BK          Initialize slave bound
S45: II7+/XS101/BK          Initialize master bound
S45: CHM/RM1/XS10           Master running
```

Input 4 off sequence - master stopped

```
S46: XS9/CHM/RM0/AX/XS10
```

Input 6 on sequence - increment

```
S65: XS9/CHS/EI3/WT16
S65: II4+/XS101/BK          Initialize slave bound
S65: II7+/XS101/BK          Else initialize slave position
S65: IO6+/IS3/XS79/XS10/BK  Else if slave running increment phase
S65: PC/VC+/II6–/ST         Else start jog forwards
S65: XS10
```

Input 6 off sequence - stop jog

```
S66: IO6+/EI/BK             If slave running, exit
S66: XS9/CHS/ST/AX/XS10     Else stop jog
```

Slave increment function for control panel support

```
S67: XS9/CHS                Go to slave axis
S67: IO6+/IS3/XS79/XS10/BK  Moving, increment phase (decrement MB)
S67: PC/VC+/XS10            Else jog forwards
```

 

Mapbase increment
```
S69: IP100
```

Input 7 on sequence - decrement
```
S75: XS9/CHS/EI3/WT16
S75: II4+/XS101/BK          Initialize master bound
S75: II6+/XS101/BK          Else initialize slave position
S75: IO6+/IS3/XS69/XS10/BK  Else if slave running decrement phase
S75: PC/VC-/II7–/ST         Else start jog backwards
S75: XS10
```

Input 7 off sequence - stop master initialize
```
S76: II5+/CHM/ST/XS10/BK    If shutdown input on, stop master
S76: XS66                   Else stop slave jog
```

Slave decrement function for control panel support
```
S77: XS9/CHS                Go to slave axis
S77: IO6+/IS3/XS69/XS10/BK  Moving, decrement phase (decrement MB)
S77: PC/VC-/XS10            Else jog backwards
```

Mapbase change
```
S79: IP-100
```

Select ratio - edit this if a different number of setups are required
```
S100: XS105                 Call ratio select sequence 105 or 106
```

Initialize unit
```
S101: IO6+/XS10/BK          If running, exit
S101: BI3/SO6/WT10/EI3/XS100  Select ratio
S101: II6+/II4+/XS102/BK    Initialize slave bound
S101: II7+/II4+/XS104/BK    Else initialize master bound
S101: XS103                 Else initialize slave position
```

Initialize slave bound
```
S102: CHS/PC/IB+/BR/SP/XS107
```

Initialize slave position
```
S103: CHS/PC/IN+            Initialize slave position
S103: IO4–/II5–/CHM/IN/RM1/SO4 Initialize master position if required
S103: XS107
```

Initialize master bound
```
S104: II5–/CHM/IB+/BR/SP/RM1/SO4
S104: XS107
```

Select ratio with BCD switch (1 of 8)

```
S105: II5+/BK                   If shutdown input on, exit
S105: SN1/DL5/CHS/IS5           Set up to increment ratio setup
S105: SO3/WT2/II5+/IP4               Test 4's bit
S105: CO3/SO2/WT2/II5+/IP2           Test 2's bit
S105: CO2/SO1/WT2/II5+/IP1           Test 1's bit
S105: CO1/WT2/DL5+              Re-enable shutdown input
```

Select ratio with BCD switch (1 of 32)

```
S106: II5+/BK                   If shutdown input on, exit
S106: SN1/DL5/CHS/IS5           Set up to increment ratio setup
S106: SO5/WT2/II5+/IP16              Test 16's bit
S106: CO5/SO4/WT2/II5+/IP8           Test 8's bit
S106: CO4/SO3/WT2/II5+/IP4           Test 4's bit
S106: CO3/SO2/WT2/II5+/IP2           Test 2's bit
S106: CO2/SO1/WT2/II5+/IP1           Test 1's bit
S106: CO1/WT2/DL5+              Re-enable shutdown input
```

Turn off run output signal

```
S107: BI3/CO6/WT10/XS10
```

Motor error sequence

```
S110: GF/BI/CO6/CHS/RM0/XS10Force both axes off
```

Autostart sequence

```
S111: EI/CO4/SO8                Set output 4 to show master not initialized
```

 

# Index

# F



# G

## H



## I



## K



## L



## M

# N



# O

# S