Quin Systems Limited

Programmable Transmission System

Reference Manual

Issue 17

November 1999

(MAN533)

Copyright Notice

Software Version

This manual reflects the following software version.

- Host software version 2.1 or higher.
- Servo module firmware version 2.1 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

List of Figures and Tables

1. Introduction

This document describes the Quin Systems digital Programmable Transmission System (PTS) range, which consists of the Q-Drive 1+1, Q-Drive MAP, MiniPTS 1+1, MiniPTS 2+1, MiniPTS 3, MiniPTS 4, PTS Mk2 and SERVOnet.

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 familiarise 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 PTS range:

The Q-Drive 1+1 and MAP consist of the SRV1+1 single axis controller module, which is a complete standalone system and is incorporated within the drive amplifier.

The MiniPTS 1+1 uses the SRV1+1 single axis controller module from the Q-Drive systems, with additional hardware to make it a complete standalone unit for use with an external drive amplifier.

The MiniPTS 2+1/3 consists of the SRV-2 two/three axis controller module, which is a complete standalone system.

The MiniPTS 4 consists of the SRV-4 four axis controller module, fitted to a custom screw terminal panel.

The multi-axis PTS Mk2 system consists of one or more SRV-4 four axis servo controller modules, together with a host processor module which supervises the individual axis cards and provides the operator interface functions.

The multi-axis SERVOnet system consists of one Machine Controller or Mini Machine Controller and any combination of Q-Drive SERVOnet, MiniPTS 3 SERVOnet or MiniPTS 1+1 SERVOnet axis controller modules.

All these systems are 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, velocity and acceleration feedforward 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 minimise any overshoot. The rate of deceleration at the end of the move is normally 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. (Note that Q-Drives use the drive resolver feedback.) 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.

A number of digital input and digital output lines are provided on all systems which may be used in various ways. 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 may also be used to allow the system to be controlled from an industrial programmable logic controller (PLC). On the standard units, the digital inputs and outputs are fully isolated, and are compatible with 24V 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. (Note that the integrated Q-Drives do not require this signal, as the controller is installed in the drive amplifier and communicates directly.) For the MiniPTS 4 optional hardware facilities allow these signals to be fully isolated as well as the digital signals, if required.

Comprehensive sequence commands allow complex operations to be completely specified. Sequences may include any valid commands or command strings. Sequences may include commands for single or multiple motors; where sequences include commands for more than one motor, operations on different motors may run in parallel or in sequence. Profile commands allow the user to define a custom velocity profile for any motor. This provides the facilities for non-trapezoidal move profiles, for example to minimise mechanical stress in the machine. Two motors may be linked together as if by a gearbox or similar linkage, but under software control. The relationship between the motor positions is completely user-defined.

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>.

As this manual covers the entire PTS range, certain functions are documented which are not available on particular PTS products. If this is the case the product name is crossed out next to the command name: e.g. ***~~Q-Drive 1+1~~*** means that the command is not available on the Q-Drive 1+1.

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. Hexadecimal number entries must begin with a numeric character or a sign, not an alphabetic character, to distinguish them from normal commands.

Certain length related parameters can be entered as floating point numbers consisting of an optional sign, one or more leading digits, a decimal point, and one or more trailing digits. Floating point values are accurate to approximately seven digits which is sufficient for most applications. Floating point numbers are displayed to the precision defined by the FP command.

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 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. The escape character may be used to stop a list command, or to exit from the list of commands given by the help command.

Any ‘+’ characters at the beginning of an input line are ignored. This is to prevent any errors when using a modem link to a remote system; the standard hangup character sequence is “+++”, and this would otherwise leave the system with two ‘+’ characters in the input buffer after the modem link is closed. Also note that if the command input buffer is busy, such as when querying a parameter and the ‘?’ prompt is displayed, then other communications such as Modbus will not proceed until the buffer is available.

The standard command set provides flexible and complete control of the system. Some of the main command categories are listed here.

- Mode commands.
  These include commands to change between and position control and motor off modes, enabling and disabling the motor drive.

- 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.

- Profile commands.
  These commands are used for the profile move facilities (Software Cam).

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

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

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

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

- Analogue control commands.
  These commands are used to set up closed loop tension control.

- Variables.
  Variables and expressions can be used in place of constants for most command parameters to increase the flexibility of the system. They are of particular use in conjunction with the Operator’s Panel.

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, multiplied by an optional user-defined scale factor. This scale factor is set by the SU set units command. Note that the range and default values are given in encoder counts, and if a scale factor other than one is used then the allowed range and default values change accordingly.

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 input and output lines are listed with the LI and LO commands.

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.

- Trigger variable.
  A trigger variable can be defined which causes a command string to be executed when the variable is updated.

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 in the motor off state, the following error message is displayed.

```
Cannot execute VC+ while in motor off
```

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
```


A sequence consists of one or more lines of commands which can be executed by issuing the appropriate XS command. Sequences and command strings on the PTS execute in parallel by sharing the available processor time. More explicit control over command execution in sequence or in parallel is controlled by the CH and CP commands, which are described later. An example of two sequences running in parallel is shown below. In this example sequence 2 executes while sequence 1 is asleep waiting for the MA command to complete.

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

Sequences are automatically split up into a number of sub-sequences which are executed on the individual channels. Before a sub-sequence can be executed it must be sent to the relevant channel. This is done explicitly by compiling the sequence using the CM command, or it can be done automatically on sequence execution.

Commands which have an expression as a parameter are always sent to the channel as a command string, since the parameter is evaluated at execution time and therefore cannot be preloaded as a sub-sequence.

Note that the above rules only apply at the channel level and do not affect operation at host level. In other words, the fact that one channel is executing a sub-sequence does not restrict operations involving any other channels.

Input line functions are executed in the same way as command strings and are subject to the same rules. A command string triggered by a variable is executed in the same way as a normal command string or an input line function and is also subject to the same rules.

Command strings are split into sections, each of which can be executed either by a single channel or by the host processor. Sections which are to be executed at channel level are sent as command strings even if they consist of a single command. This is done to keep the proper synchronisation between commands. An exception to this rule is a single channel level command which occurs at the end of the command line. In this case the command is sent as a single command and can be executed at any time provided the motor state allows it. For example the following command string is always accepted if it is executed from the command line, from an input line, or triggered by a variable.

```
CH1/SV$SPD
```

The speed at which commands are executed is influenced by whether they are executed at channel level or at host level. On the PTS Mk2 system, channel level commands are executed by the relevant four axis controller modules, while host level commands are executed by the main host processor. On the SERVOnet system, channel level commands are executed by the relevant axis controller module, while host level commands are executed by the Machine Controller. Standalone systems with a single processor still use the same system architecture, with separate host and channel tasks.

In general, channel level commands are executed rather more quickly than host level commands because there is less overhead involved. For example, an input line function which consists only of channel level commands can be dealt with at channel level and is therefore quick and repeatable. An input line function which involves host level commands requires communication between the channel and the host, and from the host back to the channel, before any commands are executed, and is therefore slower and less repeatable. Note that any command which involves the CH and CP channel change commands, or a variable or expression, has to be dealt with at host level.

To determine which commands are executed at channel level and which at host level, please refer to the command summary in section 7.1.

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.

PM **Enter privileged mode.**

The full set of commands available on the PTS 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. Unauthorised 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 unauthorised 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
>		

If the password is defined as the default carriage return only, then the system powers up in privileged mode.

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 if a password (other than just a carriage return) has been defined and saved. If no password is defined, the system powers up in privileged 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 eight 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
>		

If the password is defined as the default carriage return only, then the system powers up in privileged mode.

TX **Enter comment text.**

This command is used to define a text comment string, up to a maximum length of 20 characters. The text string is saved with the other parameters in non-volatile memory. It can be used as a short label to identify the program in memory with a particular machine or product setup.

To display the current text string, type TX<return>. The system displays the text string, followed by the ‘?’ prompt. To change the text string, enter a new string at the prompt. To leave the string unchanged, type just a <return> at the prompt. If more than 20 characters are entered, only the first 20 are stored.

SK **Set software license key.**

The PTS supports a number of optional software packages, such as the Modbus interface or the Motion Generator option. These options are chargeable, and are controlled by a license key feature to prevent them being enabled in the field without authorisation. Systems which are ordered with a particular feature will normally have the license key installed before delivery.

The SK command lists the serial number for the unit, and all currently enabled software options. It allows license key values to be entered, enabling or disabling these software options. The license key values are unique for each optional software package and for each unit, as they are calculated using the unit serial number. To upgrade a system and enable a new option, it is necessary to obtain a license key value from your sales office for the specific system.

To disable a previously enabled option, enter the option name and press Enter or CR when the system prompts for the version number.

The SK command is also used to enable or disable the software for the Operator's Panel and Mini Operator's Panel, as they use the same serial port as the various PLC interfaces. In this case the required software is enabled without a license key by entering a version number of zero.

Note that the MiniPTS 4 may also need jumper link changes to correctly configure serial port B for use with communications interfaces to PLCs such as Modbus. It is normally shipped configured correctly for use with the Operator's Panel, using an RS-485 point-to-point link. Modbus normally uses an RS-485 multi-drop link, with tristate control to disable a station that is not transmitting.

Example :

System	User
1:	SK<CR>
Serial number: 001234	
Feature Version Key	
modbus 1.1 0EE2	
New feature ?	motion
Version ?	1.1
Key ?	F0E1
OK	
1:	

4.2 Parameter File Commands

SP[bb] **Save parameters (restricted).**
Range : 8 bit binary value.
Default : 0001 1001

This command saves all the programmable parameters to nonvolatile memory. There will be a delay while the save operation takes place, depending on the number of channels in the system and 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 profiles and 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.

Saving parameters should only be done when necessary. The non-volatile memory devices used on most units (eeprom or flash rom) have a large but finite lifetime, usually expressed as a maximum number of write cycles for guaranteed operation. If the application needs to save data continuously, then use a system with a battery-backed ram option.

The SP command may be followed by an optional binary parameter to select which aspects of the current setup are to be saved. The parameter bit functions (when set to 1) are as follows.

Bit 0 Save sequences.

Bit 1 Reserved

Bit 2 Save SERVOnet node table.

Bit 3 Save parameters.

Bit 4 Save maps and profiles.

Bit 5 Save variable array values.

Bit 6 Save variable values.

Bit 7 Reserved.

If no parameter is given after the SP command, the assumed value is 11001, which saves all sequences, parameters, maps and profiles. This is for compatibility with previous versions.

For Q-Drive and MiniPTS 1+1 based systems, the SP command stops all motors before proceeding, except in the case of SP1000000 (save variables only). This is because these systems cannot run motor control and store data into nonvolatile memory at the same time. An exception is saving variables or array values only (using SP1000000 or SP100000), as they are stored in battery backed memory, and this can be done at any time. For SERVOnet systems, the SP command stops all motors before saving channel parameters, using either SP or SP1000. All other items such as sequences and maps are stored on the Machine Controller, and it is not necessary to stop the motors before they are saved.

RD[bb] **Reload stored data (restricted).**
Range : 8 bit binary value.
Default : 0001 1001

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 cleared low. 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.

The RD command may be followed by an optional binary parameter to select which aspects of the current setup are to be reloaded. The parameter bit functions (when set to 1) are described below.

Bit 0 Reload sequences.

Bit 1 Reserved

Bit 2 Reload SERVOnet node table.

Bit 3 Reload parameters.

Bit 4 Reload maps and profiles.

Bit 5 Reload variable array values.
Note that when the saved array values are reloaded, existing arrays and trigger definitions on array elements are **not** deleted.

Bit 6 Reload variable values.
Note that when the saved variables are reloaded, existing variables and trigger variable definitions are **not** deleted.

Bit 7 Reserved.

If no parameter is given after the RD command, the assumed value is 11001, which reloads all sequences, parameters, maps and profiles. This is for compatibility with previous versions.

RS[bb] **Reset to default setup (restricted).**
Range : 8 bit binary value.
Default : 0001 1001

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 (cleared).

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.

The RS command may be followed by an optional binary parameter to select which aspects of the current setup are to be reset. The parameter bit functions (when set to 1) are described below.

Bit 0 Reset sequences.

Bit 1 Reserved

Bit 2 Reset SERVOnet node table.

Bit 3 Reset parameters.

Bit 4 Reset maps and profiles.

Bit 5 Reset variable arrays.
Note that when arrays are reset, all trigger definitions on array elements are also deleted.

Bit 6 Reset variables.
Note that when variables are reset, all trigger variable definitions are also deleted.

Bit 7 Reserved.

If no parameter is given after the RS command, the assumed value is 11001, which resets all sequences, parameters, maps and profiles. This is for compatibility with previous versions.

LA[bb] **List all parameters.**
Range : 8 bit binary value.
Default : 0001 1001

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 PTS Toolkit 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 PTS at a later date if the parameter settings are lost for any reason.

Brief comment lines (using the # character) are present in LA, to indicate separate sections, such as channel parameters, sequences etc. The escape key may be used to stop the output from the LA command before it has finished.

The LA command may be followed by an optional binary parameter to select which aspects of the current setup are to be listed. The bit functions (when set to 1) are described below.

Bit 0 List sequences.

Bit 1 Reserved

Bit 2 List SERVOnet node table commands (if valid).

Bit 3 List parameters.

Bit 4 List maps and profiles.

Bit 5 List variable array values.

Bit 6 List variable values.

Bit 7 List for current channel only.

If no parameter is given after the LA command, the assumed value is 11101, which lists all sequences, parameters, maps and profiles, and the SERVOnet node table commands if the table is valid.

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.

FM **Display free memory.**

This command displays information about the memory space available for sequences, maps and profiles. For more details refer to the full description of the FM command on page 52.

ZF **Zero file structure (restricted).**

This command erases and initialises the NVM file structure. All stored data will be lost. This command is restricted, and is only available in privileged mode.

4.3 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 initialised to the current measured position when the PC command is executed.

In position control mode, an onboard relay on the servo controller is energised 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.

The PC command also controls the motor brake control signal, defined on any output line with the OB command. After the drive is enabled, the system waits for the brake delay time (set with the BD parameter), and then releases the brake output. Q-Drive systems have an internal brake control signal for use with drives that have the brake relay option fitted. If no brake output is defined, the system still waits for the brake delay time before the PC command finishes and it proceeds to the next command.

The MiniPTS 2+1 is limited to having only two channels at one time controlling real motors. If the PC command is used to put a third channel into position control, the command gives the "command not available" error message, and the channel remains in the motor off state. The MiniPTS 3 allows all three motor channels to be controlled at the same time 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 unenergised 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.

If the EA parameter is set, then the MO command performs a smooth ramp down on the analogue output, from the current output value to 0V, over the time specified by EA. The motor is held enabled until this ramp time is finished. The same ramp is applied on a motor error shutdown.

The MO command also controls the motor brake control signal, defined on any output line with the OB command. Before disabling the drive, the system engages the brake and waits for the brake delay time (set with the BD parameter). Q-Drive systems have an internal brake control signal for use with drives that have the brake relay option fitted. If no brake output is defined, the system still waits for the brake delay time before the MO command finishes and it proceeds to the next command.

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.

If the MO command is used as a motor off stop command when the system is executing a multi-channel sequence, it only puts the current channel into motor off and leaves the remainder of the multi-channel sequence running. If it is necessary to put all channels into motor off then the GF global motor off command should be used.

VMn **Set virtual motor mode (restricted).**
Range : 0 to 2

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.

Virtual mode is 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.

VM0 : The channel is set to normal mode. A real motor is controlled and a real encoder is read for position feedback.

VM1 : The channel is set to virtual motor mode. The motor and encoder are simulated internally. The motor enable relay is held in the off state, but all other commands operate normally, including the position reference and snapshot inputs.

VM2 : This is a mixture of normal and virtual modes. Demand position functions are simulated as in VM1, but measured position functions still use the real encoder data. The table below lists the commands affected.

Commands using virtual (simulated) position	Commands using real position
DD SF1, 6, 16, 17 TW bits 0, 3, 8 Master position (LW bit 0 = 0) BO, BC WA, WR, WB, WC	DP, DV SF2, 7, 10 TW bits 1, 4, 5, 6, 7, 9 Master position (LW bit 0 = 1) DR, DZ, IN, IB, DR, RT PS, DS, PO LH, LL
ZC and initialisation set both demand and measured positions. Referencing uses the real encoder position.	

Table 1: Position functions in VM2 mode

Note : The MiniPTS 2+1 is limited to having only two channels at one time controlling real motors. If the VM command is used to take a third channel out of virtual mode in any state other than motor off, the command gives the “command not available” error message, and the channel remains in virtual mode. The MiniPTS 3 allows all three motor channels to be controlled at the same time if required.

4.4 Move Commands

This section describes the basic move commands. They provide simple commands for moving the motor at a constant speed, moving from one position to another, or moving to search for a home position signal.

The speed of the motor is normally set by the SV parameter, and the acceleration and deceleration by the SA parameter. The default velocity profile (graph of velocity against time) is a trapezoidal shape, with linear acceleration and deceleration ramps. 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 a move, the motor decelerates to stop at the desired final position. The deceleration rate is usually the same as the acceleration rate, but may be set to a different rate with the SD parameter if required.

In some applications where a smoother motion is required, ‘S’ shaped ramps may be used by setting CW bit 1 to 1. The s-ramps are in fact a sine-squared velocity profile, giving very smooth acceleration and deceleration. They are scaled such that for any given SV and SA parameter values, the linear ramp and the equivalent s-ramp take the same time and cover the same distance, making it very easy to choose one or the other without changing the execution time of the programmed moves.

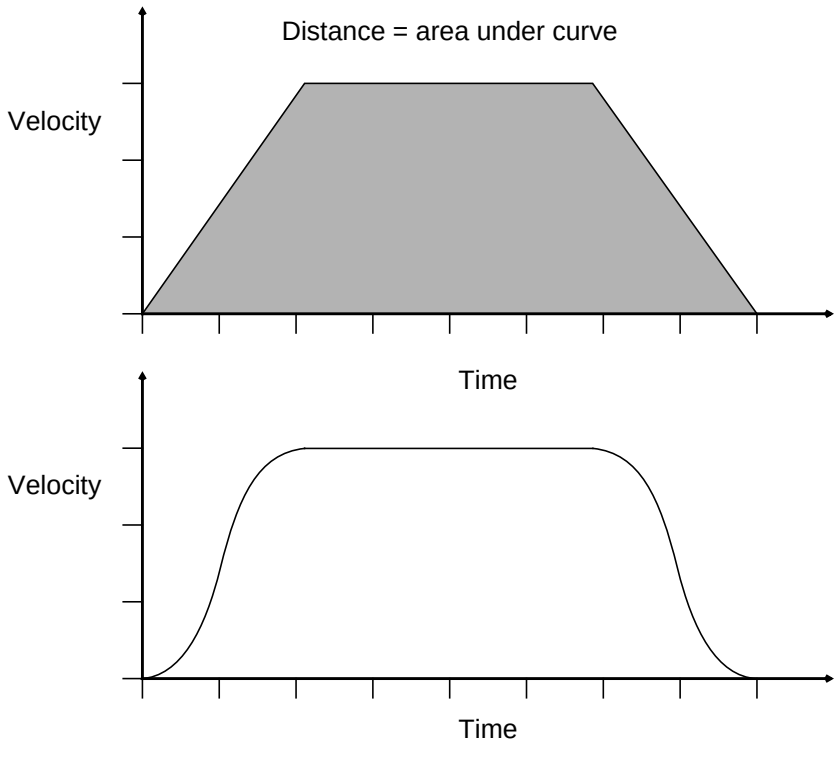


Figure 1. Trapezoidal and s-ramp move profiles.

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 one with a constant velocity segment.

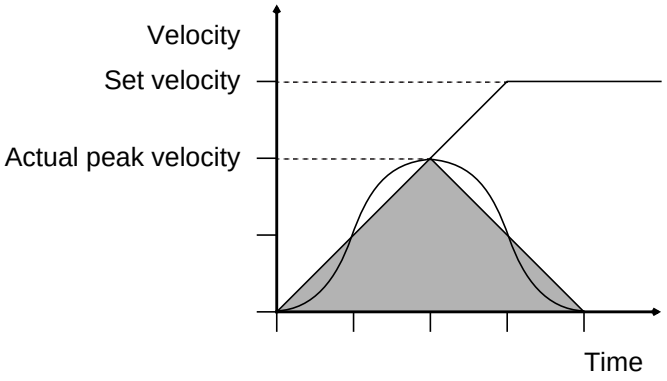


Figure 2. Triangular move profile.

Move commands can be executed from any state except motor off. This includes being able to change from other moving states to either constant velocity VC mode or to a new move with MA or MR. It also allows the target position for a move to be changed by executing a second move command with a new target position before the first move command has finished. If the axis is mapping or executing a profile, and the new move is in the same direction, then the current speed is maintained. If the axis is already executing a move or VC command, then the axis changes smoothly to the set speed for the new move command. If the new direction is opposite to the current direction, then the motor stops and reverses at the speed set by SV.

The target position for an absolute or relative move is checked against the current values of the user-defined position limits, set by the LH and LL commands. If the move would take the motor outside the set position limits, then the “target position outside limits” error message is returned, and the move is not executed. If the SB set bound value is less than the appropriate high or low limit, then the target position limit checking is disabled, as the absolute position value wraps around to zero at the bound position before reaching the limit position.

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. The position is entered in user units, which are equal to encoder counts divided by the user scale factor. The distance scale factor is set by the SU command, described in the set parameter commands section.

When the system executes an absolute or relative move command, it gives the 'M' move prompt character. The move commands may be used from any state except motor off, in which case the system returns the error message "cannot execute MA while in motor off". 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 user units.

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.

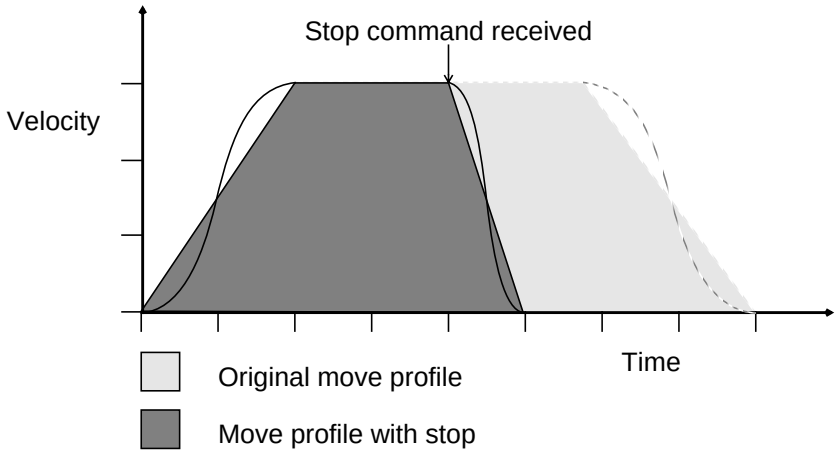


Figure 3. Move with normal stop.

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 or executing a profile, 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.

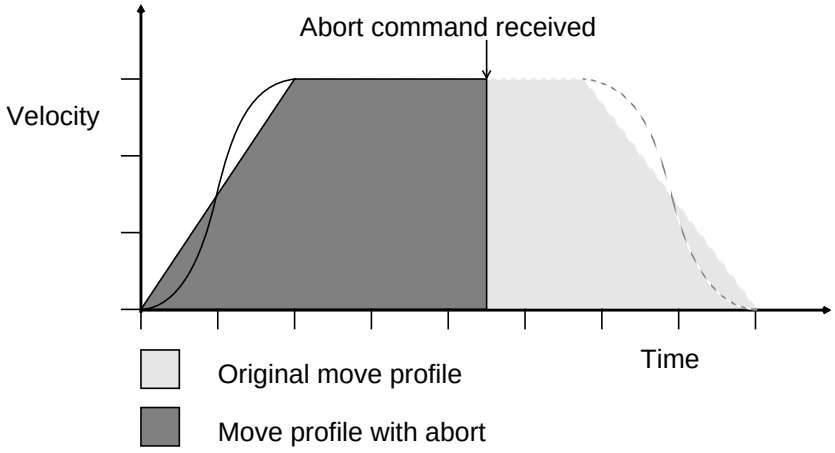


Figure 4. Move with abort.

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.

If the AB command is used when the system is executing a multi-channel sequence, it only aborts the current channel and leaves the remainder of the multi-channel sequence running. If it is necessary to abort all channels then the GA global abort command should 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 cannot be entered directly from the motor off state. If the channel is in the motor off 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.

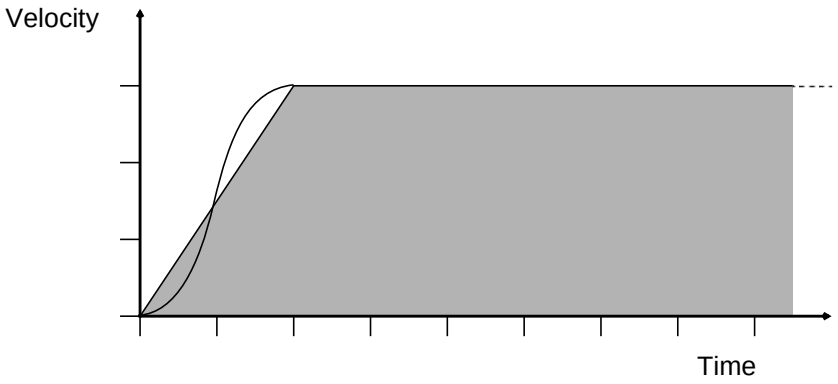


Figure 5. Constant velocity move.

IN[±] **Initialise position.**

The system performs the initialisation sequence to find a zero position reference signal. The system gives the 'I' initialise prompt character while executing the initialisation 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 (at the deceleration rate set by DC for the ST stop command), and resets the position counters as required. The motor then (optionally) moves to the new zero position, subject to any direction constraints set in the MW parameter.

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 initialisation 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 4.12 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 initialisation sequence is not executed.

Q-Drive systems normally use the resolver data from the drive for position feedback. In this case, if the IN command is executed with no reference input or marker pulse defined, it simply resets the current position to the resolver position data value returned by the drive module, without performing any movement. Once the position is reset, an MA0 command will move the motor to the new zero position. If DZ1 is set on a Q-Drive system, then the IN command performs this move to zero in order to simulate initialisation to an encoder marker pulse. If a DR input is defined, then the system performs a normal initialisation to move and search for the reference position.

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.

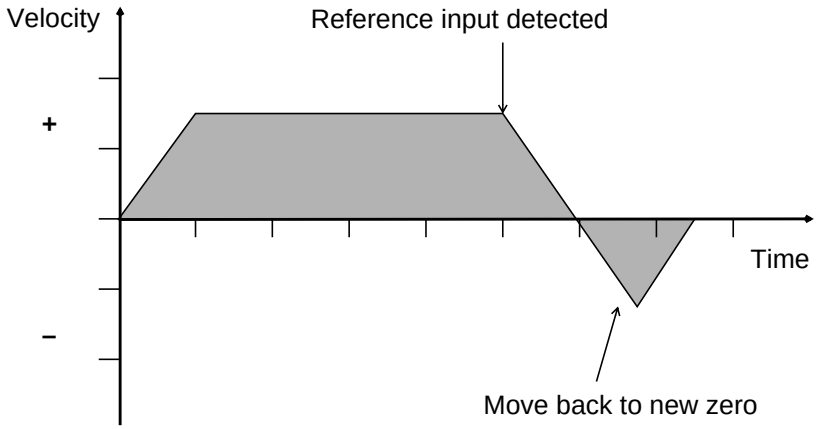


Figure 6. Initialisation to zero position.

IB[±] Initialise 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' initialise prompt character while executing the initialisation 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 to the new zero position, subject to any direction constraints set in the MW parameter.

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, in the same way as the IN command. 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 initialisation 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 **Initialise 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.

4.5 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 user units 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.

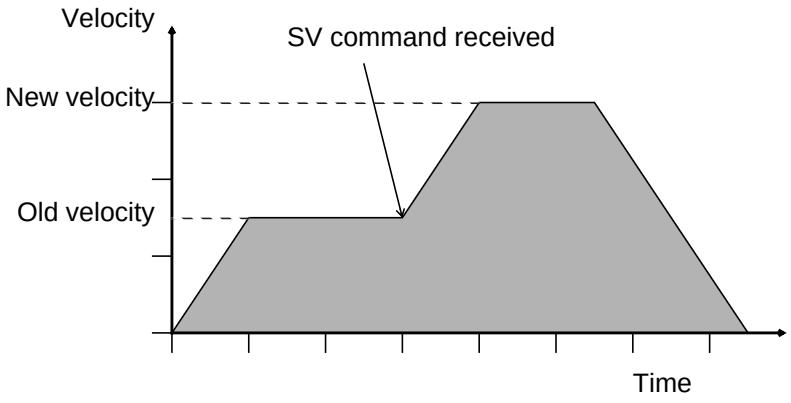


Figure 7. Move with change of velocity.

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 user units 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.

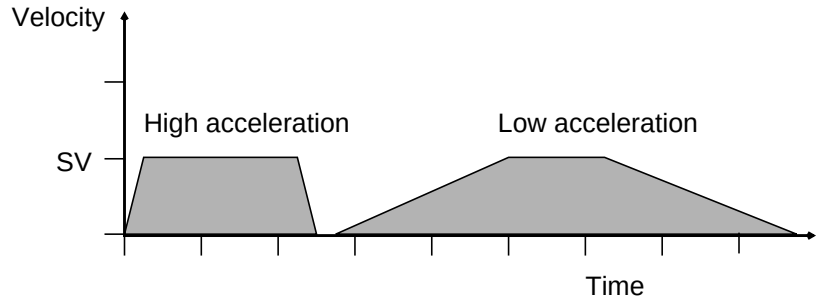


Figure 8. Set acceleration.

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.

SDnn **Set deceleration for move commands.**
Range : 0 to 2 000 000 000 (2.0E9) counts per second squared
Default : 0

This command sets the deceleration to be used at the end of a normal move command. If SD is set to zero (default) the value of SA is used for the move deceleration. By specifying a non-zero value for SD it is possible to generate non-symmetrical moves. If SV is changed during a move the value of SA is used to achieve the new speed; SD only applies to the final deceleration ramp to the target position. Note that the actual deceleration is rounded to the nearest multiple of 256 counts/second^2, and the minimum (non-zero) value is 256 counts/second^2.

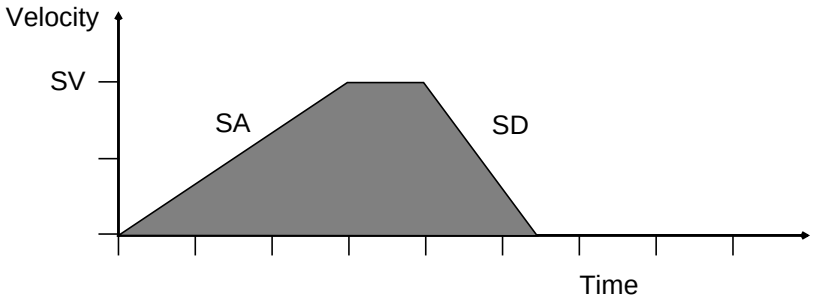


Figure 9. Separate acceleration and deceleration

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 initialisation 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 user units 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.

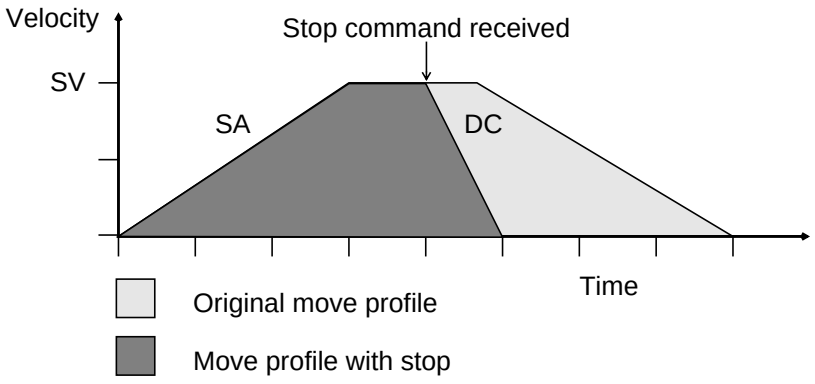


Figure 10. Effect of DC command.

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 (or jog 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.

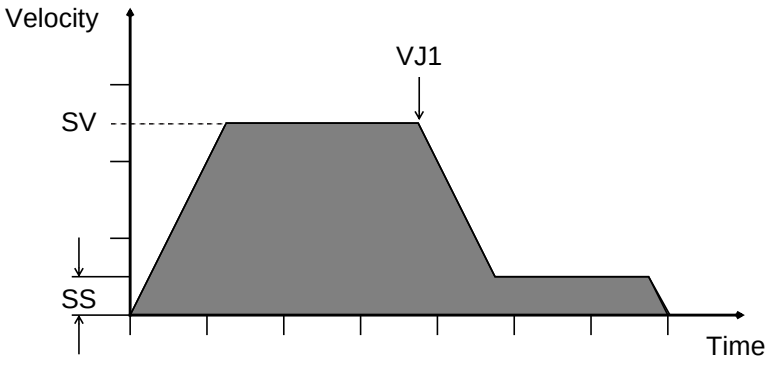


Figure 11. Move with change to slow speed.

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.

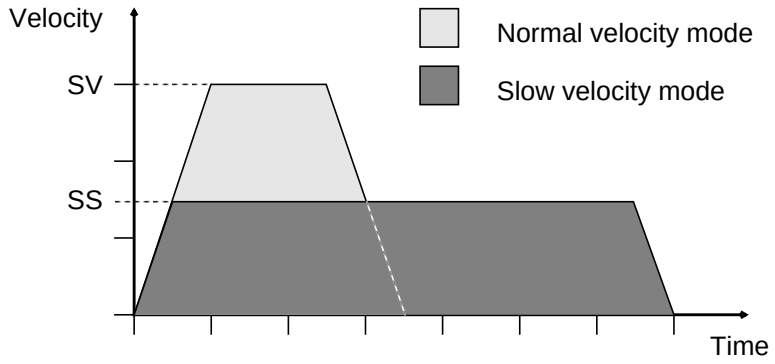


Figure 12. Normal/slow velocity mode.

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 initialised 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.

An output signal may be defined which indicates when the position error is outside the window given by SW. This is set up with the OW command, described on page 143.

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	Analogue control setpoint (AC)	±2047
3	Map base (MB)	±4000000
4	Map offset (MF)	±4000000
5	Reserved	
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 the tension control setpoint (IS2) affect the AC parameter value. If the new value is outside the allowed range, it is set to the maximum or minimum value as required.

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.

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

SUnn **Set units (restricted).**
Range : 1 to 65535
Default : 1

This command sets the scale factor between encoder counts and user distance units, or the number of encoder counts that represent one user unit. It allows length related parameters to be entered in scaled units rather than in encoder counts. This command is restricted, and is only available in privileged mode.

The scale factor and all scalable parameters may be entered as floating-point numbers (i.e. containing a decimal point). This allows scaling to be used on systems where the scaling between encoder counts and distance units is not a simple integer. When floating-point numbers are used they are only accurate to approximately 7 digits so there may some loss of accuracy when large numbers are used. The FP command is used to set the precision for displaying floating-point numbers.

Example : SU5
This sets the user scale factor to 5, such that 1 length unit equals 5 encoder counts. For example, if 1 encoder count was equal to 2 microns, then a scale factor of 5 would allow parameters to be entered in units of 0.01 mm.

Note: small, non-zero, non-integer values of SU could lead to rounding errors as all user units are converted to (integer) counts internally before use.

FPnn **Set floating point precision (restricted).**
Range : 0 to 10
Default : 6

The FP command is used to set the precision for display of floating point numbers. The parameter specifies the number of places after the decimal point. If FP is set to zero, only the integer part of the number is displayed. The FP command is **not** channel specific and only applies to scaled parameters.

Example:

System	User	Comments
`1>`	`SU10/SV1000.4`	Set velocity to 1000.4
`1>`	`FP2/SV`	Set precision to 2
`SV1000.40`		Velocity displayed to
`1>`		2 decimal places

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 4.8. 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 4.8, 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 This bit enables a low-pass filter on the encoder input speed. The filter time constant is set with the VT parameter.

Bit 1 This bit enables s-ramp acceleration and deceleration for all normal move functions, instead of the usual linear ramps.

Bit 2 This bit enables limited operation with fixed speed drives. It is available with the simple MA, MR and VC commands only. The demand position during the move is updated from the real position achieved by the fixed speed motor, so that the speed is determined by the fixed speed drive, not by the PTS. The move command finishes when the measured position reaches the move target position, within a tolerance set by the SW parameter. Thus the SW parameter allows an approximate stopping distance to be set for the fixed speed motor.
This feature allows fixed speed axes to be mixed with normal servo motors, using similar move command sequences for all axes.

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.

4.6 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. Sequences of commands for multiple motors may be defined if required. Command execution on different channels may be explicitly specified to proceed in sequence or in parallel, with each channel working independently.

All sequences are stored by the host system, and downloaded to the appropriate channel as required (this applies to all PTS systems - whether in a single box, e.g. MiniPTS 2+1, or distributed, e.g. SERVOnet). A sequence which consists only of commands for one motor is downloaded complete to the current motor channel when it is first called up, and is executed by the axis controller module itself. The sequence remains in the channel local memory for future use. This means that after the first time a sequence is used, it may be executed immediately on that channel without waiting for the sequence to be loaded from the host system every time.

A sequence which includes commands executed solely by the host system, or which includes commands for more than one motor channel and the appropriate channel change commands, is handled in a slightly different way. The sequence is first analysed by the host system, and split into its various components to be executed on each motor, and on the host. Any parts of the sequence that can be executed on a single channel are downloaded to that channel as a single-channel sequence. When the sequence is executed, the host system performs its required operations, and calls up the various sub-sequences on each of the motor channels. The local single-channel sequences are assigned sequence numbers by the host system automatically, and are not available to the user. If a system sequence is modified or deleted, the host system modifies or removes the appropriate local sequences as required. This mechanism for breaking the sequence down into its single-channel components and storing them in the channel controller’s local memory is analogous to caching mechanisms used in computers to improve the response speed of the system to commonly used commands. It provides fast execution of any sequence on the PTS, by storing the single channel sequence components locally on each channel.

This sequence analysis and downloading is completely automatic and requires no interaction on the part of the user or programmer. A sequence is said to be **compiled** when it is analysed in this way and its components are downloaded to the appropriate channels. All sequences may be compiled by using the CM compile command with no parameter, or any specific sequence may be compiled separately. Alternatively, any sequence which has not previously been compiled is automatically compiled before it is executed when the XS command is given. Note that a sequence which contains no channel change commands can only be compiled when it is executed on the current channel, since there is no information in advance to tell the system where to send the sequence components.

Commands and sequences may execute in sequence or in parallel on different channels. The default case is for all commands to be executed in sequence, except where specified explicitly by the CP change channels in parallel command. All synchronisation and correct sequencing of operations on different channels is now performed automatically. For example, consider a sequence which executes a move on channel 1, executes a move on channel 2, then returns to channel 1 to execute a profile.

```
S1: CH1/MA20000
S1: CH2/MA10000
S1: CH1/XP1
```

This sequence executes correctly when it is entered as shown here, since the system waits for each operation on a motor to complete before changing to another motor for the next operation. If it is possible to allow the two motors to operate at the same time, then the sequence may be modified as shown below.

```
S1: CP1/MA20000/XP1
S1: CP2/MA10000
```

In this case, the CP 2 command specifies that the change to channel 2 should proceed in parallel with the action on channel 1. Thus the move on channel 2 will execute at the same time as that on channel 1.

If some degree of synchronisation is required between operations taking place in parallel, then this may be specified by defining the sequence as a set of parallel sequence components which themselves execute in sequence. An example is a set of point-to-point moves in two axes on an XY table. Each two axis move consists of two moves running in parallel on two motors, but both moves must be complete on both motors before the next action can take place. The next example shows this, and also demonstrates how sequences may be nested.

```
S1: CP1/MA20000/CP2/MA10000

S2: CP1/MA0/CP2/MA0

S3: XS1/SO3/XS2/CO3
```

If more general synchronisation is required, there are commands available to manipulate signals between the motor channels and the host system. These may be used for synchronising operations on the host system and the channel controllers. They provide a signalling mechanism in both directions, both to and from the host system. These signal commands allow the user to explicitly specify that the system should wait for a previous operation to complete before proceeding to the next stage of the sequence. They also allow sequences to be started automatically on receipt of a specific signal from a channel.

ESnn **Enter sequence (restricted).**
Range : 1 to 65535

The sequence commands allow the user easily to build up complex sequences of machine operations and store them in the PTS. A stored sequence may be called up and executed with a single command. The practical limit on the number of sequences available is determined by the size of the NVM in the PTS unit, and the size of other parts of the PTS program, e.g. maps and profiles, which also take up space in NVM. This is because all sequences must reserve space in NVM for the SP command. A "working memory full" error message will be displayed when there is no more room for additional sequences.

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. Sequence nesting allocates memory as required, provided there is enough system memory available. Expressions can be nested ten deep, and there is no tradeoff between sequence nesting and expression nesting levels. 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 redefining it as an empty sequence with no commands.

Example : Entering a sequence

System	User	Comments
>	ES1<CR>	Enter sequence 1
S1:	ID/IN–<CR>	Initialise 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 65535, 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 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 65535

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. A multi-channel sequence may be aborted by using the GX global abort execution 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 **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.

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 **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.

Example :

```
1> LS10
S10: CH1/II1+/BK
S10: CH1/IN+/SO3
1> XS10/MA1000
1>
```

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

ASnn **Set autostart sequence (restricted).**
Range : 0 to 65535
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. The system also displays an “undefined sequence” error message.

All trigger variables and DI inputs are inhibited from system startup until the end of the autostart sequence. This allows trigger variables to be saved normally without any risk of them being triggered during startup before the system is completely ready.

Any MI, BI or EI commands in the autostart sequence have no effect until the end of the autostart sequence, when the automatic global inhibit is removed. If inhibit commands MV or GB are present in the autostart sequence to explicitly inhibit trigger variables or input lines, then they remain inhibited after the autostart sequence has finished. If enable commands EV or GE are put in the autostart sequence, then variables or input lines are enabled from that point.

FM **Display free memory.**

This command displays information about the memory space available for sequences, maps and profiles. It returns the global free memory space on the host processor and, for PTS Mk2 and SERVOnet systems, the local free memory space on each axis board/module. The system keeps track of the amount of spare memory each time a map, profile or sequence is entered or deleted.

It is possible under certain circumstances for the internal memory to become fragmented, when maps and profiles are being entered and deleted. This could give rise to the system reporting the “memory full” error message when the total amount of spare memory is larger than the data being entered. This occurs because the system tries to allocate a single block of memory for each map or profile. A simple solution to this problem is to save and restore the parameters using the SP and RD commands. This compacts the data and forces all the spare memory into one single block.

Example : Q-Drive 1+1, MiniPTS 1+1, MiniPTS 2+1, MiniPTS 4

```
System      User        Comments
1:          FM
Spare non-volatile memory 23271 bytes
Spare battery-backed ram 5120 bytes
Spare working memory 77576 bytes
Largest free block 75176 bytes
1:
```

Example : PTS Mk2, SERVOnet

```
1:          FM
Spare non-volatile memory 73988 bytes
Spare battery-backed ram 130048 bytes
Spare working memory 164012 bytes
Largest free block 163596 bytes
Module 1 spare memory 89132 bytes
Module 2 spare memory 89132 bytes
1:
```


CHn **Change channel.**
Range : 1 to no. of channels fitted

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.

On the MiniPTS 4, a fifth logical channel is available in addition to the four real channels. This extra channel is used as a virtual axis, for example in mapping or when using the Software Differential. It may also be used as an undriven encoder axis when the expansion board with the optional encoder interface is fitted.

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.

Note that any commands entered at the serial port or terminal are sent to the current channel, unless prefixed by a CH command. The current channel may be referred to as the focus channel, or the channel which has the terminal focus. The focus channel is changed only by a single CH command entered at the terminal. It is **not** changed by CH or CP commands executed in command strings or sequences.

CPn **Change channel in parallel.**
Range : 1 to no. of channels fitted

This command allows the user to switch between motor channels in a command string or sequence, and allow simultaneous command execution on more than one channel.

The CP command separates commands into blocks. The commands between the start of the sequence, each CP command, and the end of the sequence are treated as separate blocks of commands to be executed in sequence. The different command blocks, separated by the CP commands, are executed in parallel. The system returns an error message if a motor channel is referenced in more than one parallel block of commands.

Note that the parallel execution does not imply any synchronisation between channels. If synchronisation is required in the middle of a parallel operation, then the user may program this explicitly using the signal command facilities described later. Alternatively, the same result may be accomplished by splitting the sequence into smaller sequences, each of which is fully parallel, and nesting these sequences inside another sequence which executes them in order.

On the MiniPTS 4, a fifth logical channel is available in addition to the four real channels. This extra channel is used as a virtual axis, for example in mapping or when using the Software Differential. It may also be used as an undriven encoder axis when the expansion board with the optional encoder interface is fitted.

Example :

```
CP1
MA1000/WT256/MA0
CP2
MA2000/SV1500/MA3000
```

This shows an example of parallel command execution on two channels. The move commands on channel 2 execute at the same time as those on channel 1. See also the examples at the beginning of this section.

CM[nn] Compile sequence(s).

This command is used to download sequence definitions into their required channels. The system sequences are analysed and split into sections, each of which may be executed by only one channel. These single channel sub-sequences are then sent to the required channels and stored locally on each channel. This is the sequence caching facility described at the start of this section. When a new sequence is entered, it is normally not compiled until it is first executed. This introduces a slight delay when a sequence is first used, because it must be analysed and downloaded to the required channels before it can be executed.

The CM compile command allows any sequence to be downloaded to the appropriate channels, so that there is no delay on executing the sequence. If no sequence number is given, all sequences are compiled, provided they include the channel change commands to indicate on which channels they are used. If a sequence number is specified with the CM command, then only that sequence is compiled, together with any nested sequences that it calls. In this case a sequence with no channel change commands is compiled and sent to the current channel.

When a programmed PTS is powered up, sequences are not compiled automatically. Any sequences that should be downloaded before execution should be compiled with specific CM commands in an autostart sequence, set by the AS command. On a SERVOnet system it is good practice to issue a CM command in the autostart sequence, as this will significantly improve performance, especially during parallel map start initialisation routines, or similar.

Note that profiles and maps are stored on the host system until required by a particular channel. Any profiles or maps which are referenced by a sequence on a specific channel are automatically downloaded to the appropriate channels as part of the sequence compilation, as well as the sequence components. If the required map or profile is not yet defined, then the system returns an error message and the compilation is not completed. In this case, the sequence may be compiled again when the map or profile has been defined, or it may simply be compiled automatically when the sequence is first executed.

GS **Global stop.**

This command sends a stop command ST to all channels of a multi-channel system.

GA **Global abort.**

This command sends an abort command AB to all channels of a multi-channel system.

GF **Global motor off.**

This command sends a motor off MO command to all channels of a multi-channel system.

GX[nn] **Global abort execution.**

This command sends an AX abort execution command to all channels, and also stops any processing of commands or sequences by the host. If the system is executing some operation and a command string beginning with GX is entered, then the current operation is stopped and any commands following the GX command are executed. This provides an override facility similar to that available with ER.

If a sequence number is specified in the GX command, then that particular sequence is aborted.

USnn **Send user signal to host.**
Range : 1 to 127

This command is used in command sequences to send a user-defined signal from an axis to the host system. The parameter gives the signal number sent to the host system. This is used, for example, to indicate to the host system that an operation is complete. The signal sets a **channel-specific** internal flag in the host system. This flag is tested by the HW host wait for signal command.

It is also possible to assign a sequence to execute automatically on receiving a specified user signal from a specific channel. This is done with the SX command, described on page 58. This is used, for example, to assign a multi-channel system sequence to execute when an input line is detected on a channel. The input line is programmed with the DI command to return a user signal to the host system, and the host system assigns a sequence to that signal on that channel.

HWnn **Host system wait for user signal.**
Range : 1 to 127

This command complements the US command above. It tells the host system to wait for a specified user signal **on the current channel** before continuing on to the next step in a system sequence. The system tests an internal flag, set by the specified signal, and continues with normal execution when the flag is set by the receipt of the signal. If the signal has been received before the HW command is executed, then the system continues immediately.

ZSnn **Initialise signal number nn.**
Range : 1 to 127

This command clears an internal flag for the specified signal on the current channel. It is used to initialise the flag before use with the HW command above, and to clear it after use.

Example :

```
S1: CH1/ZS1/MA1000/MA0/US1/HW1/ZS1
S1: CH2/XP5
```

This sequence shows the use of the signal commands. The system waits for user signal 1 to be received at the end of the first command line before proceeding to the next line. In this particular example, the signal commands are not required since the system forces sequential operation anyway, but they may be useful in other synchronisation applications.

SXn/n **Set sequence to execute on a signal (restricted).**
Range : signal number - 1 to 127
sequence number - 0 to 65535

This command sets up a given sequence to execute automatically on receiving the specified signal from the current motor channel. It requires two parameter values. The first parameter specified in the SX command is the signal number used, and the second parameter is the number of the sequence to be executed on receiving the specified signal from the current channel. This can be used in conjunction with the DI and US commands to provide global input function definitions on a multi-axis system. It is also used with the DX expanded input lines command to assign sequences to the expanded input codes. To disable the sequence for a given signal, assign sequence number zero to it.

If the SX command is given with no parameter, all SX definitions are listed.

Note that each channel has its own unique set of user signals. Signals on any channel are distinct from those on all other channels. This means that the SX definitions are also channel-specific.

Example :

System	User	Comments
2>	CH1	Change to channel 1
1>	DI3-/US31	Define input 3 to send signal 31
1>	SX31/12	Assign sequence 12 to signal 31
1>		

This example shows how to set up input line 3 on channel 1 to execute sequence 12, by assigning the sequence to user signal 31, and defining input line 3 to send signal 31 when it goes to a logic low.

Example :

System	User	Comments
1>	SX	List current SX links to sequences
1/1		Signal 1 starts sequence 1
2/5		Signal 2 starts sequence 5
31/12		Signal 31 starts sequence 12
1>		

This example shows how to list the current set of sequences attached to user signals.

MUnn **Mask user signal(s).**
Range : 1 to 127

This command is used to inhibit user-defined signals on the current channel. If a signal number is given, then the specified signal is masked. If no parameter is given, all user signals on the current channel are masked.

EUnn **Enable user signal(s).**
Range : 1 to 127

This command is used to enable user-defined signals on the current channel. If a signal number is given, then the specified signal is enabled. If no parameter is given, all user signals on the current channel are enabled.

MEnn **Set motor off error sequence (restricted).**
Range : 0 to 65535
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).

UEnn **Set user error sequence (restricted).**
Range : 0 to 65535
Default : 0

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

4.7 Profile Commands

This section describes the profile commands. They provide facilities for setting up motions with user-defined velocity profiles (the Software Cam). Profile 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.

The normal move commands allow the user to easily move the motor to any possible position. However, they use the trapezoidal velocity profile defined by the set velocity and acceleration parameters. In some applications, it may be desirable to use a different velocity profile such as a cosine or parabolic profile, in order to get best results from the motor or to limit the stresses in the mechanical system. It is also not clear from the normal move parameters exactly how long a given move command takes to complete, and it may be important in some machines to specify a motion that is known to take a specific time to execute. In either of these cases the profile move facilities are very useful.

The profile commands allow a velocity profile to be completely specified by the user as a table of positions. This gives the user complete flexibility over the motor velocity and acceleration. A typical application of this is where the profile is calculated to take into account higher order derivatives of position such as “jerk”, to give a particularly smooth motion. The profile table also defines exactly the time taken for the profiled move to execute, since this is equal to the number of entries in the table multiplied by the profile step time.

Data for the profiles is entered as absolute positions, or as relative position increments. In the absolute position format, each profile table entry represents the next motor position at that time step, relative to the start of the profile. In the relative position format, each table entry represents the change in position at each time step. The selection of either absolute or relative position format is controlled by bit 4 of DW, the display options word. Profiles are now also subject to scaling by the SM parameter. This allows a profile to be defined as a template and then scaled to give the required distance.

The profile step rate, or profile velocity, is itself programmable via the PV command. The profile velocity may be defined from 1 to 256 steps per second. At the slower step rates, intermediate position values between the positions given in the profile table are calculated by interpolating linearly between the table values. This maintains the smoothness of the speed control even at the slowest step rates.

Profiles are stored in the host system until required by a specific channel. This allows a profile common to more than one channel to be defined only once, thus saving memory space. The profile data is automatically downloaded to the specific channel when required by the XP execute profile command, or it may be manually downloaded to the current channel by using the TP transfer profile command. This may be executed as part of the autostart sequence to avoid any delay the first time a profile is executed. Profiles executed from sequences are downloaded to their required channels as part of the sequence compilation process. If a DI input string contains an XP command, the specified profile is transferred to that channel when the input line is defined. If the profile has not been entered, then the transfer profile fails, the “profile is undefined or has not been transferred” error message is reported and the DI input line definition is aborted. Profiles executed in sequences are automatically transferred to the relevant channels by the sequence compilation process.

The map step value, set by the MS command, is also used to set the profile table interpolation step size. If the map step value is greater than one, the EP command prompts the user for new position values at intervals of MS. The first entry in a profile table is at step MS. This allows the user to enter positions at longer time intervals, and the system interpolates linearly between them to generate the full resolution profile. The profile data is stored at the reduced resolution specified by MS, again saving memory space. When the profile data is displayed using the LP list profile command, it is listed at the value of MS at which the data was entered. When the profile is executed, the profile step rate is effectively PV/MS steps per second, with the motor moving at a constant velocity for the duration of each step.

Profiles may be generated automatically by the optional Motion Generator package, if enabled by the appropriate software key. To make best use of the available memory in the system without reporting unnecessary “memory full” errors, profiles with numbers above 200 are **not** saved to non-volatile memory with the SP command. These may be generated on demand and use working memory, rather than filling up the memory area allocated to the saved data.

EPnn **Enter profile nn (restricted).**
Range : 1 to 255

The EP command is used to enter move profile tables into the system. The system responds with a “Pnn:” prompt for the table entries. Each entry in the table is the (signed) distance between the new position and the start of the profile, that is, the required cumulative motor position relative to the start of the profile. To end the table, make a blank entry by typing just a carriage return, and the system then returns to normal operation.

Example : A very short profile!

System	User	Comments
`>`	`EP1`	Enter profile 1
`Absolute mode`		Position format
`P1:`	`+10`	First position
`P1:`	`+40`	
`P1:`	`+100`	
`P1:`	`+180`	
`P1:`	`+270`	
`P1:`	`+350`	
`P1:`	`+410`	
`P1:`	`+440`	
`P1:`	`+450`	Final position
`P1:`	`<CR>`	End profile
`>`		Normal prompt

If bit 4 of DW is set, then the profile is entered as a table of (signed) relative position increments instead. The table values now represent the change in position at each time step, and thus the table may be viewed as a velocity profile.

Example : The same profile entered as relative position steps.

System	User	Comments
`>`	`EP1`	Enter profile 1
`Relative mode`		Position format
`P1:`	`+10`	First relative move
`P1:`	`+30`	
`P1:`	`+60`	
`P1:`	`+80`	
`P1:`	`+90`	
`P1:`	`+80`	
`P1:`	`+60`	
`P1:`	`+30`	
`P1:`	`+10`	
`P1:`	`<CR>`	End profile
`>`		Normal prompt

LP[nn] **List profile nn.**
Range : 1 to 255, or no parameter

This command allows the user to examine a profile that has previously been entered into the system. The profile table is listed on the display or terminal, one table entry per line. The system prints the profile number, the step number, and then the position value for the entry. The total time taken for the profile to execute is given by the step time multiplied by the number of steps in the profile, which is equal to the step number of the last entry in the profile table.

If no profile number is given in the command, the system lists the numbers of all profiles which are currently defined.

If bit 4 of DW is set, the profile is listed as the relative position increment at each step, instead of as the cumulative position through the table at each step. At the end of the profile it then prints the total distance for the profile.

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

Example : LP1
This lists profile 1 on the display or terminal. The output for the above example would look like this, in absolute position format.

System	User	Comments
>	LP1<CR>	User input to list profile.
Absolute mode		Position format
P1 1: +10		
P1 2: +40		
P1 3: +100		
P1 4: +180		
P1 5: +270		
P1 6: +350		
P1 7: +410		
P1 8: +440		
P1 9: +450		
>		

The output for the same example would look like this in relative position format.

System	User	Comments
>	LP1<CR>	List profile command.
Relative mode		Position format
P1 1: +10		
P1 2: +30		
P1 3: +60		
P1 4: +80		
P1 5: +90		
P1 6: +80		
P1 7: +60		
P1 8: +30		
P1 9: +10		
Total distance = 450		
>		

Example : LP

This lists all currently defined profiles on the display or terminal.

System	User	Comments
>	LP<CR>	List all profiles.
P1		Profile 1 is defined.
P3		Profile 3 is defined.
>		

XPnn[–] **Execute profile nn.**
Range : 1 to 255

This command tells the system to execute profile number nn. The 'P' profile prompt character is given while a profile is executed. The speed at which the system steps through the table (the profile velocity) is set by the PV command. Note that a profile is always executed **relative** to the current demand position. If the profile number is followed by a minus sign, the profile is executed in the reverse direction. A profile move may be aborted by using the AB, ST or MO commands. If the ST stop command is used, the motor decelerates to a stop from the current instantaneous speed at the deceleration rate set by the DC command.

On a PTS Mk2 and SERVOnet system the profile data is downloaded automatically to the current channel when the XP command is executed for the first time. This means there is a short delay between issuing the command line containing the XP command and the actual execution of the profile. This may be avoided if required by using the TP transfer profile command to transfer the profile data to the current channel in advance. Once the profile has been sent to a channel, it remains there for further use until it is deleted or changed.

If the XP command is given with no parameter, then the system displays the number of the currently executing profile.

Note that the profile positions are now scaled by the SM parameter when the profile is executed.

Example : XP1
The motor executes stored profile no. 1, at the speed set by the PV command.

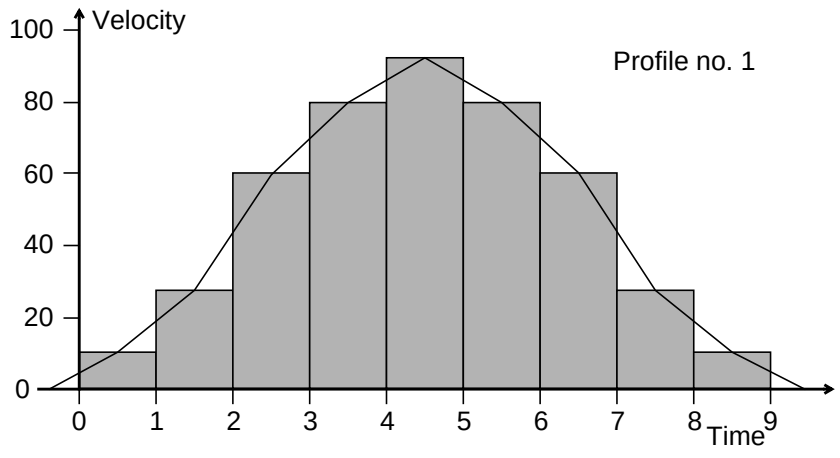


Figure 13. Profiled move.

PVn **Set profile velocity.**
Range : 1 to 256
Default : 32

This command sets the rate at which the channel steps through a profile table, known as the profile velocity. The step rate is defined in steps per second. The channel interpolates linearly between table points at the slower speeds in order to maintain smooth motion.

The profile velocity may be changed at any time, even while executing a profile.

Example : `PV16`
This sets the profile velocity, or step rate, to 16 steps per second.

TPnn **Transfer profile.**
Range : 1 to 255

This command tells the system to transfer the specified profile to the current motor channel.

The profile data is held internally as relative positions from the start of the profile.

For a PTS Mk2 and SERVOnet system, before transferring the data to the channel, the system checks whether the channel has enough free memory to take it. If not, previously transferred maps or profiles are deleted from the channel until enough space is available. If there is still not enough room for the new profile, then the system reports a “memory full” error message. Note that maps or profiles deleted from a channel in making room for a new transfer are not lost, as they are retained at host level and may themselves be transferred again.

FM **Display free memory.**

This command displays information about the memory space available for sequences, maps and profiles. For more details refer to the full description of the FM command on page 52.

4.8 Map Commands

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

The position mapping commands provide a mechanism for defining the required position of a slave motor with respect to a given master channel, for all possible positions of the master channel. 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 synchronised. The mapping itself consists of a table of slave position values,. When the map is executed on the slave axis, its demand position is calculated by using the current master position as an index into the map table, and then using the value in the table as the slave demand position.

In a simple case, this allows a wide range of linear gear ratios to be defined between the master and slave axes. In more complex applications, it allows the system to mimic the action of nonlinear systems, where the position of the slave motor has some more complicated relationship to the master axis position. This can be used to replace eccentric gearbox mechanisms, crankshaft linkages, cam operated pushrods, or almost any mechanical linkage or transmission system with equal ease, simply by defining the required map table.

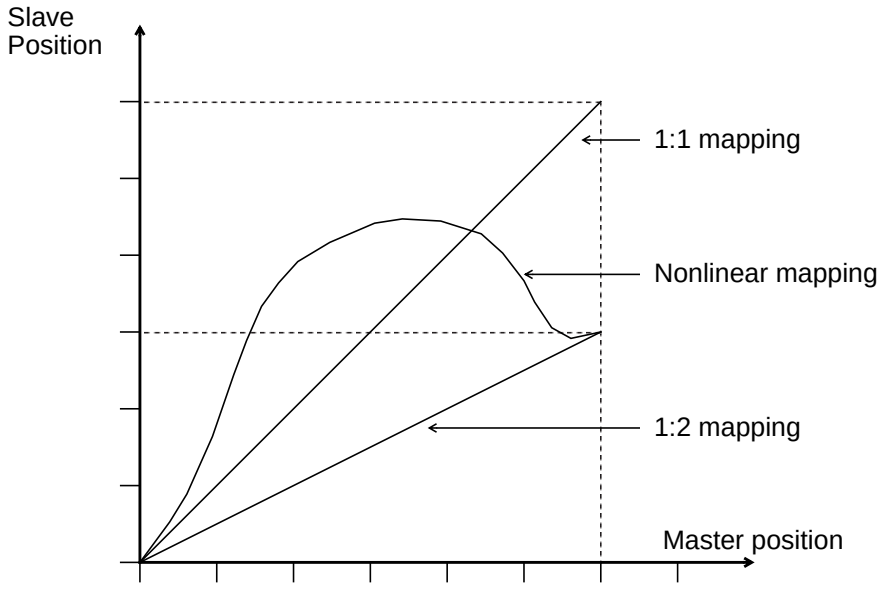


Figure 14. Simple position maps.

NOTE: A Q-Drive 1+1 only supports linear maps, i.e. XM0.

On a machine where both master and slave axes are only moving from point to point within a finite range of positions, for example on an XY table, the mapping simply needs to be defined over the range of master position actually used, and the mapping itself then defines the slave position range. This is the simplest application. In such a system, there is no need to set up the position bounds on either channel. They may be left at the default value, since the motor positions are constrained within fixed ranges and the motors cannot move continuously in one direction.

The diagram below shows a typical position mapping, where the slave channel is following some position profile relative to the master axis. This could be, for example, the path for a cutting tool on a machine where the master axis speed varies to maintain a constant linear tool speed across the material. This could not be easily achieved by other methods, which would rely on keeping the cutter motion synchronised in time with the master axis motion. This could only be set up for one specific master axis speed or profile, and would need to be reprogrammed to allow the machine to be run at a different speed. Using the position mapping mechanism, the master axis speed may be varied at any time, even during a cut, and the cutter stays on the correct path, as defined by the mapping.

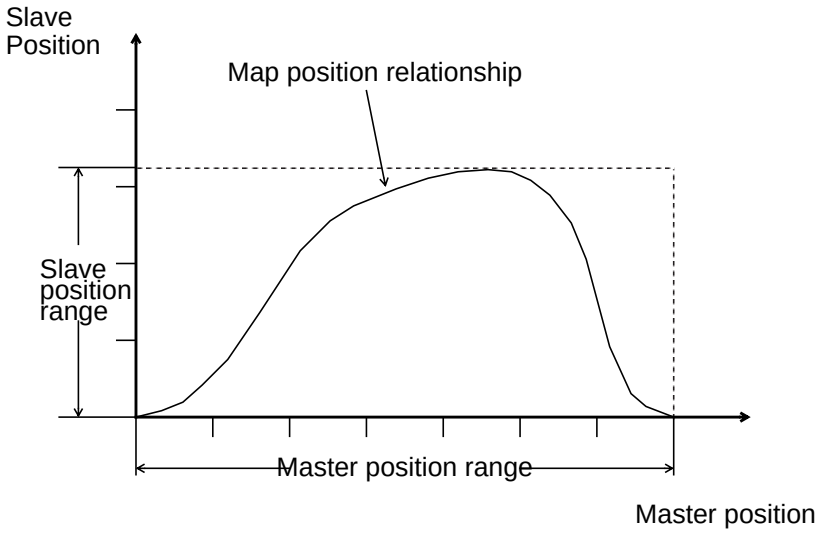


Figure 15. Position mapping over a defined range.

The example shown above could also apply to a machine where the master axis is a continuous rotary axis, such as a line shaft, with the slave axis profile repeating once for every cycle of the master axis. In this case the master bound position must be set to the master axis cycle length, and also copied to the slave axis with the MP map bound command.

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. The mapping must be defined such that the required slave axis position is continuous at the points where the master and slave axes pass their respective bound positions. If this is not done, the discontinuity in the mapping at the position bounds could give rise to very sudden changes in speed and position of the slave motor when the bound positions are reached.

If the slave position is not continuous across the map boundary, the difference between the required slave cycle length and the actual cycle length given by the mapped master position bound appears as a small shift of the slave axis relative to the master axis. This error accumulates over several cycles of the machine and behaves as an apparent drift of the slave axis. This type of problem is difficult to spot, but can be prevented by careful definition of the position map table.

The mapping should be defined over the master position range from zero to at least the master axis bound position. In addition, the master axis bound position value must be known by the slave axis, so that when the master axis position wraps around to zero at the bound position, the slave axis knows how to adjust its mapped demand position in order to maintain its speed across the master wraparound point. This is done automatically by the system when the ML map link command is used, but may be done manually if required by using the MP command. If the master axis bound value is changed while any slave channels are linked to it, the new master bound must be copied to each linked slave channel by using this command.

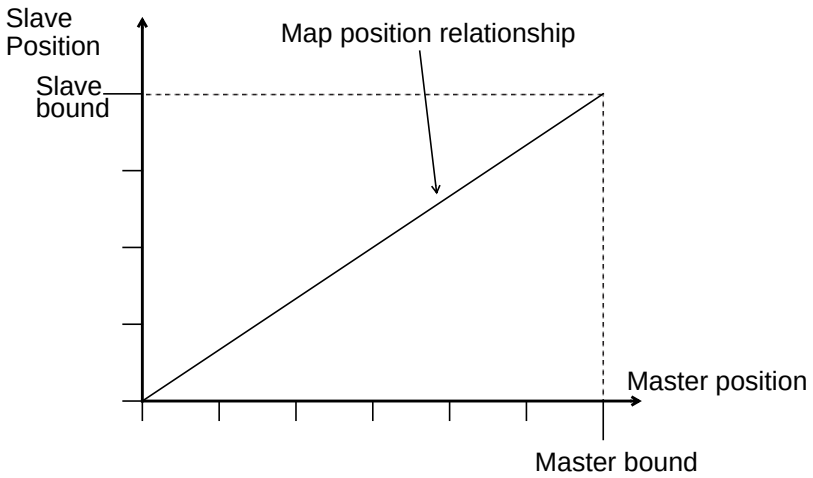
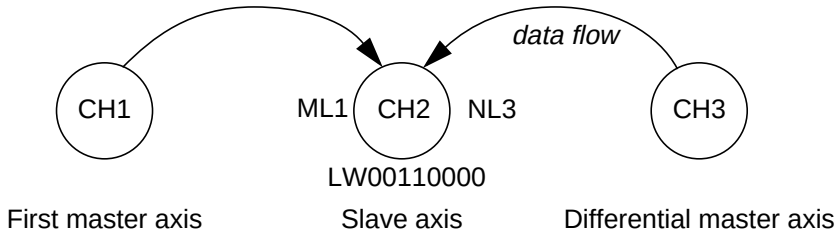


Figure 16. Position map for a cyclic machine.

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 Software Differential allows a slave motor to be synchronised so that it follows either the sum of or difference between the positions of two master axes. This is analogous to a differential gearbox. The slave axis must be linked to a master axis with the ML command, and to a second master axis (called the differential axis) with the NL command. LW bits 4–6 control how the slave axis uses the position information from the two master axes, and LW bit 0 controls whether the master or differential axis transmits its demand or actual position to slave axes.



Slave axis demand position = master position – differential master position

Figure 17. Example setup for software differential

Note that the master channels may be in any state, provided the map links are set up as required. Also note that it is important to set the LW options (on master and slave axes) before executing the ML map link command. Any change to LW is not applied until the slave channel is unlinked and re-linked to the master axis.

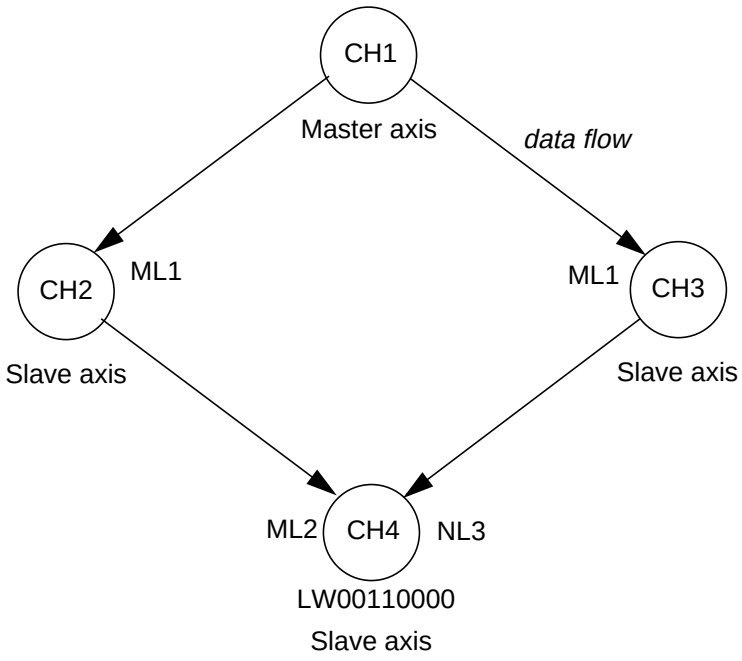


Figure 18. Example setup of software differential (complex)

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.

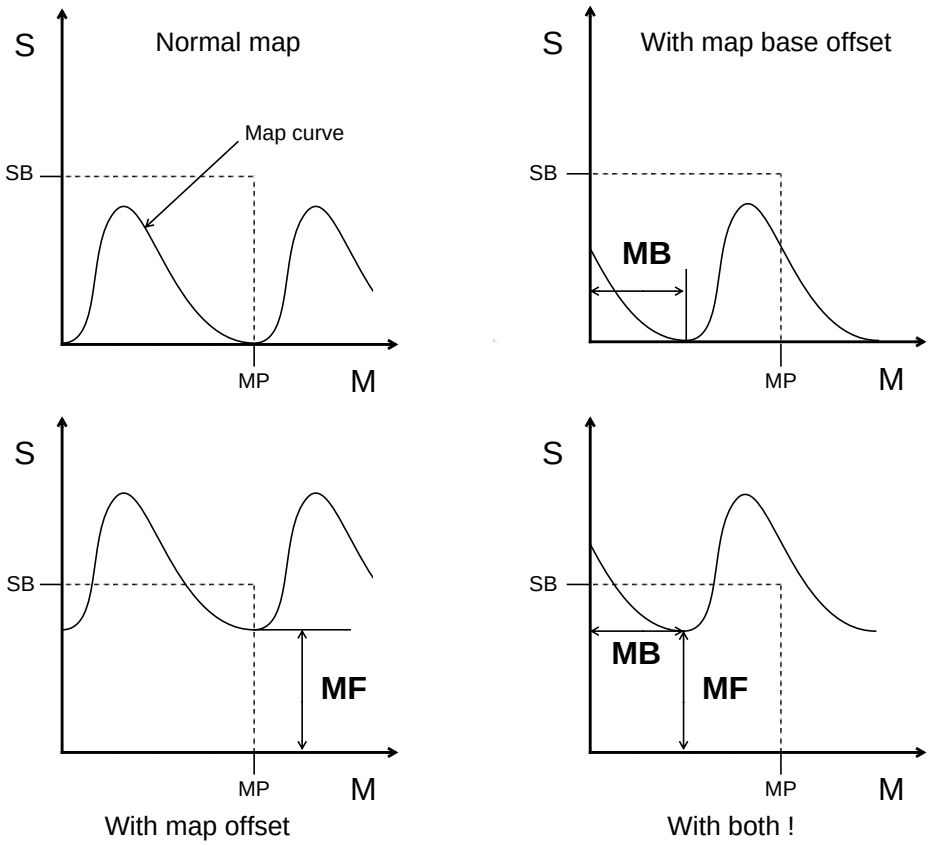


Figure 19. Effects of map base and map offset.

Data for the maps is entered as absolute slave axis positions, or as relative position offsets from the master axis position. In the absolute position format, each map table entry represents the required slave motor position at the given master axis position. In the relative position format, each table entry represents the difference between the slave axis position and the master axis position. The selection of either absolute or relative position format is controlled by bit 4 of DW, the display options word.

The various parameters governing position mapping can be summarised in the following equations. Unless noted all parameters apply to the slave axis.

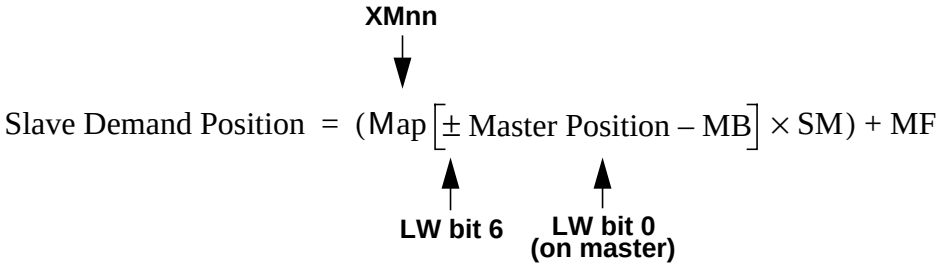
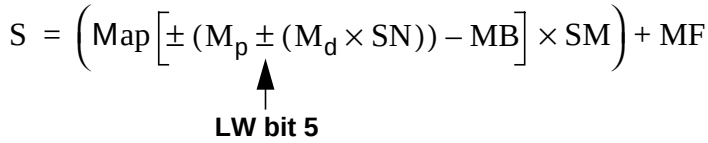


Figure 20. Position mapping as an equation



Where:
S.....Slave Demand Position
M_p.....Primary Master Position (ML)
M_d.....Differential Master Position (NL)
(notes on above figure apply as well)

Figure 21. Position mapping as an equation (differential)

Maps are stored on the host system until required by a specific channel. This allows a map common to more than one channel to be defined only once, thus saving memory space. The map data is automatically downloaded to the specific channel when required by the XM execute map command, or it may be manually downloaded to the current channel by using the TM transfer map command. This may be executed as part of the autostart sequence to avoid any delay the first time a map is executed. Maps executed from sequences are downloaded to their required channels as part of the sequence compilation process. If a DI input string contains an XM command, the specified map is transferred to that channel when the input line is defined. If the map has not been entered, then the transfer map fails, the “map is undefined or has not been transferred” error message is reported and the DI input line definition is aborted. Maps executed in sequences are automatically transferred to the relevant channels by the sequence compilation process.

The map step value, set by the MS command, is used to set the map table interpolation step size. If the map step value is greater than one, the EM command prompts the user for new slave position values at master position intervals of MS. The first entry in a map table is at master position zero. This allows the user to enter positions at larger master position intervals, and the system interpolates linearly between them when the map is executed. The map data is stored at the reduced resolution specified by MS, again saving memory space. When the map data is displayed using the LM list map command, it is listed at the default value of MS on the current channel, and the (possibly different) value of MS at which the data was entered is shown before the list of data.

Maps may be generated automatically by the optional Motion Generator package, if enabled by the appropriate software key. To make best use of the available memory in the system without reporting unnecessary “memory full” errors, maps with numbers above 200 are **not** saved to non-volatile memory with the SP command. These may be generated on demand and use working memory, rather than filling up the memory area allocated to the saved data.

EMnn **Enter map nn (restricted).** ~~***Q-Drive 1+1***~~
Range : 1 to 255

This command is used to enter position mapping tables into the system. The system gives a “Mnn xx:” prompt for each table entry, where “nn” is the map number, and “xx” is the entry number, corresponding to the master axis position for this entry. Each entry in the table is the (signed) absolute position on the slave axis at the current master channel position given by the table entry number. The master axis position is used as the index into the table to access a particular table entry. The value found at that table position is then the required slave axis position. To end the table, make a blank entry by entering just a <CR>, and the system then returns to normal operation. The mapping is accessed by means of the map number assigned by the user when it is entered.

Position map data may be entered up to the memory capacity of the system. If the system runs out of memory, it returns the “memory full” error message.

If bit 4 of DW is set to 1, then the map is entered as a table of offset values between the master axis and the slave axis positions. Each entry in the table is the (signed) difference between the master channel position and the required position of the slave channel. The master axis position is again used as the index into the table to access a particular table entry. The value found at that table position is the offset to be added to the master axis position to get the required slave axis position.

The map step value MS can be stored within the map table, ensuring that when the map is executed it always uses the intended value. To use this facility, the MS command is used as the first entry in the map table after the EM command, at the “Mnn 00:” prompt. This does not affect the default value of MS for the current channel, which is still used for listing the map with the LM command.

Example :

This example shows the entry of a map in absolute position format. It sets up a mapping with the slave position equal to twice the master position; this gives the effect of a 2:1 gear ratio between the slave and master axes.

System	User	Comments
1>	EM1<CR>	Enter map 1
Absolute mode		Position format
M1 0:	0	First slave position
M1 1:	2	
M1 2:	4	
M1 3:	6	
M1 4:	8	
M1 5:	10	
M1 6:	12	
M1 7:	14	
M1 8:	16	
M1 9:	18	
M1 10:	20	
M1 11:	<CR>	End map
1>		Normal prompt

The next example shows the same mapping entered in relative position format as a table of offsets, with the offset equal to the master position; this gives the same effect of a 2:1 gear ratio between the slave and master axes.

System	User	Comments
1>	EM1<CR>	Enter map 1
Relative mode		Position format
M1 0:	0	First offset value
M1 1:	1	
M1 2:	2	
M1 3:	3	
M1 4:	4	
M1 5:	5	
M1 6:	6	
M1 7:	7	
M1 8:	8	
M1 9:	9	
M1 10:	10	
M1 11:	<CR>	End map
1>		Normal prompt

LMnn **List map nn.** ~~***Q-Drive 1+1***~~
Range : 1 to 255

This command allows the user to examine map data previously entered into the system. The specified map is listed to the serial port, one map entry per line. Each entry is shown with its entry number, which represents the corresponding master axis position, followed by the slave position for that master position.

If bit 4 of DW is set to 1, the map is listed in relative position format as a table of position offsets between master and slave positions. The map entry value is the position offset added to the master axis position to generate the slave axis position.

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

Example : `LM1`
This will list map 1 on the display or terminal. The output for the example above would look like this, in absolute position format.

```
1>                LM1<CR>     User input to list map 1.
Absolute mode                 Position format
M1 0: 0
M1 1: 2
M1 2: 4
M1 3: 6
M1 4: 8
M1 5: 10
M1 6: 12
M1 7: 14
M1 8: 16
M1 9: 18
M1 10: 20
1>
```

This shows the same list map in relative position format.

```
1>                LM1<CR>     List map 1 command.
Relative mode                 Position format
M1 0: 0
M1 1: 1
M1 2: 2
M1 3: 3
M1 4: 4
M1 5: 5
M1 6: 6
M1 7: 7
M1 8: 8
M1 9: 9
M1 10: 10
1>
```


XMnn **Execute map nn.**
Range : 0 to 255 ***(Q-Drive 1+1 only allows XM0)***

This command puts the currently selected axis into the position mapping mode, where its demand position is calculated according to the specified mapping from another (master) channel's position. The master channel position is transferred to all slave channels; this is set up automatically when the slave channel is linked to the master channel with the ML map link command. 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.

NOTE : A map number of zero in the XM command is used for the special case of 1:1 mapping. If map zero is executed, the slave channel executes a 1:1 mapping, and it is not necessary to explicitly enter the 1:1 map data into the system. **This is the only option available for a Q-Drive 1+1**.

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 synchronised with the master axis using the Software Clutch.

The map data is downloaded automatically to the current channel when the XM command is executed for the first time. This means there is a short delay between issuing the command line containing the XM command and the actual execution of the map. This may be avoided if required by using the TM transfer map command to transfer the map data to the current channel in advance. Once the map has been sent to a channel, it remains there for further use until it is deleted or changed.

The action of the XM command is modified if tension control is on, as set by the AM1 command. When tension control is on, the map startup sequence is determined by AW bit 0. If AW bit 0 is set to 0, the slave axis goes immediately into mapping, using the software clutch, with the map ratio determined by the tension control loop. If AW bit 0 is set to 1, the slave axis moves at jog speed (set by SS) until the measured analogue input reaches the tension control setpoint, and then drops into

mapping. This is intended to initialise tension control when the master axis is stationary. The direction of the slow speed initial move is set byAW bit 6.

If either of AW bits 4 and 5 are also set to 1, then the unit automatically initialises the map scale factor before executing the map under tension control. The system measures the distance moved by the master and/or slave axes between the two positions where the analogue input high and low limits are exceeded. The ratio of these two values gives a good estimate of the initial map ratio required by the tension control loop and is used at the start of synchronisation. This allows the system to reach its correct steady state ratio much more quickly than if it starts from the default SM value, particularly if the system is not at its normal starting position.

An example where this is useful is a winding or unwinding application. In normal circumstances the system always starts with either a full or empty spool of material, and the initial scale factor is set in the SM parameter. Restarting the machine with the same spools uses the scale factor last calculated when the machine was stopped, and the tension loop restarts smoothly. However, if the machine may be started with spools that are partly filled to an unknown diameter, then the normal SM value is not at the correct value for running at the new spool diameter. The analogue range initialisation function allows the required ratio with the actual spool diameters to be measured. The machine may be started with the new spool without any large transients while the tension control loop stabilises from the initial default SM value to the new scale factor.

If the XM command is given with no parameter, then the system displays the number of the currently executing map. If no map is being executed, then the XM command returns zero.

TMnn **Transfer map.** ~~***Q-Drive 1+1***~~
Range : 1 to 255

This command tells the system to transfer the specified map to the current motor channel.

Before transferring the data to the channel, the system checks whether the channel has enough free memory to take it. If not, previously transferred maps or profiles are deleted from the channel until enough space is available. If there is still not enough room for the new map, then the system reports the “memory full” error message. Note that maps or profiles deleted from a channel in making room for a new transfer are not lost, as they are retained at host level and may themselves be transferred again.

MSnn **Set interval between position map entries.** ~~***Q-Drive 1+1***~~
Range : 1 to 65535
Default : 1

This command sets a map table step size for use when entering a simple mapping. If it is set to a value greater than 1, then when a map is entered, the user is required to only enter map data values at the given interval. The system interpolates between the values entered when the map is executed.

This function also reduces the size of memory required when saving the map table in nonvolatile memory.

Note that the interpolation step size set by the MS command also applies to profiles when entering them with the EP command or executing them with XP.

Example :
This example first sets a map step of 100, then sets up a mapping with the slave position equal to twice the master position as in the EM example. In this case the map table has been generated over a range of master axis positions from 0 to 1000 with only ten entries by the user. This map also only requires one hundredth of the storage space of the equivalent full resolution map table.

System	User	Comments
1>	MS100	Set map step to 100
1>	EM1<CR>	Enter map 1
Absolute mode		Position format
M1 0:	0	First slave position
M1 100:	200	
M1 200:	400	
M1 300:	600	
M1 400:	800	
M1 500:	1000	
M1 600:	1200	
M1 700:	1400	
M1 800:	1600	
M1 900:	1800	
M1 1000:	2000	
M1 1100:	<CR>	End map
1>		Normal prompt

MLnn **Map link slave axis to master axis nn.**
Range : 1 to same as no. of channels, but not this channel

This command links the current channel to the specified master channel in preparation for execution of a position mapping. Note that a slave channel must be linked to some master channel before executing a map, or the slave channel receives no master position data and gives the “map position update timeout” or “not linked” error message.

If the ML command is given with no parameter, the system prints the channel number of the master axis for the current channel, if it is linked.

On some PTS products, an extra logical channel is available in addition to the real channel(s). This extra channel may be used as a virtual axis, for example in mapping or when using the Software Differential, or it may be used as an undriven encoder axis if the hardware interface exists.

NLnn **Map link slave axis to differential master axis nn.**
Range : 1 to same as no. of channels, but not this channel

This command links the current channel to the specified master differential channel in preparation for execution of a software differential position mapping; see LW and the beginning of this chapter for more information.

If the NL command is given with no parameter, the system prints the channel number of the differential master axis for the current channel, if it is linked.

UL **Unlink slave axis from master axes.**

This command is used to unlink a slave axis from its master axes after finishing a position mapping. Note that channels may be left linked to their master axis if required, while other commands are executed. It is only necessary to use the UL command if the slave channel needs to be linked to a different master axis.

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

This command allows the user to modify the operation of the map link function. It also enables the "Software Differential" function. 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 link options word are described below.

Bit 0 Master axis:
This bit controls what data this axis transmits when it is a master (or differential master) axis. Normally the slave is set up to follow the master demand position, as this gives smoother motion on the slave axis. Sometimes it is more useful to set up the slave axis to follow the measured master position, particularly if the master axis has a large following error.
When set to 0, the demand position of this axis is transmitted to all slave axes.
When set to 1, the measured position of this axis is transmitted to all slave axes.

Bit 1 Master axis:
When this bit is set to 1, the master position sent to any slaves includes data from any reference corrections on the master axis. This allows the slave axis to perform the same movement as the master axis during reference correction, for example where the two axes are at each end of a flexible web material such as in a printing application.
When set to 0, reference corrections on the master axis are not seen by any linked slave axis, unless reference correction is suppressed on the master axis, or the master axis is in the motor off state.

Bit 2 Master axis:
When set to 1, this bit causes a 1 tick (4ms) delay to be applied to master position data for slave axes on the same hardware (board or module) as the master axis. This makes local slave axes behave identically to remote slave axes on separate boards or modules, which always have a 1 tick delay relative to the master axis.
When this bit is set to 0, there is no time delay between local slave and master axis positions.

Bit 3 Reserved.

Bit 4 Slave axis:
When this bit is set to 1, the Software Differential function is enabled. The master position data used by this channel is the sum or difference between the positions of two separate master channels. The slave axis is linked to the first master axis with the ML command and to the second master axis with the NL command.
When set to 0, the Software Differential function is disabled.

Bit 5 Slave axis:
This bit determines whether the current channel uses the sum or difference between the two master channel positions when the Software Differential is enabled by setting LW bit 4.
When set to 1, the differential master channel position is subtracted from the first master channel position.
When set to 0, the first and differential master channel positions are added together.

Bit 6 Slave axis:
When this bit is set to 1, the master position data (or the result of the differential sum or difference) is negated before being used by the current channel.
When set to 0, the master position data is used unchanged.

Bit 7 Reserved.

MTnn **Set map position update timeout (restricted).**
Range : 1 to 255
Default : 1 or 2 (depends on system)

When a slave channel is executing a position mapping, it expects to receive continuously a position update from the master channel. If for any reason the master axis position is not received by the slave axis controller module within the normal servo update time, the slave channel extrapolates the master position (assuming constant master velocity) and uses this to calculate its required slave position. This gives good performance in practice, and allows for variations in the host processor response time in sending the master position to all slave channels. The MT command is used to set a timeout on the master position update, specified in 1/256 second ticks (about 4ms). If the master position is not received within the specified number of ticks, the slave channel goes to the motor off state, and the system gives the “map position update timeout” error message. It should not occur during normal system operation, and may indicate that there is a problem with the hardware or the system configuration.

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 (resultant position after differential and SN scaling have been applied) before using the data for the map input. 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. It is used on the slave channel, **not** on the master. 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 realised, 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.

Note that the SM parameter is now also used to scale profiles executed with the XP command.

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.

SNn/n **Scale mapping for differential.**

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 differential mapping. It is used on the slave channel, not on the master. The value of SN is applied to the differential master position value before it is added to or subtracted from the primary master position value in the software differential. See the discussion at the beginning of this chapter for a mathematical representation of this.

Note 1. SN is stored in the PTS to 24-bit fractional precision, thus to avoid long-term errors SN should be exact at this precision.

Note 2. The scaling is applied at each crossing of the first master axis bounds; the remainders from scaling at this point are also accumulated to a 24-bit fraction only.

Note 3. Any accumulating errors are in the slave demand, thus referencing does not adjust for them.

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/MP. Because the scale factor depends on the value of MP, the BR command should only be executed after MP has been defined. This is normally done by executing the ML command. The correct sequence of operations is shown in the following example.

Example :

System	User	Comments
1>	SB4000	Set master axis bounds
1>	CH2	Change to slave channel
2>	SB5000	Set slave bounds
2>	ML1	Link slave to master axis
2>	BR	Calculate scale factor
2>	XM0	Execute map zero

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 ΔSM 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 ΔSM/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 pass the master channel position bound value to any slave channels.

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.

The master bound is set up automatically by the ML map link function, when a slave channel is linked to a master channel. However, if the master axis bound is changed while any channels are linked, each slave channel must be sent the new value of the master axis bound.

NPnn **Set differential master position bound**
Range : 1 to 4 000 000 (4.0E6)
Default : 4 000 000

This command is used to pass the differential master channel position bound value to any slave channels.

NOTE : It is essential for the slave channel to have this information so that the mapping is continuous at the point where the differential position wraps around to zero, that is, the differential bound position.

The differential bound is set up automatically by the NL map link function, when a slave channel is linked to a master channel. However, if the master axis bound is changed while any channels are linked, each slave channel must be sent the new value of the differential axis bound.

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 map 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. The alignment move may be forced to execute in one direction by setting MW bits 2 and 3 as required.
When set to 1, the "Software Clutch" facility is enabled. This is used when it is required to synchronise 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 master axis distance for this clutch acceleration ramp is defined by the CL clutch length 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 to optimise the map startup alignment move or clutch delay distance.
When set to 0, no optimisation is applied. If using an alignment move (MW bit 0 = 0), its target position is taken directly from the result of the mapping. The alignment move distance may be larger than the set bound value SB if required. If using the software clutch (MW bit 0 = 1), the distance through which the master axis moves before the clutch ramp starts depends only on the mapping, and may be larger than the master SB value.
When set to 1, the map startup is adjusted to give the shortest possible alignment move or clutch delay distance. If using an alignment move (MW bit 0 = 0), the target position for the alignment move is set to the **nearest** correct cyclic position within SB. Note that the alignment move is still subject to the direction constraints set by MW bits 2 and 3. If MW bit 2 = 0, then the move distance is limited to ±SB/2. If MW bit 2 = 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. If using the software clutch, then the distance through which the master axis moves before the clutch ramp starts is limited to one master bound.

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 mapping starts with the software clutch, such that 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 This bit enables the subtraction of the master axis position from the value found in the map table, and the map table value is used as an offset or correction to give the slave demand position. This option is provided mainly for compatibility with older systems.
When set to 1, the master position is subtracted from the map table value. The map table values represent the slave axis deviation from the master axis position.
When set to 0, the master axis position is not subtracted from the map table value, as normal. The map table values define the absolute slave axis position.

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. It is particularly useful in conjunction with the SM map scaling function, when it is not always possible to define the mapping such that the mapped master bound corresponds to the physical slave axis cycle length. In this case, set this bit to automatically set the slave axis bounds to the mapped master axis bound, and use the auto-referencing facilities to correct the motor to the actual cycle length on each cycle.
When set to 0, the bound value as set by SB is used as normally while in mapping.

Bit 7 Reserved.

CLnn **Set clutch length.**
Range : 0 to 4194304
Default : 0

This command sets the acceleration ramp length 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 XM execute map command. Instead, it goes into a holding state and waits for the master position to approach the current slave position. The system calculates the effect of a linear speed ramp on the master axis, and uses this projected ramp to find the required slave axis position. When this projected slave axis ramp start position passes the current slave position, the slave motor accelerates up to speed such that it reaches the required speed at the correct position relative to the master axis. The length of the clutch ramp is set by the CL clutch length parameter. This defines the distance on the master axis over which the slave axis accelerates from rest to the required mapped speed. Note that although CL represents a length in master axis counts, it is set on the slave axis.

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 : `CL2000`

This example shows the clutch length set to 2000 counts. Thus the slave axis will ramp up to the required mapped speed and position over a master axis distance of 2000 counts.

BAnn **Set map base advance.**
Range : 0 to 65535 (units of 15.3μs)
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.

Another use for it is when the master and slave axes should both complete one machine cycle in the same time, but the value for the bound position on the two axes is different. In this case, if the internal mapped master bound is not the same as the slave axis bound value, then the slave axis will appear to drift relative to the master axis as the machine runs. This is because the mismatch between the slave axis bound and its mapped copy of the master axis accumulates from cycle to cycle. In this case it is necessary to adjust the map table values at the master axis bound position to get the correct result.

On some systems this may not be possible because of interactions with the map scale factor or master axis division factor. In such cases, it is possible to use the value for the mapped master axis position bound together with auto-referencing on the slave axis to solve the problem. The value for the mapped master bound is read by using the GM command, and this value is sent as the bound value with the SB command. In this way, the master and slave bound values tie up exactly, and any mismatch between the slave bound value and the actual repeat length of the slave encoder is handled by the auto-referencing functions. If required, this may be performed automatically by setting bit 6 of MW, the map options word.

NOTE : The GM value is only valid when the master axis bound has been set up correctly by the ML map link command, or has been sent with the MP command, and the slave channel is executing a map.

GN **Get mapped differential axis bound position.**

This command allows the user to find the slave position which corresponds to the differential master axis bound position when executing a position mapping. It performs the same function as GM, but for the NL differential map link.

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.

FM **Display free memory.**

This command displays information about the memory space available for sequences, maps and profiles. For more details refer to the full description of the FM command on page 52.

4.9 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.

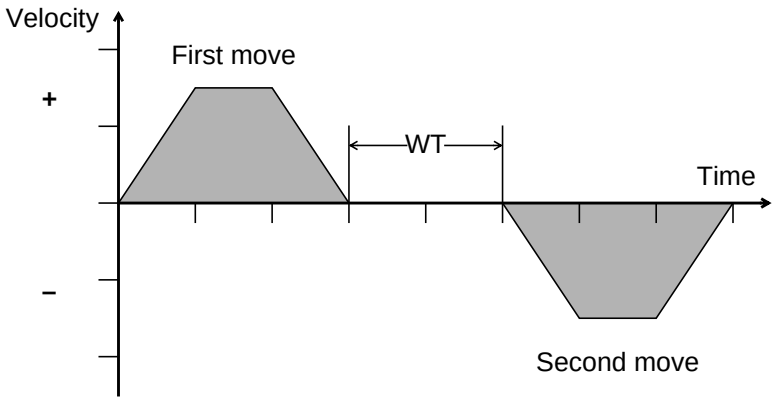


Figure 22. Wait for time.

WI[g:]n± **Wait for input line nn in group g.**
Range : 1 to 8

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.

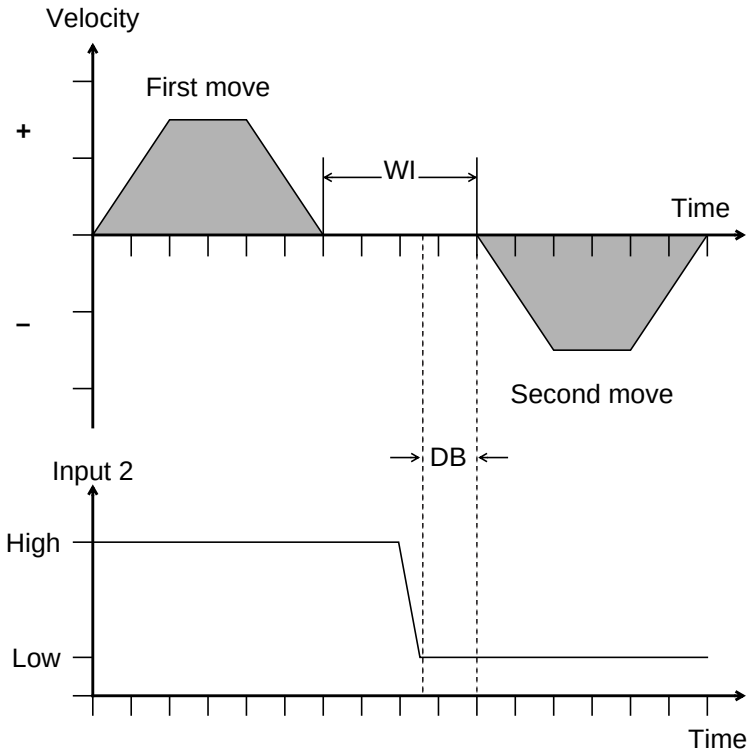


Figure 23. Wait for input line.

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.

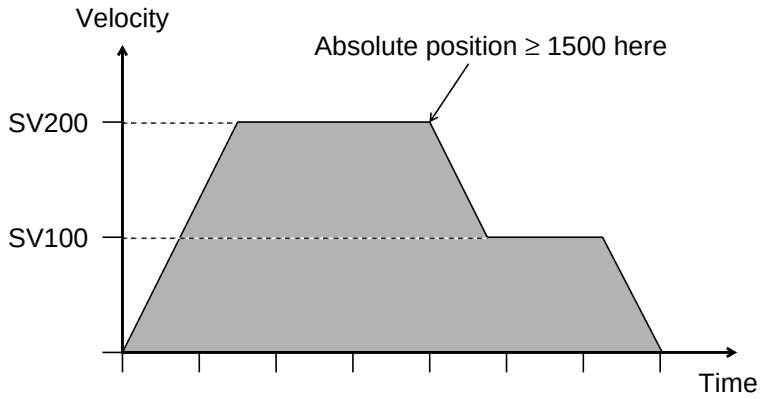


Figure 24. Wait for absolute position.

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 occurred last.

Entering position control mode	(PC)
Setting the current position	(ZC)
Start of a move	(MA, MR or VC)
Start of mapping	(XM)
Start of a profile	(XP)
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)

To reset the wait relative counter without using one of these other functions, use either WE or WT0.

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.

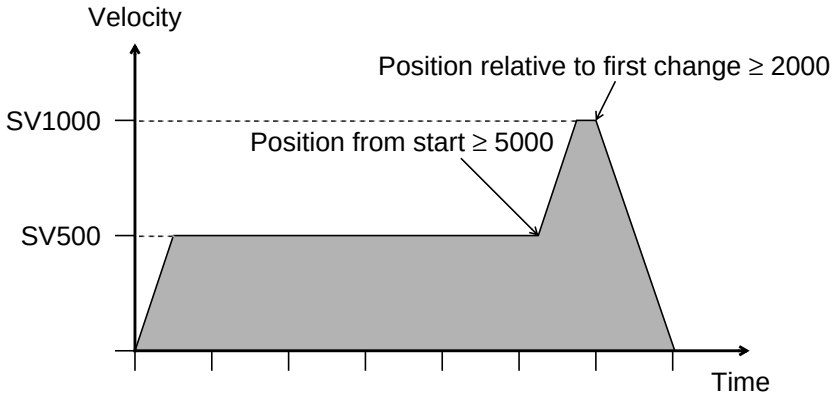


Figure 25. Wait for relative position.

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.

4.10 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 LED display. The motor error conditions are as follows.

- Motor position error.
- Motor timeout.
- Limit switch input detected.
- Motor position outside high or low position limit.
- Map update timeout.
- 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.
- Analogue input outside high or low limit.

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 4.3 for details of the MO motor off and PC position control commands. The value is defined in user units.

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 or encoder 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.

LHnn **Set high position limit (restricted).**
Range : ± 4 000 000 (4.0E6)
Default : + 4 000 000

This command sets up a user-defined limit position. If at any time the absolute position of the motor exceeds the high position limit, the system gives the "high position limit exceeded" error message and goes to the motor off state. This is similar to the action taken on detecting a limit switch input. The value is defined in user units. If the SB set bound value is less than the high limit, then the high position limit checking is disabled, as the absolute position value wraps around to zero at the bound position before reaching the high limit position.

LLnn **Set low position limit (restricted).**
Range : ± 4 000 000 (4.0E6)
Default : – 4 000 000

This command sets up a user-defined limit position. If at any time the absolute position of the motor is less than the low position limit, the system gives the "low position limit exceeded" error message and goes to the motor off state. This is similar to the action taken on detecting a limit switch input. The value is defined in user units. If the SB set bound value is less than the low limit, then the low position limit checking is disabled, as the absolute position value wraps around to zero at the bound position before reaching the low limit position.

Example : `LH10000/LL0`
This sets the high position limit to 10000 units, and the low position limit to zero. If the motor position goes outside the range 0 to 10000 units, the system gives the appropriate error message and goes to the motor off state.

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.

MTnn **Set map position update timeout (restricted).**
Range : 1 to 255
Default : 1 or 2 (depends on system)

When a slave channel is executing a position mapping, it expects to continuously receive data from the master channel. MT provides a timeout check on this. Refer to page 82 for a full description of the MT function.

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).

UEnn **Set user error sequence (restricted).**
Range : 0 to 255
Default : 0

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

EAnn **Set motor error/motor off analogue ramp time (restricted).**
Range : 0 to 2560 (× 4ms)
Default : 0

This command sets up a time for the analogue output to ramp linearly to zero on a motor error or motor off command. The default of 0 means that the analogue output is set to zero immediately. Any non-zero value gives a smooth ramp down from the current analogue output value to 0V over the set time, instead of a sudden stop. The drive enable relay is held on until the end of the ramp time. This function may be used when an abrupt stop would cause problems because of mechanical limitations.

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 When set to 1, the analogue input out of limits error is treated as a motor error, and the axis goes to the motor off state when it occurs.
When set to 0, the analogue input out of limits error is treated as a user error, and the system simply prints an error message.

Bit 4 When set to 1, reference timeout errors are suppressed, unless they are set as motor errors by EW bit 0.
When set to 0, reference timeout errors are displayed.

Bit 5 When set to 1, reference limit errors are suppressed, unless they are set as motor errors by EW bit 1.
When set to 0, reference limit errors are displayed.

Bit 6 When set to 1, reference correction overrun errors are suppressed, unless they are set as motor errors by EW bit 2.
When set to 0, reference correction overrun errors are displayed.

Bit 7 When set to 1, analogue input out of limits errors are suppressed, unless defined as motor errors by EW bit 3.
When set to 0, analogue input out of limits user errors are displayed.

LE[n] **Display last error(s).**

This command redisplays the error message(s) for the last error(s) 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.

The system continuously logs the last ten errors. The LE command takes an optional parameter which specifies how many errors (up to ten) to display. If no parameter is given, the most recent error is displayed.

4.11 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.

$$V_{out} = KP e_i + KI \sum e_i + KD(e_i - e_{i-1}) - KV(p_i - p_{i-1}) + KF(d_i - d_{i-1}) + KA[(d_i - d_{i-1}) - (d_{i-1} - d_{i-2})]$$

where KP = proportional gain constant
KI = integral gain constant
KD = differential gain constant
KV = velocity feedback gain constant
KF = velocity feedforward gain constant
KA = acceleration feedforward 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 any axis may be monitored by using another 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.

KDnn **Set differential gain constant (restricted).**
Range : 0 to 65535
Default : 0

This command sets the gain for the differential term in the controller transfer function. This term uses the differential of the position error (=rate of change of error), which represents the velocity error of the system. This is useful where the position error is changing rapidly, for example if the required motion is a step change in position.

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 feedforward gain constant (restricted).**
Range : 0 to 65535
Default : 256

This command allows the user to set the gain for the velocity feedforward 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 feedforward 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 feedforward gain. Alternatively, velocity lag may be reduced to zero by the use of integral gain, but this has other effects as well.

KAnn **Set acceleration feedforward gain constant (restricted).**
Range : 0 to 65535
Default : 0

This command allows the user to set the gain for the acceleration feedforward term in the controller transfer function. It uses the demand acceleration as opposed to the measured acceleration, and is useful when following a set position or velocity profile. The effect of KA is to provide a component of the output signal proportional to the required demand acceleration.

DK **Display system constants.**

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

- Proportional gain constant KP
- Velocity feedback gain constant KV
- Velocity feedforward gain constant KF
- Acceleration feedforward gain constant KA
- Scale units SU
- Velocity SV
- Acceleration SA
- Maximum position error SE

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 17
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	KD
6	Absolute demand position	DD
7	Absolute measured position	DP
8	Tension loop error	
9	Actual map scale factor	
10	Averaged measured speed	DV
11	Master speed	
12	Master averaged speed	
13	Tension control setpoint	AC
14	Reference error	DF
15	Snapshot position	DS
16	Demand velocity including any reference correction	
17	Demand acceleration	KA

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 (Q-Drives, MiniPTS 1+1)
0 to 4 (MiniPTS 2+1/3, MiniPTS 4 and PTS Mk2)
Default : 0

This command allocates the monitor signal for the current channel to one of the analogue outputs on the same board. For PTS Mk2 and SERVOnet systems, note that AO cannot be used between different servo controllers. The monitor function SF may be defined for the current channel 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.

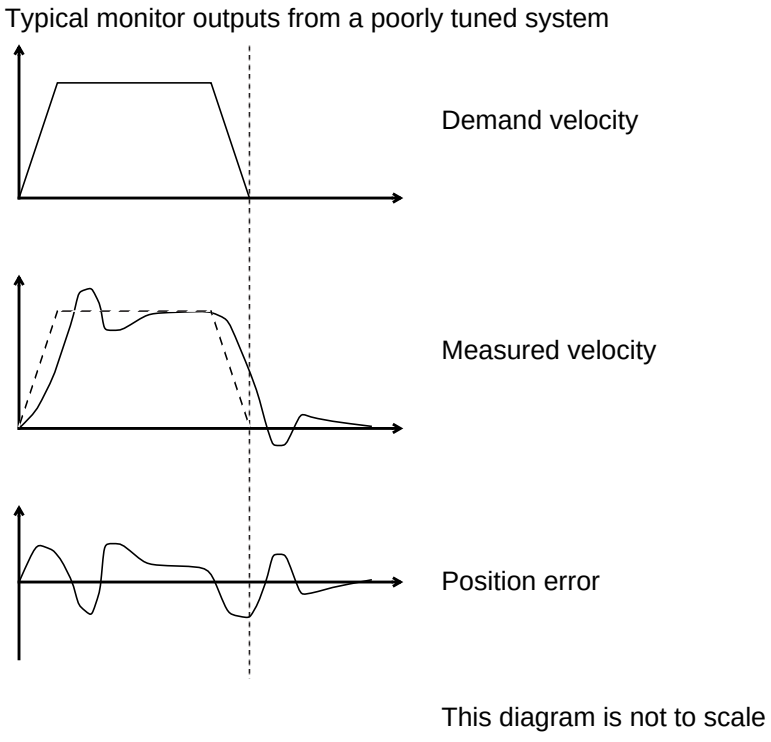


Figure 26. Monitor output functions.

4.12 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. 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.

Note that it is now possible to use referencing and snapshot functions on axes in virtual mode (VM1). They use interpolation to accurately calculate the position of the virtual axis at the point of the reference or snapshot input.

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. It may be necessary on Q-Drives to use ZC at startup, since they save position at power off.

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.

If the SB command is used on the master axis during mapping it is important to ensure that the slave channels are made aware of the change. This can be done by unlinking and linking each slave using the UL and ML commands, or by using the MP command on each slave. See section 4.8 for more details.

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.

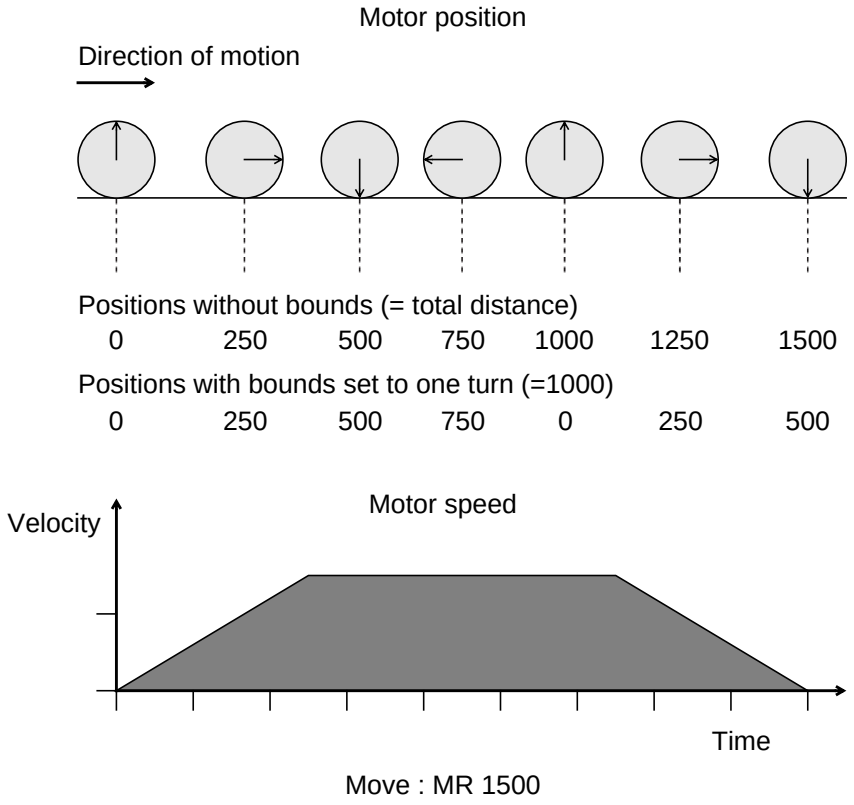


Figure 27. Position bounds.

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 : 0

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 initialise position command is executed, and when the automatic reference functions are enabled by the RM and RW commands.

Q-Drive systems normally use the resolver data from the drive for position feedback. In this case, if the IN command is executed with no reference input or marker pulse defined, it simply resets the current position to the resolver position data value returned by the drive module, without performing any movement. Once the position is reset, an MA0 command will move the motor to the new zero position. If DZ1 is set on a Q-Drive system, then the IN command performs this move to zero in order to simulate initialisation to an encoder marker pulse. If a DR input is defined, then the system performs a normal initialisation to move and search for the reference position.

DR[g:]n± **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 initialise 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 When set to 0, the minimum reference correction speed is greater than zero (actually 1 count per tick or 256 counts per second). This is to ensure that any reference correction will complete, even if the axis comes to a stop before the correction is finished.
When set to 1, the minimum reference correction speed is allowed to fall to zero if the motor speed is zero. This is used where it is undesirable for the motor to creep slowly to finish a correction when the axis is stopped.

Bit 2 When set to 0, the RA output signal is triggered when the reference is detected as valid.
When set to 1, the RA output signal is triggered at the start of the reference correction. If the correction is delayed by setting the RJ parameter, then the RA output is also delayed.

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 Enables (1) or disables (0) the reference reject output (RR function, see page 144) when a reference input fails the width test defined by the ZH, ZL, FH and FL parameters.

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 correction is enabled by bit 0 of RW, and the reference error is inside the limit defined by the SR command, then the position is corrected by this reference error. If the reference error is greater than SR (when SR>0), then the position is corrected by an amount equal to SR. Note that if the reference error is greater than FR (when FR>0) then the reference is ignored.

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 (when LR>0), 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.

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 effective value used for RJ is the remainder when RJ is divided by SB (or RL). It is not possible to defer the correction until a position beyond the next reference position.

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 initialise 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.

ZHnn **Set reference true high limit (restricted).**
ZLnn **Set reference true low limit (restricted).**
FHnn **Set reference false high limit (restricted).**
FLnn **Set reference false low limit (restricted).**
Range : 0 to 65535
Default : 0

These four parameters define limits on the width of the reference input signal seen by the DR input. They allow the system to respond to a valid reference signal only if it matches the width limits given. The reference input is accepted as valid only if it is false for a distance which is between FL and FH, and it is then true for a distance between ZL and ZH. The input is therefore recognised as valid on the trailing edge of the reference signal, when it switches from true to false. However, the reference error for initialisation and position correction purposes is calculated relative to the leading edge of the input signal, when it switches from false to true, as defined by the DR command. The distance limits are specified in encoder counts.

To disable any threshold test, set the limit value to zero. For example, to check that the reference input is false for at least 200 counts, with no maximum false distance, set FL to 200 and FH to 0. To disable reference width checking completely and return to normal operation, set all four values to zero.

Summary : reference input is accepted if
ZL <= distance with DR input true <= ZH
and FL <= distance with DR input false <= FH.

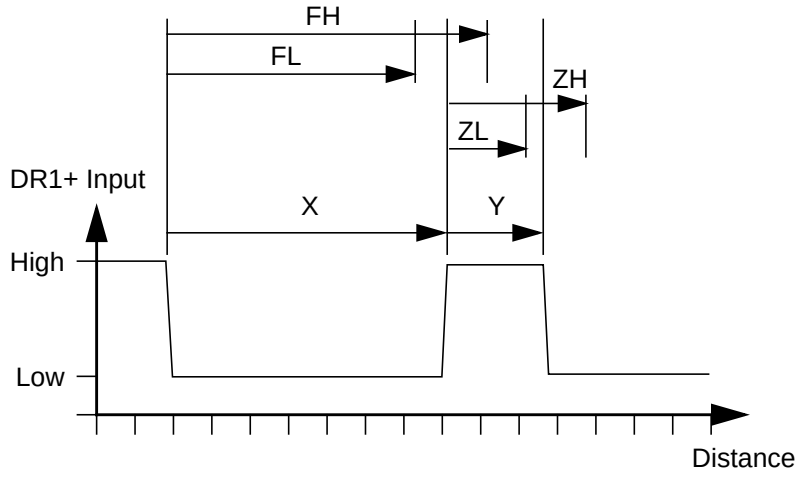


Figure 28. Reference width checking with ZH, ZL, FH and FL.

This feature is used to pick out a valid reference pattern from other signals that may be detected by the reference sensor. It is of particular use in printing and registration applications.

RA[g:]n± **Define reference accepted output (restricted).**
Range : 1 to 8

This command defines the specified output line as a reference accepted output. The output gives a 4ms pulse when a reference input signal has been accepted as a valid reference. This may be used to indicate the presence of a good product, for example when using reference width checking. This command is restricted, and is only available in privileged mode.

If the RJ parameter is set to delay correction for some distance after the reference is detected, and RW bit 2 is set to 1, then the RA output signal is also delayed until the start of the deferred correction.

RHnn **Set reference holdoff time (restricted).**
Range : 0 to 127
Default : 0

This command sets up a holdoff time on the reference input. When RH is set to a non-zero value, the reference holdoff function is enabled. When a reference input is detected, no further reference inputs are recognised until the holdoff time has elapsed. The reference holdoff function is disabled by setting RH to zero.

The reference holdoff function previously used the DB debounce time parameter, and was enabled by RW bit 6. The reference holdoff is now set by RH, independent of the DB parameter, and RW bit 6 is no longer used.

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.

4.13 Digital Inputs and Outputs.

The PTS has a number of digital inputs and digital outputs on each unit. 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.

The input and output lines on the PTS are divided into groups of eight lines, and all commands for the input and output lines take a group number prefix before the line number. This is to provide a more general input and output line numbering scheme for use across the PTS range, as different products support different numbers of I/O lines.

The extended command syntax prefixes the line number with the group number, and uses a ':' colon character to separate them. Commands which may be applied to all inputs or outputs on other units, such as RI or RO, are now applied to a group of inputs or outputs. Examples are shown below. If the group number is not specified in the command, then it is calculated from the line number to provide some backwards compatibility.

Examples :

`SO1:2`	Set output 2 in group 1 (output 2).
`SO1:`	Set all unused outputs in group 1 (outputs 1-8).
`CO2:5`	Clear output 5 in group 2 (output 13).
`RI1:1`	Read input 1, group 1 (input 1).
`RI2:1`	Read input 1, group 2 (input 9).
`RI1:0`	Read all inputs in group 1 (inputs 1–8).
`RI15`	Read input line 15 (input 7 in group 2).
`MI3:7`	Mask input line 7, group 3 (input 23).
`EI4:`	Enable all inputs in group 4 (inputs 25–32).

Group number	Input lines	Output lines
1	1–8	1–8
2	9–16	9–16
3	17–24	
4	25–32	
5	33–40	

Note: not all these I/O lines/groups are available on all PTS units.

On the Mini Machine Controller input groups 8 and 9 and output group 8 are also available corresponding to the digital I/O on the Mini Machine Controller itself. If the CANopen, Devicenet or Profibus options are enabled input and output groups 10 to 17 are available corresponding to the digital I/O for the relevant option. These additional groups are known as Host I/O since they are implemented at host level and are system wide. In other words on a Mini Machine Controller there is only one group 8 for the whole system, not one for each node as is the case for other groups. Only a subset of the I/O commands are available for use with Host I/O as follows:

BI, DI, EI, II, LI, MI, RI, CO, IO, LO, RO, SO

The PTS has a “virtual i/o” feature (MiniPTS 4 and PTS Mk2: if no I/O expansion board fitted). This makes available one extra input group and output group which are not connected to any real signals. The current states of the virtual input lines are set by the virtual output lines, as if the outputs are connected to the inputs. This allows output functions such as the PO position trigger function to be attached to an input function such as DI without requiring any real inputs and outputs.

The virtual i/o groups are numbered one higher than the maximum real i/o groups. All input and output commands are available on the virtual inputs and outputs, with the exception of reference and snapshot functions. The virtual inputs are not subject to debounce. They may be used with the timer/counter functions described later (in section 4.15 on page 150) to give very powerful combinations.

It is possible to use variables with these commands, although some care is needed to deal with the group:line numbering scheme. To use a variable in place of the group:line number in a command such as SO or CO, the value of the variable should be constructed by adding the line number to 256 × the group number, as follows:

```
$GRP=1 / $LIN=3
$OUT=(($GRP*256)+$LIN)
SO$OUT       # Sets output 1:3
```

The result of an RI or RO command can also be assigned to a variable. This allows the state of a single i/o line or a group of lines to be read into a variable and used in later calculations or decisions. Some examples are shown here:

```
$RI5=RI1:5   # Read input 1:5 into $RI5 (gives 0 or 1)
$RG2=RI2:    # Read input group 2 into $RG2 (0 to 255)
$RG1=RO      # Read default output group 1 into $RG1
$RO8=RO8     # Read output 1:8 into $RO8
```

To use a variable with the OC output code command, add 256 × the group number to the code value required. For example, to output a code value of 9 on an OX group defined on group 2:

```
$GRP=2 / $COD=9
$OUT=(($GRP*256)+$COD)
OC$OUT       # Sets output code 9 on group 2
```


SO[g:][n] **Set output line n in group g.**
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. If a group number and colon are given with no line number, or the line number is zero, then all available outputs in that group are set. No error is reported if all outputs are in use.

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

CO[g:][n] **Clear output line n in group g.**
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. If a group number and colon are given with no line number, or the line number is zero, then all available outputs in that group are cleared. No error is reported if all outputs are in use.

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

RI[g:][n] Read input line(s) in group g.
Range : 1 to 8, 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 after the group number and colon, or the line number is zero, then the system prints the current state of all 8 input lines in the specified group and a letter to show whether each line is masked (M) or enabled (E). If no group number is given, the command defaults to group 1 (inputs 1–8).

Example : `RI1:2`
This reads the state of input line 2, and prints it on the display.

System	User	Comments
`>`	`RI1:2`	Read a particular input
`0`		Input line 2 is low
`>`		Normal prompt

Example : `RI`
This reads the states of all inputs in group 1, and prints them, together with their mask/enable status.

System	User	Comments
`>`	`RI`	Read all inputs in group 1
`1 2 3 4 5 6 7 8`		Line numbers
`0 1 1 0 0 1 0 0`		Lines 2, 3, 6 are high
`E E M E M E E E`		Lines 3 & 5 are masked
`>`		Normal prompt

RO[g:][n] Read output line state(s) in group g.
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 after the group number and colon, or the line number is zero, then the system prints the current state of all 8 output lines in the specified group. If no group number is given, the command defaults to group 1 (outputs 1–8).

Example : `RO1:`
This reads the states of all outputs in group 1, and prints them.

System	User	Comments
`>`	`RO1:`	Read all outputs in group 1
`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

OC[g:]nn **Output code via expanded outputs.**
Range : 0 to maximum value possible on expanded outputs.

This command sets the expanded output lines for the current channel (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.

To use a variable to set the OC output code, add 256 × the group number to the code value required.

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

II[g:]n± **If input 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 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.

The II command may be followed by the EL command. If the input line is true the system executes the commands following II up to the EL command or end of line, whichever comes first. If the input line is false the system executes the commands following EL up to the end of line. The EL command must appear on the same line as the II command, except in a sequence, where it may also be the first command on the line following II.

Example : `II2:2-/MR-1000`
This example shows a conditional relative move. If input line 2 in group 2 is low, then the motor moves relative –1000 counts.

IO[g:]n± **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 : `IO1:3+/XS25`
This example shows a conditional sequence. If output line 3 is high, then the system executes sequence 25.

MI[g:][n] **Mask function input.**
Range : 1 to 8, 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 a group number followed by a colon is given with no line number, or the line number is zero, then all function inputs and/or expanded inputs in the specified group are masked. If no line number or group number is given, then all function and expanded inputs in all groups are masked.

BI[g:][n] Inhibit function input.
Range : 1 to 8, 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 a group number followed by a colon is given with no line number, or the line number is zero, then all function inputs and/or expanded inputs in the specified group are inhibited. If no line number or group number is given, then all function and expanded inputs in all groups are inhibited.

EI[g:][n] Enable function input.
Range : 1 to 8, 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 a group number followed by a colon is given with no line number, or the line number is zero, then all function input and/or expanded inputs in the specified group are enabled. If no line number or group number is given, then all function and expanded inputs in all groups 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 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 command, or to mask or inhibit the WI command.

GB **Global inhibit.**

This command is used to disable all defined function inputs, timer/ counter inputs, and expanded inputs. The host system uses internal MV and GB commands to inhibit all trigger variables and input lines during the execution of the autostart sequence.

GE **Global enable.**

This command is used to enable all defined function inputs, timer/ counter inputs, and expanded inputs. The host system uses internal EV and GE commands to enable all trigger variables and input lines after the autostart sequence has finished.

Any MI, BI or EI commands in the autostart sequence have no effect until the end of the autostart sequence, when the automatic global inhibit is removed (or until a GE command is executed). If a GB command is used in the autostart sequence, then all inputs will remain inhibited after it has finished executing. If a GE command is executed in the autostart sequence, then inputs are enabled from that point, provided they have not been explicitly masked or inhibited with MI or BI commands.

WI[g:]n± **Wait for input line nn.**
Range : 1 to 8

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.

4.14 Configuration Commands

The input and output lines on the PTS are divided into groups of eight lines, and all input and output configuration commands take a group number prefix before the line number. The extended command syntax prefixes the line number with the group number, and uses a ‘:’ colon character to separate them. Commands which may be applied to all inputs or outputs on other units, such as LI, are now applied to a specified group of inputs or outputs. Examples are shown below. If the group number is not specified in the command, then it is calculated from the line number to provide some backwards compatibility.

Examples :

`DL7+`	Define limit switch on input 7 in group 1.
`DI3:5+`	Define function on input 5 in group 3 (input 21).
`DE2:1-`	Define error signal on output 1, group 2.
`MG2:0011`	Define mask group for input group 2.
`LI2`	List input definitions for group 2 (inputs 9–16).
`LO1`	List output definitions for group 1 (outputs 1–8).

Group number	Input lines	Output lines
1	1–8	1–8
2	9–16	9–16
3	17–24	
4	25–32	
5	33–40	

Note 1: not all these I/O lines/groups will be available on all PTS units.

Note 2: it is now possible to use referencing and snapshot functions on axes in virtual mode (VM1). They use interpolation to accurately calculate the position of the virtual axis at the point of the reference or snapshot input.

The PTS has a “virtual i/o” feature (MiniPTS 4 and PTS Mk2: when expansion board not fitted). This makes available one extra input group and output group which are not connected to any real signals. The current states of the virtual input lines are set by the virtual output lines, as if the outputs are connected to the inputs. This allows output functions such as the PO position trigger function to be attached to an input function such as DI without requiring any real inputs and outputs.

The virtual i/o groups are numbered one higher than the maximum real i/o groups. All input and output commands are available on the virtual inputs and outputs, with the exception of reference and snapshot functions. The virtual inputs are not subject to debounce. They may be used with the timer/counter functions described later (in section 4.15 on page 150) to give very powerful combinations.

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

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 initialise 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 4.12, Reference Commands, on page 113.

DR[g:]n± **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 initialise position command is executed, and when the automatic reference functions are enabled by the RM and RW commands. See section 4.12, Reference Commands, on page 113 for more details on the reference facilities and commands. Note that only inputs 1 to 4 in group 1 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.

DL[g:]n± **Define limit switch input (restricted).**
Range : 1 to 8

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 : `DL3:7–`
This defines input line 7 in group 3 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 in group 1, previously defined as a limit switch input, to normal operation.

PS[g:]n± **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 in group 1 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.

DI[g:]n±/... Define function input (restricted).
Range : 1 to 8

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.

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

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

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

DX[g:]n± **Define expanded input lines (restricted).**
Range : 0 to 7

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 8 in the specified group is used as a strobe or trigger input. The sense of the strobe input is given by the sign after the parameter. Units with more than one input group may not use group 1 for the expanded input function, to avoid clashes with any requirement for position reference inputs, which are only available on inputs 1-4. On other input groups, lines n up to 8 are used for the expanded inputs, where n is the line number given in the DX command. To reset the expanded input definition, use “DX g:0” or “DX g:n” without the sign.

On detecting the strobe input, the remaining input lines from line 7 (most significant bit) down to the line number specified in the DX command are read as a binary code. On the PTS, it is used as a signal number, and the specified user signal is sent to the host system immediately. The host system can then take appropriate action as required. Typically, a number of sequences are assigned to execute on the appropriate range of user signals. This allows a maximum of 127 possible different operations to be controlled by the 8 input lines allowed for the DX function. 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 signal to be sent to the host system. This keeps the strobe input and the data inputs consistent.

Only one set of DX inputs is allowed on any one channel or input line group. Since the user signals are channel specific, this prevents any ambiguity in identifying the source of the user signal when the DX strobe input is triggered.

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 : `DX2:3-`

This sets up an active low expanded input group on lines 3 upwards in input group 2. When a strobe signal is detected on line 8, a user signal whose number is derived from the other input lines in the group is sent from the channel to the host system. In the example below, if user signal 5 is received then sequence 10 is executed.

System	User	Comments
1>	DX2:3-	Define expanded inputs on lines 3 upwards in input group 2
1>	SX5/10	Code 5 triggers sequence 10
1>	SX4/2	Code 4 triggers sequence 2
1>		

MG[g:]bb 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 in the specified group.

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.

The MG command applies to the specified input group only. It is not possible to specify automatic masking of inputs in a different input group. If no group number is given, the command defaults to group 1. If only the group number is given, the current MG parameter value for the specified group is displayed.

Example :

System	User	Comments
1>	DI1:1-/ST/EI2	Execute stop on input line 1
1>	DI1:2-/VC+/EI1	Execute move on input line 2
1>	MG1:00000011	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.

BG[g:]bb 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.

DE[g:]n± **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 4.10. This command is restricted, and is only available in privileged mode.

Example : `DE1:6+`
This sets up an active high error output signal on output line 6 in group 1. When an error condition is detected, the error output is set to a logic high.

OB[g:]n± **Define motor brake output (restricted).**
Range : 1 to 8

This command defines the specified output line as a motor brake output. The line is set to the specified state to engage the brake before disabling the drive when the system goes to the motor off state, either on a MO motor off command, or on detection of a motor error condition. It is cleared to the opposite state to release the brake after the drive is enabled when the axis is returned to the position control state with the PC position control command.

An optional delay time between enabling the drive and releasing the brake, and between engaging the brake and disabling the drive, is set by the brake delay time parameter BD.

This command is restricted, and is only available in privileged mode.

Example : `OB1:1+`
This sets up an active high motor brake signal on output line 1 in group 1. When a MO command is executed, the brake is engaged by setting the output to a logic high. When a PC command is executed, the brake is released by clearing the output to a logic low.

BDnn **Set motor brake delay time (restricted).**
Range : 0 to 255
Default : 0

This command sets up a delay time for the motor brake output signal. It is specified in units of 1/256 second (about 4ms). The brake delay time sets the time between enabling the drive and releasing the brake, and between engaging the brake and disabling the drive.

PO[g:]n± **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 4.16 for more details. Note that for correct operation, speed averaging with the VT command may also be required.

Example : `PO1:5–/1000/2000`
This example defines output line 5 in group 1 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.

OX[g:]n± **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. Note that the defined OX output lines are assigned to the current channel, and OC commands for these outputs must be executed on this channel.

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 “OXg:0”.

Example : `OX1:3+`
This example defines output lines 1-3 in group 1 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.

BO[g:]n± **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 : `BO2:7+`
This example defines output line 7 in group 2 as a position bound overflow output signal. Each time the system passes the position bound, a logic high pulse is output on line 7.

OW[g:]n± **Define outside window output (restricted).**
Range : 1 to 8

This command defines the specified output line as a signal that indicates when the motor position error is larger than the SW set window value. This may be useful as a continuous “in position” signal, or as a position error signal at a lower priority than the normal motor position error threshold set by SE. This command is restricted, and is only available in privileged mode.

RR[g:]n± **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 in group 1 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.

RA[g:]n± **Define reference accepted output (restricted).**
Range : 1 to 8

This command defines the specified output line as a reference accepted output. The output gives a 4ms pulse when a reference input signal has been accepted as a valid reference. This may be used to indicate the presence of a good product, for example when using reference width checking. This command is restricted, and is only available in privileged mode.

JF[g:]n± **Reference adjustment forwards output (restricted)**
Range : 1 to 8

This command defines an output line which indicates when a reference adjustment in the forwards direction is taking place. This command is restricted, and is only available in privileged mode.

JB[g:]n± **Reference adjustment backwards output (restricted)**
Range : 1 to 8

This command defines an output line which indicates when a reference adjustment in the backwards direction is taking place. This command is restricted, and is only available in privileged mode.

AE[g:]n± **Define analogue limit error output (restricted).**
Range : 1 to 8

This command defines the specified output line as an analogue input out of limits error signal. The line is set to the specified state when the analogue input value is outside the high and low limits set by the AH and AL commands, and cleared again when the analogue input returns inside these limits.

DU[g:]n± **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 recognised 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[g] **List input line definitions.**

This command lists the current definitions of the input lines for the specified group on the display. The list shows the input 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.

LO[g] **List output line definitions.**

This command lists the current definitions of the output lines for the specified group on the display. The list shows the output line number, followed by a sign (+ or –) and a letter representing its function. Lines not defined are left blank. The definitions are listed on the display or terminal, one per display line.

Examples :

```
System              User          Comments
1>                  LI1<CR>       User command to list inputs
Inputs :
1:1 – DR1                         Reference input (ch.1)
1:2 + DI3 AB                      Function input definition (ch.3)
1:3 – DI1 IN–/WT256/ID/MR–5000/ZC
1:4 + PS2                         Snapshot input (ch.2)
1:5                               Normal input lines, undefined
1:6
1:7 – DL1,2                       Active low limit switch input
1:8                               Undefined
1>

1>                  LO<CR>        User command to list outputs
Outputs :
1:1 + OX3                         Expanded output group (ch. 3)
1:2 + OX3
1:3 + OX3
1:4 – RR1                         Reference reject output (ch. 1)
1:5 – PO2 >1000 <2000             Position trigger output (ch. 2)
1:6 + DE1                         Error output (ch. 1)
1:7 + BO4                         Bound overflow output (ch. 4)
1:8 – AE2                         Analogue error output (ch. 2)
1>
```

FSnn **Select feedback encoder type (restricted).**
Range : 0 to 10
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	Applies to:
0	Quadrature ×4 (except Q-Drive)	1, 2, 3, 4, 5, 6
1	Quadrature ×2	3, 4, 5, 6
2	Quadrature ×1	3, 4, 5, 6
3	Up/down count	3, 4, 5, 6
4	Count and direction	3, 4, 5, 6
5	SSI, relative position, binary	3, 4, 5, 6
6	SSI, relative position, gray code	3, 4, 5, 6
7	SSI, absolute position, binary	3, 4, 5, 6
8	SSI, absolute position, gray code	3, 4, 5, 6
9	CANopen, relative position	1, 2, 4
10	CANopen, absolute position	1, 2, 4

Table 2: Encoder feedback options

Key :
1.....Q-Drives
2.....MiniPTS 1+1
3.....MiniPTS 2+1
4.....MiniPTS 3
5.....MiniPTS 4
6.....PTS Mk2

Note 1 : FS0 is quadrature encoder ×4 for all systems except Q-Drives. These use FS0 to select their default position source, which is resolver feedback via the drive for channel 1, and quadrature encoder ×4 for channel 2. The channel 2 encoder signals may be from the drive encoder input or encoder simulation.

Note 2 : The SSI encoder options require some jumper links to be fitted. Please refer to Appendix A. on page 229 for more details. Also note that the NB and NZ commands define the number of data bits used with an SSI or CANopen encoder.

Note 3 : For SERVOnet the rules for Q-Drives, MiniPTS 1+1 and MiniPTS 3 apply to the relevant axis controller modules.

FCnn **Set encoder feedback channel (restricted).**
Range : 0 to number of encoder inputs on board
Default: current channel number

This command sets which encoder input is used as the feedback source for the current channel. It is possible to use one encoder on multiple channels, or to cross over encoder inputs. FC0 means use the same encoder input as the channel number. This command is restricted, and is only available in privileged mode.

NBnn **Set number of bits for absolute encoder (restricted).**
Range : 12 to 24
Default : 24 (12 on a Q-Drive)

This command sets the number of data bits used when the channel is configured for use with an SSI or CANopen encoder. It allows several different types of SSI or CANopen encoder to be used in different applications, from single turn 12 bit to multiturn 24 bit models. The number of data bits may be set differently on each channel if required.

On a Q-Drive, the NB command is used to set the number of data bits per turn of the motor, as returned by the drive system. Standard drives which use a resolver for position feedback return 12 data bits per turn to the PTS module. Drives which are configured to use a SinCos encoder or similar feedback device may return up to 16 data bits per turn. The NB command is used to set the number of data bits per turn expected by the PTS module to match the drive configuration.

NZnn **Set number of leading zeros for absolute encoder (restricted).**
Range : 0 to 12
Default : 0

This command sets the number of leading (discarded) zero bits of the absolute (SSI or CANopen) encoder data format. The number of leading zeros may be set differently on each channel if required.

CF Configure port options

The CF command lists the current serial port configuration, and then prompts for a port to be changed. Port A supports both RS-232 and RS-485 operation, and also supports either hardware or software handshake in RS-232 mode. Port B supports RS-232 and RS-485 modes, but its handshake and tristate control options are determined by which particular software option is using port B, as set by the SK command.

```
1: cf
Port  Type   Handshake
 A    RS232     S/W
 B    RS485     N/A
Port ?
1:
```

On the Machine Controller and PTS MkII systems, port C is used to set the baud rate for the second CANbus port when it is enabled for SynchroLink. The first CANbus port is fixed at 500kbits/second.

If the Data Highway option is used then port D refers to the unit number.

On the PTS Mk2 and SERVOnet systems with the Machine Controller, the CF command is also used to set the Internet network number for the optional Ethernet port. The command uses port E to refer to the Ethernet port.

```
1: cf
Port  Configuration
 E    192.9.200.20
Port ?
1:
```

If the Modbus option is used then port M is used to set the unit number. If this has not been set then the unit number defaults to 1.

All changes in port configuration set by the CF command take effect next time the system is powered off and on.

4.15 Timer/Counter Functions

There are several commands available to support timer/counter functions on the digital output lines. This section describes these features briefly. The TC function may be combined with other output functions, for example to provide a defined pulse width for signalling to a PLC, or it may be used with the SO and CO commands.

It is also possible to use the timer/counter functions with external input signals. Commands are available to define a clock or trigger input, a gate input, and a reset input for any timer/counter output line.

TC[g:]n± Define timer/counter output (restricted)

Range : line number 1 to 8
count 0 to 65535 (modes 0 to 7)
0 to 31 (mode 8)
mode 0 to 8

This command defines the specified output line as a timer/counter output. If the TC command is given with the following sign, to define a new timer/counter output, then it should be followed by two parameters. The first parameter is the count value for the timer/counter, and the second value is the timer/counter mode.

The timer/counter mode values are as follows.

Mode	Operation
0	One-shot up counter
1	Cyclic up counter
2	One-shot down counter
3	Cyclic down counter
4	One-shot timer
5	Cyclic timer
6	Reserved
7	Reserved
8	Shift register

Table 3: Timer/counter modes

The output line is set true (as defined by the sense in the TC command) when the timer/counter is first triggered, and it is reset false when the timer/counter reaches its final value, set by the count parameter.

In counter modes 0 to 3, the counter is incremented or decremented when either the SO command is given for the output, or any other defined output function on the same output goes true. If it is an up counter, its initial value is zero, and its final value is given by the count parameter. If it is a down counter, its initial value is the count parameter, and its final value is zero. On the first count, the output line is set true, and the counter is started and set to its initial value. When the count is incremented or decremented to its final value, the output line is reset false, and the counter is stopped and reset to its initial value. The counter may be stopped and reset at any time by using the CO command.

In timer modes 4 and 5, the timer is triggered in the same way as in counter mode, but once triggered it counts once per tick until the final count is reached. The output line is set true when the timer is triggered, and is reset false when the timer reaches the final count.

In one-shot modes 0, 2 and 4, the timer/counter behaves as described above. In cyclic modes 1, 3 and 5, it operates in a slightly different way. When the timer/counter reaches its final count, the output line is toggled to its opposite state, the counter/timer is reset to its initial value and continues to run. The output line changes state each time the counter/timer reaches its final count.

In shift register mode 8, the timer/counter behaves as a shift register, to a maximum of 31 bits. The state of the TG gate input assigned to the timer/counter is clocked into the shift register by an active transition on the TK clock input, or by an SO command on the TC output. At the same time, the last bit in the shift register is clocked out onto the TC output line. An active transition on a TZ input or a CO command on the TC output resets the shift register contents to zero.

Examples :

```
TC1:3+/20/1
```

This defines a cyclic up counter, which toggles output 1:3 every twenty clocks. `SO1:3` increments the counter by 1. `CO1:3` stops the counter and resets it to zero.

```
RA1:4+
TC1:4+/5/4
```

These commands define a reference accepted output, and add a one-shot pulse timer to it. The timer is triggered by the RA output going true, and then holds the output true for 5 ticks (about 20ms). Thus it stretches the normal RA output time of 4ms to something a bit longer, to make it easier for it to be seen by a PLC.

```
TC1:5+/10/8
TG2:1+/1/5
TK2:2+/1/5
```

This defines a 10 bit shift register on output 1:5, with the data signal on input 2:1 and the clock signal on input 2:2. A rising edge on input 2:2 or `SO1:5`, clocks the current state of input 2:1 into the shift register, and the shift register updates output 1:5 as appropriate. `CO1:5` clears the shift register contents.

LO lists these output definitions as follows.

```
Outputs
1:1
1:2
1:3 + TC Count=00020 Mode=1
1:4 + RA1 -> TC Count=00005 Mode=4
1:5 + TC Count=00010 Mode=8
1:6
1:7
1:8
1>
```

LC[g:]l **List counter value**
Range : 1 to 8

The LC command lists the current count or time value of a timer/ counter. This may be particularly useful in applications using a counter to count some external event or signal. If the TC output is programmed as a shift register, then the current contents of the shift register are displayed, but as a numerical value, not a binary value.

TK[g:]n±/g/l Define timer/counter clock input
TG[g:]n±/g/l Define timer/counter gate input
TZ[g:]n±/g/l Define timer/counter reset input
Range : 1 to 8

The TK command defines a clock/trigger input line for a timer/counter output. The TG command defines a gate input, and the TZ command defines a reset input. Note that all three of these commands require the input group and line number on which they are defined, and the output group and line number of the timer/counter to which they are assigned.

When a clock input changes from false to true, it increments or decrements a counter, or it triggers a timer, in the same way as the SO command on the timer/counter output. When a reset input is true, the timer/counter is stopped and is reset to its initial value, in the same way as the CO command on the timer/counter output. If a gate input is defined, it allows its timer/counter to run normally when it is true, and holds it at its current value when it is false.

Example :
`TK1:2+/1/3`
This defines a clock signal on input 1:2, assigned to a timer/counter on output 1:3. LI lists this as follows.

```
Inputs
1:1
1:2 + TK -> TC1:3
1:3
...
```


4.16 Phase Advance

PAnn **Set phase advance scale factor.**
Range : 0 to 65535 (units of 15.3μs)
Default : 0

The phase advance feature is a mechanism for shifting all position trigger output signals 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.

If CW bit 0 is set to 1, then the VT filter is applied to the encoder input speed used throughout the system.

BAnn **Set map base advance.**
Range : 0 to 65535 (units of 15.3μs)
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.

4.17 Display Commands

DP **Display actual position.**

Displays current actual position, in encoder counts divided by the user scale factor.

DD **Display demand position.**

Displays current demand position, in encoder counts divided by the user scale factor.

FE **Display following error.**

Displays the closed loop position error, or following error, in encoder counts. This is provided for applications which need to read the following error into a variable.

DV **Display velocity.**

Displays the current measured velocity of the system, in encoder counts per second divided by the user scale factor. 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 4.12, and the DR and DZ configuration commands in section 4.14.

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 4.14.

DA **Display analogue input.**

Displays the current channel's analogue input signal voltage as a number in the range ±2047, corresponding to ±10V. **Note : On Q-Drives the value returned by the DA command is defined by the QA parameter (see page 183).**

The optional expansion board for the MiniPTS 4 and PTS Mk2 includes a 4:1 analogue multiplexer for the single analogue-to-digital converter. If this board is fitted, then one analogue input signal is assigned to each of the four motor channels, and the DA command displays the analogue input signal value for the current channel. If the expansion board is not fitted, then the DA command displays the value of the single analogue input signal, regardless of which is the current motor channel.

DT **Display time.**

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

TS **Time set.**

This command allows the user to set the system time. The command must be entered without a time parameter. The system displays the current time and prompts for a new time, which should be entered in the format hh[:mm[:ss]].

DM[nn] **Continuous display mode.**
Range : 1 to 32767, or no parameter

If no parameter value is given, this command turns on a continuous display of demand position, measured position, position error, and time. The data is displayed for the current selected channel. 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, recorded at the full speed system sample time.

DO **Display mode off.**

Turns off the DM continuous display. This is the default state. This command may also be used to turn off the trace mode display.

TR[nn] **Enable trace mode.**
Range : 0 to 32767, or no parameter

This command controls a continuous trace display. It allows various data values to be displayed via the serial port, similar to the DM command, but it allows data from more than one channel to be simultaneously displayed, and it supports a wider range of data values.

The data values to be traced are specified by the TW and TI commands. The TW command is local to each channel, and enables or disables channel-specific data values such as position or speed. The TI command is global, and enables or disables tracing on input and output line states.

If a parameter value is given, the system prints that number of data samples, recorded at the full speed system sample time. If no parameter value is given, then single data samples are printed continuously, one line at a time, together with a timestamp value measured in system ticks.

The trace mode display is turned off by either TR0 or DO.

TWbb **Set trace options word.**
Range : 16 bit binary value

This command enables and disables tracing for various channel data values. When tracing is enabled with the TR command, the data values set by the TW channel options and the TI input/output trace options are displayed via the serial port. Each trace display field is labelled with a prefix letter and a channel number. The TW bit functions and prefix letters are listed below:

Bit	Data value	Prefix letter
0	Demand position	D
1	Measured position	P
2	Position error	E
3	Demand velocity	V
4	Measured velocity	W
5	Average measured velocity	X
6	Reference error	R
7	Snapshot position	S
8	Reserved	U
9	Reserved	C
10	Analogue input value	A
11	Analogue loop error	L
12	Actual map scale factor (%)	F
13	Master axis position	M
14	Master axis velocity	N

TIbb **Set input/output trace options.**
Range : 16 bit binary value

This command enables and disables tracing for input and output line states. When tracing is enabled with the TR command, the data values set by the TW channel options and the TI input/output trace options are displayed via the serial port. Each input/output trace display field is labelled with a prefix letter and a group number. Note that not all PTS units support the full range of inputs and outputs listed here. The TI bit functions and prefix letters are shown below:

Bit	Data value	Prefix
0	Input group 1	I1
1	Input group 2	I2
2	Input group 3	I3
3	Input group 4	I4
4	Input group 5	I5
5	Reserved	
6	Reserved	
7	Reserved	
8	Output group 1	O1
9	Output group 2	O2
10	Output group 3	O3
11	Output group 4	O4

TF **Turn off all trace options.**

This command turns off all trace data options on all channels by resetting the TW and TI parameters to zero. It simply provides an easy way to reset all trace options without having to change to each channel individually.

DK **Display system constants.**

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

- Proportional gain constant KP
- Velocity feedback gain constant KV
- Velocity feedforward gain constant KF
- Acceleration feedforward gain constant KA
- Scale factor for length related units SU
- Velocity SV
- Acceleration (to nearest multiple of 256) SA
- Maximum position error SE

CDnn **Character delay.**
Range : 0 to 255
Default : 0

This command sets the delay between characters sent to the serial port, in units of 1/256 seconds. It allows the system to be used with terminals that do not support the xon/xoff handshake protocol, by simply inserting a time delay between each character sent from the PTS.

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 Reserved.

Bit 2 Reserved.

Bit 3 Reserved.

Bit 4 When set to 0, maps and profiles are entered and listed in absolute position format.
When set to 1, maps and profiles are entered and listed in relative position format.

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 00100000 is for a standard 80 column terminal, long error messages, and decimal input and output. This command is restricted, and is only available in privileged mode.

HE **Print help display.**

This command prints a complete list of all commands available on the system, in alphabetical order, a screenfull at a time. It pauses between each page until a character 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

```
DP Display measured position
1>
1>          CW?<CR>          Request help on CW command
CW Set control options word
   Bit Meaning when set
    0  Reserved
    1  Reserved
    2  Enable fixed speed operation
    3  Use unipolar command signal
    4  Invert command signal output
    5  Invert encoder input sense
    6  Power up in motor off state
    7  Inhibit integral gain while moving
1>
```


LE[n] **Display last error(s).**

This command redisplays the error message(s) for the last error(s) 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.

The system continuously logs the last ten errors. The LE command takes an optional parameter which specifies how many errors (up to ten) to display. If no parameter is given, the most recent error is displayed.

SYnn **Set display configuration (restricted).**
Range : 8 bit binary value.
Default : 01

This command is used to set up various terminal display options.

Bit 0 This bit enables and disables the normal state change prompts.
Setting this bit to 1 enables the state change prompts. The state changes are shown for the current focus channel, which is set by entering a single CH command at the terminal. Note that CH and CP commands in command strings and sequences do not change the terminal focus channel.
Setting this bit to 0 disables state change prompts. Other display values error messages, and the '?' parameter change prompt are always sent to the terminal, even when prompts are disabled. This is sometimes useful to stop the terminal from continuously scrolling due to state changes when manually changing parameters.

Bit 1 Setting this bit to 1 inhibits the normal echo of characters sent to the terminal. It can be useful in some applications to suppress echo, to make it easier to control the unit from some external system. This is primarily intended for use by the PTS Toolkit software, to improve file download performance.
With this bit set to 0, all characters received by the system are echoed back to the terminal.

4.18 Analogue Control

This section describes the analogue control facilities in the PTS systems.

Analogue loop control may be used for tension control applications. Closed loop tension control is performed by calculating the required speed ratio between a master and slave axis in mapping, with feedback on the analogue input. This mode is enabled by setting AM1. The system now also has two joystick control modes, where the speed or position of a motor axis is determined by the analogue input signal from a joystick. These are selected by AM2 or AM3.

Tension Control

The tension control loop reads the analogue input value at the normal system sample rate, and compares it with the analogue setpoint value. The difference between the setpoint and the input is the analogue loop error, and this is used in a closed loop control algorithm to determine the required action. The tension control on the PTS systems has two parts to its control loop; one part calculates the speed ratio between master and slave axes, while the other part calculates an additional speed component for the slave axis.

This has several advantages. Because the slave axis is mapped to the master axis at the current ratio, the slave axis will begin to move as soon as the master axis moves, without having to wait for any error to build up in the tension control loop. This avoids large transient errors as the master axis starts and stops. However, the ratio control action has no effect when the master axis is stopped, and this is where the speed calculation is useful. The tension loop error is used to calculate an additional speed component, which is added to the slave axis speed derived from the master speed and the current calculated speed ratio. At high machine speeds the ratio control term dominates, while at low machine speeds when the ratio control term acts very slowly, the speed control term remains active.

The ratio control algorithm used is of the following form.

$$\text{Ratio} = SM \times [1 + (AP e_i + AI \sum e_i + AD(e_i - e_{i-1}))]$$

where SM = default ratio
AP = proportional gain constant
AI = integral gain constant
AD = differential gain constant
e_i = tension error (= tension setpoint – measured tension)

The speed control algorithm used is as follows.

$$\text{Speed} = VP e_i + VI \sum e_i + VD(e_i - e_{i-1})$$

where VP = speed proportional gain constant
VI = speed integral gain constant
VD = speed differential gain constant
e_i = tension error (= tension setpoint – measured tension)

In closed loop control, 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 performance of the analogue control loop may be monitored by means of the analogue output monitor functions. Commands are provided to output various signals, including the analogue loop error and the current ratio, on the monitor output for viewing on an oscilloscope or chart recorder. These are described in section 4.11.

The action of the XM command is modified if tension control is on, as set by the AM1 command. When tension control is on, the map startup sequence is determined by AW bit 0. If AW bit 0 is set to 0, the slave axis goes immediately into mapping, using the software clutch, with the map ratio determined by the tension control loop. If AW bit 0 is set to 1, the slave axis moves at jog speed (set by SS) until the measured analogue input reaches the tension control setpoint, and then drops into mapping. This is intended to initialise tension control when the master axis is stopped. The direction of the slow speed initial move is set by AW bit 6.

If either of AW bits 4 and 5 are also set to 1, then the unit automatically initialises the map scale factor before executing the map under tension control. The system measures the distance moved by the master and/or slave axes between the two positions where the analogue input high and low limits are exceeded, and stores these in the AR parameters. The ratio of these two values gives a good estimate of the initial map ratio required by the tension control loop and is used at the start of synchronisation. This allows the system to reach its correct steady state ratio much more quickly than if it starts from the default SM value, particularly if the system is not at its normal starting position.

An example where this is useful is a winding or unwinding application. In normal circumstances the system always starts with either a full or empty spool of material, and the initial scale factor is set in the SM parameter. Restarting the machine with the same spools uses the scale factor last calculated when the machine was stopped, and the tension loop restarts smoothly. However, if the machine may be started with spools that are partly filled to an unknown diameter, then the normal SM value is not at the correct value for running at the new spool diameter. The analogue range initialisation function allows the required ratio with the actual spool diameters to be measured. The machine may be started with the new spool without any large transients while the tension control loop stabilises from the initial default SM value to the new scale factor.

PTS Mk2 and SERVOnet: Note that automatic measurement of the master axis AR distance is only possible when the master axis is on the same board or module as the slave axis. When the master and slave axes are on different boards, the XR command must be used on the master axis if it is necessary to measure its AR distance. This also requires the analogue input signal to be connected to both master and slave axes, and their analogue setpoint and limit values (AC, AH, AL) to be programmed to the same values.

Joystick Control

A slave axis can be put into either joystick position mode or speed mode. In the two joystick modes, the master position data for the slave axis is derived from the joystick analogue input signal. In either case this master position is subject to filtering with the BT parameter to minimise the effect of noise. AM2 selects joystick position mode, where the master position is set to the analogue input value, giving a range of 0 to 4095 counts. AM3 selects joystick speed mode, where the master speed is set to the analogue input value, giving a range of ±2047 counts per second. In addition, when joystick speed mode is selected the AD parameter is used to set a deadband around the joystick zero position, and MW bits 0, 1 and 4 are set to force speed mapping and to enable the software clutch. In either case, the slave axis can use the normal MB, MF and SM parameters to produce the required position or speed range.

AMn **Set analogue control mode.**
Range : 0 to 3
Default : 0

This command selects the analogue control mode, as shown in the following table. The AM parameter is not saved.

AM	Mode
0	Analogue control disabled
1	Tension control mode
2	Joystick position mode
3	Joystick speed mode

Table 4: Analogue control modes

The tension control loop may be enabled and disabled at any time by switching between AM1 and AM0. However, it is not recommended that the system is switched in and out of joystick control modes while running, as this is likely to cause sudden changes in position on the slave axis. Joystick control should be selected before using the XM command on the slave axis to begin following the joystick signal, and the motor should be unlinked from the joystick data with one of the stop functions before setting AM back to zero.

Setting AM1 or AM3 also sets bits 0, 1 and 4 of MW to a 1. This forces the use of the software clutch and speed mapping, as required for tension control or joystick speed mode.

APnn **Set analogue control proportional gain (restricted).**
Range : 0 to 65535
Default : 4096

This command sets the proportional gain for the tension ratio control loop. The proportional gain acts on the measured analogue error, which is calculated as the difference between the required setpoint and the value measured by the analogue input. High gain gives the system a faster response and tighter control, but if the gain is too high the system may oscillate. For best results, the proportional gain as low as possible to avoid overshoot or oscillation, while still achieving the required control response.

AInn **Set analogue control integral gain (restricted).**
Range : 0 to 65535
Default : 0

This command sets the gain for the integral term in the tension ratio control loop. When integral control is used, the system integrates the analogue error by adding the current error to a running total. Integral gain is useful to allow for long term changes in ratio, as required for tension control in winding/unwinding applications.

ADnn **Set analogue control differential gain (restricted).**
Range : 0 to 65535
Default : 0

This command sets the gain for the differential term in the tension ratio control loop. This term uses the differential of the analogue error (rate of change of error), which represents the velocity error of the system. This is useful where the analogue error is changing rapidly, and provides damping in the analogue control loop.

In joystick speed mode, set by AM3, the AD parameter defines a deadband around zero on the joystick speed value read via the analogue input. This is used to give a stable zero speed even if there is some noise on the joystick input.

VPnn **Set tension speed control proportional gain (restricted).**
Range : 0 to 65535
Default : 0

This command sets the proportional gain for the tension speed control loop. The proportional gain acts on the measured analogue error, which is calculated as the difference between the required setpoint and the value measured by the analogue input. High gain gives the system a faster response and tighter control, but if the gain is too high the system may oscillate. For best results, this parameter is normally set as low as possible, while still achieving the desired rate of response.

VInn **Set tension speed control integral gain (restricted).**
Range : 0 to 65535
Default : 0

This command sets the gain for the integral term in the tension speed control loop. When integral control is used, the system integrates the analogue error by adding the current error to a running total. Integral gain is useful to allow for long term changes in ratio, as required for tension control in winding/unwinding applications.

VDnn **Set tension speed control differential gain (restricted).**
Range : 0 to 65535
Default : 0

This command sets the gain for the differential term in the tension speed control loop. This term uses the differential of the analogue error (rate of change of error), which represents the velocity error of the system. This is useful where the analogue error is changing rapidly, and provides damping in the analogue control loop.

ACnn **Set analogue control setpoint.**
Range : ±2047
Default : 0

This command sets the setpoint for the analogue control loop. The loop error is calculated by subtracting the measured analogue input value from the setpoint. It may be incremented and decremented by the IP command when IS is set to 2.

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 main analogue output. Once set the limit is active at all times and is not specific to the analogue control mode.

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

AWbb **Set analogue control options word (restricted).**
Range : 8 bit binary value.
Default : 0

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

Bit 0 This bit controls the behaviour of the system when mapping is started with the XM command and tension control is enabled by setting AM1.
When set to 0, the slave axis goes immediately into mapping, using the software clutch, with the map ratio determined by the tension control loop.
When set to 1, the slave axis moves at jog speed (set by SS) until the measured analogue input reaches the tension control setpoint, and then drops into mapping. This is intended to initialise tension control when the master axis is stopped.
This bit set to 1 also enables the functions of AW bits 4, 5 and 6.

Bit 1 Reserved.

Bit 2 This bit controls the analogue control integral term. It allows the analogue control integral term to be turned off if required, for example if the controlled motor is disabled. This prevents the system from integrating up any static analogue error and giving a large transient when the motor starts again. In tension control applications, it affects both the ratio and speed control integral terms.
When set to 0, the analogue control integral term is disabled.
When set to 1, the analogue control integral term is enabled.

Bit 3 This bit sets the initial ratio when the slave axis is mapped to the master axis and tension control is enabled. It also allows the current ratio to be reset to the default value, as set by the SM command, at any time.
When set to 0, the initial ratio is the SM value.
When set to 1, the ratio value used previously is kept. Normally this is the last value calculated by the tension control loop before the slave axis was stopped.
When changed from a 1 to a 0, the current ratio is reset to the default value.

Bit 4 This bit controls the automatic measurement of the distance between the high and low analogue limits (the analogue range distance) on the slave axis when the XM command is executed. Refer to the AR and XR commands for more details.
When set to 1, the slave axis analogue range initialisation is enabled.
When set to 0, the slave axis analogue range initialisation is disabled.

Bit 5 This bit controls the automatic measurement of the distance between the high and low analogue limits (the analogue range distance) on the master axis when the XM command is executed. Refer to the AR and XR commands for more details.
When set to 1, the master axis analogue range initialisation is enabled.
When set to 0, the master axis analogue range initialisation is disabled.
PTS Mk2 and SERVOnet: Note that automatic measurement of the master axis AR distance is only possible when the master axis is on the same board/module as the slave axis. When the master and slave axes are on different boards/modules, the XR command must be used on the master axis if it is necessary to measure its AR distance. This also requires the analogue input signal to be connected to both master and slave axes, and their setpoint and limit values (AC, AH, AL) to be programmed to the same values.

Bit 6 This bit sets the direction of the tension takeup move when the slave axis is started with tension control enabled, and AW bit 0 set to 1.
When set to 0, the direction of the takeup move is the same as the sign of the initial tension error.
When set to 1, the direction of the takeup move is the opposite of the sign of the initial tension error.

Bit 7 This bit determines the sense of the analogue control loop.
When set to 0, an increase in the analogue error gives an increase in the output value.
When set to 1, an increase in the analogue error gives a decrease in the output value.

AHnn **Set analogue input high limit.**
Range : ±32767
Default : +2047

This command sets the high limit for the analogue input. If the analogue input value exceeds this value, then the “analogue input high limit exceeded” error message is displayed. If bit 3 of the error options word EW is set to 1, then this is a motor error, and the axis shuts down to the motor off state.

ALnn **Set analogue input low limit.**
Range : ±32767
Default : –2047

This command sets the low limit for the analogue input. If the analogue input value goes below this value, then the “analogue input low limit exceeded” error message is displayed. If bit 3 of the error options word EW is set to 1, then this is a motor error, and the axis shuts down to the motor off state.

XR **Execute analogue range distance initialisation.**

XR executes the analogue range initialisation function manually on the current axis, and stores the measured value in AR.

The analogue range initialisation function does the following:
The motor moves in one direction, chosen so as to move initially towards the setpoint, until one analogue input limit is crossed, stores the current position and stops. It then reverses and moves until the other analogue limit is crossed, stores this second position, and saves the difference between the two positions as the AR value.

It is important that the set bound values (SB) on the master and slave axes are larger than the AR distances to be measured by this function. If not, and the position value wraps around during the XR initialisation, then the difference between the high and low limit positions does not represent the distance moved between the limits, and the result stored in the AR value is wrong. This is not usually a problem since in tension control applications the axes are synchronised using speed mapping, the bounds positions are not required for position synchronisation or referencing and are set to their default maximum value.

If the current channel has tension control enabled by setting AM to 1, then the system assumes that this is a slave axis, and the XR command uses the analogue input and limit values for the current axis. If the current channel does not have tension control enabled, then the system assumes that this is a master axis. It finds the first slave axis linked to this axis on the same board/module with tension control enabled, and uses the analogue input and limit values for this axis. If no suitable slave axis is found on the same board/module linked to the current axis, with tension control enabled, then the analogue input and limits for the current channel are used.

Setting AW bits 4 and/or 5 on the tension controlled slave axis enables automatic measurement of the analogue range distance on the slave and/or master channels respectively at the start of mapping. If this is enabled on either axis, then the measured AR value(s) are used to calculate the initial map scale factor as AR(slave)/AR(master), as in the CR command. PTS Mk2 and SERVOnet: Note that automatic measurement of the master axis AR distance is only possible when the master axis is on the same board as the slave axis. When the master and slave axes are on different boards, the XR command must be used on the master axis if it is necessary to measure its AR distance. This also requires the analogue input signal to be connected to both master and slave axes, and their setpoint and limit values (AC, AH, AL) to be programmed to the same values.

ARnn **Define analogue range distance (restricted).**
Range : 0 to 65535
Default : 256

This command defines or displays the distance between the analogue input high and low limits on the current channel, called the analogue range distance. It may be used to set this value for a master axis that is not driven by the unit, or to display the value measured by the XR initialisation function.

MMnn **Define master axis analogue range distance (restricted).**
Range : 0 to 65535
Default : 256

This command defines or displays the distance moved by the current channel's master axis between the analogue input high and low limits, called the master analogue range distance. It is used by the host system to set this value for a master axis on a different board or module, allowing automatic ratio calculations in tension control applications where master and slave axes are not controlled by the same board or module. This function is of particular use on PTS Mk2 and SERVOnet systems.

CR **Calculate initial ratio from analogue range distances.**

CR calculates the ratio between the two AR values on the master and slave axes, and uses the result as the map scale factor. The ratio is calculated from the two AR values as AR(slave)/AR(master), in much the same way as the BR command does for the bounds values.

If automatic measurement of the analogue range distance is enabled on either master or slave axes, then the XM command performs an automatic ratio calculation using CR; for example:

System	User	Comments
5:	CH7<CR>	Change to master channel
7:	AM0/AH900/AL-900/PC/XR/$ARM=AR	
7>	CH1<CR>	Change to slave channel
1:	MM$ARM/AH900/AL-900/AW0xx01xxxx	
1:	AM1/PC/XM0	Ratio has been calculated
1X		

AE[g:]n± **Define analogue limit error output (restricted).**
Range : 1 to 8

This command defines the specified output line as an analogue input out of limits error signal. The line is set to the specified state when the analogue input value is outside the high and low limits set by the AH and AL commands, and cleared again when the analogue input returns inside these limits. This gives an external indication that the PTS is maintaining the analogue input within the desired limits.

DA **Display analogue input value.**

Displays the current value of the analogue input signal on the current channel as a number in the range ±2047. **Note : On Q-Drives the value returned by the DA command is defined by the QA parameter (see page 183).**

The optional expansion board for the MiniPTS 4 or PTS Mk2 includes a 4:1 analogue multiplexer for the single analogue-to-digital converter. If this board is fitted, then one analogue input signal is assigned to each of the four motor channels, and the DA command displays the value for the current channel. If the expansion board is not fitted, then the DA command displays the value of the single analogue input signal, regardless of which is the current motor channel.

4.19 Variables and the Database

The variable database is a centralised facility which is accessible to all tasks in the system and holds a set of integer variables. Because variables are generally accessible, it is possible for the user to change a variable using the Operator's Panel and for the variable to be used subsequently to set a motor parameter in the PTS. Similarly a variable can be set to some motor parameter, such as the position, which can then be displayed on the Operator's Panel. A variable can also be set up to trigger execution of a command string on the PTS. This means that a button on the Operator's Panel can be set to update a variable which in turn triggers an action on the PTS.

A variable is referred to by a short name which is assigned by the user but must keep to the following rules.

- A variable name consists of up to 3 characters which must be numbers '0-9' or letters 'A-Z' or 'a-z'.
- The name must not include punctuation characters such as '_' (underscore), '.' (dot) or any spaces.
- Upper and lower case letters are equivalent. For example 'POS' and 'pos' refer to the same variable.
- In PTS commands a variable is prefixed by '$' (dollar) to distinguish it from normal command mnemonics.

A variable can be set to a constant value using '=' (equals). For example the following command sets the variable $SPD to a value of 5000.

```
1> $SPD=5000
```

A variable can be used in place of a numeric parameter in most commands. For example the following command sets the velocity to the value of the variable $SPD which is currently 5000. If the variable has not been assigned a value, then the "undefined variable" error message is displayed.

```
1> SV$SPD
```

Conversely it is possible to query a parameter and place the result in a variable. The following example updates variable $SPD with the current velocity value.

```
1> $SPD=SV
```

A variable can be defined as a trigger variable so that when it is updated a string of commands is executed. The following example defines $SPD as a trigger variable which causes the velocity to be set to the value of $SPD each time the variable is updated.

```
1> $SPD>CH1/SV$SPD
```


Variables can be used in arithmetic expressions involving the standard operators +, -, *, /, %. The % remainder or modulo operator gives the remainder when the left operand is divided by the right. An expression must be enclosed in brackets and it may also be necessary to use further levels of brackets inside the expression to ensure that sub-expressions are evaluated correctly. In any expression the number of left brackets must always equal the number of right brackets. Expressions can be nested to a depth of ten pairs of brackets. An expression can be used as a command parameter anywhere that a simple variable can be used. The following example does a move to a position based on the product length ($LEN) and batch size (20) but allowing for bounds wraparound ($BND).

```
1> CH1/MA(($LEN*20)%$BND)
```

If the product size is 600 counts and the bounds are 5000 counts the above command would result in a move to 2000. Note the extra pair of brackets to ensure that multiplication takes place before the modulo operation.

Variables should not be used in place of parameters which include a '+' or '–' sign. In particular, setting variable $A to 4 and typing DR$A– is not the same as typing DR4– and will merely give a syntax error. In commands which take more than one parameter a variable can be used in place of either or both parameters. The following example sets map scaling to 75/100.

```
1> $PC=75
1> SM$PC/100
```

Because variables are integers, division normally gives an integer result and any remainder is lost. However, where an expression includes a floating point constant number, the result is calculated to floating point accuracy. For example if $A has a value of 5, the first example below moves to a position of 2 units whereas the second moves to a position of 2.5 units.

```
1> MA($A/2)
1> MA($A/2.0)
```

When using a variable in place of an input/output line number and group number, such as in a SO command, the required variable value is constructed by adding the line number to 256 × the group number.

```
1> $G=1/$L=4
1> SO(($G*256)+$L)       # Set output 1:4
```


Variables can also be used in logical expressions. These are used with the IF command to allow conditional execution of commands. A logical expression compares two values using the ==, !=, <, >, <=, >= operators to give a result of TRUE or FALSE. It can be combined with other expressions using the && and || operators to give compound expressions. An expression must be enclosed in brackets and it may also be necessary to use further levels of brackets inside the expression to ensure that sub-expressions are evaluated correctly. Ambiguous expressions in sequences may be resolved by the system; listing a sequence will show the result with brackets inserted as necessary.

Bit manipulation operators are also available. These allow the system to deal with binary data values such as digital input or output data. The bitwise NOT and logical NOT are both unary operators which take a single operand on the right hand side. The bitwise NOT inverts each bit of the operand, whereas the logical NOT gives either 1 (TRUE) or 0 (FALSE) depending on whether the operand is FALSE (0) or TRUE (<>0) respectively.

The shift operators shift the left operand by the number of bits given by the right operand. A left shift fills the lower bits with zeros. A left shift by 1 bit is equivalent to multiplying by 2. A right shift fills the upper bits with the sign bit, 1 for a negative value and 0 for a positive value. A right shift by 1 bit is equivalent to dividing by 2.

Bitwise AND, XOR and OR perform the standard logical operations on the two operands bit by bit. The following truth table shows the effect of the three operations.

Left	Right	AND	XOR	OR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

Table 5: Logical operations

The following table shows the valid arithmetic, logical and bitwise operators, in order of precedence. Operators of equal precedence are shown between double lines.

Symbol	Operation	Example
–	Negate	(–5) = –5
~	Bitwise NOT	(~0) = -1
!	Logical NOT	(!12345) = FALSE
*	Multiply	(5*2) = 10
/	Divide	(5/2) = 2
%	Remainder	(5%2) = 1
+	Plus	(5+2) = 7
–	Minus	(5–2) = 3
<<	Left shift	(1<<8) = 256
>>	Right shift	(512>>8) = 2
<	Less than	(5<6) = FALSE
>	Greater than	(5>5) = FALSE
<=	Less or equal	(5<=6) = TRUE
>=	Greater or equal	(5>=5) = TRUE
==	Equal to	(5==2) = FALSE
!=	Not equal to	(5!=4) = TRUE
&	Bitwise AND	(254&7) = 6
^	Bitwise XOR	(6^7) = 1
|	Bitwise OR	(6|3) = 7
&&	Logical AND	(5==4) && (5>4) = FALSE
||	Logical OR	(5==4) || (5>4) = TRUE

Table 6: Arithmetic and logical operators

Because variables and expressions are evaluated at execution time, commands using them run somewhat slower than commands using only constant parameters.

The PTS now supports arrays of variables, as well as single variables. Unlike single variables, which are created automatically when they are first written to, an array must be explicitly created before it can be used. This is done using the IA initialise array command.

```
1> IA$A[10]
```

This example creates an array $A with ten elements $A[1], $A[2], $A[3] up to $A[10]. Each element in the array can be used in the same way as a variable: as a command parameter, in expressions, etc. The following example shows a position being read into an array element and then used in an expression.

```
1> CH1/$A[1]=DP/RF(100+$A[1])
```

The array index can be a constant, a variable, or an expression. This allows many applications to be dealt with much more simply than before. The following example sets the elements of an array to a set of ascending speed values.

```
1> IA$SV[10]
1> $I=1
1> $SV[$I]=($I*1000)/$I=($I+1)/RP9
```

Array elements can also be used as trigger variables, to trigger the execution of a command string when the element is written to. The example below shows an array element being used to trigger a speed change.

```
1> $SV[1]>CH1/SV$SV[1]
```

Array declarations are saved as host parameters, similar to trigger variable definitions, so that when parameters are restored any arrays are declared before they are used in sequences. It is also possible to change the size of an array by redefining it; in this case the contents of the old array are written to the new array before the old array is deleted.

Array elements are not accessed at channel level, but are handled at host level, similar to expressions. The LV list variables command lists all existing arrays with their sizes, but not the values of their elements. An individual array element may be listed by specifying it in the LV command: for example, LV$A[1] lists the value of element 1 in the array $A. The values of all arrays and elements may be listed by LA bit 5.

Array contents are saved and restored by SP bit 5 and RD bit 5, similar to normal variables. RS with bit 5 set deletes all arrays. RD with bit 5 set does not automatically perform RS bit 5 first.

Array elements can be used with with the Operator’s Panel, MiniPanel, DeviceNet and Profibus, but not currently with Modbus or Data Highway.

$var=nn **Variable assignment.**

The equals sign '=' assigns the value or expression on its right to the variable or array element on its left. Assignment can be used to initialise a variable or to set a variable to a new value based on the value of some other variable. If '=' is followed by a command mnemonic which can be queried, the result of the query is assigned to the variable.

Example :

System	User	Comments
1>	$SPD=5000	Set variable $SPD to 5000
1>	$POS=DP	Set $POS to current position

WV$var **Wait for write to variable.**

This command tells the system to wait until a value is written to the specified variable before continuing with the command string or sequence.

$var>cmds **Define trigger variable (restricted).**

The right arrow '>' defines the variable or array element on its left as a trigger variable such that each time the variable or array element is updated (it need not change value) the commands between '>' and the end of line are executed. It is important to make sure that the triggered commands do not cause the trigger variable to be updated otherwise the system will enter an endless loop when the trigger variable is first updated.

Example :

System	User	Comments
1>	$SPD>CH1/SV$SPD<CR>	

In the example above, the variable $SPD is defined as a trigger variable so that each time it is updated, the velocity on channel 1 is set to the value of variable $SPD.

To remove the trigger variable definition the variable and '>' should be entered at the end of line with no following commands.

VX **List trigger variables.**

This command displays a list of all the trigger variables and array elements along with their associated command strings, one per line. Trigger variables are defined using '>' (right arrow). For each trigger variable the display shows the trigger variable, '>', its command string, and its mask/enable state.

Example :

System	User	Comments
1>	VX<CR>	List all trigger variables
$SPD>CH1/SV$SPD		System lists $SPD and $TR
$TR>CH2/$POS=DD		
1>		

MV[$var] Mask trigger variables.

This command inhibits a specified trigger variable, or all trigger variables if no variable name is given. The VX command shows the current mask/enable state of all trigger variables. MV also masks all array elements defined as triggers but array elements can not be individually masked.

EV[$var] Enable trigger variables.

This command enables a specified trigger variable, or all trigger variables if no variable name is given. The VX command shows the current mask/enable state of all trigger variables. EV also enables all array elements defined as triggers but array elements can not be individually enabled.

LV$var List variable value.

This command lists the value of the specified variable or array element. The display shows the variable name, '=' and the value. If LV is entered without a variable name, the system lists all the currently defined variables in alphabetical order with their values followed by all the currently defined arrays with their sizes but not their contents..

Example :

System	User	Comments
1>	LV$SPD	List value of $SPD
$SPD=5000		Current value is 5000
1>	LV	List all variables
$ABC=1		Variables are listed in order
$SPD=5000		
$TR=0		
$A[10]		Arrays listed with sizes
$B[10]		
1>		

LXexpr **List expression.**

This command displays the value of an expression or variable and is mainly useful for verifying that an expression is evaluated as expected during application development. The display echoes the LX command mnemonic followed by the numeric value of the expression.

Example :

System	User	Comments
1>	$SPD=5000	Set $SPD to 5000
1>	LX($SPD+1000/100)	
LX5010		System displays result.
1>		

In the example above, the LX command displays a result of 5010 rather than 60 because the division operator has higher precedence and is applied before the plus operator.

VE$var **Set error variable (restricted).**

This command defines the error variable for the current channel. The error variable is set to the error code each time an error occurs on its channel. Each channel has an error variable although it may be the same variable as for another channel. The error variable may be used to trigger a string of commands by defining it as a trigger variable. This allows some form of recovery action to be taken when an error occurs. Alternatively the error variable may be used to display a message on the Operator's Panel by specifying it as the associated variable to a message list. The error codes are listed in section 5.4. VE can not be used with an array element.

VS$var **Set status variable (restricted).**

This command defines the status variable for the current channel. The status variable is set to the status code each time the status changes on its channel. Each channel has an status variable although it may be the same variable as for another channel. The status variable may be used to trigger a string of commands by defining it as a trigger variable. This allows some form of action to be taken when status changes. Alternatively the status variable may be used to display a message on the Operator's Panel by specifying it as the associated variable to a message list. The status codes are listed in section 5.3. VS can not be used with an array element.

ZM[$var] **Monitor variable (restricted).**

This command is used to set up automatic monitoring of selected variables. The command ZM$VAR enables monitoring of the specified variable. Several variables may be monitored at the same time. ZO$VAR disables monitoring of a variable. When a variable is being monitored, its name and value are displayed each time it is written to. ZM with no parameter lists the names and values of all variables currently being monitored. ZO with no parameter turns off monitoring on all variables.ZM and ZO can not be used with an array element.

IFexpr **Execute commands if expression is true.**

The IF command causes the commands following it to be executed only if the expression is true (i.e. non-zero). This command affects all commands following it up to the end of line or the next EL command, whichever comes first. The expression following the IF command may consist of a simple variable, a comparison or a combination of expressions. A comparison uses the operators ==, !=, <, >, <=, >=. Expressions can be combined using the operators &&, ||.

Example: `IF($X<10)/$X=($X+1)/SV($X*1000)/WT256/RP`
This command line repeatedly increments $X, sets a new speed and waits 1 second until $X becomes equal to 10.

Example: `IF(($X>=10)&&($X<=20))/SV($X*1000)`
This example shows a more complex expression in the IF command.

EL **Else – execute commands if condition is false.**

The EL command causes the commands following it to be executed if the preceding IF or II command evaluated to false, or zero. The EL command must either appear on the same line as the associated IF or II command, or as the first command on the following line. If the EL command appears in any other position, all commands following it up to the end of line are simply ignored.

Example: `II1-/MA1000/EL/MA5000`
This command line executes a move to 1000 if input line 1 is negative or a move to 5000 otherwise.

Example: `IF($S&&($A>=100))/XS$A/EL/XS99`
This command line executes a sequence ($A) only if $S is non-zero and $A is greater than or equal to 100. Otherwise the default sequence 99 is executed.

YA–YJ **Free parameters.**

There are ten unused parameters on each channel, accessed by the commands YA, YB...YJ. They are saved to non-volatile memory by the SP command along with all other parameters. They may be used to save preset values for variables required by the application program.

IA$var[n] **Initialise variable array.**

This command declares an array of variables. Arrays must be declared befire they can be used. The size of an array can be changed by declaring it again with a new size.

Example: `IA$SV[10]`
This creates an array $SV with ten elements.

OP **Configure Operator's Panel (restricted).**

This command allows the user to access the Operator's Panel configuration menus. When OP is entered, the system displays a banner giving information about the command. The user then presses <CR> to access the configuration menus. To exit from configuration, return to the main menu and select option 0 or press <CR> to finish configuration. Then press '~' (tilde) or Escape to return to the normal PTS prompt.

Safety Note.
While the Operator's Panel is being configured, normal motor control commands cannot be processed. For this reason it is mandatory that any motors attached to the PTS are set to motor off before entering the OP command.

Please refer to the Operator's Panel Reference Manual for full details on setting up the Quin Programmable Operator's Panel.

4.20 Q-Drive Parameter Configuration

The Q-Drive 1+1, Q-Drive MAP and SERVOnet systems allow Q-Drive parameters (speed loop gains, current limit etc.) to be configured through the PTS system, rather than using a separate program.

The Q-Drive 1+1 and Q-Drive MAP are PTS systems integrated within the drive amplifier itself. This provides benefits for wiring and installation of the system. The PTS controller uses an internal high speed serial link to access all drive parameters. A separate manual, MAN431, is available and contains the information required for installing and configuring a Q-Drive.

On SERVOnet systems, all types of SERVOnet axis modules support Q-Drive configuration. The Q-Drive SERVOnet works in the same manner as the Q-Drive 1+1, using the internal high speed serial link. The MiniPTS 1+1 and MiniPTS 3 SERVOnet axis modules use a multidrop RS485 connection from their local serial port B to one or more standalone Q-Drive(s).

Thus the PTS has direct access to the configuration parameters within the drive amplifier. Two PTS commands are used to set and read these parameter values, although usually a PC based Windows program (Q-Drive Setup) is used to configure the drive.

QQnn **Query drive parameter.**
Range : 0 to 255

This function provides the ability to query any drive parameter for its current value. The meanings of all the parameters are listed in the Q-Drive Installation Manual (MAN431).

QPnn/val **Set drive parameter (restricted).**
Range : parameter number 0 to 255
value –32768 to 32767

By specifying both the parameter and value a setting in the Q-Drive can be updated. As for the QQ command, the meanings and ranges of all the Q-Drive parameters are listed in the Q-Drive Installation Manual (MAN431).

QAnn **Set Drive parameter returned by DA (restricted).**
Range : 0 to 255
Default : 0

This command allows a Q-Drive parameter to be returned by the DA command, instead of the analogue input value. Therefore, for example, the drive current could be monitored by setting QA to the appropriate parameter number, then reading DA and assigning it to a variable.

4.21 SynchroLink

SynchroLink is a method of linking individual PTS systems together to perform position mapping between motor channels on two physically separate systems. The network used is CANbus. This is similar in concept to using a link encoder or an encoder splitter, but has the advantage of passing PTS position information, complete with position bound wraparound and referencing adjustments (where applicable). Currently SynchroLink is available only on the Q-Drive MAP, MiniPTS 1+1, MiniPTS 3 and PTS Mk2 systems, plus host level linking of SERVOnet Machine Controllers. Appendix A of the SERVOnet manual (MAN529) documents hardware and PTS programming considerations for using SynchroLink.

SynchroLink commands can be used to control the LinMot linear actuators. The XN command is used to send and receive messages to and from the LinMot controller. They can also be set to follow positions broadcast across SynchroLink by a PTS master axis with LK set to 1. A full description of how to use the LinMot actuators with the PTS systems is given in a separate document.

The SynchroLink commands are also included in SERVOnet systems. This allows extra SynchroLink nodes to be added easily to an existing SERVOnet system if required, and allows LinMot actuators to be used with a SERVOnet system.

A number of Machine Controllers can be linked via SynchroLink using the second CANbus port. An axis module on one SERVOnet can then be linked to an axis module on another SERVOnet by using the MLnn:cc or NLnn:cc syntax described later in this section. The SynchroLink network is enabled on the Machine Controllers by adding the “synchro2” feature, version 0, using the SK command. The SynchroLink baud rate can be changed to 250 or 125 kbps to increase the transmission distance by configuring Port C using the CF command. In this case the number of map masters on the SynchroLink is reduced proportionately.

Note : The Q-Drive 1+1, which does not support SynchroLink, implements the CN, CK and CQ commands for use with a CANopen absolute encoder, and receives XN commands and map data for test purposes.

CQ[bb] **CAN query.**
Range : 8 bit binary value.
Default : 0

This command allows the user to query the current status of the CANbus interface used by SynchroLink. CQ without any parameters gives a short status message indicating the state of CANbus and whether this node is a clock master or clock slave. Additional information can be obtained using the optional bit functions are described below.

Bit 0 Display this nodes bus usage. The number of transmit and receive map links are displayed.

Bit 1 Find all nodes on CANbus. A report of all SynchroLink nodes that are responding on the CANbus network is produced.

Bit 2 Display global bus load. An estimation of the global CANbus load is produced from the number of masters transmitting.

Bit 3 Reserved.

Bit 4 Reserved.

Bit 5 Reserved.

Bit 6 Reserved.

Bit 7 Reserved.

CNnn **CAN node number (restricted).**
Range : 0 (off), 1 to 60
Default : 0 (off)

To turn the SynchroLink functions on a node number is needed. This uniquely identifies the PTS unit on the network. Valid node numbers are 1 to 60 inclusive. A value of 0 turns off the SynchroLink system for this PTS unit.

CKn **CAN clock send/receive (restricted).**
Range : 0 (receive) to 1 (send)
Default : 0 (receive)

All SynchroLink nodes must synchronise with each other. This is done by one node transmitting a clock signal and all other nodes receiving it. The clock master requires CK1 and all other nodes require CK0. Use the CQ function to confirm this is working.

LKn **Link as master axis.**
Range : 0 (off) to 1 (on)
Default : 0 (off)

This command determines whether the current channel transmits its position data over CANbus or not. LK1 is required for all channels that are to be used as master axes across SynchroLink.

MLnn:cc **Map link to CAN master axis.**
Range : nn 1 to 60, cc 1 to no. of channels on target unit

The ML command is extended on SynchroLink to take a node number as well as a channel number. Omitting the node number performs a ML on the local channels. All the rules governing the standard ML command apply. Additionally the master bound, MP, needs setting explicitly on the slave axis, as this information is not transmitted across SynchroLink.

NLnn:cc **Map link to CAN differential axis.**
Range : nn 1 to 60, cc 1 to no. of channels on target unit

The NL command is similarly extended on SynchroLink to take a node number as well as a channel number. Omitting the node number performs a NL on the local channels. The differential bound must be set explicitly with NP.

XNnn **Execute sequence on remote node.**
Range: 1 to 65535

Each unit in a SynchroLink network will be executing its own separate PTS programme. Providing links between these units can either be done via hardware (digital I/O lines) or by using the XN command. This allows you to execute a sequence on all other SynchroLink nodes. Note that there is no confirmation of this process so it should be used with caution.

If a remote node receives an XN command where the given sequence number is not defined on that node, then it sets the reserved variable $XN to the sequence number.

The LinMot linear actuator systems may be used with PTS systems equipped with a CANbus interface. The XN command is used to send and receive messages to and from the LinMot controller. A full description of how to use the LinMot actuators with the PTS systems is given in a separate document.

4.22 SERVOnet

SERVOnet is a PTS system using distributed control elements. A Machine Controller (or Mini Machine Controller) acts as a manager or coordinator to a number of axis controller modules (Q-Drive SERVOnet or MiniPTS 3 SERVOnet) over a high speed network based on CANbus. A separate manual for SERVOnet (MAN529) is available. Only a few extra PTS commands are required to build and maintain a SERVOnet system; all functionality is the same as other PTS systems.

SERVOnet systems also include the SynchroLink commands, described in the previous section. This allows SynchroLink nodes to be added easily to an existing SERVOnet system. It also allows the LinMot linear actuators to be used with a SERVOnet system.

CQ **CAN status query.**

A textual report on the current status of CANbus/SERVOnet is produced by this function.

SI **Configure SERVOnet node identities.**

A SERVOnet requires unique node numbers in each axis module. This can be performed both from the local serial port on the axis module and from the Machine Controller using the CN function. A textual question/ response system displays the current node number configuration and prompts for any changes.

RNnn **Reset node.**
Range : 1 to no. of axis controller module

It is possible to recover an axis controller module after a power failure or replacement of a module without restarting the whole SERVOnet. The RN function takes the module number as the parameter. It attempts to re-establish communications with the axis controller module and then restore local parameters.

The command RN61 may be used to re-establish communication with any external CANopen device. This would normally issue a reset node command to node 61, the Machine Controller. Instead, this is interpreted by the system as a special case and sends a global NMT wakeup message to any CANopen nodes on the network. A full description of the use of CANopen with the PTS systems is given in a separate document.

SERVOnet Node Table

The SERVOnet node table provides a way of mapping channel numbers to node numbers in a flexible way so that a system can be configured to include optional nodes. This is useful where a machine has certain core functions but has other parts which can be added as optional extras. The program designer can designate nodes as required or optional and write a PTS program which includes all the options. At start up the PTS scans Servonet to find out which nodes are actually present. If a required node is not found then an error is reported. If an optional node is not found then it is marked as "Asleep" and any commands issued to that channel will effectively be null commands.

LN List node table

The LN command displays the contents of the SERVOnet node table. The display shows a list of the nodes in the system along with each node's type, state, number of channels and first channel. The node type indicates whether the node is required or optional. The state shows whether the node is alive (i.e. present and working), asleep (i.e. optional and not present) or dead (i.e. required but not present). The "Chans" field shows the number of channels on the node while the "First" field shows the number of the first channel on the node.

The first line of the display shows whether the table is validated or not. When the PTS powers up, if there is not a node table saved in non-volatile memory, the system builds a node table by scanning the SERVOnet. This initial table is not validated so it can not be saved. The table is validated by entering NT and/or NN commands to specify the node configuration and can then be saved using SP100. When the system next powers up it uses the saved node table to verify that the required nodes are present and that optional nodes are accounted for. When the table has been validated the relevant NT/NN commands are listed by LA or LA100. RS100 invalidates the node table and RS100/ SP100 removes the current table from non volatile memory.

Example :

```
System    User          Comments
1>        LN
Table is validated
Node  Type      State    Chans  First
  1   Required  Alive     2      1
  2   Optional  Asleep    4      3
  3   Required  Alive     2      7
1>
```

This example shows a system with three nodes where node 2 is optional and has been removed. Because the node table specifies the number of channels on each node, the PTS knows that any commands for channels 3, 4, 5 or 6 should be dummies and that commands for channels 7 and 8 should go to node 3.

NTnn[/t] **Set node type**
Range: node number nn - 1 to 60
node type t - 0 (optional) or 1 (required)

A SERVOnet node can be marked as ether required or optional. The NT command changes the node type to 0 (optional) or 1 (required). If the command is entered without a type parameter the type of the node is displayed as a numeric value.

NNnn[/c] **Set number of channels**
Range: node number nn - 1 to 60
channels c - 1 to maximum number of channels on node

If a node is marked as optional it is necessary to specify how many channels the node has so that channel numbering on other nodes is correct when the node is removed. The NN command is used to specify how many channels a node has. If the command is entered without the second parameter the current number of channels for the specified node is displayed.

NQ[nn] **Query node/table state**
Range: 1 to 60

When entered with a node number parameter this command displays the current state of the node as a numeric value which is -1 for dead, 0 for asleep or 1 for alive. When entered without a node number parameter the command displays the state of the node table as a numeric value which is -1 for not validated, 0 for mismatch or 1 for validated and matching.

4.23 Edit Mode

Edit mode is an operating mode for the PTS which allows commands to be entered at the command line prompt without being executed immediately. Instead the commands are buffered internally and only executed when the GO command is issued to return the PTS to normal Run mode. The main use of Edit mode is for downloading commands from a remote computer. It is used automatically by the PTS Toolkit download file function. Because there are no delays in executing the commands, communications with the remote computer are more reliable.

An important advantage of Edit mode is that the commands are executed in strict order. Normally lines of commands entered at the prompt are executed in parallel so that the user always has control of the machine and is able to change parameters or abort execution while the machine is running. In Edit mode each command line is executed only when the previous line has finished.

When Edit mode is entered the sequence of events is as follows.

- The system executes the GS (Global Stop) function to terminate any command strings and/or sequences currently executing.
- While the system is in Edit mode all input functions and trigger variables are ignored so that no commands can be executed.
- All commands received are buffered for later execution.
- When all the required commands have been entered, the user enters the GO command to return to normal Run mode and execute the buffered commands.

The commands used to switch into Edit mode and back to Run mode are described below.

ED **Enter Edit mode (restricted).**

This command is used to enter Edit mode. In this mode all command execution is inhibited and commands entered at the prompt are buffered for later execution when the system returns to Run mode. When the ED command is entered at the terminal, the command line prompt changes to ED: as a reminder that the mode has changed.

GO[n] Return to Run mode (restricted).

This command is used to return to normal Run mode from Edit mode. The GO command enables normal command execution and triggers execution of the commands buffered in Edit mode. The system displays a rotating bar indicator while the buffered commands are executed line by line, and no other commands are accepted until these have finished.

The GO command may (optionally) be followed by a checksum value, as generated by PTS Toolkit. This is checked if present. If the checksum is incorrect an error message is displayed, and none of the buffered commands are executed.

Example:

The following example shows the use of Edit mode to buffer an IN command followed by DP. In addition, it shows a sequence being entered in Edit mode. The IN and DP commands are executed in strict order on exit from Edit mode.

System	User	Comments
1>	ED	Enter Edit mode
ED:	RS	Reset the system
ED:	ES10	Enter sequence 10
S10:	CH1/MA1000	
S10:	CH2/SO3	
S10:		Finish sequence 10
ED:	CH1/IN+	Initialisation commands
ED:	CH1/DP	
ED:	GO	Return to Run mode
1I		Executing buffered commands
1S		
1>		Initialisation finished
1 DP2		Executing DP command
1>	LS	List sequences
10		Sequence 10 is defined
1>		

4.24 Debug Commands

Developing an application program for the PTS is not always easy. The PTS supports parallel execution of multiple program threads, both on separate motor channels and within the main system. It is event-driven, where parts of the program are executed in response to external signals, unlike systems programmed in BASIC or ladder logic which have a main polling loop that does most of the work, testing for events and calling various subroutines as required.

It can be very difficult to see what is going on in a PTS application. There may be several sequences running on various motors, input lines and trigger variables setting off new sequences, a communication task in the background, and an operator interface task displaying variables and updating them on demand. This variety may be daunting, but it is also the source of the flexibility of the PTS system.

There are a number of commands available to display what is happening within the system while it is executing. They are documented here for use by experienced PTS programmers. Note that the information given by some of these commands is quite detailed and a thorough knowledge of the PTS system is necessary to use them effectively. Debug information messages are enclosed in curly brackets or braces {} to distinguish them from other display messages.

ZZbb **Set trace options (restricted).**

This command is used to set up trace options which print information to the terminal as commands are executed by the PTS. It takes a binary parameter, with each bit enabling and disabling particular features.
NOTE: When debug tracing is enabled, the performance of the system may be affected because of the large amount of messages displayed via the terminal. This can be improved by using the ZB command to set up a buffer for the debug data, but there will always be a reduction in performance when using the debug features.

Bit 0 Display commands
When this bit is set to 1, all commands executed by the host system or passed to the channels are printed to the terminal. Channel commands are prefixed with the channel number, and host commands with the letter ‘H’. If a parameter is sent with the command, then its value is also displayed. The trace display includes internal commands between the host and channels not normally seen by the user.

Bit 1 Reserved

Bit 2 Display variable accesses
When this bit is set, certain accesses to any variables are displayed on the terminal. A message is given for writes to any variable, execution of any trigger variable command string, and the end of any WV wait for variable command.

Bit 3 Display signals from channels
When this bit is set, user signals from the channels to the host are displayed. This includes both explicit US commands and internal signals used by the system. Internal signals occur at the beginning of a DI line which uses host commands, and at the end of command strings and sequences.

Bit 4 Display Motion Generator information
Setting this bit to 1 enables a display of all variables as they are used by the Motion Generator option, and some additional information.

Bit 5 Enable system overload error checks
This bit enables two additional error messages which can help to diagnose problems caused by overloading the system to the point where the servo loop execution time is too long. The most heavily loaded system is the MiniPTS 4, which runs the largest number of axes on a single processor and is therefore the most likely system to suffer from overload.
The “clock tick during servo loop” error message indicates that the next clock tick has occurred while the system is still executing the current servo cycle. This can occur irregularly on a system which is working correctly, as one particular servo cycle takes an unusually long time for some reason, and in this case this is not a serious problem. If it occurs often, then it indicates that the system is beginning to be overloaded, and other problems may be seen.
The “missed servo loop execution” error is an indication of a more critical problem, and is used as a system diagnostic tool during software development. It indicates that a servo loop cycle has not been executed for the previous clock tick, and this would cause unpleasant jumps or position errors on a system running a motor at any speed. It should not occur in a normally running system.

Bit 6 Display sequences, DI lines, and trigger variables
When this bit is set to 1, a message is displayed on any execution of sequences, DI lines, and trigger variables. The sequence display shows top level and nested sequences.

Bit 7 Display sequences line by line
This bit enables display messages for each line of a sequence as it is executed. This can be used to track the execution of a sequence or nested sequences and to verify the decisions made on any conditional tests.

LD[nn] List channel level sequence(s)

The debug trace data shows the internal sequence numbers, allocated by the host system to the channel level sequence components as the sequences are compiled. The LD command allows the user to list these channel level sequence components in the same way as LS lists normal sequences. This is a useful aid to interpreting the debug trace data.

ZBnn **Set output buffer for ZZ trace data (restricted).**

This command sets up a memory buffer for the trace data reported by the ZZ command. This can be used to reduce the impact on the system performance of all the trace display data. When no buffer is allocated, the system must wait for the serial port to become free before the next line of text can be sent, and this can delay the execution of commands and sequences. When a buffer is allocated with ZB, the trace data is stored in the buffer as fast as it is generated, and printed whenever the serial port is ready. This means that command execution is only delayed by the time taken to format the text and put it in the buffer, instead of having to wait for the serial port.

Example : ZB5
This command allocates 500 bytes from system memory to the ZZ trace data buffer.

ZM[$var] **Monitor variable (restricted).**

This command is used to set up automatic monitoring of selected variables. The command ZM$VAR enables monitoring of the specified variable. The system may be set up to monitor several variables at the same time, by entering separate ZM commands for each required variable. When a value is assigned to a variable which is being monitored, the variable name and its value are displayed via the terminal in the same way as for the ZZ trace function. ZM with no parameter lists the names and values of all variables currently being monitored.

To disable monitoring of a specific variable, use ZO$VAR. To disable monitoring of all variables, use ZO with no parameter. ZM can not be used with an array element.

ZO[$var] **Disable variable monitoring (restricted).**

The ZO command turns off monitoring of the specified variable. If no parameter is given, monitoring is disabled for all variables. ZO can not be used with an array element.

LGnn **Log motor trace data**
Range : 1 to 32767
DG **Display logged data**

The LG command enables logging of motor trace data, as selected by the TW and TI trace options commands. Data is stored continuously in a circular buffer, until any motor error occurs on the current channel when the data buffer is frozen. The LG parameter specifies the number of samples to be logged. Note that the TW options should only include data for channels on the same board or node as the current channel. Once a motor error occurs, the captured data may be displayed with the DG command. Another LG command must be executed to start logging data again if required.

ZT List command threads

This command displays all command execution threads (sequences or strings) that are currently running. It can be useful to check whether or not a sequence or string has stopped in the middle, waiting for some condition to become true, or if there is a problem with sequence nesting. A common error is executing a map on a slave axis using the software clutch when its master axis is stopped; the sequence stops at the XM command because the clutch does not complete until the master axis moves past the current slave axis position. The ZT command displays information about each command thread, including sequence numbers, channel numbers and channel level commands.

The information reported by the ZT command is very detailed as it is also used during software development. Some of this may be useful in debugging a user application, but certainly not all.

Example:

```
2>        ES4
S4:       CH2/ZC5000/PC
S4:       ML1/MW1/CL2000/XM0
S4:       WT256/DP/ST
S4:       <CR>
2>        CH1
1>        MA1000/MA0/RP
1M        CH2
2>        XS4
2C        ..
..        ZT
PTS command threads
===================
T1 Str0 CH1 endstr
T2 Seq4 (Lvl 1 of Str0) CH2 XD1025
```

This example shows that thread T1 is running command string Str0 on channel 1, and the host is waiting for the end of the string. Thread T2 is running sequence 4 at level 1 of string Str0 on channel 2. The host is waiting for the end of the XD1025 command, which is the channel level component of the top level sequence 4.

5. Status and Error Messages

5.1 Status Messages

This section gives the system status messages in various circumstances.

> Normal prompt
> This is the prompt character in position control mode. The system is ready for the next command.

: Motor off prompt.
This is the prompt character in the motor off state.

? New value prompt.
This character is used to prompt for a new parameter value.

A Executing map alignment.
This channel is starting to enter position mapping using an alignment move (MW bit 0 set to 0). If this prompt is visible the XM command is still executing.

C Executing software clutch.
This channel is starting to enter position mapping using the software clutch (MW bit 0 is set to 1). If this prompt is visible the XM command is still executing.

I Initialising.
The system is executing the IN initialise command.

M Moving.
The system is executing a normal trapezoidal move.

P Profile move.
The system is executing a stored profile move.

S Stopping
The system is executing a normal controlled stop.

V Velocity control mode.
The system is executing a constant velocity move.

W Waiting.
The system is waiting for some condition before continuing.

X Executing a position mapping.
This channel is linked to another master axis and is executing a position mapping (and has completed the clutch or alignment phase). 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. However, if an error occurs while the serial port is busy, then a short error message is displayed.

AH Analogue input high limit exceeded.
The high limit set on the analogue input has been exceeded.

AL Analogue input low limit exceeded.
The low limit set on the analogue input has been exceeded.

B Binary number required.
The system received a non-binary character when it expected a binary number as input. Characters allowed in a binary value are 0, 1, X, T.

CT Clutch timeout: map does not reach slave position.
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 (0–9) as input.

DF Command decoding failed.
This is an internal system error message. It indicates that there was some problem in decoding the command. It is used during development, and should not arise in a normally working system.

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

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 on another channel is assigned to this channel’s analogue output.

E Analogue output in use for position control.
The auxiliary output cannot be assigned to the specified channel's analogue output. The output is being used for normal position control.

E <cmd>: Reference inputs disabled or not defined.
The command cannot be executed because reference inputs are disabled or not defined.

E Cannot compile sequence while executing any sequence.
A sequence cannot be compiled because a sequence is already being executed on the indicated channel. This error message normally occurs when a sequence is executed without being previously compiled.

E Cannot transfer map while it is executing.
A map cannot be transferred to a channel which is already executing the same map.

E Cannot transfer profile while it is in use.
A profile cannot be transferred to a channel which is already executing the same profile.

E <cmd>: No output group defined.
The command given can only be used when there are expanded output lines defined.

E No commands after <cmd>.
This command can not be used at the end of a command line.

E No commands before <cmd>.
This command can not be used at the beginning of a command line.

E MLn: Already linked to channel n.
A slave channel cannot be linked to more than one master channel.

E MLn: Link would complete a closed loop.
Channels cannot be linked in a closed chain.

E TS: Error setting time.
The parameter entry for the TS command was not correct, or a system error occurred on attempting to set the new time.

E Invalid channel change.
Cannot change to a channel which is not fitted to the system.

E <cmd>: Sequence stack overflow.
A sequence or set of nested sequences must not call themselves; this would cause indefinite recursion.

E Only one repeat allowed in any command line.
Only one repeat command may be given on one command line.

E Command <cmd> not available.
The command given is not available on this version of the system.

E <cmd>: Not a DI input
Cannot list the input function for a line which is not a DI input.

E EL command without matching IF or II
The EL command is only valid following IF or II

F Nvm write failed.
The parameters and data could not be saved to nonvolatile memory successfully.

F Nvm verify failed.
The parameters and data saved to nonvolatile memory have not verified correctly.

F Saved data overflows nvm space.
The save parameters command has more data to save than will fit into the available nonvolatile memory space.

F Checksum error.
The calculated checksum value does not match the checksum saved with the data.

F Stored data invalid.
The stored data is invalid.

F Can’t access stored data: error n.
The system cannot access parameters stored in nonvolatile memory. The error number is an internal system error code.

F Error sending command.
An error occurred while sending the stored setup parameters to the motor channel.

FA Sequence/profile data pointer error
An internal error has occurred on the axis controller module.

FA <cmd>: String pointer error
An internal error has occurred on the axis controller module.

G Motor position error.
The system measured an instantaneous position error greater than the maximum allowed position error set by the SE command.

H <cmd>: Hexadecimal number required.
The system received a non-hexadecimal character when it expected a hexadecimal number (0–9, A–F) as input.

Ln Limit switch detected.
A limit switch input has been detected.

LH High position limit exceeded.
The motor position moved above the maximum value set by the LH command

LL Low position limit exceeded.
The motor position moved below the minimum value set by the LL command.

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.

MT Map update timeout.
The slave axis has not received any position updates from the master axis for longer than the map timeout period.

N <cmd>: Working memory full.
There is no more room in working memory for temporary storage for the given command.

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>: Target position outside limits.
The target position specified in a move command is outside the low or high position limits.

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

O <cmd>: Map is undefined or has not been transferred.
The specified map has not been entered, or has not been downloaded to the channel.

O <cmd>: Profile is undefined or has not been transferred.
The specified profile has not been entered, or has not been downloaded to the channel.

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

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

O MLn: Invalid master channel.
A slave channel cannot be linked to itself, or to a channel not fitted to the system.

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 valid reference input signals.

SY This message indicates that some internal system error has occurred. These should not arise in a normally working system, but are reported for fault finding and debugging. If the long error messages are enabled, then more detail about the error condition is given.

T Motor or encoder timeout.
No encoder counts have been received at all for the timeout period, although some were expected. This may be caused by a motor, drive, or encoder fault, drive shut down, or loss of encoder power supply.

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.

VA Undefined variable <var>.
An attempt has been made to reference a variable before it has been defined. A variable is defined by setting its value either using the PTS or the Operator's Panel.

VA Error accessing variable <var>.
The specified variable could not be accessed. This message normally indicates that the database is overloaded. The situation may be improved by reducing the access to variables to a minimum.

VA Error setting variable <var>.
The specified variable could not be set. This message normally indicates that the database is overloaded. The situation may be improved by reducing the access to variables to a minimum.

VA Attempt to divide by zero variable <var>.
An attempt has been made to divide by a variable which has a value of zero. This is not allowed as it would give an invalid result.

WF Watchdog test failed
The watchdog test has failed.

WT Watchdog timeout.
The watchdog timeout period has expired.

5.3 Status Codes

This section gives the numeric values for the status codes which are placed in the status variable defined by the VS command. The top four bits indicate the current mode, and the bottom four bits indicate additional information for some modes, particularly mapping and wait modes.

0	0x00	Idle mode (PC)
16	0x10	Constant velocity mode (VC)
32	0x20	Moving (MA, MR)
48	0x30	Executing a profile (XP)
64	0x40	Executing a position mapping (XM)
	1	Synchronised
	2	Executing alignment move
	3	Executing software clutch
80	0x50	Stopping (from ST)
96	0x60	Initialising (IN, IB)
128	0x80	Motor off (MO)
144	0x90	Waiting..
	1	..for time (WT)
	2	..for input line (WI)
	3	..for absolute position (WA)
	4	..for relative position (WR)
	5	..for reference input (WF)
	6	..for bounds wraparound (WB)
	7	..for bounds counter value (WC)

5.4 Error Codes

This section gives the numeric values (decimal and hexadecimal) for the error codes which are placed in the error variable defined by the VE command.

1	0x01	Error setting sleep time
2	0x02	Error setting timeout for channel signals
3	0x03	No data ready
4	0x04	Error testing for data ready
5	0x05	Not ready to receive data
6	0x06	Error testing ready-for-data
7	0x07	Error setting up signals
8	0x08	Error setting up event n
9	0x09	Cannot access channel 1
10	0x0A	Error closing path to n
11	0x0B	MLn: Already linked to channel n
12	0x0C	MLn: Link would complete a closed loop
13	0x0D	MLn: Cannot link channel to itself
14	0x0E	MLn: Invalid master channel
15	0x0F	Not linked to a master channel
16	0x10	<cmd>: cannot use the same channel for both master axes
17	0x11	Not linked to second master channel
18	0x12	TS: Error setting time
19	0x13	Can’t access stored data
20	0x14	Stored data invalid
21	0x15	Saved data overflows nvm space
22	0x16	Error sending command
23	0x17	Nvm write failed
24	0x18	Checksum error
25	0x19	Error setting up display buffer
26	0x1A	Error fetching display sample count
27	0x1B	Previous signal to channel still pending
28	0x1C	Map slave channel address table is full
29	0x1D	Slave channel address not found in table
30	0x1E	Device not ready
31	0x1F	Error: n
32	0x20	Error reading error status
33	0x21	Restricted command <cmd>
34	0x22	Restricted parameter <cmd>
35	0x23	Error reading status
36	0x24	Invalid channel change
37	0x25	Cannot change to CHn twice in same sequence
38	0x26	More than 1023 channel level sequences
39	0x27	Error sending output to terminal
40	0x28	Expression <expr> too deeply nested
41	0x29	Error accessing variable <var>
42	0x2A	Error setting variable <var>
43	0x2B	Undefined variable <var>
44	0x2C	Error notifying variable <var>

45	0x2D	Attempt to divide by zero variable <var>
46	0x2E	<cmd>: Working memory full
47	0x2F	<cmd>: Too many threads in use
48	0x30	Interpolation speed too high for command <cmd>
49	0x31	Interpolation radius too small for command <cmd>
50	0x32	Timeout starting interpolation command
51	0x33	Channel signal lost
52	0x34	EL command without matching IF or II
53	0x35	Unknown command <cmd> - type HE for help
54	0x36	Invalid command entry <cmd>
55	0x37	Cannot execute <cmd> while <state>
56	0x38	Cannot change <cmd> while <state>
57	0x39	<cmd>: Parameter out of range
58	0x3A	Reference timeout
59	0x3B	Reference error outside limits
60	0x3C	Reference correction overrun
61	0x3D	<cmd>: Reference inputs disabled or not defined
62	0x3E	<cmd>: Memory full
63	0x3F	<cmd>: Thread stack overflow
64	0x40	<cmd>: String memory full
65	0x41	<chan>: Failed to reach target position
66	0x42	<cmd>: Target position outside limits
67	0x43	<cmd>: Moving away from wait position
68	0x44	<cmd>: Undefined sequence
69	0x45	<cmd>: Profile is undefined or has not been transferred
70	0x46	Profile step data too large
71	0x47	<cmd>: Cannot enter or execute sequence while it is in use
72	0x48	<cmd>: Cannot transfer profile while it is in use
73	0x49	Table data pointer error
74	0x4A	<cmd>: Line already defined
75	0x4B	<cmd>: Input line too noisy
76	0x4C	<cmd>: Second position must be above first
77	0x4D	<cmd>: No output group defined
78	0x4E	<cmd>: Not a DI input
79	0x4F	<cmd>: String pointer error
80	0x50	No commands before <cmd>
81	0x51	No commands after <cmd>
82	0x52	Only one repeat allowed in any command line
83	0x53	Cannot execute command string while busy
84	0x54	Cannot execute command string on DIn while busy
85	0x55	<cmd>: Command decoding failed
86	0x56	<cmd>: Map is undefined or has not been transferred
87	0x57	Map data value too large
88	0x58	<cmd>: Cannot transfer map while it is executing
89	0x59	Clutch timeout: map does not reach slave position
90	0x5A	Host input buffer overflow
91	0x5B	Command <cmd> not available
92	0x5C	Command <cmd> not available on this channel
93	0x5D	Watchdog test failed

94	0x5E	User error code n
95	0x5F	Limit switch input detected
96	0x60	Motor position error
97	0x61	Motor or encoder timeout
98	0x62	High position limit exceeded
99	0x63	Low position limit exceeded
100	0x64	Map position update timeout
101	0x65	Map position overflow
102	0x66	Error reading encoder data
103	0x67	Error reading SSI encoder
104	0x68	Demand position change too large
105	0x69	Clock tick during servo loop
106	0x6A	Missed servo loop execution
107	0x6B	Motor error n
108	0x6C	Watchdog timeout
109	0x6D	Loss of signal
110	0x6E	Tracking error
111	0x6F	Analogue input high limit exceeded
112	0x70	Analogue input low limit exceeded
113	0x71	Cannot execute <cmd> while monitor function defined
114	0x72	<cmd>: Analogue input in use for position control
115	0x73	<cmd>: Analogue setpoint outside limits
116	0x74	Cannot execute <cmd> during analogue range initialisation
117	0x75	More than 1023 channel level strings
118	0x76	Only one pending ER allowed at a time
119	0x77	Command interpreter error - <cmd>
120	0x78	<cmd>: Stack overflow
121	0x79	<cmd>: Input line function still active
122	0x7A	<cmd>: Trigger variable function still active
123	0x7B	<n>: Position list is undefined
124	0x7C	Invalid map table
125	0x7D	<cmd>: No CAN chip fitted to board
126	0x7E	CAN chip stuck in reset
127	0x7F	No CAN network transceiver power
128	0x80	This module is CANbus OFF
129	0x81	Unable to log onto CANbus
130	0x82	<cmd>: No free CAN mailbox
131	0x83	<cmd>: No CAN mailbox assigned
132	0x84	Another module is already clock master
133	0x85	Module n is not receiving clock messages
134	0x86	Another module already uses our module number
135	0x87	XN(s) lost on CANbus
136	0x88	Module n lost RX data on CANbus
137	0x89	<cmd>: No more map links available
138	0x8A	Module n has been CANbus OFF
139	0x8B	Module n is not responding
140	0x8C	Module n has suffered power failure
141	0x8D	Module n: SERVOnet hardware error detected
142	0x8E	RN: Module n has wrong number of channel

143	0x8F	Cannot change ZB while buffer in use
144	0x90	No response from drive
145	0x91	Drive enable feedback failed
146	0x92	Drive under/over voltage (alarm 7)
147	0x93	Drive power module fault (alarm 6)
148	0x94	Drive earth fault
149	0x95	Drive over temperature (alarm 4)
150	0x96	Drive I2t limit exceeded (alarm 2)
151	0x97	Drive resolver fault (alarm 5)
152	0x98	Drive overspeed error (alarm b)
153	0x99	Drive motor thermostat (alarm 3)
154	0x9A	Drive software watchdog (alarm 9)
155	0x9B	Drive firmware not OK (alarm F)
156	0x9C	Drive parameters not OK (alarm E)
157	0x9D	Drive motor link fault (alarm C)
158	0x9E	Drive alarm register bit 9 set
159	0x9F	Drive asynchronous motor overspeed (alarm U)
160	0xA0	Drive alarm register bit 11 set
161	0xA1	Drive alarm register bit 12 set
162	0xA2	Edit mode verify failed
163	0xA3	Cannot use <cmd> to abort itself
164	0xA4	<cmd> not allowed inside a sequence
165	0xA5	Arithmetic overflow executing <cmd>

6. Interfacing

6.1 Notes on Installation

The PTS system is a sophisticated computer system, and care should be taken in all installations to protect the unit from high voltages and to minimise 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 PTS unit and the encoders to minimise 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 reinitialise 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.

PTS Mk2: A custom screw terminal block known as the Diagnostic Interface (DI-4) is available if required. It is designed specifically for use with the SRV-4 four axis controller used in the PTS. It makes the electrical installation much easier, and provides useful status and diagnostic facilities. It includes a 9-way D socket for each encoder input, with LED indication of the encoder signal states, and two part screw terminal connectors for all other signals including the 5V encoder supply input. It also provides the motor enable relays for each channel. All of the 24V digital input and output signals have LED status indicators fitted. It is highly recommended that new installations make use of this interface wherever possible.

6.2 Safety

The PTS 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
LH	Set high position limit
LL	Set low position limit
DL	Define limit switch inputs
OB	Define motor brake output
BD	Set brake delay time

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 lines per turn is seen as generating 4000 counts per turn. The encoder input signals are all fully isolated. The standard systems are designed for use with encoders having complementary line driver outputs, for 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. The table below shows the maximum encoder input count rates for the different hardware units.

Unit	Input frequency	Count rate	Max rpm with 2500 line encoder
MiniPTS 1+1	300 kHz	1.2×10^6	7200 rpm
Q-Drive	600 kHz	2.4×10^6	14400 rpm
MiniPTS 2+1 and 3 MiniPTS 4 PTS Mk2	1.2 MHz	4.8×10^6	28800 rpm

Table 7: Maximum encoder input count rate

Most PTS systems support other types of encoder if required, including absolute encoders. Q-Drives use resolver data from the drive itself for position feedback.

MiniPTS 4 : The SRVX-248/E expansion board for the PTS is fitted with an additional incremental encoder input channel. This fifth encoder channel may be used in the motor off state for synchronising to an external master axis. The fifth channel may also be programmed in virtual mode as another axis if required, but since there is no fifth analogue output it cannot be used to control a motor.

6.4 Demand Output

The normal 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 for each axis. The motor drive is normally connected such that a positive demand output signal causes the motor to move in the positive direction.

The Q-Drive systems use a digital high speed serial link between the PTS hardware and the drive amplifier module instead of the analogue demand signal. They also have a dedicated hardware drive enable signal which disables the drive in the motor off state.

6.5 Relay Contacts

The standard PTS systems use a 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.

The Q-Drive systems use an internal hardware drive enable signal to disable the drive in the motor off state, instead of the reed relay. They use a digital high speed serial link to send the setpoint to the drive, instead of an analogue signal.

6.6 Motor Brake

The PTS systems can provide a motor brake control signal, programmable on any digital output line with the OB command, which switches automatically when the motor drive is enabled and disabled with the PC and MO commands. When a motor channel is switched from motor off to position control mode, the drive is enabled before releasing the brake. When a channel is shut down to motor off, the brake is engaged before disabling the drive. The delay time between enabling the drive and releasing the brake, or engaging the brake and disabling the drive, is set by the brake delay time parameter BD.

Note that a Q-Drive fitted with the brake relay option uses an internal hardware motor brake signal and does not need to have a separate brake output line defined. There is also an interlock such that if the drive fails to enable, the brake is not released and the channel stays in motor off.

6.7 Digital Inputs and Outputs

The system has a number of digital inputs and outputs, grouped in sets of 8. 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:

Product		Digital Inputs	Digital Outputs
Q-Drive 1+1/MAP		8	8
MiniPTS 1+1		8	8
MiniPTS 2+1/3		16	8
MiniPTS 4		16	8
(with SRVX-248 expansion board)		40	16
PTS Mk2 (per axis card)		16	8
(with SRVX-248 expansion board)		40	16
(with SRVX-2424 expansion board)		40	32
SERVOnet	Q-Drive SERVOnet	8	8
	MiniPTS 3 SERVOnet	16	8

Table 8: Digital inputs and outputs on PTS systems

Only the first four inputs of group 1 of each unit have the fast response needed for use as a reference or snapshot input.

6.8 Analogue Inputs

The SRV-2 controller used in the MiniPTS 2+1 and MiniPTS 3 has three analogue inputs. The signal inputs are multiplexed, and the multiplexer output is buffered with a differential amplifier. The analogue inputs have a range of ±10V. The analogue signal levels are converted to digital values at 12 bits resolution. The DA command displays the analogue input signal value for the current channel.

The SRV-4 controller used in the MiniPTS 4 and PTS Mk2 has one analogue input. The signal input is buffered with a differential amplifier and has a range of ±10V. The analogue signal level is converted to a digital value at 12 bits resolution. This value is displayed with the DA command. The optional expansion board includes a 4:1 analogue multiplexer, providing four separate analogue inputs. If this board is fitted, then one analogue input signal is assigned to each of the four motor channels, and the DA command displays the value for the current channel.

The Q-Drives have a single analogue input with a range of ±10V, available to both motor channels. The hardware resolution is 11 bits, but the value is scaled to 12 bits (±2047) for compatibility with the other systems. The analogue input value is displayed with the DA command, unless the system has been configured to return a different value from the drive by setting the QA parameter.

The MiniPTS 1+1 has a single analogue input, available to both motor channels. The signal input is buffered with a differential amplifier and has a range of ±10V. The analogue signal level is converted to a digital value at 12 bits resolution. This value is displayed with the DA command on either channel.

6.9 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 low 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.

6.10 Reference Inputs

A reference input is required on each motor channel during the IN initialisation 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 initialisation 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) on group 1 of each module or four axis card by using the DR command. A dedicated encoder marker input is also available, enabled by the DZ command. See section 4.12 for more details on the full range of reference facilities.

6.11 Serial Communications

The serial link uses RS-232 signal levels as standard, but may be configured for RS-485 if required. 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. All the PTS systems use xon/xoff software handshake by default on the programming terminal port, but they also support hardware handshake using RTS and CTS signals.

Q-Drive 1+1, MiniPTS 2+1/3, PTS Mk2: The serial ports are set up by the CF command (configure serial port options), described earlier on page 149.

MiniPTS 4: To enable hardware handshake, fit a jumper link between J10 pins 1-2 on the SRV-4 board. Refer to the Installation Manual or Hardware Manual for details of jumper link locations.

For operation using hardware handshake, connect the PC, with hardware handshake enabled in the settings for the Windows Terminal or PTS Toolkit software, using a cable with the RTS and CTS lines connected. Then power up the system. Pin 8 (CTS) on the terminal port must be pulled high (logically ON) when the system is powered on for hardware handshake to be enabled. Connecting to the PC first should ensure that this is correct.

To return to software handshake using xon/xoff, remove the PC connection and power the system off and back on. The system detects that the CTS line is low and disables hardware handshake. If there is any risk that a PC may be connected using software handshake, with a cable configured for hardware handshake, then reconfigure the serial port for software handshake.

IMPORTANT
The PTS must be set for software handshake if the connection to the PC is removed for any length of time. If the connection needs to be removed without powering off the PTS, then the RTS and CTS pins of the terminal port should be linked immediately after the PC serial cable is disconnected. This should only be used as a temporary measure until the PTS can be safely powered off and reset to software handshake.

6.12 I/O Expansion Boards

Two optional expansion boards are available for the SRV-4 controller board, used in the MiniPTS 4 and PTS Mk2, which provide some additional hardware facilities.

The SRVX-248 expansion board provides an extra 24 inputs and 8 outputs, giving a total of 40 inputs and 16 outputs. The SRVX-2424 expansion board provides an extra 24 inputs and 24 outputs, giving a total of 40 inputs and 32 outputs. Both boards also have an analogue multiplexer providing four analogue input channels. When one of these boards is fitted, the input/output commands and configuration commands allow the use of higher group numbers for programming these extra lines, and one analogue input signal is assigned to each of the four motor channels.

The SRVX-248/E expansion board is fitted with an additional incremental encoder input channel, for use in the MiniPTS 4. This fifth encoder channel may be used in the motor off state for synchronising to an external master axis. The fifth channel may also be programmed in virtual mode as another axis if required, but since there is no fifth analogue output it cannot be used to control a motor.

7. Summary

7.1 Commands

Note that some commands are restricted. These commands can be used only in privileged mode. Commands which are executed at host level are followed by (host). Commands which are affected by scaling are followed by (scaled).

Miscellaneous commands

VN	display Version Number and revision date	(host)
PM	set to Privileged Mode	(host)
NM	set to Normal Mode	(host)
PW	set PassWord	(restricted)(host)
TX	enter comment TeXt	(host)
SK	set Software license Key	(restricted)(host)
CF	ConFigure serial port options	(restricted)(host)
OP	configure Operator's Panel	(restricted)(host)

Parameter file commands

SP[bb]	Save Parameters to nonvolatile memory	(restricted)(host)
RD[bb]	Read Data from nonvolatile memory	(restricted)
RS[bb]	ReSet complete setup to defaults	(restricted)(host)
LA[bb]	List All parameters	(host)
CS	CheckSum test	(host)
FM	display Free Memory	(host)
ZF	Zero File system in nvm	(restricted)(host)

Mode commands

MO	set to Motor Off	
PC	set to Position Control mode	
VMn	set Virtual Motor mode	(restricted)
EAnn	set motor off/Error Analogue ramp time	(restricted)
OB[g:]n±	define motor Brake Output	(restricted)
BDnn	set Brake Delay time	(restricted)

Move commands

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

Set parameter commands

SVnn	Set Velocity	(scaled)
SAnn	Set Acceleration	(scaled)
SDnn	Set Deceleration at end of move	(scaled)
DCnn	set DeCeleration for stop command	(scaled)
SSnn	Set Slow jog speed	(scaled)
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	
SUnn	Set Units	(restricted)(host)
FPnn	Set Floating point Precision	(restricted)(host)
MWbb	set Move/map options Word	(restricted)
CWbb	set Control Word	(restricted)

Sequence commands

ESnn	Enter Sequence	(restricted)(host)
LS[nn]	List Sequence	(host)
XSnn	eXecute Sequence	(host)
RP[nn]	RePeat command line	
ER	End Repeat	
AX	Abort command eXecution	
BK	BreaK out of sequence	(host)
ASnn	set AutoStart sequence	(restricted)(host)
FM	display Free Memory	(host)

Multi-channel commands

CHn	CHange channel	(host)
CPn	Change channel in Parallel	(host)
CM[nn]	CoMpile sequence(s)	(host)
GS	Global Stop	(host)
GA	Global Abort	(host)
GF	Global motor oFf	(host)
GX[nn]	Global abort eXecution	(host)
USnn	send User Signal	
HWnn	Host system Wait for user signal	(host)
ZSnn	Zero user Signal	(host)
SXn/n	set Sequence to eXecute on signal	(restricted)(host)
MUnn	Mask User signal	(restricted)(host)
EUnn	Enable User signal	(restricted)(host)
MEnn	set Motor Error sequence	(restricted)(host)
UEnn	set User Error sequence	(restricted)(host)

Profile commands

EPnn	Enter Profile	(restricted)(host)
LP[nn]	List Profile	(host)
XPnn[–]	eXecute Profile	
PVn	set Profile Velocity	
SMnn/nn	Scale profile	
TPnn	Transfer Profile	(host)
FM	display Free Memory	(host)

Map commands

EMnn	Enter Map	(restricted)(host)
LM[nn]	List Map	(host)
XMnn	eXecute Map	
MSnn	set Map Step	(host)
TMnn	Transfer Map	(host)
MLnn	Map Link to master channel	(host)
NLnn	map link to differential channel	(host)
UL	UnLink from master channel	(host)
LWbb	set Link options Word	(restricted)(host)
MTnn	set Map update Timeout	(restricted)
MB±nn	set Map Base offset	
MF±nn	set Map oFfset	
SMnn/nn	Scale Map	
SNnn/nn	Scale differeNtial	
AVn	set map base/offset/scale Adjustment Velocity	
MPnn	set Master Position bound	(restricted)
NPnn	set differeNtial Position bound	(restricted)
BR	set map scale factor from Bounds Ratio	
MWbb	set Map/move options Word	(restricted)
CLnn	set software Clutch Length	
BAnn	set map Base Advance	
BTn	set map Base advance master speed averaging Time	
GM	Get Mapped master bound position	
GN	Get mapped differeNtial bound position	
GW	Get Wraparound offset value	
FM	display Free Memory	(host)

Wait commands

WTnn	Wait for Time	
WI[g:]n±	Wait for Input line	
WA±nn	Wait for Absolute position	(scaled)
WR±nn	Wait for Relative position	(scaled)
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)(scaled)
TOnn	set TimeOut	(restricted)
LH±nn	set High position Limit	(restricted)(scaled)
LL±nn	set Low position Limit	(restricted)(scaled)
RTnn	set Reference Timeout	(restricted)
MTnn	set Map update Timeout	(restricted)
EAnn	set motor off/Error Analogue ramp time	(restricted)
MEnn	set Motor Error sequence	(restricted)(host)
UEnn	set User Error sequence	(restricted)(host)
DL[g:]n±	Define Limit switch input	(restricted)
DE[g:]n±	Define Error output	(restricted)
AE[g:]n±	define Analogue limit Error output	(restricted)
EWbb	set Error options Word	(restricted)
LE[n]	display Last Error(s)	(host)

Gain commands

KPnn	set Proportional gain constant	(restricted)
KInn	set Integral gain constant	(restricted)
KDnn	set Differential gain constant	(restricted)
KVnn	set Velocity feedback gain constant	(restricted)
KFnn	set velocity feedForward gain constant	(restricted)
KAnn	set Acceleration feedforward gain	(restricted)
FE	display Following Error	
DK	Display system constants	(host)

Encoder input commands

FSnn	Feedback Select encoder type	(restricted)
FCnn	Feedback Channel	(restricted)
NBnn	set Number of Bits for absolute encoder	(restricted)
NZnn	set Number of Zeros for absolute encoder	(restricted)
VMn	set Virtual Motor mode	(restricted)

Analogue output commands

OLnn	set analogue Output Limit	(restricted)
EAnn	set motor off/Error Analogue ramp time	(restricted)
DU[g:]n±	Define Unipolar direction control output	(restricted)
UTnn	set Unipolar direction output delay Time	(restricted)
AOn	set Auxiliary Output channel	(restricted)
SFn	Set monitor Function	(restricted)
KMnn	set Monitor output gain constant	(restricted)
OMnn	set Offset on Monitor output	(restricted)

Reference commands

ZC[nn]	Zero position Counters or set position	(scaled)
SBnn	Set Bound position	(restricted)(scaled)(host)
BO[g:]n±	define Bound Overflow output	(restricted)
BCnn	set Bounds Counter	
DZn	Define Zero marker input	(restricted)
DRn±	Define Reference input	(restricted)
RLnn	set Reference repeat Length	(restricted)
RF±nn	set Reference oFfset	(restricted)
RMn	set continuous Reference Mode on/off	
RWbb	set Reference options Word	(restricted)
RHnn	set Reference Holdoff time	(restricted)
SRnn	Set maximum Reference correction	(restricted)
FRnn	set Filter on Reference error	(restricted)
LRnn	set Limit on Reference error	(restricted)
RTnn	set Reference Timeout	(restricted)
RVnn	set Reference correction Velocity	(restricted)
RJ±nn	set Reference adJustment position	(restricted)(scaled)
ZHnn	set reference true High limit	(restricted)
ZLnn	set reference true Low limit	(restricted)
FHnn	set reference False High limit	(restricted)
FLnn	set reference False Low limit	(restricted)
RA[g:]n±	define Reference Accepted output	(restricted)
RR[g:]n±	define Reference Reject output	(restricted)
JF[g:]n±	define ref. adJustment Forwards output	(restricted)
JB[g:]n±	define ref. adJustment Backwards output	(restricted)
WF	Wait for reFerence signal	
DF	Display reFerence error	

Input/output commands

SO[g:][n]	Set Output line(s)	
CO[g:][n]	Clear Output line(s)	
OC[g:]nn	Output Code value on expanded outputs	
RI[g:][n]	Read Input line(s)	(host)
RO[g:][n]	Read Output line state(s)	(host)
II[g:]n±	If Input true do command line	(host)
IO[g:]n±	If Output true do command line	(host)
MI[g:][n]	Mask Input(s)	
BI[g:][n]	inhiBit Input(s)	
EI[g:][n]	Enable Input(s)	
GB	Global inhiBit all inputs	
GE	Global Enable all inputs	
WI[g:]n±	Wait for Input line	

Configuration commands

DZn	Define Zero marker input	(restricted)
DRn±	Define Reference input	(restricted)
DL[g:]n±	Define Limit switch input	(restricted)
DI[g:]n±	Define function Input	(restricted)
DX[g:]n±	Define eXpanded input lines	(restricted)
PSn±	define Position Snapshot input	(restricted)
MG[g:]bb	define input Mask Group	(restricted)
BG[g:]bb	define input inhiBit Group	(restricted)
DE[g:]n±	Define Error output	(restricted)
OB[g:]n±	define motor Brake Output	(restricted)
BDnn	set Brake Delay time	(restricted)
PO[g:]n±	define Position trigger Output	(restricted)(scaled)
OX[g:]n±	define eXpanded Output lines	(restricted)
BO[g:]n±	define Bound Overflow output	(restricted)
RA[g:]n±	define Reference Accepted output	(restricted)
RR[g:]n±	define Reference Reject output	(restricted)
JF[g:]n±	define ref. adJustment Forwards output	(restricted)
JB[g:]n±	define ref. adJustment Backwards output	(restricted)
OW[g:]n±	define Outside Window output	(restricted)
AE[g:]n±	define Analogue Error output	(restricted)
DU[g:]n±	Define Unipolar direction control output	(restricted)
UTnn	set Unipolar direction output delay Time	(restricted)
DBnn	set input DeBounce time	(restricted)
LI[g]	List Input line definitions	(host)
LO[g]	List Output line definitions	(host)
FSnn	Feedback Select encoder type	(restricted)
FCnn	Feedback Channel	(restricted)
NBnn	set Number of Bits for absolute encoder	(restricted)
NZnn	set Number of Zeros for absolute encoder	(restricted)
CF	ConFigure serial port options	(restricted)(host)
OP	configure Operator’s Panel	(restricted)(host)

Timer/counter functions

TC[g:]n±	define Timer/Counter output	(restricted)
LC[g:]n	List Counter value	
TK[g:]n±	define Timer/counter clocK input	(restricted)
TG[g:]n±	define Timer/counter Gate input	(restricted)
TZ[g:]n±	define Timer/counter reset input	(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	(scaled)
DD	Display Demand position	(scaled)
FE	display Following Error	
DV	Display current Velocity	(scaled)
DF	Display reFerence error	
DS	Display position Snapshot data	
DA	Display Analogue input	
DT	Display Time	(host)
TShh:mm:ss	Time Set	(host)
DM[nn]	set continuous Display Mode on	(host)
DO	Display mode Off	(host)
TR[nn]	enable/disable TRace mode	(host)
TWbb	set Trace options Word	
TIbb	set Trace Input/output options	
TF	Trace options oFf	(host)
DK	Display system constants	(host)
CDnn	set Character Delay	(host)
DWbb	set Display Word	(restricted)(host)
HE	print HElp display	(host)
LE[n]	display Last Error(s)	(host)
SYbb	configure System displaY	(restricted)(host)

Analogue control commands

AMn	set Analogue control Mode	
APnn	set tension ratio control Proportional gain	(restricted)
AInn	set tension ratio control Integral gain	(restricted)
ADnn	set tension ratio control Differential gain	(restricted)
VPnn	set tension speed loop Proportional gain	(restricted)
VInn	set tension speed loop Integral gain	(restricted)
VDnn	set tension speed loop Differential gain	(restricted)
ACnn	set Analogue Control setpoint	
OLnn	set analogue Output Limit	(restricted)
AWbb	set Analogue control options Word	(restricted)
ARnn	set/display Analogue Range distance	(restricted)
MMnn	set/display Master analogue range distance	(restricted)
CR	Calculate Ratio from AR values	
XR	eXecute analogue Range initialisation	
AHnn	set Analogue input High limit	
ALnn	set Analogue input Low limit	
AE[g:]n±	define Analogue limit Error output	(restricted)
DA	Display Analogue input	

Variable commands

$var=nn	assign value to variable	(host)
WV$var	Wait for write to Variable	(host)
$var>cmds	define trigger variable	(restricted)(host)
VX	list trigger variables	(host)
MV[$var]	Mask trigger Variable(s)	(host)
EV[$var]	Enable trigger Variable(s)	(host)
LV[$var]	List Variable(s)	(host)
LX(expr)	List eXpression	(host)
VE$var	define Error Variable	(restricted)(host)
VS$var	define Status Variable	(restricted)(host)
ZM[$var]	Monitor variable	(restricted)(host)
ZO[$var]	turn variable monitor Off	(restricted)(host)
IF(expr)	execute commands IF expression true	(host)
EL	ELse command	(host)
IA	Initialise Array of variables	(host)
FPnn	Set Floating point Precision	(restricted)(host)
OP	configure Operator's Panel	(restricted)(host)

Q-Drive configuration commands

QQnn	Query Q-drive parameter	
QPnn/nn	set Q-drive Parameter to value	(restricted)
QAnn	set Q-drive parameter for DA	(restricted)

SynchroLink commands

CQ[bb]	Query CANbus/SynchroLink status	(host)
CKn	enable/disable CAN clocK	(restricted)(host)
CNn	Configure Node number	(restricted)(host)
MLnn:nn	Map Link	(host)
NLnn:nn	differeNtial Link	(host)
XNnn	eXecute remote sequence	(restricted)(host)

SERVOnet commands

CQ	Query CANbus/SERVOnet status	(host)
SI	configure SERVOnet node Identities	(restricted)(host)
MLnn:nn	Map Link	(host)
NLnn:nn	differeNtial Link	(host)
XNnn	eXecute remote sequence	(restricted)(host)
RNnn	Reset Node	(restricted)(host)
LN	List SERVOnet Node table	(host)
NTnn[/t]	set Node Type	(host)
NNnn[/c]	set Number of channels	(host)
NQ[nn]	Query Node/table state	(host)

Edit mode commands

ED	enter EDit mode	(restricted)(host)
GO[n]	return to run mode	(restricted)(host)

Debug commands

ZZbb	set command trace options	(restricted)(host)
LD[nn]	List channel level sequences	(host)
ZBn	set Buffer for ZZ trace data	(restricted)(host)
ZM[$var]	Monitor variable	(restricted)(host)
ZO[$var]	turn variable monitor Off	(restricted)(host)
LGnn	LoG motor trace data	(host)
DG	Display loG of motor trace data	(host)
TWbb	set Trace options Word	
TIbb	set Trace Input/output options	
ZT	list command Threads	(restricted)(host)

7.2 Prompts and Status Messages

>	normal prompt in position control mode
:	motor off prompt
?	parameter value prompt
A	Aligning into position mapping
C	Clutching into position mapping
I	Initialising to reference position
M	Moving to new position
P	executing a Profile move
S	Stopping under normal deceleration
V	Velocity control mode
W	Waiting
X	executing a position mapping
ED:	system in Edit mode

7.3 Error Messages

AH	Analogue input High limit exceeded
AL	Analogue input Low limit exceeded
B	Binary number required
CT	Clutch Timeout
D	Decimal number required
DF	command Decoding Failed
E	Error - unrecognised command, invalid parameter, command not allowed at this time, 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
LH	High position Limit exceeded
LL	Low position Limit exceeded
MO	Map demand position Overflow
MT	Map update Timeout
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
SY	System error
T	motor Timeout
U	line in Use
VA	VAriable error
WF	Watchdog test Failed
WT	Watchdog Timeout

A. Using Absolute Encoders

A.1 Introduction

The PTS systems have the option to use absolute shaft encoders as the position feedback device. This section describes how to configure the sytems for use with absolute encoders, and what changes are made in the software when the absolute option is selected.

MiniPTS 2+1, MiniPTS 3, MiniPTS 4 and PTS Mk2 systems support the use of absolute encoders with an SSI interface. MiniPTS 1+1, MiniPTS 3 systems (both standalone and SERVOnet) and Q-Drive 1+1, MAP, and SERVOnet systems support absolute encoders with a CANbus interface, using the CANopen protocol.

A.2 Axis Configuration for SSI Encoder

The SSI encoder option is enabled by setting the FS parameter on the required motor channel, and by installing jumper links as follows.

MiniPTS 2+1, 3 and SERVOnet: : J4
MiniPTS 4, PTS Mk2 : J9

Mode	SSI encoder	Normal encoder
Channel 1	Link 1–2, 3–4	Remove 1–2, 3–4
Channel 2	Link 5–6, 7–8	Remove 5–6, 7–8
Channel 3	Link 9–10, 11–12	Remove 9–10, 11–12
Channel 4 *	Link 13–14, 15–16	Remove 13–14, 15–16

* MiniPTS 4 and PTS Mk2 only.

Please refer to the relevant hardware manual for more configuration details.

A.3 Axis Configuration for CANopen Encoder

The CANopen encoder option is available on the MiniPTS 1+1 and MiniPTS 3/ SERVOnet, and the Q-Drive 1+1/MAP/SERVOnet. One CANopen encoder only can be assigned to each controller or axis module. Any channel may be set to read the CAN encoder position by using the FS command. The CANopen encoder (which uses standard CAN messages) is set to have the same node number as that used by its allocated axis module/PTS controller (which uses extended CAN messages). The node number of the axis controller module is set by the CN command or by setting the SERVOnet module number. One module in a SynchroLink network needs to provide a CAN sync clock; this is enabled by CK1. In a SERVOnet system this is automatically provided by the machine controller.

The CANopen encoder must meet these specifications:

- Comply with CANopen CiA DS-406 for Encoder, class C1.
- Hardware compatible with CAN V2.0A active and CANV2.0B passive at 500kbit/s bus speed.
- Support the minimum CANopen bootup with node guarding and a predefined connection set (CiA DS-301).
- Be pre-configurable and maintain these settings, even without electrical power.

The CANopen encoder must be configured as follows:

- CANopen node number set to the same value as the corresponding PTS device.
- Transmit position data synchronously on receipt of sync. message, using PDO2, 1 transmission every sync. message.

A.4 Encoder Feedback Options

Encoder feedback options are selected by the FS parameter on each axis. If the option selected is for an absolute encoder, either SSI or CANbus, then the NB parameter sets the number of bits used, and the NZ parameter sets the number of additional leading data bits ignored. This allows each axis to be configured for absolute encoders from 12 bit single turn up to 24 bit multiturn. Note that each axis may be set for a different encoder feedback option.

A.5 Using the Absolute Encoder for Position Feedback

If an incremental position mode is selected by the feedback select parameter FS, then the PTS system behaves in exactly the same way as for normal incremental encoders. The absolute encoder is simply used to measure the change in position at each sample, not the absolute position.

If an absolute position mode is selected, then the system behaviour is modified to make use of the absolute position information from the encoder. In this case, the absolute position from the encoder is used as the feedback position for the control closed loop.

The PTS system allows most of the normal facilities to be used in absolute position feedback modes. The main areas where changes apply are listed here.

- The RF reference offset parameter is used to define an offset applied to the absolute position data received from the encoder. The current absolute position is given by the sum of the encoder position and RF. This allows the absolute zero position to be defined as required, without having to align the encoder to any specific position.
- The SB parameter is available and bounds wraparound operates as normal. This allows the absolute encoder to be used in multi-turn applications.
- The IN command is used to reset the current position to the absolute position value given by the sum of the encoder data and the reference offset RF. This value is then subject to wraparound by calculating the remainder when divided by the SB parameter, and the absolute position is set to this remainder.
- The IB initialize bounds and WF wait for reference commands are not available.
- The BL backlash correction parameter has no effect.
- All referencing facilities are disabled. Reference parameters may be set and read as normal, but referencing cannot be enabled, and has no effect.
- The reference timeout check is not executed, and the RT parameter has no effect.
- The ZC command is available, and operates as normal. It allows the current absolute position to be defined as required, without having to align the encoder to any specific position. The position value given with the ZC command is again subject to wraparound by taking the remainder when divided by SB.

A.6 Speed Limits with Absolute Encoders

When an absolute encoder option is selected, and the number of data bits is less than 15, the system imposes a lower maximum speed limit on the SV and SS commands. This is because the system must correctly identify the direction of motion for any change in position, and thus the maximum change in position in any one sample is plus or minus half the number of counts from the encoder. The maximum values allowed for the SV and SS commands are shown in the table below.

Number of data bits	Maximum speed (counts per second)
24–15	4 000 000
14	2 000 000
13	1 000 000
12	500 000

Table 9: Speed limits with absolute encoders

The speed limits at the lower encoder resolutions are equivalent to a maximum encoder speed of about 7000 r.p.m.

A.7 Connections

SSI encoders are connected to the same connectors on the PTS as normal incremental encoders. The table below gives the details for connecting a typical SSI encoder to the PTS.

Signal	9 way socket pin number
CLK +	1
DATA +	2
not used	3
Encoder + supply	4
Screen	5
CLK –	6
DATA –	7
not used	8
Encoder 0V	9

Table 10: SSI encoder connections

CANopen encoders are connected to the CANbus cable, often via a 9 way D-type, which has the following connections:

Signal	9 way pin number
reserved	1
CAN L (signal)	2
CAN ground (0V)	3
SERVOnet continuity line	4
CAN shield	5
CAN ground (0V)	6
CAN H (signal)	7
SERVOnet error line	8
CAN V+ (12V)	9

Table 11: CANopen encoder connections

Entries in bold are the minimum required for CANopen, but cable continuity for the remaining six wires must be maintained for SynchroLink and SERVOnet to work. Note that the entries in italics are specific to SERVOnet and should not be connected to CANopen devices. The CAN data rate used for SynchroLink and SERVOnet is 500kbit/s.

A.8 Suggested SSI Encoders

Type no.	Manufacturer		
ROC 424	Dr. Johannes Heidenhain GmbH Dr.-Johannes-Heidenhain-Straße 5 D-8225 Traunreut	Telephone Fax.	+49 (0) 86 69 / 31-0 +49 (0) 86 69 / 50 61
AG 661	Max Stegmann GmbH D-7710 Donaueschlingen Dürrheimer Straße 36 Postfach 1560	Telephone Fax.	+49 (0) 771 / 807-0 +49 (0) 771 / 807100
GEL 150 range	Lenord, Bauer & co. GmbH Dohlenstraße 32 D-4200 Oberhausen 11 (Königshardt)	Telephone Fax.	+49 (0) 208 99 63-0 +49 (0) 208 67 62 92

U.K. distributors for these are shown below.

Manufacturer	U.K. Distributor		
Heidenhain	Heidenhain (G.B.) Ltd. 200 London Road Burgess Hill West Sussex RH15 9RD	Telephone Fax.	+44 (0) 1444 247711 +44 (0) 1444 870024
Stegmann	Stegmann U.K. Ltd. 413 Warwick Road Birmingham	Telephone Fax.	+44 (0) 121 333 5551 +44 (0) 121 333 5501
Lenord + Bauer	Motor Technology (UK) Ltd. Motec House Chadkirk Industrial Estate Romiley Stockport Cheshire SK6 3LE	Telephone Fax.	+44 (0) 161 427 3641 +44 (0) 161 427 1306

Note that usually the SSI interface option, the number of counts per turn and the number of turns must all be specified.

A.9 Suggested CANopen Encoder

Type no.	Manufacturer		
AWC58	Fraba Sensorsysteme		
	Schanzenstraße 35	Telephone	+49 (0) 221 96213-0
	D-51063 Köln	Fax.	+49 (0) 221 96213-20

The U.K. distributor for this is shown below.

Manufacturer	U.K. Distributor		
Fraba	Servodynamic Ltd.		
	Cherry Orchard		
	Codsall Wood		
	Wolverhampton	Telephone	+44 (0) 1902 847331
	WV8 7QR	Fax.	+44 (0) 1902 847387

B. LED Status Codes

B.1 Introduction

The PTS is fitted with 7 segment LED displays, which are used to indicate the state of each axis, and also to indicate some error conditions on any axis. These displays are useful to show any activity on the unit, and to identify that an error has occurred, without a terminal or host system connected to the serial port.

The following pages show the status and error code values currently used.

B.2 Status Codes

The LED digits normally show the current status for all axes. The status codes are updated any time an axis changes state. 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.

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

B.3 Error Codes

If a motor error occurs on any axis, then the LED display indicates an error code. The error code remains on the LED display until the next PC command is executed on the channel which detected the error. System errors, used during software development, are not normally seen on a running system, but are described here for completeness.

MiniPTS 1+1 and Q-Drive systems: All motor errors display ‘E’, or limit switches ‘L’. System faults display an ‘F’.

MiniPTS 2+1 and 3: The top digit displays ‘E’, or ‘L’, the other two digits display the error code as in the table below. System faults display ‘F’ with the error code.

MiniPTS 4 and PTS Mk2: The top digit shows the number of the channel where the error has occurred, while the remaining digits show the error code.

Prefix	Error type	Code	
nE (n = channel no.)	Errors	01	Position error
		02	Timeout error
		03	High position limit
		04	Low position limit
		05	Reference timeout
		06	Reference out of limits
		07	Reference overrun
		08	Watchdog timeout
		09	Map update timeout
		10	Map position overflow
		11	Analogue input high limit
		12	Analogue input low limit
		13	Encoder counter fault
		14	SSI encoder too noisy
		15	Demand position change too large
nL (n = channel no.)	Limit switch	xx	Limit switch input detected First digit = input group number Second digit = line number
FA	Fault	00	Bus error
		01	Address error
		02	Illegal instruction
		03	Divide by zero
		04	Check exception
		05	Privilege violation
		06	Trace exception
		07	Opcode 1010 exception
		08	Opcode 1111 exception
		09	Breakpoint exception
		10	Unitialised interrupt vector
		11	Spurious interrupt exception
		12	Unexpected autovector interrupt
		13	Unexpected exception
		14	Unexpected trap
		15	QSPI mode fault

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

