Motion Control System

Software User Manual

Revision 16.1

20 February 2003

PAN CONTROLS LIMITED

DRUMMORE
DOUNE
PERTHSHIRE
FK16 6AX
SCOTLAND

Telephone +44 (0)1786 850261
Fax +44 (0)1786 850387
email: sales@pancontrols.com
internet: http://www.pancontrols.com

Table of Contents

1 DOCUMENT MANAGEMENT

1.1 Release (major issue changes)

Issue	Date	Comments	Software
1	June, 1994	Early release	
2	August, 1994	Stepper control	
3	September, 1994	Multi-axis moves	
4	October, 1994	Analogue i/p	V 1.9
5	November, 1994	Scaled channel linking	V 1.18
6	January, 1995	Rev D Host control board	V 1.24
7	February, 1995	Separate Hardware manual	V 1.25
8	June, 1995	Flexible hardware configuration (HW)	V 1.30
9	June, 1996	Rev E of PC3/100; data logging	V 1.42
10	July, 1996	Hardware Handshake for terminal port	V 1.44
11	November, 1997	Revision F. host control hardware	V 152
12	December,1998	Position gain maps. Event sequences	V 153.8
13	May, 1999	4Mb Flash, second serial port	V 154.0
14	February, 2001	Printer & Smart card options	V 156.1
15	March, 2002	Expanded memory options	V 157.5
16	January, 2003	Numerous additional commands	V 157.8

1.2 Critical changes to software

The purpose of this section is to highlight changes to software revisions, where there could be compatibility problems with previous versions.

1.2.1 Version 152.2 (November 1997)

Bit 14 of the DW Display Options word (Hardware handshaking option), has now been transferred to bit 4 of the GW Global Control Word. In addition ,it's sense has been changed, so that there is now hardware handshaking with bit 4 cleared. Parity and checksum checking have been added to the global control word, and for Revision F of the PC3/100 boards, the default is for parity, checksum checking, and hardware handshaking. Revision E have these bits clear by default. For more information see the description of the global control word (page 26).

1.2.2 Version 154.0 (May 1999)

The RM (reference mode) command has been removed. The same functions are now carried out by the EI (Enable input) and MI (Mask input) commands, when the inputs referred to have been defined for reference activities.

1.2.3 Version 154.4 (August 1999)

The EO command has been changed from echo mode off to Enter Offset table. The EM (Echo Mode command) has been changed to accept a parameter 1 or 0. This defines the state of the echo mode.

1.2.4 Version 154.5 (August 1999)

The RW command has a new default value of 0000 0001 0000 0000 (positive numbers only). Previously bit 8 had a default value of zero. If bit 8 is required to be zero, it has to be explicitly set to zero.

1.2.5 Version 154.9 (October 1999)

The SK, EK and MK commands have had keys re-allocated to them. This means it is possible that a programme written for an earlier version of software will need to be changed. For example EKK now enables the right hand display key; previously it enabled the escape key.

1.2.6 Version 155.5 (March 2000)

The IW (inch wait) been re-name IP (inch pause). This is to allow a new IW (interpolation word) command. The SU(Set Units) command has been removed. The variable system allows scaling to be performed more flexibly.

1.2.7 Version 155.9 (September 2000)

Bit 2 of GW (global control word) now controls the cursor on the operator interface display. Its previous use for defining whether there is an external operator interface is redundant.

1.2.8 Version 157.0 (June 2001)

The hardware control word (HW) has been re-organised to accommodate 2 slave boards.

1.2.9 Version 157.2 (October 2001)

The position trigger output command now uses the ":" character as a delimiter instead of the "/" character. This is to allow the PO command to be used within sequences. The PO command now allows variables to be used.

CHV (select motor channel variable) has now been changed to CHB. This is to allow CHV to be used for setting the channel number from a variable.

1.2.10 Version 157.3 (October 2001)

EB and LB commands for the inter-channel bounds system has been removed.

1.2.11 Version 157.8 (October 2002)

The EM command has been changed from echo mode to execute event sequence after motion complete. The EM (Echo Mode command) has been re-named EX.

The FN command has been modified to only allow positive step values between 1 and 127. This will be internally converted to a negative number if the start value is greater than the finish value.

1.2.12 Version 157.9 (February 2003)

The EX and BE commands have been replaced with the extended OW command.

1.3 Key additions to software

The purpose of this section is to highlight additions to the language in software revisions.

1.3.1 Version 156.5
Added DIL, RIL, MID, EID input latching commands

1.3.2 Version 157.0
Allowed up to 2047 sequences

1.3.3 Version 157.1
Added CA camera commands

1.3.4 Version 157.2
Fixed XT command

1.3.5 Version 157.3
Added EJ, LJ, RY, IY, CY, SY commands
Addedd FC, FT, UC commands
Changed PO to work in sequences n.b. delimeter changed to : from /
Fixed bug to allow neg numbers with ME
Fixed bug in TG. Added camera timeout to error messages
Fixed bug in SPV - used to overwrite sequences in RAM
Allowed XL to operate with very small moves in one axis

1.3.6 Version 157.4
Added PT, SG, WP commands
Added SPP command
Fixed bug in DIx for resetting DI line from within sequence

1.3.7 Version 157.5
Allowed 128K of programme space
Fixed bug when ST used with v large SA and v small SV
Fixed bug with multi-axis motor position error (interpreter crashed previously)

1.3.8 Version 157.6
Allowed averaging for analogue inputs

1.3.9 Version 157.7
Added NV, ZW, PB, RM, RR, OS, TA commands

1.3.10 Version 157.8
Added RQ commands
Fixed bug in FN for-next loop with variables
Allowed 8 and 16 bit word parameters to output to variable
EM re-named EX
Added EM Execute event sequence after motion
Added ZM Zero block of memory
Added XF continuous variable offset command
Improved IRQ masking for FO flashing output (ref Jode bug)
Added VBD option to display bit pattern of variable
Added VZ multiply and divide function (allowing 64-bit intermediate result)
Allowed background channel control (e.g. “>2") for large number of commands.

1.3.11 Version 157.9
Added GP & GC cartesian & polar transforms
Extended OW to allow parity etc for all serial ports
Extended BD to allow control of baud rates for all serial ports
Removed EX echo mode (replaced by extended OW)
Removed BE echo mode for secondary port (replaced by extended OW)
Allowed optioning rounding for VR square root function

2 INTRODUCTION

This document describes the Pan Controls motion control system.

The system controls servo motors with position feedback, or stepper motors with or without position feedback. The system is supplied on circuit boards which can be installed in a 6U height 19" sub-rack system. It may be set up with a standard computer terminal. It includes comprehensive software to allow the user to control the motors using simple high level commands.

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 carry out training to experiment to familiarise themselves with the facilities available. This is best done on an off-line test machine which is not directly linked into a production unit.

3 SYSTEM OVERVIEW

The control systems are based on microprocessors running on Euromodule sized circuit boards, using established hardware and software technology. The system is designed to be able to provide an integrated solution to a wide range of motion control situations. In particular, Stepper motors (open or closed loop) and position feedback based servo drives can be mixed at will.

3.1 Circuit boards

There are several different circuit boards which have been developed together with their associated software. These consist of:

3.1.1 Host control board (PC3/100, Revision F). This provides feedback position control for up to 2 axes. It is based on the Motorola MC68HC16Z1 microprocessor. Its functions include:

- (i) Up to 4 channels D to A output (12 bits resolution). All channels are isolated against high voltage. Bipolar -10/+10v output.
- (ii) 8 channels A to D input (10 bits resolution). These channels are not isolated, and must be in the range 0-5v.
- (iii) 2 channels Quadrature & ref input. Isolated RS-422 inputs (differential signals).
- (iv) 16 channels of general purpose parallel inputs (4 of which may be used for reference inputs). Isolated inputs. Default to 24v DC, with selectable alternative voltages. 8 of these inputs can be re-defined as outputs.
- (v) 16 channels of general purpose parallel outputs. Isolated output. Output voltage supplied externally. 8 of these outputs can be re-defined as inputs.
- (vi) 4 asynchronous serial ports. TTL signals to daughter board. One of these is also taken to an isolated RS-485/422 circuit for use on a proprietary bus, or general purpose RS-422.
- (vii) Real time clock.
- (viii) Hardware watchdog.
- (ix) 1 dedicated Asynchronous RS-232 serial port (set-up terminal)
- (x) Memory sockets to allow a mix of SRAM and Flash memory. Also serial EEPROM and NOVRAM.
- (xi) Programmable logic to be used for i/o line logic and address & chip select decoding.
- (xii) Synchronous & asynchronous extended G-64 bus interface (with option for 1 or 2 MHz operation). Allows synchronous serial/8-bit data bus/chip select connections to slave control board
- (xiii) CAN bus.
- (xiv) Scanning keypad interface.
- (xv) LCD/VFD character display interface.
- (xvi) Two channels SSI serial absolute position feedback interface.
- (xvii) Two channels remote resolver absolute position feedback interface.
- (xviii) Smart card facility for parameter storage.
- (xix) Sites for 2 single Euromodule sized daughter boards.
- (xx) Site for one front edge daughter board.

3.1.2 Slave control board (PC3/110, Revisions A-B). This provides feedback position control for up to 2 axes. It is a slave to the central control board (and uses the central control board's microprocessor). Its functions include:

- (i) 4 channels D to A output (12 bits resolution). All channels are be isolated against high voltage. Bipolar (-10/+10v or unipolar (0/+10v) selectable.
- (ii) 2 channels Quadrature & ref input. Isolated inputs, with differential signals from 0 to 24 volts.

(iii) 8 channels of general purpose parallel inputs (8 of which may be used for reference inputs). Isolated inputs. Default to 24v DC, with selectable alternative voltages.
(iv) 8 channels of general purpose parallel outputs. Isolated output. Output voltage supplied externally.
(v) 4 asynchronous serial ports. TTL signals to daughter board.
(vi) Synchronous serial/8-bit data bus/chip select connections to host control board

3.1.3 Slave control board (PC3/110, Revision C). This provides feedback position control for up to 3 closed loop servo axes, and up to 4 stepper axes. It is a slave to the central control board (and uses the central control board's microprocessor). Its functions include:
(i) 4 channels D to A output (12 bits resolution). All channels are be isolated against high voltage. Bipolar (-10/+10v or unipolar (0/+10v) selectable.
(ii) Hardware watchdog.
(iii) 3 channels Quadrature & ref input. Isolated inputs, with differential signals from 0 to 24 volts.
(iv) Up to 32 channels of general purpose parallel inputs (4 of which may be used for reference inputs). Isolated inputs. Default to 24v DC, with selectable alternative voltages.
(v) Up to 16 channels of general purpose parallel outputs. Isolated output. Output voltage supplied externally.
(vi) Synchronous serial/8-bit data bus/chip select connections to host control board

3.1.4 Operator interface board or single axis control board (PC3/120). This is a microprocessor driven circuit board, based on a Motorola 68HC11 microprocessor. It communicates with the host control board by an asynchronous serial port. Its functions include:
(i) Up to 2 channels D to A output (12 bits resolution). Bipolar -10/+10v output.
(ii) 8 channels A to D input (8 bits resolution). These channels are not isolated, and must be in the range 0-5v.
(iii) 2 channels Quadrature & ref input. RS-422 inputs (differential signals).
(iv) 4 channels of general purpose parallel inputs (2 of which may be used for reference inputs). 5v un-isolated inputs.
(v) 4 channels of general purpose parallel outputs. 5v un-isolated outputs.
(vi) 2 asynchronous serial ports. One of these is also taken to an isolated RS-485/422 circuit for use on a proprietary bus, or general purpose RS-422.
(vii) Hardware watchdog.
(viii) Scanning keypad interface.
(ix) LCD/VFD character display interface.
(x) EEPROM facility for parameter storage.
(xi) Site for daughter board.

3.1.5 Serial driver/display board (PC3/130). This is a daughter board to the host/slave control board, or the operator interface board. It provides up to 4 isolated RS-422 duplex channels and up to 4 seven segment L.E.D. driver units (each with up to 8 digits).

3.1.6 Auxiliary control board (PC3/135). This is a daughter board to the host/slave control board, Its functions include:
(i) 4 channels D to A output (12 bits resolution). All channels are isolated against high voltage. Bipolar (-10/+10v or unipolar (0/+10v) selectable.
(ii) 2 channels Quadrature & ref input. Isolated RS-422 inputs (differential signals).

(iii) 16 channels of general purpose parallel inputs (4 of which may be used for reference inputs). Isolated inputs. Default to 24v DC, with selectable alternative voltages. 8 of these inputs can be re-defined as outputs.
(iv) 16 channels of general purpose parallel outputs. Isolated output. Output voltage supplied externally. 8 of these outputs can be re-defined as inputs.
(v) Hardware watchdog.
(vi) Programmable logic to be used for i/o line logic and address & chip select decoding.
(vii) Two channels SSI serial absolute position feedback interface.
(viii) Two channels remote resolver absolute position feedback interface.

3.1.7 Analogue input board (PC3/137). This is a daughter board to the PC3/153 backplane board. Its functions include:
(i) 8 channels A to D input (10 bits resolution). All channels are isolated against high voltage. Bipolar (-10/+10v or unipolar (0/+10v) selectable.

3.1.8 L.E.D. display board (PC3/140). This consists of 5 seven segment L.E.D. displays (14.2mm). It is connected to the serial driver/display board by means of a ribbon cable.

3.1.9 L.E.D. display board (PC3/141). This consists of 8 seven segment L.E.D. displays (14.2mm). It is connected to the serial driver/display board by means of a ribbon cable.

3.1.10 Counter/timer board (TIO-33). This is used for controlling open loop stepper drives. It is connected by means of a DIN 41612 connector at the rear of the board.

3.1.11 G-64 bus, 2 slot (PC3/152). This is used for a connecting host control (PC3/100) to a slave control (PC3/110) board.

3.1.12 G-64 bus & power supply, 3 slot (PC3/153). This is used for a connecting host control (PC3/100), Revision F and later, to a slave control (PC3/110) board. Its functions include:

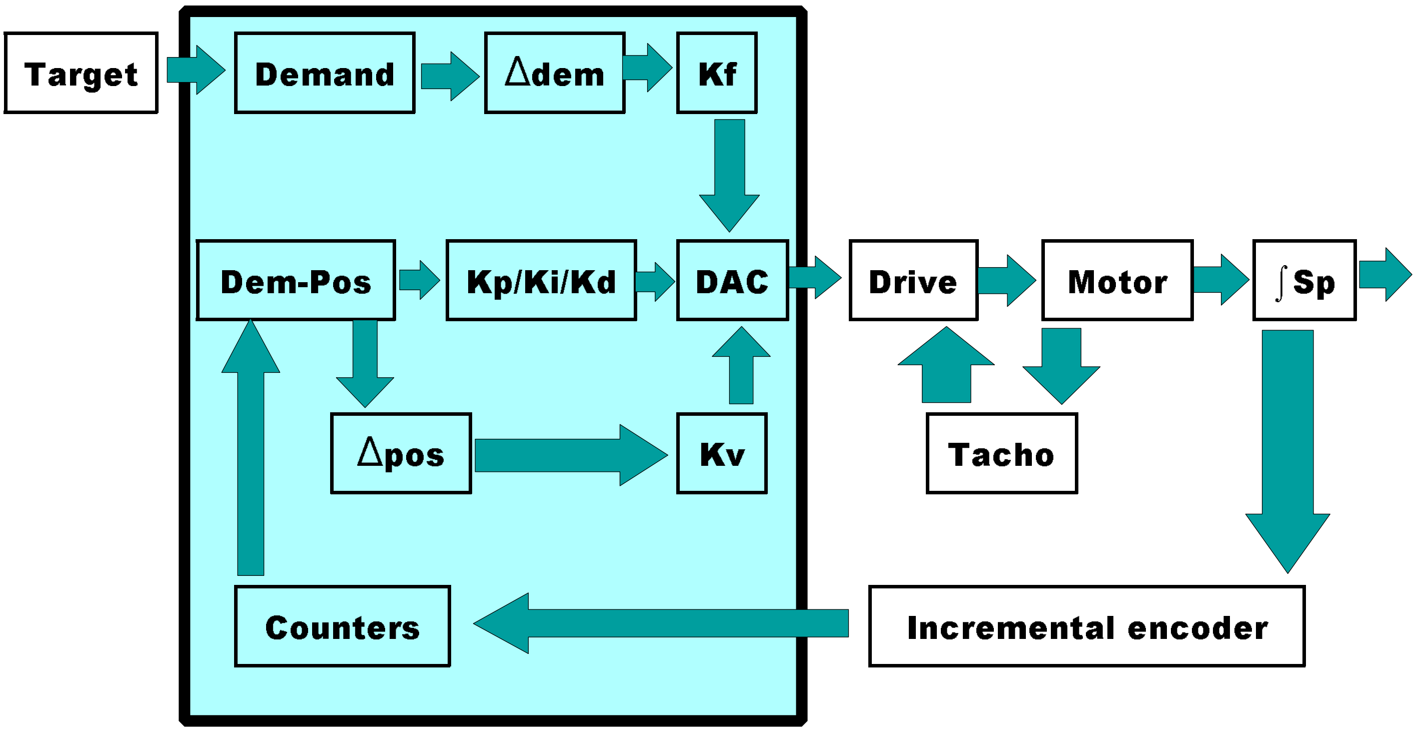
(i) +5v, +15v, -15v power supply for PC3/100 & PC3/110 control boards.
(ii) +5v isolated power supply for incremental encoders.
(iii) +7v un-regulated power supply for operator interface.
(iv) Isolated RS485-422 port for connection to Operator Interface or serial bus.
(v) Isolated CAN bus port.
(vi) Infra -Red or RS-232 asynchronous serial port.
(vii) 2 asynchronous serial ports which can be configured as RS-422 or RS-232.
(viii) Socket for isolated analogue input board (8 channel).
(ix) 16 channels of isolated stepper circuits.
(x) Standard LCD/VFD character display port.
(xi) Keypad interface port.

3.1.13 G-64 bus & power supply, 2 slot (PC3/154). This is used for a connecting host control (PC3/100), Revision F and later, to a slave control (PC3/110) board. Its functions include:

(i) +5v, +15v, -15v power supply for PC3/100 & PC3/110 control boards.
(ii) +5v isolated power supply for incremental encoders.
(iii) +7v un-regulated power supply for operator interface.
(iv) Isolated RS485-422 port for connection to Operator Interface or serial bus.
(v) Isolated CAN bus port.
(vi) RS-232 asynchronous serial port.

(vii) 2 asynchronous serial ports which can be configured as RS-422 or RS-232.
(viii) Socket for isolated analogue input board (8 channel).
(ix) Standard LCD/VFD character display port.
(x) Keypad interface port.

4 GENERAL DESCRIPTION - CONTROLLER



Control board (PC3/100) - 1 axis

This section gives an overview of the operating principles and a brief description of the facilities of the digital motor control system.

The hardware resides on double height Euromodule circuit boards, connected by means of a G-64 computer bus. The host processor, and the motion control interface for two axes are taken from an PC3/100 circuit board. Three PC3/110 boards used in conjunction with a PC3/100 controls an additional six axes. The serial interface for the outstations is taken from a daughter board, and the diagnostic terminal interface comes from the PC3/100 processor board. The eight axes of positional control will be operated from one PC3/100-PC3/110 set of boards. The tension control will be operated from a single PC3/100 board.

The control system works on the basis of continually sampling information and performing a control algorithm at a defined timer interval (known as the sampling time). This sampling time is set to be 1÷256th second. In other words, the time interval between updating calculations is about 3.9ms. The control software uses two key pieces of information to generate a positional error. These are the current demanded position, and the current measured position. The demanded position is calculated by the controller on the basis of a target positional move. For example, if the controller is asked to move a distance of 4000 encoder counts at a speed of 500 counts/sec and an acceleration of 2000 counts/sec^2, it can generate a velocity profile in terms of the desired position at every 1÷256th second. If the system is under control but not moving, there will still be a demand position, but it will not change.

This positional error information is used as the basis for a *PID* (proportional, integral, and derivative) calculation, whose output is fed to a Digital to Analogue Converter (DAC). The analogue voltage is sent to an analogue drive as a velocity command.

Most high performance drive systems will have an analogue velocity control loop built in. This takes the form of a Tacho-generator on the end of the motor shaft, which generates a voltage proportional to its velocity, and an associated gain potentiometer on the drive. This allows a stable, high gain system to be set-up.

There are two additional control terms built in to the software, which can be useful under particular circumstances. The first of these is a digital velocity feedback system. This is only really useful where no tacho feedback is available, and is bound to be of a lower performance that an analogue system due to the limitations of digital sampling. The second is called velocity feed-forward, and is proportional to the rate of change (derivative) of the internally generated demand position signal. This feature is particularly useful for helping a mechanical system to anticipate changes in velocity.

The system is intended for use with digital incremental position encoders which provide two signals in quadrature. The encoder interface multiplies the resolution of the encoder by four, such that each complete cycle of the encoder signals represents four counts.

The encoder signals are decoded and counted by hardware on the PC3/1x0 board. The software converts these to numbers which represent the measured position. This signal is then used to compare with the demanded position information, as described above.

The system is set up by high level commands from a serial link. Most commands are two letters, sometimes followed by a numerical parameter. These commands can be built up into programmes which can then be stored on non-volatile memory (FLASH) on the system. The motors may be controlled using simple proportional control, where the demand signal depends on only the position error. The proportional gain constant is set by the user. It is also possible for the user to set gain constants for integral feedback, differential feedback, velocity feedback, and velocity feed-forward terms, providing very flexible control over the system transfer function.

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

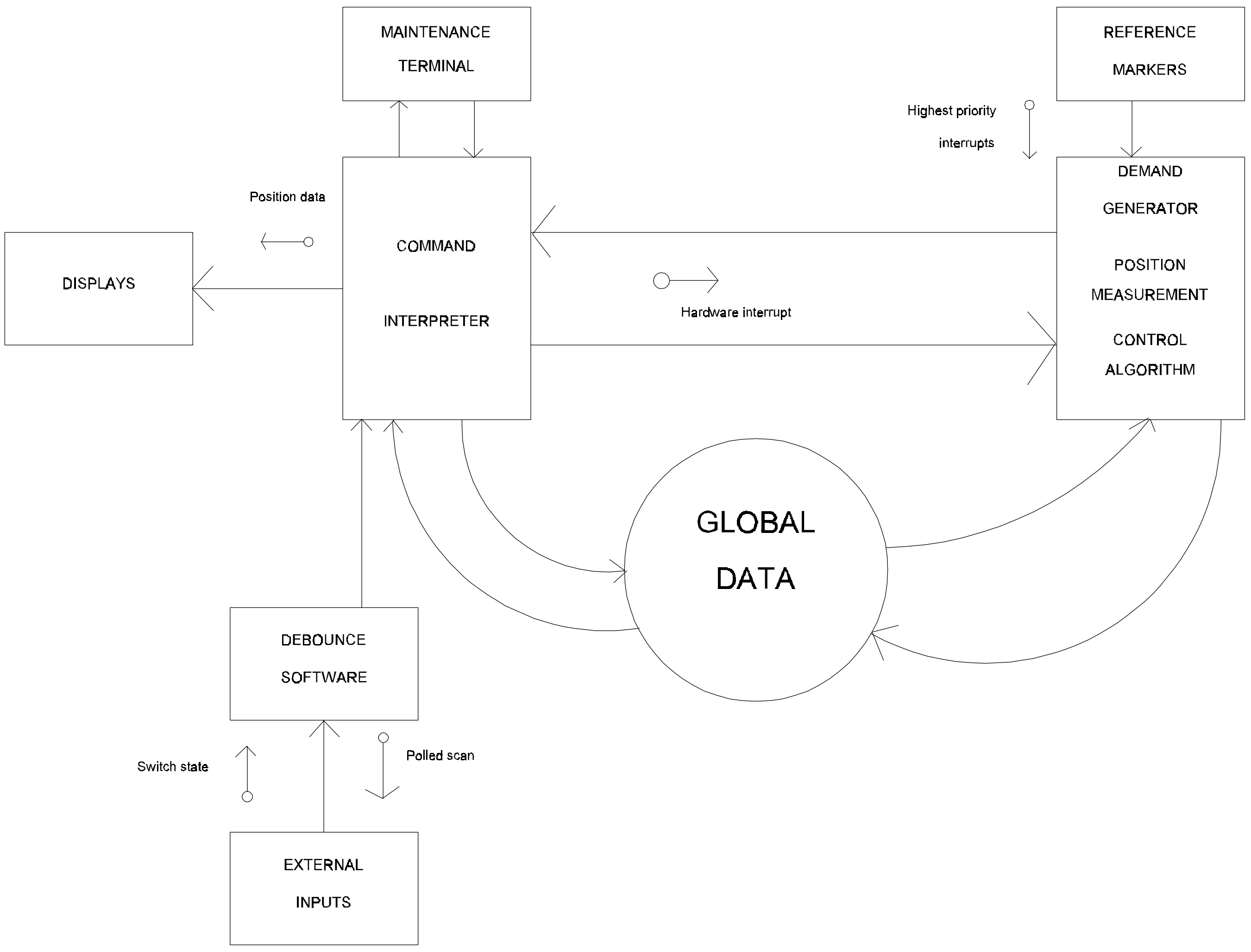
The motors may be controlled at a constant velocity instead of controlling the motor position. In velocity control mode, the system accelerates the motor until it reaches the specified system velocity, and then maintains that velocity. The motor may be stopped with the normal deceleration, or may be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

The system has up to 32 digital input and 32 output lines, 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 set and cleared, and can be used to control external relays or valves, or just be used for status indication. They may also be used to allow the system to be controlled from an industrial programmable logic controller (PLC).

The facilities allow the user to define move profiles other than a trapezoidal or triangular profile, in order to follow a specific motion, or to mimic some mechanical system. Sequences of commands may also be defined by the user. In addition, the profiles may be operated simultaneously on axis 1 and 2, allowing complex axis movement profiles.

The use of variables allows a fixed programme to operate in a flexible manner.

5 GENERAL DESCRIPTION - SOFTWARE FUNCTIONALITY



Data flow diagram

The control software is highly deterministic, being interrupt driven by a hardware timer. The control algorithm is implemented every time this hardware timer generates an interrupt (every 1÷256 second). In addition, polling of the external inputs is carried out during the same servo loop closure process. Serial character receipt and transmission is also handled by the interrupt mechanism, and a special high priority interrupt is used for reference marker detection. This gives a response time of better than 15µs.

The command interpreter is executed in background mode, together with display routines.

The modular approach to the software means that system can be easily adapted to particular situations.

6 MODIFICATIONS TO THE USER PROGRAMME

The user programme consists of sequences of instructions which are stored in non-volatile memory on the Pan controller.

To make the management of the construction of the user programme simple, a system has been developed to allow the user to develop and modify a master copy of the user programme on a personal computer.

The sequence of programme development can be summarised as follows:

1. Develop programme on personal computer using any editor which will generate an ASCII file (i.e. no embedded control codes).
2. Download programme from personal computer to Pan controller (to the controller's RAM - volatile memory).
3. Save the programme from the controller's RAM to FLASH (non-volatile memory).

This procedure means that a master copy of the programme is always maintained on a personal computer, and can be copied and maintained for archiving purposes.

A utility programme (called "PANTERM") is supplied to run on an MS-DOS based personal computer. This enables the personal computer to emulate a terminal, and to allow file transfer. The personal computer must have a serial port ("COM1", "COM2", "COM3", or "COM4").

It is suggested that a batch file be set up to enable the user to enter the programme by entering a single name (e.g. control). This will start the "PANTERM" programme, and prompt the user to press a key to start terminal emulation.

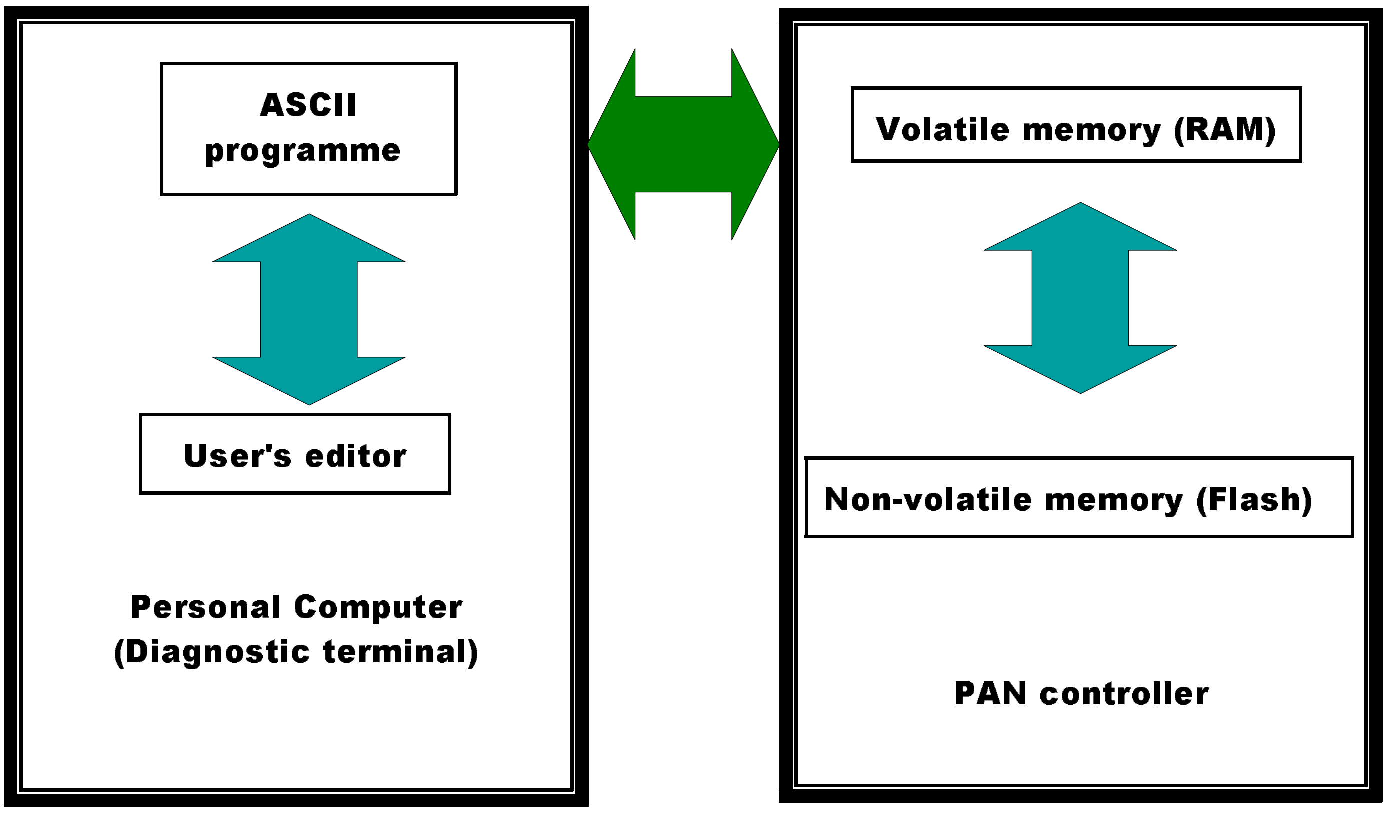
If the Pan controller is connected to the personal computer and switched on, then any keys which are pressed on the personal computer will be echoed back to the screen. If characters are not echoed, then there is something wrong with communications. The leads should be checked. Also, the "PANTERM" configuration can be modified by pressing <ALT> C. A sample screen might appear as follows:

```
                    Configuration
Communications Port (1 - 4):                     1
Baud Rate:                                       9600
Serial mode; RS232, 422 or 485:                  232
File loading delay (milliseconds):               0
Communication mode: Transparent or Checksum:     C
Software or Hardware handshaking:                H
Colour or Mono display:                          C
Parity: None or Even:                            E
Editor name:                                     ed
```

In order to modify a source programme, it is necessary to access the editor by pressing <ALT> E. The user is then prompted to enter a file name (e.g. "ICI6.1"). The editor then enables the user to move around the programme, using the cursor keys and the <Page Down> & <Page Up> keys.

Modifications can be made until the user is satisfied. The user can then leave the editor by pressing <esc> followed by E (to Exit and save), followed again by <esc>.

In order to download a source programme from the personal computer to the Pan controller, it is first necessary to ensure that the controller is in the "privileged mode" (see command "PM"). It is suggested that the user resets the controller to its default state by using the "RS" command, particularly when doing a final installation. This will enable a definite checksum to be established on a particular machine. This means that any subsequent changes (accidental or intentional) can be identified. If the file loading mode is set to C (checksum), The programme is downloaded by pressing <ALT> U. The user is then prompted to enter a file name (e.g. "ICI6.1"), and the programme will then be transferred. If the programme uses stored profiles, these will need to be downloaded in the same way (e.g. ICI6.PRF).


ASCII programme
User's editor
Personal Computer (Diagnostic terminal)
Volatile memory (RAM)
Non-volatile memory (Flash)
PAN controller

PANTERM programme maintenance environment

7 COMMAND DESCRIPTION

7.1 General

This section gives full details of all the system commands and syntax. Numeric parameters are denoted by "$_{nn}$". Parameters entered as a binary string ("0"s and "1"s) are denoted by "$_{bb}$". Note that individual bits of a binary number can be replaced with either an "X" (no change to existing value) or a "T" (toggle bit). All commands are terminated by a carriage return (CR) or by a carriage return and line feed (LF). The system responses are all followed by (CR)(LF).

Numeric parameters are input and output in either decimal or hexadecimal. Commands are available to set the system to use one or the other. Decimal numbers are output by the system as signed seven digit numbers. Hexadecimal numbers are output in 24 bit two's complement format as six hex digits with no sign. Leading zeros are not suppressed in numeric outputs. 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. Any number may be entered as a variable (a to z) by suffixing the command with a "V" followed by the variable. In addition, commands which output numeric data may be re-directed to a variable by suffixing the command with an "O" followed by the variable.

The normal character set consists of the letters A-Z and a-z, the numbers 0-9, "+", "-", and space. Multiple command sequences may be entered as one command line, with the individual commands separated by a delimiter character. Any printing character not in the normal character set (except for "&" or ":") may be used as a delimiter between commands, such as the "*", ".", and "/" characters. The maximum input line length is 255 characters. Backspace or delete may be used to remove characters from the current input line. Other non-printing characters are simply echoed, and have no effect.

The escape character (Hex 1B) will exit from any command which is producing a long list of many pages. The tilde (~) character (Hex 7E) will cause the system to halt any current activity and perform a soft reset.

The commands allow very flexible control of the system. They fall broadly into the following categories.

(a) Boot programme commands.
Commands to operate on the boot programme, to allow maintenance of the main programme.

(b) Miscellaneous.
Commands to change between channels, and to handle the stored setup data.

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

(d) Move commands.
These include commands to move to absolute and relative positions, to find the zero reference position, to move at a constant velocity, and to stop the move either normally or abruptly.

(e) Set parameter commands.
These commands set up a wide range of system parameters, including the velocity and acceleration of the normal moves, and setting up the creep and deadband facilities.

(f) Sequence commands.
These include commands to enter, list, and execute complex command sequences.

(g) Profile commands.
These commands are used for the profile move facilities ("Software Cam"). Profiles can be executed simultaneously on channels 1 and 2.

(h) Wait commands.
These commands may be used in command sequences to wait until a condition is true before executing the next command in the sequence.

(i) Error trapping.
These commands set up the system error conditions.

(j) Gain commands.
These include commands to set up the constants used in the closed-loop control algorithm.

(k) Digital input and output commands.
These are commands to directly control the input and output lines.

(l) Reference commands.
These include commands to set up continuous position correction on the reference input signal.

(m) Configuration commands.
These commands allow the user to configure the digital input and output lines for various automatic functions.

(n) Display commands.
These include commands to display parameter values and status information via the serial port.

(o) Variable commands.
These commands allow the user to set up and read values by means of variables.

(p) Conditional commands.
These commands allow the system to test certain conditions, and to execute commands depending on whether or not those conditions are met.

(q) Remote communication network commands.
These commands allow the system to communicate bi-directionally with other devices and controllers on a serial network.

(r) Data logging commands.
These commands allow the system to perform high speed data logging of a number of variables, for later analysis.

The command reference section (7.2) 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. Note that the range and default values are given in encoder counts, and if a scale factor 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 all the input and output lines are listed with the LI command.

Many commands that affect the behaviour of the system are *restricted*, or *privileged*, and can be used only in privileged mode after entering a password. This allows the system to be configured as required by the Control Engineer or Systems Engineer, while preventing access to the more fundamental setup parameters by the machine operator. 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 non-volatile 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 manufactured default configuration.

7.2 Command Reference

7.2.1 Boot programme commands

The boot programme is designed so that it is normally invisible to the operator. It consists of a small area of code which can operate autonomously, and allow upgrades to the main programme by the use of the PANTERM programme on a personal computer. This system is installed on hardware revisions E and later of the PC3/100 board. The boot programme is the first piece of code to be executed by the microcontroller, and its first task is to check that there is a valid main programme installed on the board. If the main programme is installed and valid, it is executed immediately; otherwise the controller will enter the boot programme, and await instructions from the terminal port. If no terminal is connected, the two L.E.D.'s on the top edge of the board will indicate the current state of the board. If the top L.E.D. is not flashing, then there is a hardware fault. If the second L.E.D. is illuminated, then the boot programme is running; otherwise the main code is operating. The boot programme can be entered from the main programme by issuing the "XB" execute boot command.

BD$_n$ Set baud rate.
Range : 0 to 11
Default : 0

Sets a baud rate for the terminal port. It is important to remember that having changed the baud rate on the controller, the terminal device will not communicate with the controller until its own baud rate has been correspondingly changed. The list below shows the available baud rates. It also indicates whether a particular baud rate is available on the standard serial port of a personal computer. This is useful when using a terminal emulation program such as PANTERM.

value	0	9600 baud (available on personal computer)
	1	14400 baud (available on personal computer)
	2	19200 baud (available on personal computer)
	3	28800 baud (available on personal computer)
	4	38400 baud (available on personal computer)
	5	57600 baud (available on personal computer)
	6	76800 baud
	7	115200 baud (available on personal computer)
	8	230400 baud
	9	7200 baud (available on personal computer)
	10	4800 baud (available on personal computer)
	11	2400 baud (available on personal computer)

BK Backup main programme (low level code).

This command allows the user to keep a copy of the currently installed main programme. It sends a complete copy of the code down the serial port in the form of Motorola S0, 2, and 8 records. This can be stored on the hard disk of a personal computer using the PANTERM programme. One might want to make a copy of the existing main programme before up-grading to a new version.

EN Erase non-volatile memory.

This command erases all the non-volatile memory areas with the exception of the boot code itself. It should be used with extreme caution, since the existing main programme **and** its associated stored parameters will both be erased.

WARNING: this command destroys the existing low-level code. It is important that the user is aware of the implications of using this command.

HE Print help display.

This command prints a complete list of all commands on the system, in alphabetical order, a screenful at a time. It pauses between each page until a character is received. Help on a single command is displayed if the command mnemonic followed by "?" is entered.

Example:

System	User	Comments
B>	UP?<CR>	Request help on UP
UP Upload new code version		Single line help

UP Upload new main programme (low level code).

This command allows the user to upgrade the currently installed main programme, in the form of Motorola S0, 2 and 8 records. A file must have been created in advance and placed in the relevant directory of a personal computer, together with the PANTERM programme. The command first erases the existing main programme, and as a result it should be used with extreme caution. It then reads in the S-record file, and does appropriate checksum checking as it does so. If there is any checksum failure during uploading, then no programme is loaded. If the programme is loaded successfully, it is saved, together with a checksum to verify the integrity of the complete code.

WARNING: this command destroys the existing low-level code. It is important that the user is aware of the implications of using this command.

XF Execute final running code.

This command allows the user to execute the main programme from the boot programme. It first does a checksum check on the integrity of the code, and if it passes, it then executes it. This operation would happen by default when the unit is switched on or reset if there is a valid main programme installed.

VN Display version number.

This command prints information about the version of software fitted to the system. It gives the version number of the firmware, its revision date, and some configuration information.

7.2.2 Miscellaneous commands

CH$_n$ Select motor channel $_n$.
Range : 1 to n (where n=Max no of channels)
Default : 1

This command allows the user to switch between any of the motor channels. If in position control mode, a channel which is not selected remains under servo control with proportional and integral control active, to maintain its current position. The total number of available motor channels is determined by HW, the hardware setup word (see section 7.2.14, page 107).

CHB Select motor channel variable.
Range : 0 to n (where n=Max no of channels)
Default : 0

This command sets channel variable. This allows any channel related command to operate on the channel defined by the channel variable, using the "^" suffix. This happens regardless of the setting of the CH command (without the B).

Note: Channel related data can also be displayed and changed by using the “>” operator to explicitly describe which channel is being referred to. This can be particularly useful where sequences can be initiated asynchronously.

Example : FM>3:Vh
This sets the link multiplier for channel 3 to a value determined by variable h.

Example : CHB3/FM^Vh
This sets the link multiplier for channel 3 to a value determined by variable h. It performs exactly the same operation as the previous example.

AP±$_{nn}$ Add channel dependent parameter.
Range : -2,147,483,647 to +2,147,483,647

Allows a channel dependent variable to be set up.

Example : AP>2:300/AP>1:150/FM<
This sets up FM on channel 1 to be 150, and on channel 2 to 300.

Example : IVC345/IVD440/AP>2:VC/AP>1:VD/FM<
This uses variables to set up FM on channel 1 to be 440, and on channel 2 to 345.

BV$_n$ Select slave board version numbers.
Range : 0 to 255
Default : 1

This command allows the user to set up the software for different slave board revisions, when it is not possible for the software to automatically identify revision numbers. A parameter of 0 indicates Slave board revision A, and 1 indicates revision B.

VN Display version number.

This command prints information about the version of software fitted to the system. It gives the version number of the firmware, its revision date, and some configuration information.

SP(V/S/P) Save parameters. (restricted)

This command saves all the programmable parameters in non-volatile memory. There may be a short delay while the save operation takes place. The saved parameters become the new defaults, used by the system on power-up. The SP command also saves any profiles sequences, and variables A-Z. At the end of the save operation, the system calculates a cyclic redundancy check byte (CRC) on the saved data, which is then saved in non-volatile memory as well. This allows the saved data to be verified at any time by comparing the stored CRC byte with a calculated one. If the saved data has changed at all, the stored CRC will not be the same as the calculated CRC. If the save operation fails for any reason, then an "F" error message is returned. In this case, please contact your sales office. This command is restricted, and is only available in privileged mode.

The optional V suffix allows the numeric variables (%) to be saved. This works independently of the SP command without a suffix, and allows selective saving of variables during a programme.

The optional S suffix allows the sequences alone to be saved. This works independently of the SP command without a suffix, and allows selective saving of sequences.

The optional P suffix allows the parameters alone to be saved. This works independently of the SP command without a suffix, and allows selective saving of parameters. This includes items such as system gains, upper case variables, etc.

NOTE(1): The system can be set back to its default state regardless of the saved parameters by putting link 29 on the PC3/100 control board before switching on.

NOTE(2): If sequences are being saved, the system will be locked for approximately a second, whilst the memory map is adjusted. It is important that motion is not taking place when this operation is carried out. This happens if SPS or SP with no suffix is used.

CS Checksum test.

This command is used to verify the data stored in the non-volatile memory. The system calculates a new CRC value for the stored data, and displays it. It then compares the new value with the CRC value that was stored with the data when it was saved. If the values are different, an "F" fail error message is displayed. If the CRC test fails, it indicates that the stored data has changed since it was saved. If this occurs, please contact your supplier.

RD Reload stored data. (restricted)

This command reloads all the parameters, input and output line definitions, sequences and profiles from the stored setup in the non-volatile memory. If the stored data checksum is not correct, then this command returns the 'F' failed error message, and the stored data values are not loaded.

RS(F/V/S) Reset to default setup. (restricted)

This command resets all the parameters, variables except numeric type, input and output line definitions, sequences and profiles to their default settings. If the optional "F" (Full reset) parameter is omitted, then the HW hardware setup word (page 107), DW display word (page 119, 120, 121, 157), and GW global control word (page 26) values are retained at the settings which they were at before RS was entered. If the optional "V" parameter is used then the numeric variables (% type) are all reset to zero. If the optional "S" parameter is used then the sequences are all deleted.

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

HR_{nn} Set Hard Reset sequence or initiate Hard Reset.
Range : 0 to 255
Default : 0

This command locks the software watchdog in order to force a system reset (after about 8 seconds). If a parameter is entered, an auto-start sequence for startup after software watchdog reset can be entered. This could be the same as the AS auto-start sequence, but could be some alternative sequence to ensure a safe re-start.

LA List all parameters.

This command lists all the parameters(with the exception of the HW hardware setup word and DW display word), input and output line definitions, sequences and profiles to the serial port in a suitable format for entering parameters etc. at a later date. If the system is connected to an MS-DOS based personal computer running the PANTERM programme, the parameters can be recorded on disk for backup purposes and loaded into another control system to duplicate parameter setting from one machine. Note that bits 8 and 9 of the GW (page 26) global control word can be used to select whether the LA command outputs I/O commands and sequences/profiles.

LC Limit for cosine interpolation error.
Range : 0 to 255
Default : 0

This command sets a limit in ADC units for the error in cosine calculation. The sequence of the calculation is that analogue channel 1 is sampled first. Two angular positions are calculated, and two target voltages are calculated and compared with the measured voltage on analogue channel 2. If the measured value falls outside of a band defined by the LC parameter, then no correction is done to the motor position. If it falls inside one of the bands, then that defines which of the 2 angular positions are to be used to set the motor position.

UP Upload new main programme (user level code).

This command allows the user to upgrade the currently installed user level programme, in the form of an ASCII file. The file must have been created in advance and placed in the relevant directory of a personal computer, together with the PANTERM programme. The command first erases the existing user level programme, and as a result it should be used with caution. It then reads in the file, and does appropriate checksum checking as it does so. If there is any checksum failure during uploading, then an error message is displayed. If the programme is loaded successfully, it can be saved to non-volatile memory by using the SP command.

NOTE: this command does not destroy non-volatile code or data. The SP command must be used to transfer the programme to non-volatile memory.

BR(D/P) Binary record of user level programme and data.

This command allows the user to record the binary data for a user programme and data. BRP outputs the currently loaded programme and parameters. The can be saved to a file using the <Alt>R record facility on PANTERM. Likewise BRD outputs the numeric variables. The files generated from this programme are in the form of Motorola S-records.

NOTE: the file collected with this command is only useable with the version of low level code under which it was recorded.

BU Binary upload of a file.

This command allows the user to upload binary data for a user programme and data from a personal computer to the controller. The <Alt>U facility must be used to transfer a file.

NOTE: the file transferred with this command is only useable with the version of low level code under which it was recorded.

BH Breakdown of current Hardware.

This command displays the hardware which the software has recognised when the system has been switched on. This information is saved, together with the stored parameters. If the hardware found does not match the stored parameters, then the system will be reset, and a warning will be sent to the screen. This command can be used to output a 16-bit binary code to a variable, using the O suffix.

Bit	
1	Quad Universal Asynchronous Receiver/Transmitter (QUART)
2	Versatile Interface Adaptor (VIA)
3	Programmable Timer Module (PTM) no 1
4	Programmable Timer Module (PTM) no 2
5	Digital to Analogue Converter (DAC) no 1
6	Digital to Analogue Converter (DAC) no 2
7	Controller Area Network (CAN) bus
8	Real Time Clock (RTC)
9	Non Volatile Random Access Random Access Memory (NOVRAM)
10	Electrically Erasable Programmable Read Only Memory (EEPROM)
11	Operator Interface
12	Keypad Interface
13	Display unit
14	Not used
15	Not used
16	Not used

BK Backup main programme (low level code).

This command allows the user to keep a copy of the currently installed main programme. It sends a complete copy of the code down the serial port in the form of Motorola S0, 2, and 8 records. This can be stored on the hard disk of a personal computer using the PANTERM programme. One might want to make a copy of the existing main programme before up-grading to a new version.

PD_n Set factor for position feedback division.
Range : 0 to 8
Default : 0

This command sets the division factor for position feedback. This can be used when the resolution of the encoder is higher than is required. The actual position data is divided by 2^n. The largest division factor is 256.

Example : PD 2
This sets the position feedback division factor to $2^2 = 4$.

MP_n Set factor for position feedback multiplier
Range : 1 to 255
Default : 1

This command sets the multiplication factor for position feedback. The actual position data is multiplied by n in conjunction with being divided by two to the power of the PD parameter. The largest multiplication factor is 255.

Example : MP 5
This sets the position feedback multiplication factor to 5.

EV"ccc" Enter a user software revision no string.

This stores a character string (enclosed by double quotes, maximum 16 characters). This can be useful for identifying the current setup, using the VN command.

XB Execute boot programme.

This command allows the user to execute the boot programme from the main programme. It then enables the user to upgrade a new main code version to the unit.

OT Output the time counter.

Prints the current value for the time counter variable.

TW_{nn} Wrap for time counter input.
Range : 1 to 2,147,483,647
Default : 2,147,483,647

This command sets the value at which the time counter wraps to zero.

TI	Initialise ADC offset. Under normal conditions, there may be some constant offset in the strain gauge amplifier which causes the tension to be displayed and controlled at a value different to the actual tension. The TI command sets the system up to allow for this (assumed constant) offset in all subsequent tension control operations. It must be used every time the system is powered on, when the system is in not in tension control mode, and there is no load on the strain gauge. The measured value will then be subtracted from all future tension measurements. If this command is used on a regular basis (e.g. whenever the system is switched on), it will have the effect of removing any drift in the analogue strain gauge amplifier which may have arisen with time.
CA...	Camera commands. The CA command acts as a prefix to send various commands down the auxiliary RS-232 serial port to a Keyence CV-501 series vision system.
CAI	Camera initialise. This initialises the serial port associated with the camera.
CAS	Camera stop. This shuts down the serial port associated with the camera.
CAO	Stop camera run mode, and enter camera programme mode.
CAR	Enter camera run mode.
CAPW$_v$	The camera programme number is changed to the number specified by the variable $_v$.
CAUW$_v$	The camera window number is changed to the number specified by the variable $_v$.
CAPR$_v$	The camera's current programme number is read and put in variable $_v$.
CAUR$_v$	The camera's current window number is read and put in variable $_v$.
CAT$_{v1}$:$_{v2}$	A trigger signal is sent to the camera. The Y and X co-ordinate data is placed in variables $_{v1}$ and $_{v2}$ respectively.
CAM$_{v1}$:$_{v2}$	A re-output signal is sent to the camera. The last Y and X co-ordinate data is placed in variables $_{v1}$ and $_{v2}$ respectively.
CAWT$_v$	The maximum wait time (timeout) for the camera is read from variable $_v$. (value in system ticks).
CADO	Debug mode on. Characters received from camera are echoed to terminal port.
CADF	Debug mode off.

GW$_{bbbb}$ Set global control word. (restricted)

Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).

Default : 0000 0000 0000 0000 (PC3/100, Revision F)
0000 0000 0011 1000 (PC3/100, Revision E)

This command allows the user to write a value into the global system control word. Note that the leading zeros may be omitted. The global control word allows various components of the system to be enabled and disabled, as required. The global control word bit functions are described below.

bit	Bit set	Bit cleared
0	Vacuum Fluorescent display brightness BR0 (see table below)	
1	Vacuum Fluorescent display brightness BR1 (see table below)	
2	Flashing cursor on display	No cursor on display
3	No parity for comms.	Even parity for comms
4	Software handshaking.	Hardware handshaking.
5	Enable logic	Disable logic
6	Enable extended sequence memory	Disable extended sequence memory
7	Reserved for future expansion.	
8	Omit I/O definitions for LA command	Include I/O definitions for LA command
9	Omit sequences for LA command.	Include sequences for LA command
10	Omit variables from LA command.	Include variables for LA command
11	Skip waiting for command to be completed	Wait for command to be completed
12	Reserved for future expansion.	
13	2 line display.	4 line display
14	Logged pos data signed 16-bit	Logged pos data unsigned 16-bit
15	AI command bi-polar	AI command uni-polar

Bit 1 (BR1)	Bit 0 (BR0)	Brightness
0	0	100%
0	1	75%
1	0	50%
1	1	25%

Bit 6	Bit 7	
0	0	Normal Serial communications
1	0	Low priority serial communications
0	1	High priority serial communications

Under normal conditions, the serial communications use the same interrupt level as the servo loop closure.

Note that bit 11 currently operates for the PB command only.

BW_{bbbb} Set boot options control word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0000 0000 0000 0000

This command allows the user to write a value into the boot options control word. Note that the leading zeros may be omitted. The boot options control word allows various components of the system to be enabled and disabled, as required. The boot options control word bit functions are described below.

		Bit set	Bit cleared
bit	0	1 MHz E clock	2 MHz E clock
	1	No check for operator interface	Check for operator interface
	2	High speed operator interface	Low speed operator interface
	3	No check for smart card interface	Check for smart card interface
	4	Reserved for future expansion.	
	5	Reserved for future expansion.	
	6	Reserved for future expansion.	
	7	Reserved for future expansion.	
	8	Reserved for future expansion.	
	9	Reserved for future expansion.	
	10	Reserved for future expansion.	
	11	Reserved for future expansion.	
	12	Reserved for future expansion.	
	13	Reserved for future expansion.	
	14	Reserved for future expansion.	
	15	Reserved for future expansion.	

OA_n/Z/M/L Initialise ADC offsets to value
Range : -512 to 512
Default : 0

This variation of the ID command is used for setting up the Analogue inputs for use with the Sine/Cos option of the link word. The Z, M, & L suffix refers to zero point, maximum, and lower (Min) points. The suffix $_n$ (either 1 or 2) defines which analogue channel is being set up.

MC(C,I,R,W)$_{n1}$ Memory card operations

This command allows the user to use the smart card interface system. The card can be used for storing between 1 and 8191 numeric variables.

MCC performs an integrity check on the card and returns a 1 or 0 in variable a. This can be used to check the validity of a smart card before trying to read it. Note that the integrity check is automatically performed before performing a read.

MCI performs a check to see if the card is inserted, and returns a 1 or 0 in variable a as an indication. This can be used to check whether the user has inserted a smart card, and provide intelligent prompts on an operator interface.

MCR $_{n1}$: $_{n2}$ performs an integrity check on the card, and if successful, then reads the card into $_{n1}$ (range 1 - 8191) numeric variables, starting with variable number % $_{n2}$. Variable a indicates the integrity of the card (0 or 1) as in MCC. The range of $_{n2}$ is from 0 to 24576.

MCW $_{n1}$: $_{n2}$ performs a write operation on the card. n1 (range 1 - 8191) numeric variables starting from % $_{n2}$ are written to the smart card. The range of $_{n2}$ is from 0 to 24576.

TAG/R/S/A$_n$/B$_n$ Set tangent control axes. (restricted)

Range : 0 to 8.
Default : 0

This command allows the user to set up an axis to operate as a tangent with respect to two other axes. It is designed to be able to control the direction of a cutting blade on an x-y table so that the blade is always pointing in the direction of its motion (which is controlled by the x and the y axes).

In order for the system to operate, there are several conditions which need to be met:

- The TAA and TAB channel parameters need to be set up (preferably lower numbers than the tangent controlled axis, so that the calculations can be performed within the same servo loop closure operation).
- The TAA channel number must be lower than the TAB channel number.
- The tangent controlled axis need to be set so that it operates with only positive numbers (bit 8 of RW, see page 91).
- The bounds value needs to be set so that the SB value corresponds to 360°.

The TAG option starts the motion of the tangent axis, imposing the SA and SV parameters. TAS stops the tangent motion.

The TAR option forces a 360° rotation of the tangent axis.

7.2.3 Mode commands

LM_n Link current axis to axis no $_n$.
Range : 1 to 2×n (where n=No of control boards) or 99
Default : 0

Links the motion of the current axis to axis no $_n$. The defined axis and the current axis will be linked by a ratio defined by the FD and FM parameters. If 0 is specified then there will be no link to the current axis. If a velocity link is defined (using the LW parameter), the current axis will increase in velocity by the defined acceleration rate until it reaches the desired velocity (i.e. the velocity of the channel being followed, multiplied & divided by its FD and FM parameters). If the channel which is being followed is in the "Motor Off" state, then its derived velocity (this can be averaged using the VT parameter) will be used. If it is under control, then the demand velocity will be used. The ST command will cause the motor to decelerate from its current velocity to a standstill, using a deceleration rate defined by the SA parameter. If a position link is defined (bit 0 of the link word, LW), the current axis number must be higher than that which it is being linked to. This is because of the order of the servo loop closure calculations. Please note that under position linking, the bounds of the slave and master axes must correspond to the FM and FD parameters. Also, since absolute positions are linked, the axes must be aligned before issuing the LM command.

The parameter 99 is a special case which links the current axis to 2 analogue inputs which de-code a sine & cosine signal. Bit 8 of the link word LW needs to be set to allow the 99 parameter to be entered.

FD_n Set link factor for division.
Range : 0 to 32 or 1 to 2,147,483,647
Default : 0

This command sets the division factor for linking one axis to another. If bit 2 of the link word (LW) is one, then the actual position data is divided by 2^n in conjunction with being multiplied by the FM parameter. The largest division factor is 2^{16} (65536).

Example : FD 4
This sets the link division factor to $2^4 = 16$.

If bit 2 of the link word (LW) is zero, then the actual position data is divided by n in conjunction with being multiplied by the FM parameter. The largest division factor is 65535.

FM_n Set link factor for multiplication.
Range : 1 to 2,147,483,647 or -2,147,483,647 to +2,147,483,647
Default : 256

This command sets the multiplication factor for linking one axis to another. The actual position data is multiplied by n in conjunction with being divided by two to the power of the FD parameter. The largest multiplication factor is 65535. If bit 7 of LW is set, then negative multipliers are allowed. Note that several channels can be set up simultaneously using the AP command in conjunction with the FM< command (page 20).

Example : FM 9
This sets the link multiplication factor to 9.

Example : FM 23/LW0/FD5
This sets the link multiplication factor to 23, and the division factor to $2^5 = 32$. The resultant factor is therefore 23÷32 = 0·71875.

LW_{bbbb} Link motions control word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0

This command allows the user to set various motion linking control options. Note that the leading zeros may be omitted. The link motions control word bit functions are described below.

		Bit set	Bit cleared
bit	0	Position link	Velocity link
	1	Link to master demand position	Link to master measured position
	2	Only allow 0 to 16 for FD (power of 2 mode)	Allow any FD divisor
	3	Positive & negative tables generated for PGC	Only one table generated for PGC
	4	16 bit gain tables generated	32 bit gain tables generated
	5	Change reference error by cumulative gains	True reference error
	6	Limit velocity and acceleration	No limit to velocity and acceleration
	7	Allow negative multiplier for FM	Positive multiplier only for FM
	8	Link motion to 2 analogue sine/cosine inputs	Normal
	9	Use lookup table for sine/cosine decoding	Normal
	10	Use global position for offset table decoding	Normal
	11	Generate position gain table on master	
	12	Use modified master position (master axis)	Normal
	13	Use modified master position (slave axis)	Normal
	14	Reserved for future expansion.	
	15	Reserved for future expansion.	

Note: If bit 13 is set, bit 12 must be set on LW for the master axis and bit 0 must be clear on the slave.

CL_{pn} Lower bound for link Correction. (restricted)
Range : 1 to ±4 000 000 (4.0E6)
Default : 0

This command sets the lower position bound for link correction. See the PG command below for details on how link correction operates.

CU_{pn} Upper bound for link Correction. (restricted)
Range : 1 to ±4 000 000 (4.0E6)
Default : 0

This command sets the upper position bound for link correction. See the PG command below for details on how link correction operates.

PG_{pn}(D) Position Gain (correction) Value. (restricted)
Range : 1 to ±4 000 000 (4.0E6)
Default : 0

This command sets the value for positional correction required, in conjunction with the CL and CU commands. When the CP command is executed, or on power-up, a table of offsets will be generated This can be displayed using the D option, and shows the incremental position for every calculated position of the slave between the CL and CU bounds. Note that three tables can be generated. These are differentiated by an A, B, or C suffix. A and B refer to continually operating gain tables, and C refers to a one off offset correction. It is effectively enabled by the PGC command, unless the parameter is zero. If the system is currently in one-off table bounds, it will not perform the correction until the next time that the system appears in these bounds. Note that the bounds for a one off table can be superimposed on one of the A or B type tables. However A type bounds cannot be superimposed on B type, and vice-versa. A and B tables can only be used simultaneously if their bounds do not overlap.

There are 2 special case suffix options, which allow one-shot (C type) tables to be re-used without having to re-calculate the table. These are P (prime positive), and N (prime negative). Note that the N suffix is only allowed if bit 3 of the LW link word is set. A one shot table can be shared between different channels provided that bits 3 & 4 of LW are the same, and PGC and CUC-CLC are the same.

Note that number of table entry points created is equal to the CU minus CL parameters minus the PG parameter. Thus a negative gain value uses more table points than a positive one. If the gain value is between -32767 and 32767, it is sensible to set bit 4 of the LW link word, This uses 16 bit values instead of 32 bit values for the tables.

Normally these tables are designed to be used on the slave axis. For example the slave may be a print cylinder which is designed to follow an upstream part of the machine. The pitch of the material may be 120% of the drum circumference, so it is necessary to accelerate the drum over the period when it is not printing. The link word and the tables must be set up on the respective slave channels.

There is an option to use this table mechanism on the master. This allows an indexing mechanism to relate to a master’s absolute position. For example a stitching head is driven by a crank, and for 56% of the rotation of the crank, material can be fed through the head. In this instance the stitching head is the master with bit 12 of LW set. The material feeder is a slave with bit 13 of LW set. In this case, modified incremental position data is passed to the slave (“velocity link”) for the position range of the master, defined by CU and CL.

Example : CH1/SB4000/RW100000000/PGA-600/CUA1500/CLA1000

$CP_{(1)}$ Compile position gain tables. (restricted)

This command compiles all position gain tables. Since it clears the link table memory area before starting the operation, all axes must be unlinked before executing this command.

ZG_n Set Zero Gain distance for position gain table (restricted)
Range : 1 to ±4 000 000 (4.0E6)
Default : 0

This command allows a period of stationary "dwell" during a negative position gain table. It sets the slave distance over which the dwell takes place. Note that this refers to the slave distance and not the master (i.e. after the Multiply and divide factors have been applied, but before any correction table adjustment).

PC Enter position control mode.

This command puts the system back into the normal state with the motor position continuously controlled, after an MO command has been executed, or a position error abort has occurred. The prompt character "n>" is returned in position control mode (where "n" is the current channel number). This is the normal default state entered by the system on power-up, unless specified otherwise by the control word.

In position control mode, an onboard relay on the hardware 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.

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 `:' 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 drive completely. If the drive is not disabled in the motor off state, then it is likely that the motor position will drift, due to some offset in the drive circuits, since the motor position is not controlled in this state.

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

GF Global Motor Off.

All channels which are stopped and under position control will be changed to their motor off state.

PW Set password. (restricted)

This command allows the user to set the privileged mode password. The system replies "Enter password: ", and the user should then type in the new password. The new password is limited to a maximum of ten characters. It is saved in non-volatile 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
1>	PW<CR>	Privileged mode command
Enter password:	(password)	The password is not echoed
O.K.		Password accepted
1>		

PM Enter privileged mode.

For most applications, it is only necessary to make full use of the command set 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 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 "OK" message, and goes into privileged mode. If the password is incorrect, the system sends an "E" error message and stays in normal mode.

Example :

System	User	Comments
1>	PW<CR>	Set password command
Enter password:	(password)	The new password is echoed
1>		

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 into normal mode.

TC Enter tension control mode.

This command puts the system back into the mode when the control algorithm is calculated using tension as the feedback. (instead of position). The velocity parameter is used as the desired tension. Likewise the maximum error ("SE") parameter is used to monitor the tension error. The prompt character "Tn>" is returned in position control mode. This is the normal default state entered by the system on power-up, unless specified otherwise by the control word.

In the tension control mode, an onboard relay on the hardware 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.

CI_n Enter transparent Communication with operator Interface

This command allows the terminal port to communicate directly with the operator interface command interpreter. This allows a complete system to be set up and tested from the host control board. The asynchronous serial port to be used is defined by the $_n$ parameter. All commands with the exception of CX are handled transparently.

CX_n Remote reset of operator interface

This command performs a remote reset of the operator interface, even if it is currently in a locked state waiting for characters from the controller. The asynchronous serial port to be used is defined by the $_n$ parameter.

ZW_{bbbb} Set position record control word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0000 0000 0000 0000

This command allows the user to write a value into the position record control word. Note that the leading zeros may be omitted. The position record control word allows various components of the system to be enabled and disabled, as required. The position record control word bit functions are described below.

		Bit set	Bit cleared
bit	0	Record channel 1 position	Do not record channel 1 position
	1	Record channel 2 position	Do not record channel 2 position
	2	Record channel 3 position	Do not record channel 3 position
	3	Record channel 4 position	Do not record channel 4 position
	4	Record channel 5 position	Do not record channel 5 position
	5	Record channel 6 position	Do not record channel 6 position
	6	Record channel 7 position	Do not record channel 7 position
	7	Record channel 8 position	Do not record channel 8 position
	8	Reserved for future expansion.	
	9	Reserved for future expansion.	
	10	Reserved for future expansion.	
	11	Reserved for future expansion.	
	12	Reserved for future expansion.	
	13	Reserved for future expansion.	
	14	Reserved for future expansion.	
	15	Reserved for future expansion.	

$PB_{n1}:_{n2}$ Enter playback position mode.

This command is used to enter the mode for describing position continuously from numeric variables, starting with variable $_{n1}$, and optionally finishing with $_{n2}$. If $_{n2}$ is not entered, then playback carries on until the variable number reaches 32767, or PB is executed without a parameter. When the playback is finished an "F" is displayed on the screen. When the RR record rate is greater than 1, the positions are linearly interpolated between points. Note that the interpreter mode can be defined using GW bit 11 (page 26). If GW bit 11 is clear, then the interpreter returns to its top level, allowing playback to be stopped using PB. If bit 11 is set, then the interpreter waits until the playback has finished until returning to the top level.

PBC Continue playback position mode.

This command is used continue the playback position mode using the pointers as they were last used when PB was executed without a parameter.

PBD/d Set current playback mode divide factor
Range : 0 to 16
Default : 0

This command is used to set up a divide factor to be applied to the first (lowest number) channel (D) or the second channel (d) of the playback data.

The following formula is applied to the playback data:

$$Pos = Dat \times PBM \div \left(2^{PBD}\right) + PBO$$

where Dat is the raw data, PBM is the multiplier, PBD is the divisor, and PBO is the offset.

PBM/m Set current playback mode multiply factor
Range : 1 to 65535
Default : 1

This command is used to set up a multiply factor to be applied to the first (lowest number) channel (M) or the second channel (m) of the playback data.

PBO/o Set current playback mode offset value
Range : 0 to ±4,000,000
Default : 0

This command is used to set up a offset value to be applied to the first (lowest number) channel (O) or the second channel (o) of the playback data.

PBP Display current playback mode pointer.

This command is used to display the playback position mode pointer value. This can be useful when entering and leaving playback position mode.

RM$_{n1}$:$_{n2}$ Enter record position mode.

This command is used to enter the mode for recording position continuously to numeric variables, starting with variable $_{n1}$, and optionally finishing with $_{n2}$. If $_{n2}$ is not entered, then recording carries on until the variable number reaches 32767, or RM is executed without a parameter. When the recording is finished an "F" is displayed on the screen.

Example :

System	User	Comments
1>	RR10	Set record rate to 40 ms
Enter password:	RM%1:%50:	Start recording from variable 1 to 50
1>		

RMC Continue record position mode.

This command is used continue the record position mode using the pointers as they were last used when RM was executed without a parameter.

RMP Display current record mode pointer.

This command is used to display the current record mode pointer value. This can be useful when entering and leaving record position mode.

RR_{nn} Set record position time rate $_{nn}$.
Range : 0 to 8
Default : 5

This command sets the rate at which the system records (using RM) and plays back (using PB) sets of position data (as defined by ZW). The step rate is 2^n steps per second, where n is the value of RR. The fastest step rate is 256 steps per second, or one step every 4ms. The system interpolates linearly between table points at the slower speeds in order to maintain smooth motion.

The record position rate may be changed at any time, even when a recording or playback session.

Note that the RQ record distance rate must be set to zero in order for the RR time rate to be set.

Example : ZW11/RR 3
This sets the record rate to 2^3 = 32 sets (motor channels 1 & 2) per second.

RQ_{nn} Set record position distance rate $_{nn}$.
Range : 0 to 65535
Default : 0

This command sets the distance rate at which the system records (using RM) and plays back (using PB) sets of position data (as defined by ZW). The step rate is in interpolated position increments

The record position rate may be changed at any time, even when a recording or playback session.

Note that if RQ is set to a value other than zero, then record mode will be performed using the RQ position data rate rather than the RR record time rate.

Example : ZW11/RQ100
This sets the record rate to be 1 set (motor channels 1 & 2) every 100 encoder counts.

7.2.4 Move commands

AB(K,O) Abort, emergency stop.

The motor stops immediately, ignoring the system acceleration. This may be used instead of the ST command, where an abrupt stop is required. The AB command may also be used to break out of sequences. The optional K parameter enables a selective abort from any pending keypad operations (e.g. VK), whilst maintaining all other current system operations. The optional O parameter performs a complete reset of the operator interface. Likewise AB without the K parameters maintains any current keypad operations, whilst stopping all motion operations.

GA Global abort, emergency stop.

All motors stop immediately, ignoring the system acceleration. This may be used instead of the ST command, where an abrupt stop is required. The GA command may also be used to break out of sequences.

ST_{nn} Stop.

The motor stops under controlled deceleration, set by the SZ command. The stop command may be used during a normal move or constant velocity move to decelerate the motor to a stop. The ST command may also be used to break out of sequences. The optional parameter allows the stop to finish at a defined absolute position.

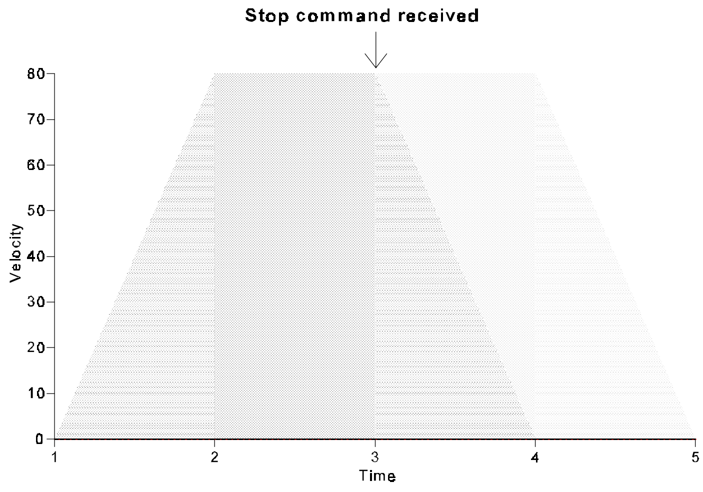


Figure 4 Move with normal stop

GS Global Stop.

All channels currently running (whose bits are set in MW) motor stop under controlled deceleration, set by the channels' respective SZ command. The stop command may be used during a normal move or constant velocity move to decelerate the motor to a stop. The GS command may also be used to break out of sequences.

MA±$_{nn}$ Move to absolute position ±$_{nn}$.
Range : ± 4 000 000 (4.0E6) encoder counts.

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

Example : MA +2000
The motor moves to absolute position 2000.

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 position. The move distance is entered in user units.

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

MM$_{nn}$ Prepare move ±$_{nn}$ units on current channel (multi-axis move)
Range : ± 8 000 000 (8.0E6) encoder counts.
Default : 0

The system sets up a distance for a multi axis move. The parameter is the distance which will be used by the MX command to execute an MX command. A mixture of relative and absolute moves may be specified by the appropriate bits in the MW word. The move distance is entered in user units.
For circular interpolation, the MM value depends on what bit of IW has been set. If bit 0 is set, then MM is the destination position. If bit 1 is set, then MM is the number of degrees for the move; it should be set the same for both channels.

MX Execute multi-axis move

The system performs a multi axis move. The command operates globally, moving all axes defined by the MW word. The distances moved are defined by the previously set up MM parameters. The start of the moves is synchronized, but the finish is determined by the parameters on each channel. A mixture of relative and absolute moves may be specified by the appropriate bits in the MW word. The move distance is entered in user units.

Example :
```
CH2/MM-3000
CH3/MM500
MW100 0000 0110
MX
```

The motor on channel 2 moves 3000 units from its current position in the negative direction and simultaneously, the motor on channel 3 moves to an absolute position 500.

MW_{bb} Set multi axis move word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0

This command allows the user to write a value into the global multi axis move word. Note that the leading zeros may be omitted. The move word defines which axes will be executed during a MX or VX multi axis move operation. The multi axis move word bit functions are described below.

		Bit set	Bit cleared
bit	0	Enable MX/VX move for ch 1	Disable MX/VX move for ch 1
	1	Enable MX/VX move for ch 2	Disable MX/VX move for ch 2
	2	Enable MX/VX move for ch 3	Disable MX/VX move for ch 3
	3	Enable MX/VX move for ch 4	Disable MX/VX move for ch 4
	4	Enable MX/VX move for ch 5	Disable MX/VX move for ch 5
	5	Enable MX/VX move for ch 6	Disable MX/VX move for ch 6
	6	Enable MX/VX move for ch 7	Disable MX/VX move for ch 7
	7	Enable MX/VX move for ch 8	Disable MX/VX move for ch 8
	8	Enable Abs MX or -ve VX move for ch 1	Enable Rel MX or +ve VX move for ch 1
	9	Enable Abs MX or -ve VX move for ch 2	Enable Rel MX or +ve VX move for ch 2
	10	Enable Abs MX or -ve VX move for ch 3	Enable Rel MX or +ve VX move for ch 3
	11	Enable Abs MX or -ve VX move for ch 4	Enable Rel MX or +ve VX move for ch 4
	12	Enable Abs MX or -ve VX move for ch 5	Enable Rel MX or +ve VX move for ch 5
	13	Enable Abs MX or -ve VX move for ch 6	Enable Rel MX or +ve VX move for ch 6
	14	Enable Abs MX or -ve VX move for ch 7	Enable Rel MX or +ve VX move for ch 7
	15	Enable Abs MX or -ve VX move for ch 8	Enable Rel MX or +ve VX move for ch 8

WM_{nn} Wait for $_{nn}$ system ticks before moving current channel.
Range : 0 - 65535 system ticks.
Default : 0

This allows a time delay between an MX command and current channel actually starting to move.

VC± Move at constant velocity.

The system accelerates the motor at the system acceleration until it reaches the system velocity, in the specified direction. It then controls the motor at constant velocity, until asked to stop by the ST command. Velocity control mode can only be entered from position control mode, and not directly from the motor off state.

Example : SA 1000/ SV 2000/ 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.

VX Move (multi-axis) at constant velocity.

The system accelerates the motor channels which have been defined in the multi-axis move word (MW) at the system acceleration until it reaches the system velocity, in direction which have been specified in the multi-axis move word. It then controls the motors at constant velocity, until asked to stop by the GS (all selected channels) or ST (currently selected channel) command. Velocity control mode can only be entered from position control mode, and not directly from the motor off state.

Example : MW11/CH1/SA 1000/ SV 2000/CH2/ SA500/SV6000/ VX
This command sequence enables multi axis velocity control on channels 1 and 2. It sets the acceleration on channel 1 to 1000 units per second squared, the velocity to 2000 units per second, and on channel 2 to 500 and 6000 respectively. It then accelerates to the set velocity in the positive direction.

AC_{nn} Set arc centre coordinate for current channel for interpolated circular move
Range : ± 8 000 000 (8.0E6) encoder counts.
Default : 0

The system sets up a co-ordinate for the centre of an interpolated circular move. This should be set up on both the axes which together will create a circular interpolated move.

XC+/- Execute interpolated circular move

The system starts a circular move in the positive (clockwise) or negative (anti-clockwise) direction. The destination is defined by the MM parameter. If it is found that the current position is not on the arc of the same circle as that defined by the MM and AC parameters, then an error is returned.

XL Execute interpolated linear move

The system starts a linear move. The destination is defined by the MM parameter. The current position is the starting point.

$GC_{v1}:_{v2}$ Transform from Polar to Cartesian

The system performs a 2 dimensional transform from a Polar to Cartesian co-ordinate system. The 2 axes are defined by bits 0-7 of MW (move word). The origin for both the Polar and the Cartesian definition is defined in the AC parameter for each of the two channels (the lowest channel number representing the x, and the higher number representing the y). The radius is defined in variable $_{v1}$, and the angle (in degrees) × 100 is defined in variable $_{v2}$. The Cartesian result is added to the AC zero, and placed in the MM parameter for the relevant channel.

$GP_{v1}:_{v2}$ Transform from Cartesian to Polar

The system performs a 2 dimensional transform from a Cartesian to Polar co-ordinate system. The 2 axes are defined by bits 0-7 of MW (move word). The origin for both the Polar and the Cartesian definition is defined in the AC parameter for each of the two channels (the lowest channel number representing the x, and the higher number representing the y). The AC zero is subtracted from the MM parameter for the relevant channel to generate Cartesian co-ordinates. The radius is placed in variable $_{v1}$, and the angle (in degrees) × 100 is placed in variable $_{v2}$.

IW$_{bb}$ Set interpolation move word. (restricted)
Range : 0000 0000 to 1000 0000 (binary).
Default : 0

This command allows the user to write a value into the interpolation move word. Note that the leading zeros may be omitted. The move word defines how a 2-axis circularly interpolated move is to be defined. Only one bit may be set at any one time. The interpolation move word bit functions are described below.

		Bit set	Bit cleared
bit	0	Start point, end point, centre point	Normal
	1	Start point, arc, centre point	Normal
	2	Reserved for future expansion	
	3	Reserved for future expansion	
	4	Reserved for future expansion	
	5	Reserved for future expansion	
	6	Reserved for future expansion	
	7	Reserved for future expansion	

QC$_{nn}$ Prepare relative stop (from creep) position on current channel (multi-axis stop)
Range : 0 or min defined by SS & SZ to 8 000 000 (8.0E6) encoder counts.
Default : 0

The system sets up a relative distance for a multi axis stop, using an input line, defined by the DQ command. The parameter is the relative distance which will be used to stop the current axis when an input line is seen. The DQ stop for a particular motor channel is enabled by setting a non-zero value to the QC parameter. Please note that the stop is only performed when running at constant creep velocity (as defined by the SS parameter). The minimum non-zero value for the QC parameter is defined by SS and SZ, and will be displayed if a non-zero value less than the minimum is entered. If SS or SZ for the current channel are changed, then the QC parameter is set to zero.

QV$_{nn}$ Prepare relative stop position on current channel (multi-axis stop)
Range : 0 or min defined by SV & SZ to 8 000 000 (8.0E6) encoder counts.
Default : 0

The system sets up a relative distance for a multi axis stop, using an input line, defined by the DQ command. The parameter is the relative distance which will be used to stop the current axis when an input line is seen. The DQ stop for a particular motor channel is enabled by setting a non-zero value to the QV parameter. Please note that the stop is only performed when running at constant maximumm velocity (as defined by the SV parameter). The minimum non-zero value for the QV parameter is defined by SV and SZ, and will be displayed if a non-zero value less than the minimum is entered. If SV or SZ for the current channel are changed, then the QS parameter is set to zero.

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. The motor accelerates to the system velocity in the specified direction. When the system detects a reference input signal, it resets the position counters to zero immediately. The motor then decelerates to a stop and moves back to the new zero position.

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, and bit 4 of RW which defines whether any reference input is valid, or only a combination of them. The 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 read the reference command section (section 7.2.13, page 90) later in this manual.

If no reference input or marker input is defined, then the IN commands returns the E error message, and the initialisation sequence is not executed.

Example : RW0 / IN +
The motor moves in the positive direction until the reference input is seen. It then stops, and in this example it moves back in the negative direction to the reference position.

ID Initialise DAC 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 allow 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 position control mode, to set the actual position as close as possible to the required position. This can be done automatically by using the autostart sequence facility. 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 after a move remains in the move state without returning to the normal position control mode. 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.

If this command is used on a regular basis (e.g. whenever the system is switched on), it will have the effect of removing any drift in the analogue amplifier which may have arisen with time.

IDZ/M/L_n Initialise ADC offsets.

This variation of the ID command is used for setting up the Analogue inputs for use with the Sine/Cos option of the link word. The Z, M, & L suffix refers to zero point, maximum, and lower (Min) points. The suffix $_n$ (either 1 or 2) defines which analogue channel is being set up.

OA_n/Z/M/L Initialise ADC offsets to value
Range : -512 to 512
Default : 0

This variation of the ID command is used for setting up the Analogue inputs for use with the Sine/Cos option of the link word. The Z, M, & L suffix refers to zero point, maximum, and lower (Min) points. The suffix $_n$ (either 1 or 2) defines which analogue channel is being set up.

IP_{nn} Set inch pause time.
Range : 0 to 65,535
Default : 0

This command sets a maximum time for an input line to be set while inching.

IL_{nn} Set inch distance.
Range : 0 to 4,000,000
Default : 0

This command sets the distance for an input line to be set while inching.

IM± Execute an inch move.

This command executes a move whose distance is defined by the "IL" parameter. There is a special case if "IM" is defined in a "DI" input line string. In this case if the input is not reversed within the "IW" inch wait tine parameter, the axis will continue moving until the input is reversed.

Example : IM+
This executes a move from the current position by a relative distance defined by the "IL" parameter.

7.2.5 Set parameter commands

SW$_{nn}$ Set Window. (restricted)
Range : 0 to 65535
Default : 10

This command sets a window around the required final position of a move. Its operation is such that the system finishes a move and returns the prompt character to the user only when the motor is within this window. Note that when using a narrow window, it is important that the DAC offset on the current channel 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 stay in the move state without returning the normal prompt. If this occurs, the ST or AB commands may be used to get back to the normal state. Note that the window is specified in encoder counts. This command is restricted, and may only be used in privileged mode.

Example : SW 25
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.

SB$_{nn}$ Set position overflow bound. (restricted)
Range : 1 to 4 000 000 (4.0E6)
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 by using the DC command, and may be reset to zero by the RC command.

The bound position defined by the SB command is also used as the expected reference position when the system is set up to continuously monitor the reference input. Refer to section 7.2.13 (page 90)for more details about the reference commands.

Example : SB1000
This sets the position bounds to ±1000 counts. An application of this is 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 1200 is executed, the final position is 200. The motor has moved a total distance of 1200 counts as required, but the final position is the remainder when divided by the bound value. If a move from zero to -1200 is executed, the final position is -200. 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.

GB_{nn} Set global position overflow bound. (restricted)
Range : SB value to 4 000 000 (4.0E6)
Default : 4 000 000

This command sets upper and lower bounds on the global absolute position of the system. It allows an overall position measurement which could be a complete cycle of a machine, whilst the local SB bounds values could be set to an encoder revolution. The global position can be accessed using the DPG command.

BB_{nn} Set bound counter overflow bound. (restricted)
Range : 0 to 4 000 000 (4.0E6)
Default : 0

This command sets upper and lower bounds of the bounds counter. The number of times the bound counter has wrapped can be read using the DCB display bounds counter overflow counter command.

SV_{nn} Set velocity (speed).
Range : Creep velocity (SS) to 800 000 (8.0E5)
Default : 1024

This command is used to set the system velocity, in user units per second. It may be used when the motor is stationary, or when moving at constant velocity after a VC command.

Example : SV 5000
This sets the system velocity to 5000 units per second.

SA_{nn} Set acceleration.
Range : 1 to 2 000 000 000 (2.0E9)
Default : 1024

This command sets the normal system acceleration to the specified value, in user units per second squared. Note that the acceleration is rounded to the nearest multiple of 256, giving a resolution of 256 counts.

Example : SA 10000
This sets the system acceleration to 10000 units per second squared.

SZ_{nn} Set deceleration.
Range : 0 to 2 000 000 000 (2.0E9)
Default : 0

This command sets the normal system deceleration to the specified value, in user units per second squared. If zero is entered, then the current acceleration rate will be used for decelerating. Note that the acceleration is rounded to the nearest multiple of 256, giving a resolution of 256 counts.

Example : SZ 2000
This sets the system deceleration to 2000 units per second squared.

SC$_{nn}$ Set creep distance.
Range : 0 to 65535
Default : 0

The normal trapezoidal velocity profile for a position move can be modified to include a slow speed creep to the final required position. The creep distance is the distance from the final position over which the system moves at the slow creep speed, set by the SS command. This may be used to minimise overshoot at high speeds and accelerations.

Example : SC 200
This command sets the creep distance to 200 units. A position move command will now start to decelerate earlier than normal, such that the system reaches the slow creep speed at least 200 units before the final required position.

SS$_{nn}$ Set slow creep speed.
Range : 1 to system velocity (SV)
Default : 32

This command allows the user to set the speed of the slow creep to the final position, if required. It is specified in the same units as the system velocity.

Example : SS 100
This sets the slow creep speed to 100 units per second.

SD$_{nn}$ Set deadband.
Range : 0 to 65535
Default : 0

The system normally controls the position of the motor continuously, whether moving or stopped. This command allows the user to set up a deadband about the nominal position, within which the system will not control the position of the motor. This may be used, for example, to prevent hunting in systems with mechanical backlash. The deadband only becomes active after the system has reached the required position and the settling time, set by the SL command, has expired.

Example : SD 100
This sets the deadband to 100 units. The system will now only control the position of the motor if it moves more than 100 units away from the required position.

SL$_{nn}$ Set settling time.
Range : 0 to 65535
Default : 256

This command sets the time that the system waits, after reaching its required position, before the deadband becomes active. It is specified in units of 1/256 seconds.

Example : SL 128
This sets the settling time to 128x1/256 = 0.5 seconds.

TH±$_{nn}$	Set high tension limit. (restricted) Range : ± 4 000 000 (4.0E6) Default : + 15 000

This command sets the upper limit for the system tension set point. The units are defined by the calibration of the analogue input and the TM tension multiplier parameter.

Example : SL 13200
This sets the upper tension limit to 13.2 Kg force, assuming that the TM parameter has been set up so that one unit represents 1 gramme force.

TL±$_{nn}$	Set low tension limit. (restricted) Range : ± 4 000 000 (4.0E6) Default : + 3 000

This command sets the lower limit for the system tension set point. The units are defined by the calibration of the analogue input and the TM tension multiplier parameter.

Example : SL 2500
This sets the lower tension limit to 2.5 Kg force, assuming that the TM parameter has been set up so that one unit represents 1 gramme force.

TM$_{nn}$	Set tension multiplier. (restricted) Range : 1 to 65535 Default : 1

This command sets the multiplier factor for converting measured tension to grammes force.

Example : TM 75
This sets the tension multiplier to a value of 75.

MT$_{nn}$	Set minimum tension threshold. (restricted) Range : 0 to 65535 Default : 0

This command sets minimum tension value (measured tension × TM tension multiplier). If the system tension is below this value, tension control will stop, and both channels 1 and 2 will enter the motor off state.

Example : MT 50
This sets the tension multiplier to a value of 50 grammes force, assuming that the TM parameter has been set up so that one unit represents 1 gramme force.

HT$_{nn}$ Set maximum tension threshold. (restricted)
Range : 0 to 65535
Default : 3,000

This command sets maximum tension value (measured tension × TM tension multiplier). If the system tension is above this value, tension control will stop, and both channels 1 and 2 will enter the motor off state.

Example : HT 2800
This sets the tension multiplier to a value of 2.8 Kg force, assuming that the TM parameter has been set up so that one unit represents 1 gramme force.

BC Set backlash compensation distance.
Range : 0 to 65535
Default : 0

This command sets up a backlash compensation facility. It applies only to the MA and MR point-to-point move commands. It defines an extra distance that the motor moves each time it reverses direction, thus taking up any slack or backlash between the motor and the final output. It is defined in encoder counts.

Example : BC 20
This sets the backlash compensation distance to 20 counts. Each time the motor changes direction on successive move commands, the first move in the new direction is extended by 20 counts to take up the backlash.

ZC(G)[$_{nn}$] Zero position counters or set position.

If a position value is given, the system sets the current demand position to the given (absolute) value. If no value is given, it sets the current demand position to be zero absolute position. The ZC command may be used at any time. The optional G suffix allows the global position to be reset to zero.

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

Example : ZC8000
This defines the current demand position as position 1000

RC Reset position overflow counter.

This command resets the position overflow counter to zero. The overflow is incremented when the position exceeds the upper bound, and is decremented when the position passes the lower bound.

TS$_{dd:mm:yy:hh:mm:ss}$ Time set.

This command allows the user to set the system time. It expects the time to be in the format $_{dd:mm:yy:hh:mm:ss}$.

DB(K)$_{nn}$ Set input debounce time. (restricted)
Range : 0 to 255
Default : 5

This command sets up a debounce time for all the digital inputs. It is specified in 1÷256 second (about 4ms) ticks. 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. The suffix K allows the keypad debounce time to be adjusted.

NOTE: The debounce value does not apply to reference 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 : DB 2
This sets the debounce time to about 8 ms (2 samples).

7.2.6 Sequence commands

This section describes the sequence facilities. 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 an "N" no room error message.

ES_{nn} Enter sequence.
Range : 1 to 2047

This command is used to enter sequences into the system. The system responds with a "S_{nn}:" prompt for the sequence entries. Each entry in the sequence is any valid command line. Command sequences on one command line are accepted as one sequence entry. Sequence entries may also include commands to execute other sequences and profiles. To end the sequence, make a blank entry by just entering a (CR), 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. If bit 6 of the global control word (GW) is set, and sequences 1001 to 2047 are put in a second memory page. This can be useful if there are a lot of large sequences to be stored. However it should be noted that this second memory page is also used for the data logging facilities. The second sequence memory page should therefore not be used if data logging is to be used.

Example :

System	User	Comments
1>	ES 1 (CR)	Enter sequence 1
S1 1:	ID/ IN - (CR)	Initialise position
S1 2:	MA100/ WT256/ MA0/ WT256/ RP3 (CR)	Do this 3 times
S1 3:	MA 2000 (CR)	A single move
S1 4:	(CR)	End sequence
1>		Normal prompt

LS_{nn} List sequence.
Range : 1 to 255

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. The sequence may be listed continuously, or the system can print one line at a time and wait for the user to press the (CR) key before printing the next line. This is useful when using the system with a membrane keyboard or a hand-held terminal which only has a small number of display lines. This list pause facility is controlled by one of the flag bits in the DW command.

Example : LS 1
This will list sequence 1 on the display or terminal. The output for the sequence given above would look like this.

```
1>LS 1 (CR)                          User input to list  sequence.
S1: ID/IN-
S1: MA100/WT256/MA0/WT256/RP3
S1: MA2000
1>
```

XS_{nn} Execute sequence.
Range : 1 to 2047

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 will abort automatically if any error condition occurs. A sequence may be aborted by using the AB command. The ST stop command may also be used to stop the currently executing move command.

Example : XS 3
The system executes stored sequence no. 3.

RP_{nn} Repeat.
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 system will repeat indefinitely.

Example : MA 2000, MA 0, RP5
This moves the motor to position 2000 and then back to position 0, repeated 5 times.

ER(F) 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 stop and abort commands, which stop the system 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.

The optional F parameter allows the termination of a for-next loop next time the loop is completed (See page 108).

HS_{nn} Display history of sequences executed.
Range : 1 to 255, or no value

Displays a list of the numbers of the most recent $_{nn}$ sequences executed in inverse chronological order. If no parameter is given, all of the last 256 most recently executed sequences are listed.

$SK(_{A-Z})_{nn}$ Execute sequence on keypad entry event.
Range : 1 to 2047 or no value

This command sets up the system to execute sequence number $_{nn}$ as soon as the enter key has been pressed on the keypad (after a VK command, page 135). This allows for the system to operate in an asynchronous manner. The optional prefix parameter (A-Z) allows non-numeric keys to start events at any time. These keys can be selectively masked using the EK and MK commands (page 86). A table of parameters to define which keys are to be used can be found in section 12.3 (page 183). If no parameter is entered, then all the above keys are disabled.

Example : SK 56
The system executes stored sequence no. 56 after a variable has been entered using the VK command.

Example : SKB 23
The system executes stored sequence no. 23 after the F2 key has been pressed.

NOTE: The keys which are controlled by this command have been changed in software version 154.9.

CE_{nn} Execute sequence continuously as timed event.
Range : 0 to 2047

This command sets up the system to execute sequence number $_{nn}$ as soon as the time defined by the PL parameter has passed. This sequence will be repeated indefinitely at the defined time interval until CE0 or an AB command is entered. Note that the time interval relates to the start of the sequence, even if the sequence takes a significant time to execute.

Example : PL512/CE 12
The system executes stored sequence no. 12 after continuously every 2 seconds.

PL$_{nn}$ Set parameter for looping continuous event
Range : 1 to 65535
Default : 256

This command sets the time parameter for the CE continuous sequence command. The units are system ticks (1÷256 second).

Example : PL256
This sets the delay between continuous sequence events to 1 second.

ED$_{nn}$ Execute sequence event after time delay.
Range : 0 to 2047

This command sets up the system to execute sequence number $_{nn}$ as soon as the time defined by the PE parameter has passed. This sequence will be executed once only.

Example : PE64/ED 20
The system executes stored sequence no. 64 after a period of 1/2 second..

PE$_{nn}$ Set parameter for ED event
Range : 1 to 65535
Default : 256

This command sets the time parameter for the ED timed sequence event command. The units are system ticks (1÷256 second).

Example : PL192
This sets the delay between continuous sequence events to 1•5 seconds.

EE$_{nn}$ Execute sequence on error condition.
Range : 1 to 2047, or no value

This command sets up the system to execute sequence number $_{nn}$ as soon as a system error state is encountered. This allows for system to interrogate the system error and display an appropriate message.

Example : EE 123
The system executes stored sequence no. 123 after an error state has been reached.

EM$_{nn}$ Execute sequence when Move condition complete
Range : 0 to 2047

This command sets up the system to execute sequence number $_{nn}$ as soon as the currently executing motion is complete.

NE_{nn} Execute sequence after snapshot event. (local)
Range : 0 to 255

This command sets up the system to execute sequence number $_{nn}$ as soon as a snapshot event has taken place. This command needs to be reset after each operation.

Example : NE 55
The system executes stored sequence no. 55 after an a snapshot event has taken place.

RE_{nn} Execute sequence after reference event. (local)
Range : 0 to 255

This command sets up the system to execute sequence number $_{nn}$ as soon as a reference event has taken place. This command needs to be reset after each operation.

Example : RE 2
The system executes stored sequence no. 2 after an a reference event has taken place.

GE_{nn} Execute sequence after position Gain event. (local)
Range : 0 to 255

This command sets up the system to execute sequence number $_{nn}$ as soon as an offset gain event has taken place.

MF Display free memory.

This command displays information about the memory space available for sequences and profiles. This command returns the amount of free space as a number of bytes. The maximum space available is about 20 kbytes.

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 an "N" out of memory error message when the total amount of spare memory is larger than the data being entered. This occurs because the system allocates a contiguous 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 :

System	User	Comments
1>	MF<CR>	Enter MF command
6538		System returns number of bytes free
1>		

AS(R)$_{nn}$ Set autostart sequence.
Range : 0 to 2047
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 non-volatile memory. If no sequence number is given, the system prints the current autostart sequence number. If the "R" option is entered, an auto-start sequence for startup after software watchdog reset can be entered. This could be the same as the AS auto-start sequence, but could be some alternative sequence to ensure a safe re-start.

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.

HR$_{nn}$ Set Hard Reset sequence or initiate Hard Reset.
Range : 0 to 255
Default : 0

This command locks the software watchdog in order to force a system reset (after about 6 seconds).

XT Exit from current sequence.

This command tells the system to exit from the sequence which is currently running. If this is a sequence which has been called from another sequence, then the calling sequence will continue execution.

GL_n Go to line in sequence
Range : 1 to no of lines in current sequence

This command allows looping to a specified line number within a sequence. Note that it can only be executed from within a sequence.

Example:
A sequence might consist of the following:

```
CO3/MR5000
SO4/WT500/II5-/XT
GL1
```

This will continue looping to line 1 until input line 5 is negative, when the XT command will escape from the sequence.

WS_{nn} Wait until stopped before executing sequence.
Range : 1 to 255

This command sets up the system to execute sequence number $_{nn}$ as soon as the current channel has stopped. This state can be setup for more than one channel. Note that if motors are being mapped to others, only the master channel is considered to be moving. i.e. if WS parameters have been set up for motors which are mapping to others, the sequence(s) will be executed immediately. This command enables an input line to execute commands which are only legal when a channel has stopped, without having to perform involved tests. Note that once a WS sequence has been executed, the parameter is reset to zero, and has to re-enabled.

Example : DI3-/CH2/ST/WS 34/CH1

```
ES34
MI8/CH2/LM1
```

The system sets up input line so that when it goes to the negative state it executes stored sequence no. 34 after channel has stopped. Sequence 34 links the motion of channel 2 to channel 1

7.2.7 Logic commands

This section describes the logic commands. They provide facilities for defining logic functions which will run asynchronously with the rest of the controller's functions, in a similar way to a programmable logic controller (PLC).

Logic functions are set up by entering logic definition equations (EJ command). These are executed sequentially, line by line. There are inputs, outputs (which can be logical inputs or outputs), and imaginary nodes (which can also be logical inputs or outputs). There are 3 logic operators: AND (*), OR (+), and exclusive OR (^).

In addition there are 32 timers and 32 counters. A timer or counter can be loaded with a value between 1 and 65535 when an input node or output is in a given state. A timer then decrements every 0.1 sec, and it can be defined what the logical consequence is when it reaches zero. A counter operates in the same way, except that when an associated input changes from low to high the counter is decremented.

EJ$_{nn}$ Enter logic definition $_{nn}$.
Range : 1 to 255

This command is used to enter logic definition sequences into the system. The system responds with a "L$_{nn}$:" prompt for each line entry. Each line is a single logic sequence, which is parsed from left to right. The last entry of each line must be an output or output node. Inputs can be a physical input (I), physical output (O), a logical node (N), a zero counter (C), or a zero timer (T), and they must be entered in upper case. Logic operators are placed after the inputs, which are to be operated on. Up to 8 inputs can be entered on one line. However only one logic operation and one output can be on one line. This limitation is imposed for efficiency, and can be overcome by the use of nodes and multiple line logic sequences. To end the sequence, make a blank entry by just entering a (CR), and the system then returns to normal operation. The logic definitions are accessed automatically, and sequentially (i.e. logic definition 1 followed by 2 etc). Logic definition numbers 1 to 10 are executed every 1/256th second (i.e. high performance), and number 11 to 255 are executed in background mode. It is important that there should not be conflicting logic between high performance (1-10) and normal (11-255) logic sequences.

Example:

System	User	Comments
1>	EJ 3<CR>	Enter logic definition 1
L3:	I1/I2/I4/*/N2	I/p 1 is ANDed with i/p2 and i/p4 to node 2
L3:	N2/I6/^/O2	Node 2 is exclusive ORed with i/p 6 to o/p 2
L3:	(CR)	End logic equation
1>		Normal prompt

FC$_{n1}$:$_{n2}$:I$_{n3}$±:O$_{n4}$±:I$_{n5}$± Fill logic counter value.

This command is used to enable counter number $_{n1}$(1-32) to be set to a value $_{n2}$ (1-65535). This happens when Input number n3 changes to a + or - state. The counter will then be decremented whenever input $_{n5}$ goes to a defined state, and on completion output $_{n4}$ will go to the defined state. If the command is used in the form FC$_{n1}$<CR>, then the current value of counter $_{n1}$ is output. A counter set-up and its association with inputs and outputs can be reset by using the form FC$_{n1}$:R<CR>. The current setup for a counter can be displayed using FC$_{n1}$:D<CR>.

FT$_{n1}$:$_{n2}$:I$_{n3}$±:O$_{n4}$± Fill logic timer value.

This command is used to enable timer number $_{n1}$(1-32) to be set to a value $_{n2}$ (1-65535). This happens when Input number n3 changes to a + or - state. The timer will then be decremented every 101.5ms, and on completion output $_{n4}$ will go to the defined state. If the command is used in the form FT$_{n1}$<CR>, then the current value of timer $_{n1}$ is output. A timer set-up and its association with inputs and outputs can be reset by using the form FT$_{n1}$:R<CR>. The current setup for a counter can be displayed using FC$_{n1}$:D<CR>.

LJ$_{nn}$ List logic definition.
Range : 1 to 255

This command allows the user to examine a logic definition that has previously been entered into the system.

Example : LJ 3
This will list logic definition 3 on the display or terminal. The output for the definition given above would look like this.

```
1>LJ 3 (CR)                    User input to list logic definition.
L3 1: I1/I2/I4/*/N2
L3 2: N2/I6/^/O2
1>
```

RY$_{nn}$ Read node state(s).
Range : 1 to 32

This command reads the current state of an imaginary node. Nodes can only be set and cleared using logic equations. The current state is printed as a "0" or "1" on the display. A "0" represents a logic low, and a "1" represents a logic high. If no node number is given in the command, the system displays the current state of all nodes.

IY$_{nn}$± If node true do command line.
Range : 1 to 32

This command allows the programmer to specify that a command or command line is conditional on the current state of an imaginary node. Nodes can only be set and cleared using logic equations. If the node specified in the IY command is in the specified state (the condition is true), then the remainder of the command line is executed. If the node is not in the specified state, the remainder of the command line is skipped, and execution proceeds to the next line of input. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on input line states.

SY_{nn}	Set node $_{nn}$.
	Range : 1 to 32

This command sets up the specified node to a logic high. If no node number is set then all available nodes are set to a logic high.

Example : SY 15
This sets up node 15 to a logic high.

CY_{nn}	Clear node $_{nn}$.
	Range : 1 to 32

This command clears the specified node to a logic low. If no node number is set then all available nodes are set to a logic low.

Example : CY 17
This sets up node 17 to a logic low.

7.2.8 Profile commands

This section describes the profile facilities. 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 an "N" no room error message.

The normal move commands allow the user to easily move the motor to any possible position. However they are limited to using the trapezoidal or sinusoidal velocity profile defined by the set velocity and acceleration parameters. In some cases it may be desirable to use a different velocity profile such as parabolic. 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 either as absolute or 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.

The profile step rate, or profile velocity, is itself programmable by means of the PV command. The fastest profile velocity is 256 steps per second, or approximately one step every 4ms. The profile velocity may be reduced by factors of 2, that is to half speed, quarter, eighth, etc., down to the slowest step rate of 1 step 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 lowest step rates.

EP_{nn} Enter profile $_{nn}$.
Range : 1 to 255

This command is used to enter profile tables into the system. The system responds with a "P_{nn}:" prompt for the table entries. Each entry in the table is the (signed) distance between the new position and the previous position, that is, the required cumulative motor position relative to the start of the profile. To end the table, make a blank entry by entering just a (CR), and the system then returns to normal operation. The profile is accessed by means of the profile number assigned by the user when it is entered.

Example:

System	User	Comments
1>	EP 1<CR>	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
1>		Normal prompt

If bit 4 of the Display Word (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
1>	EP 1<CR>	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
1>		Normal prompt

LP$_{nn}$ List profile $_{nn}$.
Range : 1 to 255

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 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. The profile table may be listed continuously, or the system can print one line at a time and wait for the user to press the (CR) key before printing the next line. This is useful when using the system with a membrane keyboard or a hand-held terminal which only has a small number of display lines. This list pause facility is controlled by one of the flag bits in the DW command.

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 the Display Word (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.

Example : LP 1
This will list profile 1 on the display or terminal. The output for the example profile given previously would look like this, in absolute position format.

```
System              User          Comments
1>                  LP 1<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
1>
```

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

```
System              User          Comments
1>                  LP 1<CR>      User input to list profile
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
```

1>

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

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

XP_{nn} [-,$_{r}$;$_{nn}$ [-], $_{r}$] 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 SA command.

This command has special delimiters, which allow a repeat parameter after a comma, and up to four separate profiles to be executed in sequence, each profile separated by a semi-colon. By using a single XP command for multiple profiles and repeats, the system stays in the profile state until the repeated profile is completed, without returning to the idle state in between. This maintains a continuous smooth operation without any pause in between profiles for command interpretation.

Example : XP 2
The motor follows the stored profile no. 2, at the speed set by the PV command.

Example : XP 1,2;2,3
The motor follows the stored profile no. 1 three times, followed by the stored profile no 2 four times, at the speed set by the PV command.

$XP_{nn}\&_{nn}$ Execute simultaneous profiles $_{nn}$ and $_{nn}$.
Range : 1 to 255

This command tells the system to execute profile number $_{nn}$ on the foreground channel simultaneously with a second (can be the same) profile $_{nn}$ on the background channel. The speed at which the system steps through the tables (the profile velocity) is set by the PV command. Note that the profiles are always executed relative to the current position. A profile move may be aborted by using the AB command. If one or both of the profile numbers are followed by a minus sign, the respective profile is executed in the reverse direction.

Example : XP 2
The motor follows the stored profile no. 2, at the speed set by the PV command.

PV_n Set profile velocity.
Range : 0 to 8
Default : 5

This command sets the rate at which the system steps through a profile table. The step rate is 2^n steps per second, where n is the value of PV. The fastest step rate is 256 steps per second, or one step every 4ms. The system 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 when executing a profile.

Example : PV 5
This sets the profile velocity, or step rate, to 2^5 = 32 steps per second.

$EO_{nn}(:_x)$ Enter Offset correction table $_n$
Range : 1 to 255

This command is used to enter offset correction tables into the system. The system responds with a "O_{nn}:" prompt for the table entries. Each pair of entries in the table is the nominal position and the (signed) offset from that position. Note that the first and last entry points should have offset values of zero, and that there need to be at least 4 pairs of values. The system interpolates between the nominal positions to create a required offset. To end the table, make a blank entry by entering just a (CR), and the system then returns to normal operation. A table can be entered automatically by using the optional suffix of a ":" followed by a number (or "v" followed by a variable) which indicates the number of entry pairs to be loaded. These are copied from numeric variables starting from %0. Once the table has been entered in this way, the numeric variables can be used for other purposes. The correction table is accessed by means of the table number assigned by the user when it is entered.

Normally, the bounds would define the correction range, and typically a machine would have cyclic imperfections, the cycle being defined by the bounds. When this is not the case, a table can be created to go over a bounds by setting bit 14 of the control word for the current channel. In this case the position within a table is calculated by adding the current demand to the bounds × bounds counter plus the EZ zero offset value for the current channel. Note that the bounds counter needs to be zeroed at a known position for this mode to work correctly.

LO_{nn} List Offset correction table $_n$
Range : 1 to 255

This command is used to enter offset correction tables into the system. The system responds with a "O_{nn}:" prompt for the table entries. Each pair of entries in the table is the nominal position and the (signed) offset from that position.

XO_{nn}(F) Execute offset correction table $_{nn}$.
Range : 1 to 255

This command tells the system to execute an offset correction table number $_{nn}$. It should be noted that this is a local command, and several motor channels can individually execute the same global correction table. The optional F suffix allows the correction table to be disabled. Bit 10 of the LW link word can be set to use the absolute position (as viewed from DPG) to index into the table.

EZ_n Enter zero offset for EO offset table.
Range : 1 to 4 000 000 (4.0E6)
Default : 4 000 000

This command sets the value which is added to the current demand plus the bounds × bounds counter when bit 14 of the local control word is set. See the EO command above for more details.

7.2.9 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 will execute the following commands. The system returns a "W" status message to indicate that it is waiting. If the position specified in a wait for position command is outside the range of the previous move command, then the system gives an "O" error message to indicate that the position was out of range.

WT$_{nn}$ Wait for time.
Range : 0 to 2,147,483,647

This command tells the system to wait for the given time, in units of 1÷256 seconds, before proceeding to the next command.

Example : MA 2000/ WT 512/ MA 0
This command sequence tells the system to move to position 2000, wait there for 2 seconds, and then move to position 0.

WI$_{nn}$± Wait for input line $_{nn}$.
Range : 1 to 16×n (where n=No of control boards)

This command tells the system to wait until the specified input line goes to the specified state. Note that if the specified input line has been defined as some other function input, the system returns the "U" line already used error message.

Example : MA 5000/ WI 2 -/ MA 0
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.

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 an "O" error message to indicate that the position was out of range.

Example : SV 200/ MA 2000/ WA 1500/ SV 100
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.

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 the last position used in a command.

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.

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.

Example : RW 10100001 / WF / RW1 / SO3
This command string sets up the initial RW such that the reference input behaves as a fast ZC input. It then waits for the first reference input to be detected, and changes RW to the normal auto-correction setting. Finally an output line is set to indicate that the unit has initialised 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 it the bound overflow counter increments (or decrements) by the specified count 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.

WK($_{nn}$) Wait for keypad entry

This command sets the system into the wait state, until an value has been entered using the VK command (page 135). An optional parameter can be entered, which corresponds to a code for the particular key (see appendix, page 182). The system will then wait until the key referred to has been pressed.

WN Wait for network acknowledge response

This command sets the system into the wait state, until an acknowledge from a recently sent network command is received, or a time has elapsed corresponding to the network timeout (TN) parameter.

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>		

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 motor moves back to zero immediately.

WS_{nn} Wait until stopped before executing sequence.
Range : 1 to 255

This command sets up the system to execute sequence number $_{nn}$ as soon as the current channel has stopped. This state can be setup for more than one channel. Note that if motors are being mapped to others, only the master channel is considered to be moving. i.e. if WS parameters have been set up for motors which are mapping to others, the sequence(s) will be executed immediately. This command enables an input line to execute commands which are only legal when a channel has stopped, without having to perform involved tests. Note that once a WS sequence has been executed, the parameter is reset to zero, and has to re-enabled.

Example : DI3-/CH2/ST/WS 34/CH1

```
ES34
MI8/CH2/LM1
```

The system sets up input line so that when it goes to the negative state it executes stored sequence no. 34 after channel has stopped. Sequence 34 links the motion of channel 2 to channel 1

7.2.10 Error trapping

SE$_{nn}$ Set maximum position error. (restricted)
Range : 0 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 "G" error message, decelerates to a stop and enters the motor off state. The system must be returned to the position control mode before any further motion commands are accepted by the system. See section 7.2.3 (page 29) for details of the MO motor off and PC position control commands. The value is defined in user units.

Example : SE 500
This sets the maximum position error to 500 units.

TO$_{nn}$ Timeout. (restricted)
Range : 1 to 65535
Default : 32

This command sets a timeout value, in units of 1÷256 seconds. When a move command is executed, if the motor does not move for a period that exceeds the timeout, then the system will print a "T" error message and go to the motor off state. The system must be returned to position control mode before any further move commands are accepted.

Example : TO 512
This sets the timeout to 2 seconds.

LH±$_{nn}$ 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 "LH" 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.

LL$\pm_{nn}$ 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 "LL" 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.

Example : LH 10000/ LL 0
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.

RT$_{nn}$ Set reference timeout. (restricted)
Range : 0 to 255
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 bound position, and cleared each time a valid reference input is detected. If the counter reaches the "RT" value, the system gives the "RT" error message. The reference timeout function may be disabled by setting it to zero if it is not required.

EW$_{bb}$ Set local error options word. (restricted)
Range : 0000 0000 to 1111 1111 (binary)
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.
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.
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 correction overrun error is treated as a user error, and the system simply prints an error message.
3	Reserved for future expansion.
4	Reserved for future expansion.
5	Reserved for future expansion.
6	Reserved for future expansion.
7	Reserved for future expansion.

LE Display last error

This command redisplays the error message for the last error detected by the system. It is useful for finding an error message which has stopped the system when there is not normally a display connected to the machine, or to display the long error message for an error which has been reported with a short error message. This is done by setting bit 5 of DW before executing LE. If The "O" (output to variable) option is used, then a channel number (most significant 16 bits) followed by and error number (least significant 16 bits) is transferred to that variable. This can be interpreted by interrogating the variable.

EG_{bb} Set global error options word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0000 0000 0000 0000

This command allows the user to write a value into the global error options word. 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 set	Bit cleared
bit	0	No error messages sent	Error messages sent
	1	DE command responds to all errors	Only responds to motor errors
	2	Reserved for future expansion	
	3	Reserved for future expansion	
	4	Reserved for future expansion	
	5	Reserved for future expansion	
	6	Reserved for future expansion	
	7	Reserved for future expansion	
	8	Reserved for future expansion	
	9	Reserved for future expansion	
	10	Reserved for future expansion	
	11	Reserved for future expansion	
	12	Reserved for future expansion	
	13	Reserved for future expansion	
	14	Reserved for future expansion	
	15	Reserved for future expansion	

7.2.11 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.

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

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

The dynamic behaviour of the system depends on these gain constants, and on the mechanical characteristics of the system being controlled. Tuning the control system to get the best performance on a particular mechanical setup requires setting up these gain constants. Some of the gain terms in the control algorithm may be disabled by setting appropriate bits in the control word. For more details see the CW command description below.

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

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

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

If bit 15 of the control word (CW) is set, the scaling changes as follows:

$$V_{out} = Err \times \frac{KP}{65536} \times \frac{10}{2048}$$

The performance of the system may be monitored by means of the auxiliary analogue output channel. 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.

CW$_{bbbb}$ Set control word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0000 0000 0100 1011

This command allows the user to write a value into the system control word for the current motor channel. Note that the leading zeros may be omitted. The control word allows various components of the demand signal to be disabled if required, and allows the sense of the encoder input and of the command signal 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 set	Bit cleared
bit	0	Enable integral control	Disable integral control
	1	Enable velocity feed-forward control	Disable velocity feed-forward control
	2	Enable differential control	Disable differential control
	3	Enable velocity feedback control	Disable velocity feedback control
	4	DAC output/stepper direction reversed	DAC output/stepper direction normal
	5	Encoder input reversed	Encoder input normal
	6	Power up in motor off state	Power up in position control mode
	7	Integral term active when static only	Integral term active continuously
	8	Sine profile moves	Trapezoidal profile moves
	9	Stepper encoder monitor	No stepper encoder monitor
	10	Stepper motor control	Servo motor control
	11	Disable position feedback	Enable position feedback
	12	Enable bounds checking	Disable bounds checking
	13	Enable measured pos for SN	Enable demand pos for SN
	14	Bound count for offset table	Normal offset table
	15	Divide gain factors by 256	Normal gain factors

The default control word value of 00000000 01001011 allows integral, velocity feedback, and velocity feed-forward control, and the channel powers up in the motor off state.

Example : CW 0000 0000 0000 0001
This enables proportional and integral control only.
(Proportional control is always enabled.)

The stepper motor options for channels one and two each use two of the isolated output lines. See the hardware specification for full details.

WD_{nn} Set stepper direction delay. (restricted)
Range : 0 to 255
Default : 10

This command sets the delay (in units of 1÷256 second) between changing the stepper direction line and starting motion pulse outputs.

LV_{nn} Set lower stepper resonance velocity bound. (restricted)
Range : 0 to 800 000 (8E5)
Default : 0

This command sets the lower bound for the stepper stepper resonance zone. This zone defines a range of velocities which are never used due to electrical and/or mechanical resonance in a stepper system. It should be used in conjunction with the UV command, and the parameter for LV cannot be set higher than the current UV parameter.

UV_{nn} Set upper stepper resonance velocity bound. (restricted)
Range : 0 to 800 000 (8E5)
Default : 0

This command sets the upper bound for the stepper stepper resonance zone. This zone defines a range of velocities which are never used due to electrical and/or mechanical resonance in a stepper system. It should be used in conjunction with the LV command, and the parameter for UV cannot be set lower than the current LV parameter.

KP_{nn} 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.

KI_{nn} 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 1.

KD_{nn} 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. In practice, if the system is anywhere near correct tuning, the position error is small and the rate of change of error is smaller still, so that the differential gain only has a limited effect on the system.

KV_{nn} 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 to 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.

KF_{nn} Set velocity feed-forward gain constant. (restricted)
Range : 0 to 65535
Default : 0

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

IT_n Set integration time constant. (restricted)
Range : 0 to 2
Default : 0

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

Code	Time constant	Scale factor
0	1÷256	256
1	1	1
2	256	1÷256

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

SF_n Set monitor output function. (restricted)
Range : 0 to 8
Default : 0

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

Code	Function	Associated gain term
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	
7	Absolute measured position	
8	Global output value	AO

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. Code number 8 allows the user to set up an independently calculated value. This can then be sent out using the AO command.

AO$_n$ Set analogue output to global value.
Range :0 to ± 32767
Default : 0

This command selects a particular global value to output on the auxiliary output channel. The first parameter defines the notional channel number, and the second defines the value of the output. Note that the D to A convertor has a resolution of 12 bits, and that actual values will be rounded to the nearest digit. The actual output voltage will depend on the KM and OM values. Whilst this command is channel dependent, by using the ^ character, the channel can be pre-defined using the CHB command. This then allows the global use of this command, regardless of the current active motor channel.

KM$_{nn}$ 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. The monitor output signals are also independent of the settings of the gain term enable bits in the control word.

OM±$_{nn}$ 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

7.2.12 Digital inputs and outputs

Revision E and later versions of the PC3/100 host control board have several i/o options. The basic system has 8 inputs and 8 outputs. This can be expanded to 16 inputs and 16 outputs. Alternatively, the system can be set to 24 inputs and 8 outputs. In this configuration, please note that inputs and outputs are mixed on connector CN7. If in doubt about the current hardware setup, use the BH command to give a breakdown of the current hardware setup.

SO$_{nn}$ Set output line $_{nn}$.
Range : 1 to 16×n (where n=No of control boards)

This command sets up the specified output line to a logic high. Note that the output state is maintained until superseded by another command for the same output line. If no output line number is set then all available output lines are set to a logic high.

Example : SO 11
This sets up line 11 to a logic high.

CO$_{nn}$ Clear output line $_{nn}$.
Range : 1 to 16×n (where n=No of control boards)

This command clears the specified output line to a logic low. Note that the output state is maintained until superseded by another command for the same output line. If no output line number is set then all available output lines are set to a logic low.

Example : CO 7
This sets up line 7 to a logic low.

CM$_{nn}$ Complement output line $_{nn}$.
Range : 1 to 16×n (where n=No of control boards)

This command complements the current state of the specified output line. Note that the output state is maintained until superseded by another command for the same output line. If no output line number is set then all available output lines are complemented.

Example : CM 5
If line 5 is initially logic high, this command sets up line 5 to a logic low. Likewise, if line 5 is initially logic low, this command sets up line 5 to a logic high.

SY$_{nn}$ Set node $_{nn}$.
Range : 1 to 32

This command sets up the specified node to a logic high. If no node number is set then all available nodes are set to a logic high.

Example : SY 15
This sets up node 15 to a logic high.

CY_{nn} Clear node $_{nn}$.
Range : 1 to 32

This command clears the specified node to a logic low. If no node number is set then all available nodes are set to a logic low.

Example : CY 17
This sets up node 17 to a logic low.

$FO_{nn\pm}$ Flash output line $_{nn}$.
Range : 1 to 16×n (where n=No of control boards)

This command sets up the specified output line to "flash" on and off at a rate defined by the "FR" flash rate command. Note that the output state is maintained until superseded by another command for the same output line. If no output line number is set then all available output lines are set to a logic high.

Example : FO 28+/ FO3-
This sets up line 28 to flash at a rate defined by the FR command. At the same time, line 3 flashes in opposition to output line 3 (i.e. when line3 three is on, line 28 is off, and vice-versa).

FR_{nn} Set flash rate $_{nn}$ for output lines.
Range : 1 to 65535

This command sets up the flash rate in ticks (1÷256 sec). It is used in conjunction with the "FO" flash output line command. Note that the time defined is for the ON period, and that the total flash cycle time is twice the time defined.

Example : FR 256
This sets up the flash output timer to have a cycle time of 2 seconds.

OC_{nn} Output code via expanded output group.
Range : 0 to maximum value possible on defined output group.

This command sets the expanded output line group (as defined by the OX command) to the given code data value. If the group was defined as active low, the data is inverted. 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 this command is used when there is no output group defined, and "E" error message is returned. If the parameter value given cannot be represented as a binary number with the number of lines in the output group, then the "O" error message is returned.

Example : OC 5
This sets the output group lines to the binary value 0101 (=5_{10}).

RI$_{nn}$ Read input line (s).
Range : 1 to 16×n (where n=No of control boards)

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 in the command, the system displays the current state of all available input lines. If the parameter is replaced by "O" followed by a variable name, and some input lines have been set up for variable input (using the VI command), then the named variable will be set to a number corresponding to the binary input.

Example : RI 3
This reads the state of input line 3, and prints it on the display.

```
System          User            Comments
1>              RI3             Read a particular input
0                               Input line 3 is low
1>                              Normal prompt
```

Example : VI2/6+; RIOB
The first command (VI...) sets up the inputs so that lines 2 to 6 will form a 5-bit binary number, with positive inputs giving 1's in the resultant binary number. The second command (RIOB) reads the state of input lines 2 to 6, and puts the result in variable B.

Example : RI
This reads the state of all 8 input lines, and displays them.

```
System          User            Comments
1>              RI              Reads all inputs
 1  0                           Line 1 is low
 2  1                           Line 2 is high...
 3  1
 4  0
 5  0
 6  0
 7  1
 8  0
1>                              Normal prompt
```

RO$_{nn}$ Read output line state(s).
Range : 1 to 16×n (where n=No of control boards)

This command reads the current state of the specified output line, reads its current state, 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 in the command, the system displays the current state of all available output lines.

Example : RO 6
This reads the state of output line 6, and prints it on the display.

```
System          User            Comments
1>              RO6             Read a particular output
0                               Output line 6 is low
1>                              Normal prompt
```

Example : RO
This reads the state of all output lines, and displays them.

```
System          User            Comments
1>              RI              Reads all outputs
 1  1                           Line 1 is high
 2  0                           Line 2 is low...
 3  1
 4  0
 5  1
 6  1
 7  1
 8  0
1>                              Normal prompt
```

RY$_{nn}$ Read node state(s).
Range : 1 to 32

This command reads the current state of an imaginary node. Nodes can only be set and cleared using logic equations. The current state is printed as a "0" or "1" on the display. A "0" represents a logic low, and a "1" represents a logic high. If no node number is given in the command, the system displays the current state of all nodes.

IY$_{nn}$± If node true do command line.
Range : 1 to 32

This command allows the programmer to specify that a command or command line is conditional on the current state of an imaginary node. Nodes can only be set and cleared using logic equations. If the node specified in the IY command is in the specified state (the condition is true), then the remainder of the command line is executed. If the node is not in the specified state, the remainder of the command line is skipped, and execution proceeds to the next line of input. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on input line states.

II$_{nn}$± If Input true do command line.
Range : 1 to 16×n (where n=No of control boards)

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. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on input line states.

Example : DP/II6-/MR20000/SO2
This displays the current position, and if input line no 6 is negative, the system moves 20000 counts, and sets output line 2. If input line 6 is positive, the current position is displayed, and the remainder of the line is ignored.

IO$_{nn}$± If Output true do command line.
Range : 1 to 16×n (where n=No of control boards)

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. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on output line states.

Example : DP/IO3+/MR5000/CM6
This displays the current position, and if output line no 3 is positive, the system moves 5000 counts, and complements output line 6. If output line 3 is negative, the current position is displayed, and the remainder of the line is ignored.

IY$_{nn}$± If node true do command line.
Range : 1 to 32

This command allows the programmer to specify that a command or command line is conditional on the current state of an imaginary node. 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. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on input line states.

Example : DP/II6-/MR20000/SO2
This displays the current position, and if input line no 6 is negative, the system moves 20000 counts, and sets output line 2. If input line 6 is positive, the current position is displayed, and the remainder of the line is ignored.

MI$_{nn}$ Mask function input.
Range : 1 to 16×n (where n=No of control boards), or no parameter

This command is used to disable the action of defined function inputs or any expanded input group lines. It allows several input lines to selectively lock out defined actions, depending on the current function activated. For example, a machine start sequence assigned to a function input may disable itself once the machine has started, until the stop sequence assigned to another sequence 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. Disabled inputs are enabled again by the EI command. If a line number is given as a parameter, then the specified line is disabled. If no line number is given, then all function inputs and/or expanded group inputs are disabled.

EI$_{nn}$ Enable function input.
Range : 1 to 16×n (where n=No of control boards), or no parameter

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

NOTE: The disable/enable action applies **only** to inputs defined as function inputs with the DI command, or as expanded group inputs with the DX command. Input lines which are not currently defined for any particular function may be masked or enabled, but this has no effect unless the lines are subsequently defined as DI or DX inputs. It is not possible to disable other types of defined inputs with the MI command, or to inhibit the WI command.

MK$_{nn}$ Mask keypad input.
Range : A to Z, or no parameter

This command is used to disable the action of certain keys on the keypad. A table of parameters to define which keys are to be masked can be found in section 12.3 (page 183). If no parameter is entered, then all the above keys are masked.

NOTE: The keys which are controlled by this command have been changed in software version 154.9.

EK$_{nn}$ Enable keypad input.
Range : A to Z, or no parameter

This command is used to enable the action of certain keys on the keypad. It allows the keypad to start events, in conjunction with the SKn command (page 53) at any time. A table of parameters to define which keys are to be enabled can be found in section 12.3 (page 183). If no parameter is entered, then all the above keys are enabled.

NOTE: The keys which are controlled by this command have been changed in software version 154.9.

RK$_{nn}$ Read keypad input.
Range : A to Z

This command is used to read the state of a keypad input A-Z (as defined above in the EK command). It returns 1 when the specified key is depressed, and 0 when it is released. This output can be re-directed to a variable with the O suffix. It can be used in conjunction with an SK sequence to distinguish between the press and release function. A table of parameters to define which keys are to be read can be found in section 12.3 (page 183).

DI$_{nn}$±/! Define input line function. (restricted)
Range : 1 to 16×n (where n=No of control boards)

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. Alternatively the sign may be replaced by a "!" character. This indicates the action when a previously defined input line reverts to its normal 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. The length of each definitions is limited to 80 characters. If there is not enough memory for the new line definition, the system will return an "N" no room error message. Note that a line that has been defined as a function input cannot be set or cleared. A function input line may be returned to normal operation by entering this command without the sign and the function text. This command is restricted, and is only available in privileged mode.

If a line is defined as an input, and 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 : DI1+/ AB
This defines line 1 as a single command, such that when line 1 goes to a logic high, the system executes the AB command.

Example : DI2-/ ID/IN-/WT256/MR-5000/ZC
DI2!/ST
This defines line 2 as a command sequence, such that when it goes to a logic low, the system executes the given sequence. It initialises the DAC offset, initialises the position to the reference position, waits for 1 second, moves -5000 units, and zeros the position counters at this position. When the input line reverts to a logic high, the ST command is executed. If the system is still in motion, it will then stop.

Example : DI 3
This returns line 3, previously defined as a function input, to normal operation.

DIL$_{nn}$± Define input line to be latched. (restricted)
Range : 1 to 16×n (where n=No of control boards)

This command defines a specified input line to be latched next time it goes to the state defined by the ± parameter. The latch can be read using the RIL command. When the DIL command is executed, the RIL state is set to the opposite sign as the DIL parameter. At the first transition to the sign defined by the DIL parameter the RIL state follows to this sign. The RIL state is reset at the next occurrence of a DIL command. If DIL is executed without a sign parameter, it removes that line from a latch definition, and hence the MID command will not have any effect on it.

RIL$_{nn}$± Read input line latch.
Range : 1 to 16×n (where n=No of control boards)

This command reads an input line latch which must have been set up using the DIL command, as described above.

MID	Mask input line latch. Range : 1 to 16×n (where n=No of control boards) This command masks any input lines which have DI strings associated, and have also been defined to be latched using the DIL command. It differs from the MI command in that a pending DI command string will be executed after the EID command has been executed. Pending DI strings will executed in the order of receipt of signal, if there is more than one DI string waiting to be executed,
EID	Enable input line latch. Range : 1 to 16×n (where n=No of control boards) This command enables any input lines which have DI strings associated, and have also been defined to be latched using the DIL command. It differs from the EI command in that a pending DI command string will be executed after the EID command has been executed. Pending DI strings will executed in the order of receipt of signal, if there is more than one DI string waiting to be executed,

7.2.13 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 marker and use it to automatically adjust the absolute position of the system. The position of the reference input 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. For more details on the SB set bound command, refer to section 7.2.5 (page 45).

RW_{bb} Set reference options word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary)
Default : 0000 0001 0000 0000

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.
When set to 0, disables the position correction but still allows the measurement of reference error.

1 When set to 1, limits the position correction to a maximum value set by the SR command, if the reference correction is enabled by setting RW bit 0 to a 1.
When set to 0, allows the full correction to be made regardless of the maximum value set by SR.

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

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

4 Reserved for future expansion.

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.

6 This bit defines the action taken if the reference error is greater than the maximum value set by the SR command and bit 1 of RW is set to enable this limit.
When set to 1, if the reference error is greater than the maximum value set by the SR command, the system corrects by this maximum value. The "RL" reference out of limits error is reported, and may set the channel to the motor off state if required.
When set to 0, a reference error greater than the maximum value is ignored completely and its value is discarded. This also inhibits the reference out of limits error message, and does not update the reference error displayed by the DF command.

7 When set to 1, the system always sets the position to zero on detecting a reference signal.
When set to 0, the system defines the reference position or ±SB, whichever is closest. 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.

8 When set to 1, the system always uses positive numbers when displaying its position. This means, that while the system is moving in the negative direction past the zero position, it will wrap to the current bounds value set using the SB command.
When set to 0, the system will display negative numbers as it passes the zero position moving in the negative direction. It will only wrap when it reaches either the positive or negative bounds value.

9 When set to 1, the system allows the crossing of bounds for an absolute move.
When set to 0, the system always stays within the current bound setting. The practical use of this bit is when a system has had its bounds set to a machine cycle, and it is necessary to move the shortest distance. By setting bits 8 and 9, the system will always take the shortest route for an absolute move.

10 When set to 1, enables the position correction on detecting a snapshot signal.
When set to 0, disables the position correction but still allows the measurement of snapshot position.

11 When set to 1, limits the position correction to a maximum value set by the PR command, if the reference correction is enabled by setting RW bit 10 to a 1.
When set to 0, allows the full correction to be made regardless of the maximum value set by PR.

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

13 Reserved for future expansion.

14 This bit defines the action taken if the snapshot error is greater than the maximum value set by the PR command and bit 10 of RW is set to enable this limit.
When set to 1, if the snapshot error is greater than the maximum value set by the PR command, the system corrects by this maximum value. The "RL" reference out of limits error is reported, and may set the channel to the motor off state if required.
When set to 0, a reference error greater than the maximum value is ignored completely and its value is discarded. This also inhibits the reference out of limits error message, and does not update the snapshot error displayed by the DS command.

15 Reserved for future expansion.

Example : RW 11/ SR20
This enables the position correction on detection of a reference input, limits the allowed correction to a maximum of 20 encoder counts.

Example : RW 10100001
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 the ZC command, but is effective on the fly at any time.

SJ±$_{nn}$ Set deferred adjustment position. (restricted)
Range : ± 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 SJ command defines a position which the motor must pass before the correction due to the reference signal takes place. 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 SJ position is set to a value which is greater than the bound (set by SB), the reference correction will never take place.

SR$_{nn}$ Set maximum reference correction. (restricted)
Range : 0 to 65535
Default : 65535

This command, when enabled by bit 1 of RW, limits the maximum allowed reference correction to the specified number of encoder counts. It may be used to eliminate false reference signals at positions far away from the expected reference position, or to allow the position reference facilities to be used even when the machine cycle length is not the same as the distance between reference marker signals.

When a reference signal is seen, the reference error is calculated as the difference between the zero position defined by the reference input, and the zero position or nearest bound position as measured by the normal system encoder counters. If enabled by bit 0 of RW and inside the limit defined by the SR command, the position is corrected by this reference error. If the reference error is greater than the maximum limit, the action taken depends on bit 6 of RW. If it is clear, then the position is not corrected, the out of limits error reference value is discarded, and the reference is ignored completely. If it is set, the position is corrected by an amount equal to the maximum correction limit.

If required, the system may be programmed to generate a motor error when the reference error is outside this limit. This facility is enabled by setting bit 1 of EW, the error options word. When this is set, a reference error greater than the reference set by SR gives a "RL" reference limit error message, and the channel goes to motor off. Note that the reference error and the out of limits error are only reported if RW bit 6 is set to 1 to enable correction by up to the defined limit. If RW bit 6 is set to 0, then reference errors outside this maximum limit are discarded and ignored completely, and no error is reported.

RF±$_{nn}$ Set reference offset. (restricted)
Range : ± 4 000 000 (4.0E6)
Default : 0

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

Example : CH1/RF500
This sets the reference offset on channel 1 to 500 counts. This means that the position where the reference input signal is seen is defined as absolute position 500, and not zero.

RV$_{nn}$ 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 unnecessary 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÷256 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 "RO" 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.

RL$_{nn}$ Set reference repeat length. (restricted)
Range : 0 to 4 000 000 (4.0E6)
Default : 0

This command sets the reference repeat length for the current motor channel. This is the position at which the system 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.

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 for 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 then gives a useful reference error measurement which may be used for correction if required.

PJ±$_{nn}$ Set snapshot deferred adjustment position. (restricted)
Range : ± 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 PJ command defines a position which the motor must pass before the correction due to the reference signal takes place. 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 PJ position is set to a value which is greater than the bound (set by SB), the reference correction will never take place.

PR$_{nn}$ Set maximum snapshot reference correction. (restricted)
Range : 0 to 65535
Default : 65535

This command, when enabled by bit 1 of RW, limits the maximum allowed reference correction to the specified number of encoder counts. It may be used to eliminate false reference signals at positions far away from the expected reference position, or to allow the position reference facilities to be used even when the machine cycle length is not the same as the distance between reference marker signals.

When a reference signal is seen, the reference error is calculated as the difference between the zero position defined by the reference input, and the zero position or nearest bound position as measured by the normal system encoder counters. If enabled by bit 0 of RW and inside the limit defined by the VP command, the position is corrected by this reference error. If the reference error is greater than the maximum limit, the action taken depends on bit 6 of RW. If it is clear, then the position is not corrected, the out of limits error reference value is discarded, and the reference is ignored completely. If it is set, the position is corrected by an amount equal to the maximum correction limit.

If required, the system may be programmed to generate a motor error when the reference error is outside this limit. This facility is enabled by setting bit 1 of EW, the error options word. When this is set, a reference error greater than the reference set by VP gives a "RL" reference limit error message, and the channel goes to motor off. Note that the reference error and the out of limits error are only reported if RW bit 6 is set to 1 to enable correction by up to the defined limit. If RW bit 6 is set to 0, then reference errors outside this maximum limit are discarded and ignored completely, and no error is reported.

SG$_{nn}$ Set snapshot goal value. (restricted)
Range : 0 to current bounds setting
Default : 0

This command converts the DS snapshot value from an actual position to an error with reference to a goal value. This will automatically correct for bounds transitions and makes the snapshot averaging more useful. For example, if a bounds is set 2000, a desired position is 34, and the snapshot shows 1995, the DS will show 1995 if SG is 0. If SG is 34, then DS will show 39. Note that bit 8 of the reference word must be set for this to work correctly (positive positions only).

PF±$_{nn}$ Set snapshot reference offset. (restricted)
Range : ± 4 000 000 (4.0E6)
Default : 0

This command sets the offset for the snapshot reference position of the current motor channel. It defines the absolute position for the snapshot reference input signal.

Example : CH1/PF500
This sets the reference offset on channel 1 to 500 counts. This means that the position where the reference input signal is seen is defined as absolute position 500, and not zero.

BS$_{nn}$ Set no of bounds for deferred snapshot correction.
Range : 0 - 5
Default : 0

This command sets a number of bounds delay before implementing a snapshot correction.

VP$_{nn}$ Set snapshot reference correction velocity. (restricted)
Range : 0 to 8
Default : 0

This command sets the correction speed for any snapshot reference error. It is used to make large reference error corrections less harsh by spreading the correction over several time steps. If VP is set to zero, then the reference error correction is performed immediately in one step. If VP 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 VP. This means that the reference correction speed scales automatically with the machine speed, such that the value of VP may be chosen for correct operation at full machine speed without causing unnecessary quick corrections at lower machine speeds.

At the maximum value of VP=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 VP is reduced by one, the correction speed is halved, down to the minimum value of VP=1, when the correction speed is 1÷256 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 "RO" 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 VP is too small and should be increased.

7.2.14 Configuration commands

DR$_n$± Define reference input. (restricted)
Range : 1 to 4

This command defines the sense of the position reference input for the current motor channel. The system looks for the specified change in the reference input when the IN initialise position command is executed, and when the automatic reference facilities are enabled by the RW command. See the Reference Commands section (7.2.13, page 90) for more details on the reference facilities and commands. This command is restricted, and is only available in privileged mode.

Example : DR 2+
This specifies that the reference position is detected by a transition on input line no 1 for the current channel.

DZ$_n$±(D) Define zero marker input on/off. (restricted)
Range : 0 to 2

This command defines whether the encoder zero marker input is on or off. The sense of 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 one, the zero marker input is routed through the Quart device. If it is two, the VIA device is used, producing a higher priority (and hence higher response time). 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 RW command. The optional D suffix displays the last zero marker reference position error. For more details of the operation of the reference inputs, see the Reference Commands section (7.2.13, page 90). This command is restricted, and is only available in privileged mode.

EC$_{nn}$± Define external counter input. (restricted)
Range : 1 to 4

This command an input line to be an external counter, whose frequency must not be greater than 100Hz. The counter has a maximum value defined by the WX parameter. The counter subsequently wraps to zero. The counter can be read at any time by the use of the RX command. The external counter function uses the same mechanism as the reference input function to get an accurate count on an input signal. An EC 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 : EC 1-
This specifies that the input line 1 is to be an external counter.

RX Read external counter input.

This command reads an external counter value.

WX_{nn} Wrap for external counter input.
Range : 1 to 2,147,483,647
Default : 2,147,483,647

This command sets the value at which an external counter wraps to zero.

$ZX[_{nn}]$ Zero external counter input or set counter input.
Range : 0 to 2,147,483,647

This command reads sets an external counter value to zero.

If a value is given, the system sets the current external counter to the given (absolute) value. If no value is given, it sets the external counter to be zero. The ZX command may be used at any time.

$AA_a:_{nn}$ Analogue input averaging term.
Range($_a$): 1 to 2
Range($_{nn}$): 0 to 12

This command is used in conjunction with the AL and AG commands. It provides a rolling sum on analogue input no $_a$. The number of terms in the sum is 2^{nn}, and so this also provides a gain of 2^{nn}. The output of this sum is then multiplied by the AG gain constant, before being converted to a position.

AI_{nn} Read analogue input.
Range : 1 to 4

This command reads an external analogue input. Variables can be fed with the analogue input value by using a colon delimiter.

Example : AI2:g
This sets variable g to the current analogue input on channel 2

AG_{nn} Set analogue input gain constant. (restricted)
Range : 0 to 1048575
Default : 1

This command sets the analogue input proportional gain of the system. It is to be used in conjunction with the AL command. Note that this parameter is local to a particular motor channel.

AL_n Set analogue input to link to motion position.
Range : 1 to 2

This links the external analogue input on analogue channel number $_n$ to the demand position on the current axis. Note that this must be defined with the AW command. When the command is executed, the current analogue input is defined as the current demand position. The analogue gain constant (AG) is used to define the gain between the analogue input and the demand position. More than one axis may be linked to an analogue input channel.

ALV(-)$_n$ Set analogue input to link to motion velocity.
Range : 1 to 2

This links the external analogue input on analogue channel number $_n$ to the demand velocity on the current axis. Note that this must be defined with the AW command. When the command is executed, the current analogue input is defined as the current velocity. The analogue gain constant (AG) is used to define the gain between the analogue input and the velocity. More than one axis may be linked to an analogue input channel. The optional - sign allows the sense of the analogue input signal to be negated.

AU$_n$ Unlink analogue input from motion.
Range : 1 to 2

This unlinks the external analogue input $_n$ from motions which have been set up using the AL and AW commands.

AW$_{bb}$ Set analogue input link word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0

This command allows the user to write a value into the global analogue input link word. Note that the leading zeros may be omitted. The analogue input word defines which axes will be linked to analogue input channel 1 after an AL analogue link command axis move operation. The analogue input link word bit functions are described below.

		Bit set	Bit cleared
bit	0	Link ch 1 to analogue i/p ch1	No link to analogue channel 1
	1	Link ch 2 to analogue i/p ch1	No link to analogue channel 1
	2	Link ch 3 to analogue i/p ch1	No link to analogue channel 1
	3	Link ch 4 to analogue i/p ch1	No link to analogue channel 1
	4	Link ch 5 to analogue i/p ch1	No link to analogue channel 1
	5	Link ch 6 to analogue i/p ch1	No link to analogue channel 1
	6	Link ch 7 to analogue i/p ch1	No link to analogue channel 1
	7	Link ch 8 to analogue i/p ch1	No link to analogue channel 1
	8	Link ch 1 to analogue i/p ch2	No link to analogue channel 2
	9	Link ch 2 to analogue i/p ch2	No link to analogue channel 2
	10	Link ch 3 to analogue i/p ch2	No link to analogue channel 2
	11	Link ch 4 to analogue i/p ch2	No link to analogue channel 2
	12	Link ch 5 to analogue i/p ch2	No link to analogue channel 2
	13	Link ch 6 to analogue i/p ch2	No link to analogue channel 2
	14	Link ch 7 to analogue i/p ch2	No link to analogue channel 2
	15	Link ch 8 to analogue i/p ch2	No link to analogue channel 2

DL$_{nn}$± Define limit switch input. (restricted)
Range : 1 to 16×n (where n=No of control boards)

This command defines the specified input line as a limit switch input for the current motor channel. 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 an "L$_n$" error message to indicate that a limit switch input was detected, and goes to the motor off state. Note that a line that has been defined as a limit switch input cannot be set or cleared. A line which has been defined as a limit switch input may be returned to normal operation by entering this command without the sign. Note that only lines 7 to 14 may be defined as limit switch inputs, and not lines 1 to 6. This command is restricted, and is only available in privileged mode.

Example : DL 8 -
This defines input line 8 as an active low limit switch input. The system detects a limit switch when line 8 goes to a logic low.

Example : DL 4
This returns line 4, previously defined as a limit switch input, to normal operation.

PS$_n$± Define position snapshot input. (restricted)
Range : 1 to 4

This command defines the specified input line as a position snapshot input for the system's current motor channel. 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. A position snapshot input line may be returned to normal operation by entering this command without the sign. A single input line can be defined as a position snapshot for different motor channels. This command is restricted, and is only available in privileged mode.

Example : CH2/PS 4 +
This defines that the snapshot position for motor channel 2 is detected on a low-to-high transition on input line 4.

WPL/U Define window for position snapshot input. (restricted)
Range : 1 to 4 000 000 (4.0E6)
Default : 0

This command defines a window for the system to scan for a position snapshot input. WPL defines the lower limit and WPU defines the upper limit. The system checks every 100µs to see if the system in within the position window. If this condition is matched, then the interrupt mechanism is enabled for the defined window.

DI$_{nn}$±/! Define input line function. (restricted)
Range : 1 to 16×n (where n=No of control boards)

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. Alternatively the sign may be replaced by a "!" character. This indicates the action when a previously defined input line reverts to its normal 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. The length of each definitions is limited to 80 characters. If there is not enough memory for the new line definition, the system will return an "N" no room error message. Note that a line that has been defined as a function input cannot be set or cleared. A function input line may be returned to normal operation by entering this command without the sign and the function text. This command is restricted, and is only available in privileged mode.

If a line is defined as an input, and 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 : DI1+/ AB
This defines line 1 as a single command, such that when line 1 goes to a logic high, the system executes the AB command.

Example : DI2-/ ID/IN-/WT256/MR-5000/ZC
DI2!/ST
This defines line 2 as a command sequence, such that when it goes to a logic low, the system executes the given sequence. It initialises the DAC offset, initialises the position to the reference position, waits for 1 second, moves -5000 units, and zeros the position counters at this position. When the input line reverts to a logic high, the ST command is executed. If the system is still in motion, it will then stop.

Example : DI 3
This returns line 3, previously defined as a function input, to normal operation.

DQ$_{n}$± Define multi-axis stop input. (restricted)
Range : 1 to 4

This command defines the specified input line as a trigger for a multi-axis stop for the system. It is used in conjunction with the QS commands. The system monitors the specified input and stores the absolute position value for every channel whose QS parameter is greater than zero at that time. The system then stops each channel at a relative offset position defined by the local QS parameter. This allows the system to stop at a precise position with reference to an external position, and can be useful with machines which are passing a material with a reference mark printed on, and a stop position is required at a fixed offset from the reference mark. The multi-axis stop function uses the same mechanism as the reference input function to get an accurate measurement of position on an input signal. A DQ 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 : DQ 3 -
This defines that the multi-axis stop is performed on a high-to-low transition on input line 3.

DH$_{nn}$±±/$_{s1}$/$_{s2}$ Define inching input. (restricted)
Range : 1 to 16×n (where n=No of control boards)

This command defines a specified input line to provide an inching move, as defined by the IL inch distance parameter. The first sign specifies the active state of the input, such that the system executes the function when the input changes to the specified state. The second sign defines the direction of motion with reference to the inch distance. S1 specifies a starting sequence, to precede the inch move; S2 specifies a sequence which is executed when the inch move has completed. If either or both the starting and finishing sequence are not required, a null sequence(0) should be entered. If the input line is held in an active state for longer than the IP pause time, then the axis will move continuously until the button is released.

Example : DH 8 -+/5/7
This defines Input line 8 as an active low inch switch input. The system will execute sequence 5, followed by an inch move of the amount defined in the IL command, followed by sequence 7, when line 8 goes to a logic low.

DX$_{nn}$± Define expanded input line. (restricted)
Range : 0 to 7

This command sets up a group 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 is used as a strobe or trigger input. The sense of the strobe input is given by the sign after the parameter. Lines 8 downwards are used for the expanded input function. To reset the expanded input group use the command "DX0".

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. This value is used as the number of a sequence, which is executed immediately if it is defined. This allows a maximum of 127 possible different operations to be controlled by lines 1 to 7 on a single controller. If the strobe input is active low, as defined by the sign in the command, then the data lines are also inverted when deriving the number of the sequence to be executed. This keep the strobe input and data inputs consistent.

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

Example : DX 4-
This sets up an active low expanded input group on lines 4 upwards, as above. When a strobe signal is detected on line 8, the unit looks for a sequence number derived from the other input lines in the group, and if it is defined, then it is executed. Thus if the input lines 7-4 are in the binary pattern 0111 when the strobe input is seen, then sequence 7 is executed ($0111_2=7_{10}$).

Line	State	Bit Value	Decimal Value
8	High->low		
7	High	0	0
6	Low	1	4
5	Low	1	2
4	Low	1	1
3			Total 7
2			
1			

$SI_n\pm$ Set input to stop current motor channel. (restricted)
Range : 1 to 4

This command defines the specified input line as a way of stopping the current channel. The sign defines which logic transition is used to initiate the stop. The system monitors the defined input and starts to decelerate **immediately** the specified change of state takes place. This command is useful when a repeatable distance between seeing a input and coming to rest is required. It is therefore preferable to using a "DI string" for this purpose. The SI function uses the same mechanism as the reference input function to get an accurate measurement of position on an input signal. An SI 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 : SI1-
This defines that if the motor in the current channel is moving, it will decelerate to stop on a high-to-low transition on input line 1.

DB$_{nn}$ Set input debounce time. (restricted)
Range : 0 to 255
Default : 5

This command sets up a debounce time for all the digital inputs. It is specified in 1÷256 second (about 4ms) ticks. 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 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 : DB 2
This sets the debounce time to about 8 ms (2 samples).

DE$_{n}$± Define error output line. (restricted)
Range : 1 to 16×n (where n=No of control boards)

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. Error conditions which are signalled in this way are as follows.

(a) Exceeding maximum position error.
(b) Exceeding maximum timeout.
(c) Detection of a limit switch input.
(d) Motor position outside position limits.

Any optional errors enabled by setting bits in the error word EW also cause the error output signal to switch. More details of the error conditions and commands to set them up are given in section 7.2.10 (page 71). This command is restricted, and is only available in privileged mode.

If bit 1 of the global error word EG is set, the line is set to the specified state when the system detects any error condition, and is cleared on receiving the next valid command.

Example : DE 4+
This sets up an active high error output signal on output line 4. When an error condition is detected, output line 4 is set to a logic high.

PO$_n$± Define position trigger output. (restricted)
Range : 1 to 16×n (where n=No of control boards)

This command defines the specified output line as a position trigger output. The PO command **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.

Example : PO5-: 1000: 2000
This example sets up an output as a position trigger, such that it goes low between positions 1000 and 2000.

Variables can be used to define position trigger points.

```
Example: IV%34:245/IV%43:900
         PO2+:v%34::v%43:
```

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 that 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$_n$± Define expanded output line group. (restricted)
Range : 0 to 8 (or 16 if slave board(s) present)

This command sets up a group of outputs 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 output group, and the sign determines whether or not the output data should be inverted.

To reset the expanded output definition, use "OX0". If any of the outputs required for the expanded output group function are already defined as some other function the "U" error message is returned. This command is restricted and may only be used in privileged mode.

Example : OX3+
This example defines output lines 1-3 as an active high expanded output group. This allows output codes from 0 to 8 to be put on these lines with the OC command.

BO$_n$± Define bound overflow output. (restricted)
Range : 0 to 6

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 about 1ms. 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 : BO5+
This example defines output lines 5 as a position bound overflow output signal. Each time the signal passes the position bound, a logic high pulse is output on line 5.

LI List I/O line definitions.

This command lists the current definitions of the I/O lines on the display. Lines defined as function inputs are shown as "$_{nn}$: ± I (function)", and lines defined as limit switch inputs are shown as "$_{nn}$: ± L", where $_{nn}$ is the I/O line number. Lines not defined as either function or limit switch inputs are left blank. The I/O line definitions are listed on the display or terminal, one per display line. They may be listed continuously, or the system can print one line at a time and wait for the user to press a key before printing the next line. This is useful when using the system with the membrane terminal, which only has a two line display. This list pause facility is controlled by one of the flag bits in the DW command.

Example : LI
This will list the I/O line definitions on the display or terminal. The display for some of the above definitions would look like this.

```
1>LI (CR)                        User input to list  sequence
Inputs:
1: + AB                          Function input definition
2: - IN-/WT256/ID/MR-5000/ZC
3:                               Normal I/O lines, undefined
4:
5:
6:
7: - L                           Active low limit switch inputs
8: - L
Outputs:
1:
2:
3:
4:
5:
6:
7:
8:
1>
```

HW$_{bbbb}$ Set hardware setup word. (restricted)
Range : 0000 0000 0000 0001 to 8 channels set or D.
Default : 0000 0000 0011 0000

This command allows the user to write a value into the hardware setup word (global). Note that the leading zeros may be omitted. The hardware setup word allows various combinations of stepper and closed loop servo control to be implemented. In the default state, there are 2 servo channels on the Host Control board (PC3/100). Please note that this command has significantly changed at version 149 and 157.0 of the control software.

		Bit set	Bit cleared
bit	0	Enable host stepper no 1	Disable host stepper no 1
	1	Enable host stepper no 2	Disable host stepper no 2
	2	Enable host stepper no 3	Disable host stepper no 3
	3	Enable host stepper no 4	Disable host stepper no 4
	4	Enable host servo no 1	Disable host servo no 1
	5	Enable host servo no 2	Disable host servo no 2
	6	Enable slave 1 stepper no 1	Disable slave 1 stepper no 1
	7	Enable slave 1 stepper no 2	Disable slave 1 stepper no 2
	8	Enable slave 1 stepper no 3	Disable slave 1 stepper no 3
	9	Enable slave 1 stepper no 4	Disable slave 1 stepper no 4
	10	Enable slave 1 servo no 1	Disable slave 1 servo no 1
	11	Enable slave 1 servo no 2	Disable slave 1 servo no 2
	12	Enable slave 1 servo no 3	Disable slave 1 servo no 3
	13	Enable slave 2 servo no 1	Disable slave 2 servo no 1
	14	Enable slave 2 servo no 2	Disable slave 2 servo no 2
	15	Enable slave 2 servo no 3	Disable slave 2 servo no 3

The actual allocation of hardware to channel number is done automatically, and can be confirmed by using the "HWD" command. The maximum total number of channels is currently set at 8.

7.2.15 Loop commands

FN$_{v1n1:n2:n3}$ For variable $_{v1}$ = $_{n1}$ to $_{n2}$ step $_{n3}$ do command line.
Variable range : a to z or A to Z
$_{n1}$ & $_{n2}$ range : -2,147,483,647 to +2,147,483,647
$_{n3}$ range : 1 to +127

This command allows the programmer to use a variable (lower case letter type) to control a loop within a line of a sequence. More complex operation may be performed by calling sequences from the line. It allows numeric variables to be indexed from the loop variable (if a lower case variable is used). The step parameter, $_{n3}$, is optional, and when entered, its sign must be positive. This command can be nested, but note that if the same variable is used for different interacting FN loops, there is scope to get into indefinite loops. The constants $_{n1}$, $_{n2}$, and $_{n3}$ can be replaced by variables. A for next loop can be terminated by the ERF command (see page 53).

Example:

```
1>FNa2:4/OVa                          User input to start loop
+0000000002
+0000000003
+0000000004
1>

1>IVb8/IVc2/IVd2/FNab:c:d/OVa         User input to start loop using variables
+0000000008
+0000000006
+0000000004
+0000000002
1>

1>IV%20:10/IV%22:4/iv%24:2
1>FNc%20::%22::%24:/ovc               User input to start loop using numeric variables
+0000000010
+0000000008
+0000000006
+0000000004
1>

1>IV%20:10/IV%22:4/iv%24:2            User input to start loop using numeric variables
1>FNc%20::%22::%24:/vmc%22:d/ovc/iv%c:vd/ov%c:     Note the use of the loop variable
                                                   for indexing
+0000000010
+0000000040                           Value of variable %10
+0000000008
+0000000032                           Value of variable %8
+0000000006
+0000000024                           Value of variable %6
+0000000004
+0000000016                           Value of variable %4
1>
```

GL_n Go to line in sequence
Range : 1 to no of lines in current sequence

This command allows looping to a specified line number within a sequence. Note that it can only be executed from within a sequence.

Example:
A sequence might consist of the following:

```
CO3/MR5000
SO4/WT500/II5-/XT
GL1
```

This will continue looping to line 1 until input line 5 is negative, when the XT command will escape from the sequence.

7.2.16 Conditional commands

II_{nn}± If Input true do command line.
Range : 1 to 16×n (where n=No of control boards)

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. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on input line states.

Example : DP/II6-/MR20000/SO2
This displays the current position, and if input line no 6 is negative, the system moves 20000 counts, and sets output line 2. If input line 6 is positive, the current position is displayed, and the remainder of the line is ignored.

IO_{nn}± If Output true do command line.
Range : 1 to 16×n (where n=No of control boards)

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. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on output line states.

Example : DP/IO3+/MR5000/CM6
This displays the current position, and if output line no 3 is positive, the system moves 5000 counts, and complements output line 6. If output line 3 is negative, the current position is displayed, and the remainder of the line is ignored.

GX_{v1} If the external counter is greater than $_{v1}$, then do the rest of the command line.

This command executes the remainder of the current command line, if the external counter is greater than variable $_{v1}$.

Example : GX R/CO5
If variable "R" is 34 and the external counter is 5, the output line 5 is cleared.

XG_{v1} If variable $_{v1}$ is greater than the external counter, then do the rest of the command line.

This command executes the remainder of the current command line, if the variable $_{v1}$ is greater than the external counter.

Example : XG B/MA500
If variable "B" is 23 and the external counter is 400, the system moves to absolute position 500.

XE_{v1} If variable $_{v1}$ is equal to the external counter, then do the rest of the command line.

This command executes the remainder of the current command line, if the variable $_{v1}$ is equal to the external counter.

Example : XE A/DT
If variable "A" is 400 and the external counter is 400, the current time is displayed.

IA If network acknowledge received, do command line.

This command allows the programmer to specify that a command or command line is conditional whether an acknowledge has been received from the serial bus as a result of a recent command on the bus. If the acknowledge signal has been received, 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. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on input line states.

Example : DP/IA/MR20000/CO5
This displays the current position, and if an acknowledge signal has been received as a result of a command sent on the serial bus, the system moves 20000 counts, and clears output line 5. If an acknowledge signal has not been received, the current position is displayed, and the remainder of the line is ignored. Note that this condition could also be met if a command has been sent on the bus with no acknowledge received and a time has elapsed corresponding to the network timeout (TN) parameter.

$IC_{n1:v1}$ If bit $_{n1}$ of variable $_{v2}$ is clear, then do the rest of the command line.

This command executes the remainder of the current command line, if bit number $_{n1}$ in variable $_{v1}$ is clear. This command can be used in conjunction with the VBC or VBS bit manipulation command to control flow through a programme.

Example : IC 3:b/MR -25000/SO5
If bit 3 in variable "b" is clear, then the system executes a move of -25000 counts, and then sets output line 5.

IS$_{n1:v1}$ If bit $_{n1}$ of variable $_{v2}$ is set, then do the rest of the command line.

This command executes the remainder of the current command line, if bit number $_{n1}$ in variable $_{v1}$ is set. This command can be used in conjunction with the VBC or VBS bit manipulation command to control flow through a programme.

Example : IS 17:%35/MR-25000/SO5
If bit 17 in variable "%35" is clear, then the system executes a move of -25000 counts, and then sets output line 5.

IE$_{v1v2}$ If $_{v1}$ is equal to $_{v2}$, then do the rest of the command line.

This command executes the remainder of the current command line, if variable $_{v1}$ is equal to variable $_{v2}$.

Example : IE NA/MR 5000/CO3
If variable "N" is 16 and variable "A" is 16, then the system executes a move of 5000 counts, and then clears output line 3.

IF$_{v1v2}$ If $_{v1}$ is not equal to $_{v2}$, then do the rest of the command line.

This command executes the remainder of the current command line, if variable $_{v1}$ is not equal to variable $_{v2}$.

Example : IF tO/MA 0
If variable "t" is 516 and variable "O" is 516, then the system moves to an absolute position of 0 counts.

IG$_{v1v2}$ If $_{v1}$ is greater than $_{v2}$, then do the rest of the command line.

This command executes the remainder of the current command line, if variable $_{v1}$ is greater than variable $_{v2}$.

Example : IG VW/IVV0
If variable "W" is 12 and variable "V" is 13, then 0 is allocated to variable V.

GT$_{v1}$ If variable $_{v1}$ is greater than the time counter, then do the rest of the command line.

This command executes the remainder of the current command line, if the time counter is greater than variable $_{v1}$.

Example : GT X/SO12
If variable "X" is 544 and the time counter is 300, the output line 12 is set.

TG_{v1} If the time counter is greater than $_{v1}$, then do the rest of the command line.

This command executes the remainder of the current command line, if the variable $_{v1}$ is greater than the time counter.

Example : TG Z/MA0
If variable "Z" is 544 and the time counter is 700, the system moves to absolute position 0.

TE_{v1} If variable $_{v1}$ is equal to the time counter, then do the rest of the command line.

This command executes the remainder of the current command line, if the variable $_{v1}$ is equal to the time counter.

Example : TE a/DP
If variable "a" is 352 and the time counter is 352, the current position is displayed.

7.2.17 Display commands

DP(G)	Display actual position.

Displays current position, in encoder counts. The optional G suffix displays the current clobal position.

DD	Display demand position.

Displays current demand position, in encoder counts.

DV	Display velocity.

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

VT_n	Set velocity averaging time constant Range : 0 to 6 Default : 0

The VT command sets up an averaging mechanism, such that the number of speed samples doubles for each increment of the value of VT. When VT is zero, no averaging takes place. When VT is 6, 2^6 (64) samples are averaged over a period of half a 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.

DC	Display position bound overflow count.

Displays the current value of the position bound overflow count. Each time the motor passes the bound in the positive direction, this counter is incremented. When the motor passes the negative bound in the negative direction, the counter is decremented. For more information see the SB command in section 7.2.5 (page 45). The count value can be reset using the RC command

DF(P)	Display reference position error.

This command displays the last measured absolute position error relative to the reference input for this channel, in encoder counts. The optional P suffix allows the system to show the extrapolated demand position from the last measured position. This is actually the last measured reference position error added to the previous regular servo loop calculation of measured position minus demand position. It should be borne in mind that whilst the last servo loop calculation could be up to 3ms out of date, the difference between demand and measured position is unlikely to have changed significantly. For more details on the reference commands see section 7.2.13, and the DR and DZ commands in section 7.2.14 (page 96).

RA$_n$ Set reference averaging constant
Range : 0 to 5
Default : 0

The RA command sets up an averaging mechanism, such that the number of averaging samples doubles for each increment of the value of RA. When RA is zero, no averaging takes place. When RA is 5, 2^5 (32) samples are averaged. The system keeps a running average of the reference error which is updated at each time an external reference signal is seen, so that the latest average reference error is always available using the DF command.

PT$_n$ Set position snapshot averaging constant
Range : 0 to 5
Default : 0

The PT command sets up an averaging mechanism, such that the number of averaging samples doubles for each increment of the value of PT. When PT is zero, no averaging takes place. When PT is 5, 2^5 (32) samples are averaged. The system keeps a running average of the reference error which is updated at each time an external reference signal is seen, so that the latest average reference error is always available using the DS command.

DS Display snapshot position data.

This command displays the absolute position measured when a snapshot input signal was detected. For more details see the PS command in section 7.2.14 (page 100).

DT Display time.

Displays current time, in dd:mm:yy:hh:mm:ss format. If the O suffix is used (to output to a variable), then the seconds from startup value is sent to the defined variable.

EX$_n$ Echo mode.
Range : 0 to 1

This command defines whether characters are to be echoed back to the display terminal. The default state is 0 (echo mode enabled). When set to 1 the echo mode is switched off.

HC(N/F)$_{nn}$ Display history of commands executed.
Range : 1 to 255, or no value

Displays a list of the most recent $_{nn}$ commands executed in inverse chronological order. If no parameter is given, all of the last 256 most recently executed sequences are listed. In addition, together with each command, there is listed a time stamp and calling sequence. The time stamp is to a resolution of 1÷256 second. This can be useful for debugging a system, particularly when there is a large amount of asynchronous activity. The optional N or F parameters can be used without a numeric parameter to switch the logging activity on or off respectively. For example, HCF could be used within a repeat loop to block logging of large amounts of useless information, and HCN after the repeat loop to restore the logging of useful information.

HS(N/F)$_{nn}$ Display history of sequences executed.
Range : 1 to 255, or no value

Displays a list of the numbers of the most recent $_{nn}$ sequences executed in inverse chronological order. If no parameter is given, all of the last 256 most recently executed sequences are listed. The optional N or F parameters can be used without a numeric parameter to switch the logging activity on or off respectively. For example, HSF could be used within a repeat loop to block logging of large amounts of useless information, and HSN after the repeat loop to restore the logging of useful information.

HI Display history of PC3/100 control board.

Displays various aspects of the history of the control board. This includes:

1 Date/time the board was originally manufactured.
2 Date/time the board was last switched on (only if real-time clock present).
3 Total time the board has been switched on (only if NOVRAM present).
4 Total number of times that the board has been switched on (only if NOVRAM present).
5 Hardware (e.g. slave boards) attached to the system.
6 Status of the above data (validity of checksum, only if NOVRAM present).

DK Display system constants.

The system displays various parameter values, in the following order:
- Proportional gain constant
- Integral gain constant
- Velocity feedback gain constant
- Velocity feed-forward gain constant
- Scale factor for length related units
- Velocity
- Acceleration (to nearest multiple of 256)
- Maximum position error

DM Continuous display mode.

Turns on a continuous display of demand position, measured position, position error, and time.

DO Display mode off.

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

CD_{nn} Character delay, terminal.
Range : 0 to 255
Default : 0

This command sets the delay between characters sent to the serial terminal port, in units of 1/256 seconds. It allows the system to be used with terminals that do not support XON/XOFF protocols.

DN Use decimal numbers for input and output.

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

HN Use hexadecimal numbers for input and output.

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

$VOA..D_{n1:n2:n3:n4}$ Sends a variable continuously to display.

This command sets up the system to allow display a variable number $_{n1}$ ($ type) continuously, at a rate determined by the UR command. The suffix $_{n2}$ defines the line number on the display, $_{n3}$ defines the starting character number within line $_{n2}$, and $_{n4}$ defines the field width. This will carry on updating the display until VO is executed without a parameter. System variables (see section 7.2.18.2) can be displayed using the VO command. Up to 4 separate variables can be simultaneously displayed using the A, B, C, or D suffixes.

Example : UR50/VOA33
This will continuously send the measured position from channel 1 to the display at an update rate of once every 50 ticks.

UR_n Set update rate (ticks) for continuous variable display.
Range : 0 to 255
Default : 128

This the update rate, in system ticks, for the VO command. This is the rate at which the VO display is updated. e.g. If a value if 2 is set, the display will be updated every 2 ticks.

$XDA..D_n$ Set display division factor.
Range : 0 to 32
Default : 0

This command sets the division factor for the VO display. The actual data is added to the XF parameter, and then divided by 2^n in conjunction with being multiplied by the XM parameter. The largest division factor is 2^{32} (65536).

Example : XDB 5
This sets the VOB display division factor to $2^5 = 32$.

$XFA..D_n$ Set display offset value.
Range : -2,147,483,647 to +2,147,483,647
Default : 0

This command sets the offset value for the VO display. The actual data is added to the XF parameter $_n$, and then divided by 2 two to the power of the XD parameter in conjunction with being multiplied by the XM parameter.

Example : XFC 3500
This sets the VOC display offset value to 3500.

$XMA..D_n$ Set display multiplication factor.
Range : 1 to 2,147,483,647
Default : 1

This command sets the multiplication factor for the VO display. The actual data is added to the XF parameter, and then multiplied by n in conjunction with being divided by two to the power of the XD parameter.

OW$_{bb}$ Output options word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0000 0001 0001 0001

This command allows the user to set various auxiliary serial port options. Note that the leading zeros may be omitted. The display option word bit functions are described below.

		Bit set	Bit cleared
bit	0	Terminal port communications on	Terminal port communications off
	1	Auxiliary port 1 communications on	Auxiliary port 1 communications off
	2	Auxiliary port 2 communications on	Auxiliary port 2 communications off
	3	Auxiliary port 3 communications on	Auxiliary port 3 communications off
	4	Terminal echo on	Terminal echo off
	5	Auxiliary port 1 echo on	Auxiliary port 1 echo off
	6	Auxiliary port 2 echo on	Auxiliary port 2 echo off
	7	Auxiliary port 3 echo on	Auxiliary port 3 echo off
	8	Terminal parity on	Terminal parity off
	9	Auxiliary port 1 parity on	Auxiliary port 1 parity off
	10	Auxiliary port 2 parity on	Auxiliary port 2 parity off
	11	Auxiliary port 3 parity on	Auxiliary port 3 parity off
	12	Terminal parity odd	Terminal parity even
	13	Auxiliary port 1 parity odd	Auxiliary port 1 parity even
	14	Auxiliary port 2 parity odd	Auxiliary port 2 parity even
	15	Auxiliary port 3 parity odd	Auxiliary port 3 parity even

NOTE: Auxiliary port 1 is a 9 pin male RS-232 socket on the left hand side of the motherboard.
Auxiliary port 2 is a 9 pin female RS-232 or RS422 socket on the left hand side of the motherboard at the bottom.
Auxiliary port 3 is a 9 pin female RS-232 or RS422 socket on the left hand side of the motherboard one up from the bottom.

DW$_{bbbb}$ Display options word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0000 0000 0010 0010

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

		Bit set	Bit cleared
bit	0	40 column display	80 column display
	1	PANTERM terminal	Standard terminal
	2	List pause on	List pause off
	3	Pads lines to 16 chars	Prints <CR><LF> after each line
	4	Relative profiles	Absolute profiles
	5	Verbose error messages	One or two char error messages
	6	Hexadecimal input & output	Decimal input & output
	7	Debug display	Normal display
	8	Reserved for future expansion	
	9	Conditional tests in wait state.	Conditional tests - immediate execution.
	10	Output to serial ports for SN.	No output to serial ports for SN.
	11	Output to LED displays for SN.	No output to LED displays for SN.
	12	Reserved for future expansion	
	13	Always output required data	Normal operation
	14	Reserved for future expansion	
	15	No characters sent to terminal.	Characters sent to terminal.

The default value of 100010 is for a PANTERM 80 column terminal, and uses decimal input and output. It produces verbose error messages. This command is restricted, and is only available in privileged mode.

Note that bit 13 should be used in conjunction with bit 15 in order to have an effect. The result of combining bits 13 & 15 is to only transmit data of requested items (e.g. DP, OV, etc). This can be useful when the system is being controlled on the terminal port from a PLC. It is subtly different from using the EM1 (echo off command), since the echo off command sends prompts, error messages, etc.

Example : DW 101
This sets the display options for use with a 40 column display terminal, and turns the list pause on. The list pause is used when listing sequences and profile tables on a display with a small number of display lines.

XW_{bbbb} auXiliary output word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0000 0000 0000 0000

This command allows the user to set hidden outputs to the terminal display. These can be used to give a continuously updated display using the PANTERM communications programme (version 2.1 and later). The update rate of this display is controlled using the XU command. The various parameters which can be displayed are as follows:

		Bit set	Bit cleared
bit	0	Channel 1 position output	No channel 1 position output
	1	Channel 2 position output	No channel 2 position output
	2	Channel 3 position output	No channel 3 position output
	3	Channel 4 position output	No channel 4 position output
	4	Channel 5 position output	No channel 5 position output
	5	Channel 6 position output	No channel 6 position output
	6	Channel 7 position output	No channel 7 position output
	7	Channel 8 position output	No channel 8 position output
	8	Display inputs	No input display
	9	Display outputs	No output display
	8	Reserved for future expansion	
	9	Reserved for future expansion	
	10	Reserved for future expansion	
	11	Reserved for future expansion	
	12	Reserved for future expansion	
	13	Reserved for future expansion	
	14	Reserved for future expansion	
	15	Reserved for future expansion	

XU_n Set update rate (ticks) for continuous auxiliary output display.
Range : 0 to 255
Default : 128

This the update rate, in system ticks, for the XW command. This is the rate at which auxiliary data for the display terminal is updated. e.g. If a value if 2 is set, the display will be updated every 2 ticks.

BDA/B/C/D$_n$ Set baud rate.
Range : 0 to 11
Default : 0

Sets a baud rate for the terminal port(A), or auxiliary ports 1(B), 2(C), or 3(D). It is important to remember that having changed the baud rate on the controller, the terminal device will not communicate with the controller until its own baud rate has been correspondingly changed. The list below shows the available baud rates. It also indicates whether a particular baud rate is available on the standard serial port of a personal computer. This is useful when using a terminal emulation program such as PANTERM.

Please note that the terminal being used must be capable of operating at the baud rate set. If the reset button is pressed during the third ON period of the flashing L.E.D. after power up, then the system will reset to its default values (including the baud rate).

value	0	9600 baud (available on personal computer)
	1	14400 baud (available on personal computer)
	2	19200 baud (available on personal computer)
	3	28800 baud (available on personal computer)
	4	38400 baud (available on personal computer)
	5	57600 baud (available on personal computer)
	6	76800 baud
	7	115200 baud (available on personal computer)
	8	230400 baud
	9	7200 baud (available on personal computer)
	10	4800 baud (available on personal computer)
	11	2400 baud (available on personal computer)

HE Print help display.

This command prints a complete list of all commands on the system, in alphabetical order, a screenful at a time. It pauses between each page until a character is received. Help on a single command is displayed if the command mnemonic followed by "?" is entered.

Example:

System	User	Comments
1>	DP?<CR>	Request help on DP
DP Return current position		Single line help

LE Display last error

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

If The "O" (output to variable) option is used, then a channel number (most significant 16 bits) followed by and error number (least significant 16 bits) is transferred to that variable. This can be interpreted by interrogating the variable, based on the table of error codes (page184):

Example : LEo%1:

```
    Ov&2:          # Shows channel no
    Ov&3:          # Shows error no
```

TD Display measured tension

Displays current tension, in measured units multiplied by the tension factor (TM).

TR Display required tension ("set point")

Displays current required tension ("set point"), in units defined by the TL and TH parameters.

ON$_{nn}$ Output to numeric display.
Range : ± 32767

Sends numeric value $_{nn}$ to numeric L.E.D. display unit.

OB$_{nn}$ Output brightness value for numeric display.
Range : 1 to 15

Sets a brightness for the L.E.D. display unit.

OE Clear output of L.E.D. display unit.

Clears the output of the L.E.D. display unit.

OP$_{nn}$ Set decimal point position for L.E.D. display unit.

Sets the decimal point to be $_{nn}$ digits from the right of the display.

OD_n Set number of digits for L.E.D. display unit.
Range : 1 to 8
Default : 4

This the number of digits to be displayed on the display unit. If the OP parameter is greater than 0, then the decimal point takes up one digit space, and the OD parameter must be one less than the actual number of digits.

CB_{nn} Character delay, auxiliary serial port.
Range : 0 to 255
Default : 0

This command sets the delay between characters sent to the auxiliary serial port, in units of 1/256 seconds. It allows the system to be used with devices that do not support XON/XOFF protocols.

$SM_n/_v$ Send a variable $_v$ to serial port number $_n$.

This sends a number together with a variable code and a checksum to a serial port. It is to be received by another unit which is executing the RB command.

SN_n Send continuous display of position data to serial port number $_n$.

This sends continuous display of position data to a serial port number $_{n.}$ The data can be either demand position or measured position, as defined by bit 13 of the control word (page 76). The channels whose position data is sent are defined using the MS map output (channel dependent) command. Each bit of data is sent in the form of a variable, together with a variable code and checksum. It is to be received by another unit which is executing the RB command. Note that bits 10 and 11 of the display word (page 119, 120, 121, 157) must be set as required. In addition the CB character delay parameter must be set to an appropriate value. The variables used are as follows:

Channel 1	Variable S
Channel 2	Variable T
Channel 3	Variable U
Channel 4	Variable V
Channel 5	Variable W
Channel 6	Variable X
Channel 7	Variable Y
Channel 8	Variable Z

CN_n Stop continuous display of position data to serial port number $_n$.

This is the complement of the SN command listed above.

MS_n Map position output to serial port number $_n$.

This command defines which serial port(s) will be used for outputting position data for the current channel. It is described in more detail under the SN command.

RB_n Receive a variable from serial port number $_n$.

This receives a number together with a variable code and a checksum to from a serial port. It is designed to accept data sent using the SM command. If no valid data arrives within the timeout period defined by the TB command, the system will retry for a number of retries defined by the NR command.

NR_n Set number of retries for RB command.
Range : 0 to 255
Default : 3

This sets the number of times the system will retry to receive data using the RB command, or from the operator interface.

TB_n Set timeout period for RB command.
Range : 1 to 65535
Default : 2048

This the timeout period, in system ticks, for the RB command, and for operator interface communications. If this period elapses after the RB command or operator interface communications without a valid response, the system will retry for the specified number of retries (NR).

BE_n Set echo mode on for serial port number $_n$.

BN_n Set echo mode off for serial port number $_n$.

LD_n Set L.E.D. display number
Range : 1 to 4
Default : 1

This defines the L.E.D. display unit which will be accessed using the display commands above.

MD(F,N,L)$_{n(:c)v1}$"ccc"   Send a character string to the LCD/VFD display.

This sends a character string (enclosed by double quotes) to the line number $_{n}$ of the LCD/VFD display, starting at character number $_{c}$ (optional). Variables $_{v1}$ can also be displayed before or within text. Any number of variables can be used in this way. Either or both the character string and the variable can be entered. If the command is entered with no parameter, the display is cleared. If the command is entered with a line number, and no other parameter, the line referred to is cleared. If the command is entered with a line number, character number, and no other parameter, the line referred to is cleared starting from the character number entered. The optional F suffix allows actual screen update to be suppressed, whilst a series of other screen related commands are executed, until an MDN command is executed. This technique can cut the amount of writing to the display considerably. The L suffix allows a parameter (range 0-9) to define the number of digits to be displayed when variables are sent. This allows a number, which represents a decimal fraction, to be correctly aligned. A number will be padded with leading zeros so that the total number of digits is equal to the parameter set with the L parameter. Alternatively, the "^" character can be used within a normal MD string to indicate that the following number will re-allocate the number of leading zeros for subsequent variables. This allows one MD command to send several variables to the display, each with different numbers of leading zeros.

The table below shows the displayed symbols when the matching characters are enclosed in single quotes ('). Where a ✖ is indicated, the character is illegal.

sp	!	"	#	$	%	&	'	(	)	*	+	,	_	.	/
sp	○	┌	┘	\	•	ヲ	✖	ィ	ゥ	|	ォ	ャ	ュ	ョ	ッ
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
			▌	█	▌				♪	°c	°f	▾	▸	◂	▴
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Ä	À	Å	á	å	E	Ö	ö	Ø	ø	Ű	ű	\	≠	~	§
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
Æ	æ	£	Pt	●	○	♦	◊	│	⊊		≤	/	↩	↑	↓
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
α	ä	β	E	μ	σ	ρ	g	√	-1	j	x	ȼ	ꝁ	ñ	ö
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
→	"	θ	∞	Ω	ü	Σ	π	$\overline{X}$	←				÷	✖	✖

ME(R,E)$_{n1}$:$_{n2}$:$_{n3}$:$_{n4}$:$_{v1}$:$_{v2}$:$_{v3}$ Sends a variable to the LCD/VFD display.

This sends a variable $_{v1}$ to line number $_{n2}$ of the LCD/VFD display, starting at character number $_{n3}$. The first parameter $_{n1}$ allows a number (1 to 8) of variable "sites" to be set up for editing. When the MEE command is executed, the flashing cursor starts, and the up and down arrow keys can be used to index through any ME variables displayed on the screen. The parameter $_{n4}$ specifies the maximum number of characters allowable for a field entry. When the enter key is pressed, any of the 8 variable sites which have been set up will be read from the screen to their respective variables, and the cursor will go off. The optional $_{v2}$ and $_{v3}$ parameter allow a variable to display and receive numbers to the right of a decimal point. $_{v2}$ indicates the variable to used for the number to the right of the decimal point, and $_{v3}$ indicates the number of digits to the right of the decimal point. The SK command or WK13 (wait fo enter key) can be used to trigger another event after the screen has been read.

The optional R suffix allows all the variable sites to be read without the enter key being pressed. It also resets the variable site pointers. This reset operation can be performed by using ME followed by <CR> only.

AN$_{vn}$ Allows alpha-numeric characters to be input to variable.
Range($_{n}$) : 1 to 20

This displays a range of alpha-numeric characters on lines 1 to 3 of the display. Line 4 is used for inputting a string of characters to one ore more numeric variables (%type). Each % type variable can accommodate 4 characters. $_{v}$ points to the starting numeric variable to be filled, and $_{n}$ indicates the maximum number of characters to be input, and from which the number of variables to be used can be calculated. The arrow keys can be used to move the cursor over a character; if the ● key is then pressed, the character under the cursor will be placed above the ▲ character on the third line. The backspace key can be used to delete the character above the ▲ character. The ▲ character can be moved to the left and right by using the left and right up arrow keys under the display. The <ESC> key can be used to abort the current operation without changing the variables. Entering the <enter> key clears the assigned variable(s) and places the preceding characters in the variable(s). The contents of %variables which have strings of characters can be displayed using the DA command.

DA$_{vn}$:$_{n1}$:$_{c}$ Display alpha-numeric characters from variable(s).
Range($_{n2}$) : 1 to 20

This sends a character string to the line number $_{n1}$ of the LCD/VFD display, starting at character number $_{c}$. The string consists of $_{n}$ characters starting from % type variable pointed to by variable $_{v}$. Each variable can store up to 4 characters.

PP(A,C,D,E,L,N)$_{\text{"CCC" n(:c)v1}}$"ccc" Send a character string to the printer.

This sends a character string (enclosed by double quotes) to an ASL AP24XS-40 panel-mounting printer. The principle of using this command is similar to the MD command. However the position of the printout is only determined by the width of the last field to be printed. There is no specific position parameter, and nothing is actually printed until the PPL (new line) command is issued. This prints the contents of the buffer, sends a line feed and resets the current column position to 1. There are several variants of the PP command, defined by the suffix.

PPA $_{vn}$ followed by a variable, followed by a number, allows an alpha-numeric variable to be printed. The alpha-numeric variable has to be in a block of numeric variables whose starting address is defined by variable $_{v}$. The number of characters to be printed is defined by the numeric parameter $_{n}$.

Example : ivt1000/ppat9

This prints 9 characters of the alpha-numeric variable starting at %1000.

PPC can be used to cancel any current printing activity. This can be useful if a lot of information was sent to the printer which needs to be stopped. The printer has an 8 KB buffer, so a lot of information can be queued for printing inadvertently.

PPD disables the printer serial port, and PPE enables it.

PPL issues prints the contents of the current line buffer, and sends a new line command.

PPN sets up an internal variable in the controller to define the number of columns for variables which are subsequently sent to the printer. Variables are automatically aligned to the right hand side of the number of columns defined.

PP without a letter suffix can be used to send character strings and variables to the printer buffer in the same way as the MD command.

OS(M/C/R)($_{\text{"s"}}$) Output a string.

Prints a string.

Example : OS”testing”
This displays the word “testing” on the display terminal. This is useful for de-bugging a system, and allows a meaningful display to be set up on the terminal.

The “M” option allows communication with a serial Modem. Note that bit 0 of OW should be set to 0, and bit 3 of GW should be set to 1 (no parity).

The “C” option allows the system to operate with a modem in connect mode.

The “R” option resets the modem flags.

7.2.18 Variable commands

The system allows the use of integer variables. This enables fixed programmes to be modified merely by changing one or two key parameters. This can be particularly useful for controlling machine tools. The variables are 32 bits long and are signed. There are 32280 of them; 26 of them are designated by the letters A-Z(case sensitive), and are saved in non-volatile memory with the SP command. Another 26 are volatile, and are designated by the lower case letters a-z(case sensitive). There are 256 volatile $ variables (0 to 255), some of which are designated as system variables, and are read-only, please see section 7.2.18.2 on system variables (page 137). The remaining 32768 are non-volatile and are designated by a % character followed by a number from 0 to 32767, followed by a : full colon if the variable name is not the end of the variable command. The numeric % variables can be selectively saved using the V suffix on the SP command (page 21). In addition % variables can be accessed indirectly by using lower case variables.

Better use can be made of memory space byte using numeric variables in 16-bit (&), 8-bit (|), or 8-bit unsigned (@) form. These access the same memory area as the standard 32-bit (%) variables, and care must be used to ensure that the correct areas are being accessed. For example variable %2 can be accessed by variable &4 and &5, or variables |8, |9, |10 and |11.

There are two extensions to inputting and outputting data:

1 V for inputting a parameter from a variable
2 O for outputting a parameter to a variable

Example: MRVC
This moves relative an amount defined in variable C.

Example: TOO%23
Put the current value of Timeout into volatile variable no %23.

IV$_{v1}$±$_{nn}$ Input a variable.
Range : -2,147,483,647 to +2,147,483,647

Allows a variable $_{v1}$ to be set up.

Example: IVC 12000
This sets up variable "C" to a value of 12,000.

Example: IVa 24
IV %a:345
This sets up variable "a" to a value of 24, and then variable %24 to a value of 345..

OV($_{"s"}$)$_{v1}$ Output a variable.

Prints a value for variable $_{v1}$.

Example : OV%36
This displays the value of variable "%36" on the display terminal. The optional string can be placed between inverted commas before the variable name. This is useful for de-bugging a system, and allows a meaningful display to be set up on the terminal.

SPV Save numeric variables. (restricted)

This command saves all the numeric variables to non-volatile memory. There may be a short delay while the save operation takes place. This works independently of the SP command without a suffix, and allows selective saving of variables during a programme. The saved variables become the new defaults, used by the system on power-up. At the end of the save operation, the system calculates a cyclic redundancy check byte (CRC) on the saved data, which is then saved in non-volatile memory as well. This allows the saved data to be verified at any time by comparing the stored CRC byte with a calculated one. If the saved data has changed at all, the stored CRC will not be the same as the calculated CRC. If the save operation fails for any reason, then an "F" error message is returned. In this case, please contact your sales office. This command is restricted, and is only available in privileged mode.

NV$_{v1}$:$_{v2}$ Display range of numeric variables.

Prints values for variables %$_{v1}$ through to %$_{v2}$.

Example : NV%100:%169:
This displays the value of variable "%100" through to "%169" on the display terminal in a format as shown below:

```
000100 +0000004683 +0000000025 -0000005630 +0000000000 +0000000000
000105 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000110 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000115 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000120 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000125 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000130 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000135 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000140 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000145 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000150 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000155 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000160 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
000165 +0000000000 +0000000000 +0000000000 +0000000000 +0000000000
```

If $_{v2}$ is omitted, then the system will display variables starting with $_{v1}$ until <ESC> is pressed.

If both $_{v1}$ and $_{v2}$ are omitted, then the system will display variables starting with variable %0 until <ESC> is pressed.

VBD/C/S$_{n1}$:$_{v1}$ Display, Clear or Set a bit in a variable.

This command allows individual bits within a variable to be set or cleared. This command can be useful for directing the flow of instructions in conjunction with the condition bit test commands IC and IS.
Example :

```
1>VBS13:F           Set bit 13 of variable F.
1>VBDF              Display bit pattern for variable F
Bit no 32                    16              1
          0000000000000000 0001000000000000
1>
```

$\mathrm{VD}_{v1v2v3v4}$ Divide a variable ($_{v1} \div {}_{v2} \rightarrow {}_{v3}$).

This commands divides $_{v1}$ by $_{v2}$, and assigns the quotient to $_{v3}$. The remainder is assigned to $_{v4}$.

Example : VD AX%154:Z
If variable "A" is 15 and variable "X" is 3, then variable "%154" will be assigned the value 3, and Z will be assigned the value 0.

VM_{v1v2v3} Multiply a variable ($_{v1} \times {}_{v2} \rightarrow {}_{v3}$).

This commands multiplies $_{v1}$ by $_{v2}$, and assigns the result to $_{v3}$.

Example : VM GIB
If variable "G" is 5 and variable "I" is 3, then variable "B" will be assigned the value 15.

$\mathrm{VZ}_{v1v2v3v4v5}$ Multiply and Divide a variable ($_{v1} \times_{v2} \div {}_{v3} \rightarrow {}_{v4}$).

This commands multiplies $_{v1}$ by $_{v2}$ and divides the result by $_{v3}$, and assigns the quotient to $_{v4}$. The remainder is assigned to $_{v5}$. The intermediate result is a 64-bit number (-9,223,372,036,854,755,808 to +9,223,372,036,854,755,808), but the divisor must be set so that the final result does not overflow a 32-bit number (-2,147,483,647 to +2,147,483,647). This function can be useful for performing high resolution number conversion, and helps to overcome the limitations of integer arithmetic.

Example : VZ bAX%154:Z
If variable "b" is 2, "A" is 15 and variable "X" is 3, then variable "%154" will be assigned the value 6, and Z will be assigned the value 0.

VA_{v1v2v3} Add a variable ($_{v1} + {}_{v2} \rightarrow {}_{v3}$).

This commands adds $_{v1}$ to $_{v2}$, and assigns the result to $_{v3}$.

Example : VA XTO
If variable "X" is 16 and variable "T" is 5, then variable "O" will be assigned the value 21.

$\mathrm{VR}_{v1v2}(_{v3})$ Square root of variable ($\sqrt{}_{v1} \rightarrow {}_{v2}$).

This command takes the square root of variable $_{v1}$ and assigns the result to $_{v2}$. The remainder is assigned to $_{v3}$. If $_{v3}$ is omitted, then $_{v2}$ is assigned the rounded square root.

Example : VR EDF
If variable "E" is 18, then variable "D" will be assigned the value 4, and variable "F" will be assigned the value 2.

Example : VR ab
If variable "a" is 21, then variable "b" will be assigned the value 5.

VS_{v1v2v3} Subtract a variable ($_{v1}$ - $_{v2}$ -> $_{v3}$).

This command subtracts $_{v2}$ from $_{v1}$, and assigns the result to $_{v3}$.

Example : VS HBZ
If variable "H" is 27 and variable "B" is 8, then variable "Z" will be assigned the value 19.

AV_{v1v2} Absolute value of variable (ABS($_{v1}$) -> $_{v2}$).

This command takes the absolute value of variable $_{v1}$ and assigns the result to $_{v2}$.

Example : AV B%220
If variable "B" is -357, then variable "%220" will be assigned the value 357.

Example : AV ZJ
If variable "Z" is 24, then variable "J" will be assigned the value 24.

$TVs/S/c/C/t/T_{v1v2}$ Trigonometric functions([Arc]Sine/[Arc]Cosine/[Arc]Tangent($_{v1}$) -> $_{v2}$).

This command performs a trigonometric function on variable $_{v1}$ and assigns it to variable $_{v2}$. If the suffix is S, C, or T, variable $_{v1}$ is the number of degrees x 100. The result is the sine/cosine/tangent x 1000000. When the result is ∞, this is represented by 2,147,483,647 (the largest number which can be assigned to a variable).

Example : TVC c%45:
If variable "c" is 5000 (i.e. 50°), then variable "%45" will be assigned the value 642788.

If the suffix is s, c, or t, variable $_{v1}$ is the sine/cosine/tangent x 1000000. The result is the number of degrees x 100 (between 0 and 9000).

$IC_{n1:v1}$ If bit $_{n1}$ of variable $_{v2}$ is clear, then do the rest of the command line.

This command executes the remainder of the current command line, if bit number $_{n1}$ in variable $_{v1}$ is clear. This command can be used in conjunction with the VBC or VBS bit manipulation command to control flow through a programme.

Example : IC 3:b/MR -25000/SO5
If bit 3 in variable "b" is clear, then the system executes a move of -25000 counts, and then sets output line 5.

$IS_{n1:v1}$ If bit $_{n1}$ of variable $_{v2}$ is set, then do the rest of the command line.

This command executes the remainder of the current command line, if bit number $_{n1}$ in variable $_{v1}$ is set. This command can be used in conjunction with the VBC or VBS bit manipulation command to control flow through a programme.

Example : IS 17:%35/MR-25000/SO5
If bit 17 in variable "%35" is clear, then the system executes a move of -25000 counts, and then sets output line 5.

IE_{v1v2} If $_{v1}$ is equal to $_{v2}$, then do the rest of the command line.

This command executes the remainder of the current command line, if variable $_{v1}$ is equal to variable $_{v2}$.

Example : IE NA/MR 5000/CO3
If variable "N" is 16 and variable "A" is 16, then the system executes a move of 5000 counts, and then clears output line 3.

IF_{v1v2} If $_{v1}$ is not equal to $_{v2}$, then do the rest of the command line.

This command executes the remainder of the current command line, if variable $_{v1}$ is not equal to variable $_{v2}$.

Example : IF tO/MA 0
If variable "t" is 516 and variable "O" is 516, then the system moves to an absolute position of 0 counts.

IG_{v1v2} If $_{v1}$ is greater than $_{v2}$, then do the rest of the command line.

This command executes the remainder of the current command line, if variable $_{v1}$ is greater than variable $_{v2}$.

Example : IG VW/IVV0
If variable "W" is 12 and variable "V" is 13, then 0 is allocated to variable V.

GT_{v1} If variable $_{v1}$ is greater than the time counter, then do the rest of the command line.

This command executes the remainder of the current command line, if the time counter is greater than variable $_{v1}$.

Example : GT X/SO12
If variable "X" is 544 and the time counter is 300, the output line 12 is set.

TG_{v1} If the time counter is greater than $_{v1}$, then do the rest of the command line.

This command executes the remainder of the current command line, if the variable $_{v1}$ is greater than the time counter.

Example : TG Z/MA0
If variable "Z" is 544 and the time counter is 700, the system moves to absolute position 0.

TE$_{v1}$ If variable $_{v1}$ is equal to the time counter, then do the rest of the command line.

This command executes the remainder of the current command line, if the variable $_{v1}$ is equal to the time counter.

Example : TE a/DP
If variable "a" is 352 and the time counter is 352, the current position is displayed.

MV$_{v1v2v3}$ Move a block of numeric variables.

This commands moves a block of $_{v1}$ % type variables from address $_{v2}$ to $_{v3}$.

Example : iva20/ivb150/ivc250/MVabc
Move a block of 20 variables from %150 to %250.

ZM$_{v1v2v3}$ Set a block of numeric variables [to zero].

This commands sets a block of $_{v1}$ % type variables from address $_{v2}$ to a value of $_{v3}$. If $_{v3}$ is omitted, then a value of zero is assumed.

Example : ivx20000/ivy9000/ivz35/ZMyxz
Set a block of 9000 variables from %20000 to %28999 to a value of 35.

OT Output the time counter.

Prints the current value for the time counter variable.

TW$_{nn}$ Wrap for time counter input.
Range : 1 to 2,147,483,647
Default : 2,147,483,647

This command sets the value at which the time counter wraps to zero.

ZT[$_{nn}$] Zero time counter or set time counter.
Range : 0 to 2,147,483,647

This command reads sets the time counter value to zero.

If a value is given, the system sets the current external counter to the given (absolute) value. If no value is given, it sets the external counter to be zero. The ZT command may be used at any time.

RB_n Receive a variable from serial port number $_n$.

This receives a number together with a variable code and a checksum to from a serial port. It is designed to accept data sent using the SM command. If no valid data arrives within the timeout period defined by the TB command, the system will retry for a number of retries defined by the NR command.

$SM_n/_v$ Send a variable $_v$ to serial port number $_n$.

This sends a number together with a variable code and a checksum to a serial port. It is to be received by another unit which is executing the RB command.

$VK_{n1:v1:v2:v3}$ Set a variable from the keypad.

This command waits for an input from the keypad. It displays the current value of the variable $_{v1}$ on the left hand side of line number $_n$ of the LCD/VFD display, and echoes the keys entered on the right hand side. Only decimal numbers are allowed. The Enter key causes the number entered to be assigned to the variable $_{v1}$. "<-" will perform a destructive backspace, and "ESC" will cancel the current entry. If a second variable, v2, is specified, the system will assume that a two part number, separated by a decimal point is to be entered. The third variable is used as a counter to specify the number of decimal points entered (particularly useful when the decimal section begins with a 0). This can be useful when entering parameters with decimal points to the system. If the command is entered with no variable, any pending keypad input is cancelled.

$VI_{n1}/_{n2}\pm$ Set system to input variable from input lines.

This command sets up the system to allow input lines to set a variable using the RI command (with O followed by a variable name). The sign defines the sense of the inputs, with reference to the binary number which is allocated to a variable using the RI command. If no parameter is entered the current VI state is displayed.

$VOA..D_{n1:n2:n3:n4}$ Sends a variable continuously to display.

This command sets up the system to allow display a variable number $_{n1}$ ($ type) continuously, at a rate determined by the UR command. The suffix $_{n2}$ defines the line number on the display, $_{n3}$ defines the starting character number within line $_{n2}$, and $_{n4}$ defines the field width. This will carry on updating the display until VO is executed without a parameter. System variables (see section 7.2.18.2) can be displayed using the VO command. Up to 4 separate variables can be simultaneously displayed using the A, B, C, or D suffixes.

Example : UR50/VOA33
This will continuously send the measured position from channel 1 to the display at an update rate of once every 50 ticks.

UR_n	Set update rate (ticks) for continuous variable display. Range : 0 to 255 Default : 128 This the update rate, in system ticks, for the VO command. This is the rate at which the VO display is updated. e.g. If a value if 2 is set, the display will be updated every 2 ticks.
$XDA..D_n$	Set display division factor. Range : 0 to 32 Default : 0 This command sets the division factor for the VO display. The actual data is divided by 2^n in conjunction with being multiplied by the XM parameter. The largest division factor is 2^{32} (65536). Example : XDB 5 This sets the VOB display division factor to 2^5 = 32.
$XFA..D_n$	Set display offset value. Range : -2,147,483,647 to +2,147,483,647 Default : 0 This command sets the offset value for the VO display. The actual data is added to the XF parameter $_n$, and then divided by 2 two to the power of the XD parameter in conjunction with being multiplied by the XM parameter. Example : XFC 3500 This sets the VOC display offset value to 3500.
$XMA..D_n$	Set display multiplication factor. Range : 1 to 2,147,483,647 Default : 1 This command sets the multiplication factor for the VO display. The actual data is multiplied by n in conjunction with being divided by two to the power of the XD parameter. Example : XMC 9 This sets the VOC display multiplication factor to 9.

7.2.18.2 System Variables

Certain variables are used by the system, and can be interrogated at any time. They cannot be written to, and will incur an error if an attempt is made to write to any of them.

7.2.18.2.1 Channel status variables

The channel status is allocated to variables $1 through to $8 (for motor channels 1 to 8 respectively). The variable value indicates the following information about the current motor channel:

0	Not moving
1	Accelerating
2	Moving at constant velocity
3	Decelerating to new velocity
4	Decelerating to creep velocity
5	Creeping at constant velocity
6	Waiting to stop (DQ i/p line or ST with parameter)
7	Stopping (final deceleration)
8	Stopped after move

7.2.18.2.2 Channel mode variables

The channel mode is allocated to variables $9 through to $16 (for motor channels 1 to 8 respectively). The variable value contains the following bit values which indicate the following information about the current motor channel:

Bit	
1	Constant velocity mode (VC)
2	Moving (MR or MA)
3	Executing a profile
4	Executing position mapping
5	Stopping
6	Initialising
7	Inching
8	Motor off (MO)

7.2.18.2.3 Channel error mode variables

The channel mode is allocated to variables $17 through to $24 (for motor channels 1 to 8 respectively). The variable value indicates the following information about the current motor channel:

7	Limit switch error
8	Reference timeout error
9	Reference error outside limits
10	Reference error correction over-run
11	Position error
12	Timeout error
13	High position limit error
14	Low position limit error
15	Bounds limit error

7.2.18.2.4 Channel bounds counter variables

The channel bounds counter is allocated to variables $25 through to $32 (for motor channels 1 to 8 respectively).

7.2.18.2.5 Channel position variables

The channel position is allocated to variables $33 through to $40 (for motor channels 1 to 8 respectively).

7.2.18.2.6 Channel velocity variables

The channel velocity is allocated to variables $41 through to $48 (for motor channels 1 to 8 respectively).

7.2.18.2.7 Channel position error variables

The channel position error is allocated to variables $49 through to $56 (for motor channels 1 to 8 respectively).

7.2.19 Remote communication network commands

The system is designed to communicate bi-directionally with other devices and controllers on a serial network. Each device or controller must have a unique address (set up with the Network Address command), otherwise there will be unpredictable results.

There is a token passing system which is initiated by setting bit zero of the serial network control word (NW).

NA_n Set a serial network communication address
Range : 1 to 128
Default : 1

Sets up a serial network address to be the value $_n$.

Example : NA 12
This sets up the network address for the current controller to be 12.

NS Display status of the serial communication network.

This command display the current state of the serial network.

NW_{bbbb} Serial network control word. (restricted)
Range : 0000 0000 0000 0000 to 1111 1111 1111 1111 (binary).
Default : 0

This command allows the user to set various serial network configuration options. Note that the leading zeros may be omitted. The serial network control word bit functions are described below.

		Bit set	Bit cleared
bit	0	Join serial network	Exit from serial network
	1	Reserved for future expansion.	
	2	Reserved for future expansion.	
	3	Reserved for future expansion.	
	4	Reserved for future expansion.	
	5	Reserved for future expansion.	
	6	Reserved for future expansion.	
	7	Reserved for future expansion.	
	8	Reserved for future expansion.	
	9	Reserved for future expansion.	
	10	Reserved for future expansion.	
	11	Reserved for future expansion.	
	12	Reserved for future expansion.	
	13	Reserved for future expansion.	
	14	Reserved for future expansion.	
	15	Reserved for future expansion.	

$NC_{v1}\,{}_{c}/_{a}$ Send a network command on the serial network

This sends a command $_c$ to a device at address $_a$ on the serial network, using a parameter from variable $_{v1.}$

TN_n Set timeout period for response time from network
Range : 1 to 65535
Default : 2048

This the timeout period, in system ticks, for the WN command. If this period elapses after transmitting to the serial bus without an acknowledge packet, the system will clear an internal flag, which indicates that the system is waiting for acknowledgement from the network. If the system is currently waiting with the WN command, it will stop waiting.

IA If network acknowledge received, do command line.

This command allows the programmer to specify that a command or command line is conditional whether an acknowledge has been received from the serial bus as a result of a recent command on the bus. If the acknowledge signal has been received, 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. This could either be the next line of a sequence, or new input commands.

This command can be used within sequences to construct multiple conditions, based on input line states.

Example : DP/IA/MR20000/CO5
This displays the current position, and if an acknowledge signal has been received as a result of a command sent on the serial bus, the system moves 20000 counts, and clears output line 5. If an acknowledge signal has not been received, the current position is displayed, and the remainder of the line is ignored. Note that this condition could also be met if a command has been sent on the bus with no acknowledge received and a time has elapsed corresponding to the network timeout (TN) parameter.

WN Wait for network acknowledge response

This command sets the system into the wait state, until an acknowledge from a recently sent network command is received, or a time has elapsed corresponding to the network timeout (TN) parameter.

7.2.20 Data logging commands

AR Abort Recording of data.

This command aborts a current recording session if the parameter defined by the MN command has not been reached.

NL$_n$ Set number of channels to log.
Range : 1 to 4
Default : 1

This command sets the total number of channels to be logged.

IR Initialise Recording of data.

This command starts recording of data, as defined with the LF, NL, MN, and LG commands. When the recording is finished an "L" is displayed on the screen.

LF$_n$ Set logging frequency.
Range : 0 to 65504
Default : 0

This command sets the frequency of sampling of the pre-defined variables. The frequency is (8192÷LF) Hz. If LF is set to zero, then no logging will take place. The values currently allowable are 1, 2, 4, 8, 16, 32, and multiples of 32 up to 65504.

MN$_n$ Set Maximum Number of stored samples.
Range : 1 to 80000
Default : 1

This command sets the maximum number of samples to be recorded during a logging session. This can be over-ridden by using the AR command. Note that the maximum no of recordings per channel is the MN parameter ÷ the NL parameter.

UL Unload Logged data.

This command sends recorded data in a formatted ASCII format down the terminal port.

LG$_{n}$:$_{o}$:$_{p}$ Define logging channel. (restricted)
Range ($_{n}$) : 1 to 4 (logging channel)
Range ($_{o}$) : 1, 2 or 4 (type of data)
Range ($_{p}$) : 1 to 4 (data channel)

This command defines a logging channel. The first parameter can be any number between 1 and 4, and defines the order in which the data will be unloaded using the UL command. The second parameter can be any number between 1 and 3, and defines the type of data to be logged:

value	1	Analogue channel
	2	Position channel
	4	Position channel with velocity calculation (as defined by the VT parameter for the particular channel)

The third parameter can be any number between 1 and 4, and defines the actual position or analogue channel to be logged to the current logging channel.

If no parameter is entered, the setup for the number of channels defined with the NL command is displayed.

DG$_{n}$± Define logging output line. (restricted)
Range : 1 to 16×n (where n=No of control boards)

This command defines the specified output line as a data logging indicator. The line is set to the specified state when the system starts data logging (from the IR command), and is cleared to the opposite state when logging has completed.

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

Example : DG 3-
This sets up an active low data logging output signal on output line 4. When an logging starts, output line 4 is set to a logic low. It reverts to a logic high when logging has finished.

7.3 Status and Error Messages

7.3.1 Status messages

This section gives the system status responses in various circumstances.

>	Normal prompt character in position control mode. The system is ready for the next command.
:	Prompt character in motor off state.
?	Prompt character for new parameter value.
I	Initialising to reference position.
M	Moving to new position.
P	Profile move. The system is executing a stored profile.
S	Stopping under normal deceleration.
V	Velocity control mode. The system is executing a constant velocity move.
W	Waiting. The system is waiting for some condition before continuing.
T	Tension control mode.

7.3.2 Error messages

This section gives the various error messages produced by the system.

B	Binary number required. The system received a non-binary character when it expected a binary number as input.
D	Decimal number required. The system received a non-decimal character when it expected a decimal number as input.
E	Error in command. The system received a command which was not recognised, not allowed at this time, or had an invalid parameter.
F	Failed parameter save or checksum test. The parameters and data saved in non-volatile memory have not verified correctly. Please contact your sales office.
G	The previous move command was aborted when the instantaneous position error was greater than the maximum allowed position error set by the SE command.
H	Hexadecimal number required. The system received a non-hexadecimal character when it expected a hexadecimal number as input.
L	Limit switch detected or position limit exceeded.
N	No room. The space available for input function strings, sequences or profiles is full.
O	Out of range. The value entered was outside the allowed range for the command.
R	Restricted command. This command is available only in privileged mode.
T	Timeout. The last move command was aborted when the system detected a timeout error.
U	Line already in use. It is not possible to set or clear an output line that has been defined as an error output, or to redefine an input line that is already defined as a reference, limit switch or function input.

8 INTERFACING

8.1 Notes on installation

The system relies on the position information from the incremental encoders, and any noise on the encoder signals can give rise to errors in the absolute position. Care must be taken in installation of the control module and the encoders to minimise any noise on the encoder signal lines. The encoder interfaces on the PC3/1x0 board have differential input stages for use with encoders with complementary outputs, 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. The system will be set up so 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 re-initialise the absolute position.

The digital input and output lines are designed to be used with 24 volt logic levels. These lines are optically isolated from the microprocessor based circuits. This provides protection and allows higher voltage signals to be used for greater noise immunity.

8.2 Safety

The Pan control system provides many safety facilities, and it is recommended that these are used in addition to external safety systems such as hard-wired limit switches. Pan Controls 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, LL	Set position limits
DL	Define limit switch inputs

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

8.3 Indicator L.E.D.'s

The control boards have three indicator L.E.D.'s to indicate various system functions.

(i) The top indicator is used to show whether the system is functioning correctly. For correct operation this flashes on and off equally once per second. If there is a hardware fault, or if the saved parameters do not match the hardware found, this L.E.D. will flash on and off, with a different time period for on and off. If a terminal is connected to the RS-232 port, it will be possible to identify what the problem is.

(ii) The second indicator shows the state of the on-board watchdog timer, which is in turn connected to the on-board relays. When the watchdog is activated, the L.E.D. is off.

(iii) The third indicator shows the presence of a 5v power supply. It should be noted that for the analogue part of the board to work correctly, +15v and -15v supplies are also required.

8.4 Position Encoders

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 for both channels 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 1,000 counts per revolution is seen as generating 4,000 counts per revolution. The maximum count rate is 10^6 counts per second (1 MHz), giving a maximum encoder cycle rate of 250 kHz. On a 1000 line encoder, this is equivalent to a maximum speed of 250 revolutions per second, or 15,000 r.p.m.

The encoder inputs have differential input circuits for use with encoders with complementary output signals. The encoder signals are complementary signals with line driver outputs, which gives good noise rejection.

8.5 Demand outputs

The demand outputs to the high power drives are analogue signals with a range of ±10V, at 12 bits resolution. These outputs are switched directly to 0V in the motor off state. The motor drives should be connected such that a positive demand output signal causes the motor to move in the positive direction.

8.6 Relay Contacts

The relays which switch the demand outputs to 0V in the motor off state have a spare set of changeover contacts. These may be used to derive inhibit signals to the motor drives in the motor off state, or for example to switch a joystick onto a drive input to allow manual control of the motor.

8.7 Digital Input/Output Lines

The control system has eight isolated input and eight isolated output lines per pair of axes. Inputs may be programmed as a signal to execute a user-defined command sequence, or as limit switch inputs. Outputs may be controlled directly from command sequences if required. In addition, there is a dedicated error output line which may be programmed to give an indication of any serious system error condition. All the input lines indicate high if left unconnected.

8.8 Operation of Limit Switches

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

If a limit switch is operated, the system will stop the motor immediately and go into the "motor off" state. The system displays an "L_{nn}" error message to indicate that a limit switch has been detected. All limit switches should be wired such that operation of any switch gives an error signal to the system.

8.9 Reference inputs

The reference inputs are used during the IN initialisation sequence to define the zero reference position for each motor. They may also be used to continuously update the system absolute position from the external zero reference if required. They are connected to marker signals from the position encoders. The reference inputs have differential input circuits similar to the encoder inputs.

The IN initialisation sequence is as follows.

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

The sense of the reference input for the current channel is programmed by using the DR command.

8.10 Serial Communications

8.10.1 Diagnostic terminal

The serial link to the diagnostic terminal uses RS-232 signal levels. The serial word format used is 8 data bits, 1 stop bit, and even parity. The baud rate is set by default to 9600 baud. The baud rate and parity may be changed if required.

The serial interface is buffered in software and echoes back the characters as they are received. By default, it uses hardware handshaking, and is designed to communicate with the PANTERM programme for a personal computer. However, it can be set up to use XON/XOFF software handshake, where it sends XOFF to signal that its input buffer is becoming full, and sends XON when it is ready for more characters. Note that if the XOFF is ignored, the buffer may overflow and characters will be lost. The system also responds to XON/XOFF to control its output.

The characters normally used for the XON/XOFF protocol are DC1 ($11 hex, cntl-Q) for XON, and DC3 ($13 hex, cntl-S) for XOFF.

8.10.2 Outstations

The serial links to the outstations use RS-485 signal levels. This has a high noise immunity, and allows long transmission lengths if this should be required in the future. The wiring is be arranged in a daisy chain configuration, allowing the use of multi-drop protocol.

9 ELECTRICAL CHARACTERISTICS

Power Supplies :

Open frame linear regulated unit - encoders
115V or 240V 50Hz mains at 1A max.
Open frame linear regulated unit - control system
+5V ±0.1V at 2.00A
+15V ±0.5V at 500mA
-15V ±0.5V at 100mA

Serial link (diagnostic terminal):

Signal levels	RS-232
Baud rate	9600 (default) - 115,200
Data format	8 data bits, 1 stop bit, even parity

Serial link (outstations):

Signal levels	RS-422/485
Baud rate	9600
Data format	8 data bits, 1 stop bit, no parity

Encoder inputs :

Input impedance	6kΩ
Input signal levels	+5V (set by value of resistor network)
Input cycle rate	250 kHz max

Track A input leads track B input for positive movement

Command signal outputs :

Isolated output range	±10V
Resolution	12 bits

Digital Inputs (isolated):

Input signal levels	+24V (set by resistor network)
Input current	10 ma typical at selected input voltage

Digital Outputs (isolated NPN Darlington outputs):

Load current	100ma maximum

Relay contacts :

Rated load	1A 60V d.c. 0.3A 110V a.c. (resistive)
	0.5A 60V d.c. 0.2A 110V a.c. (inductive)
Carry current 2A	
Switch voltage max	60V d.c. 125V a.c.
Switch current max	2A d.c. 1A a.c. (resistive)
	1A d.c. 0.5A a.c. (inductive)
Switch power max	60W 60VA (resistive)
	30W 30VA (inductive)
Inductive load power factor	0.4 max
Contact resistance	50 mΩ max

10 SUMMARY

10.1 Commands

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

Miscellaneous commands

AP±$_{nn}$	Add channel dependent parameter.	
BH	Breakdown of current Hardware	
BK	Backup main programme (low level code)	
BV$_{n}$	Select slave board version numbers	
BW$_{bbbb}$	Set boot options control word.	(restricted)
CA	Camera commands.	
CH$_{n}$	select CHannel	
CS	CheckSum test	
EV"ccc"	Enter a user software revision no string	
GW$_{bbbb}$	Set global control word.	(restricted)
HR$_{nn}$	Set Hard Reset sequence or initiate Hard Reset	
LA	List all parameters	
MC(C,I,R,W)$_{n1}$	Memory card operations	
MP$_{n}$	Set factor for position feedback multiplier	
OA$_{n}$/Z/M/L	Initialise ADC offsets to value	
OT	Output the time counter	
PD$_{n}$	Set factor for position feedback division	
RD	Read data from nonvolatile memory	(restricted)
RS	ReSet complete setup to defaults (restricted)	
SP	Save Parameters	(restricted)
TAG/A/B$_{n}$	Set tangent control axes.	(restricted)
TI	Initialise ADC offset	
TW$_{nn}$	Wrap for time counter input	
UP	Upload new main programme (user level code).	
VN	display Version Number and revision date	
XB	Execute boot programme	

Mode commands

CI$_{n}$	Enter transparent Communication with operator Interface	
CL$_{n}$	Lower bound for link Correction.	(restricted)
CP	Compile position gain tables.	(restricted)
CU$_{n}$	Upper bound for link Correction.	(restricted)
CX$_{n}$	Remote reset of operator interface	
EM$_{n}$	Echo mode	
FD$_{n}$	Set link factor for division	
FM$_{n}$	Set link factor for multiplication	
GF	Global Motor Off.	
LM$_{n}$	Link current axis to axis no $_{n}$	
LW$_{bbbb}$	Link motions control word. (restricted)	
MO	set to Motor Off	
MW$_{bb}$	Set multi axis move word.	(restricted)
NM	set to Normal Mode	

$PB_{n1}:_{n2}$	Enter playback position mode.	
PC	set to Position Control mode	
PG_n(D)	Position Gain (correction) Value.	(restricted)
PM	set to Privileged Mode	
PW	set PassWord	(restricted)
RM	Enter record position mode.	
RQ_{nn}	Set record position distance rate $_{nn}$.	
RR_{nn}	Set record position time rate $_{nn}$.	
TC	Enter tension control mode	
WM_{nn}	Wait for $_{nn}$ system ticks before moving current channel.	
ZW_{bbbb}	Set position record control word.	(restricted)

Move commands

AB	ABort, emergency stop	
AC_{nn}	Set arc centre coordinate for current channel for interpolated circular move	
GA	Global abort, emergency stop	
GS	Global Stop.	
ID	Initialise Demand signal offset	
IL_{nn}	Set inch distance	
IM±	Execute an inch move	
IN±	INitialise to reference position	
IP_{nn}	Set inch pause time.	
IW_{bb}	Set interpolation move word. (restricted)	
$MA\pm_{nn}$	Move to Absolute position	(scaled)
MM_{nn}	Prepare move $\pm_{nn}$ units on current channel (multi-axis move)	
$MR\pm_{nn}$	Move Relative to current position	(scaled)
MX	Execute multi-axis move	
OA_n/Z/M/L	Initialise ADC offsets to value	
QC_{nn}	Prepare relative stop (from creep) position on current channel (multi-axis stop)	
QV_{nn}	Prepare relative stop position on current channel (multi-axis stop)	
ST	STop with normal deceleration	
$TC_{v1}:_{v2}$	Transform from Polar to Cartesian	
$TP_{v1}:_{v2}$	Transform from Cartesian to Polar	
VC±	set to Velocity Control mode	
VX	Move (multi-axis) at constant velocity	
XC+/-	Execute interpolated circular move	
XL	Execute interpolated linear move	

Set parameter commands

BB_{nn}	Set bound counter overflow bound.	(restricted)
BC_{nn}	set Backlash Compensation distance	(scaled)
DB_{nn}	set input DeBounce time	(restricted)
HT_{nn}	Set maximum tension threshold.	(restricted)
MT_{nn}	Set minimum tension threshold.	(restricted)
RC	Reset bound overflow Count	
SA_{nn}	Set Acceleration	(scaled)
SB_{nn}	Set position Bounds	(restricted)(scaled)
SC_{nn}	Set Creep distance	(scaled)
SD_{nn}	Set Deadband	(scaled)
SL_{nn}	Set settLing time	
SS_{nn}	Set Slow creep speed	(scaled)
SV_{nn}	Set Velocity	(scaled)
SW_{nn}	Set Window on final position	(restricted)
SZ_{nn}	Set deceleration	(scaled)
$TH\pm_{nn}$	Set high tension limit	(restricted)
$TL\pm_{nn}$	Set low tension limit	(restricted)
TM_{nn}	Set tension multiplier	(restricted)
$TS_{hh:mm:ss}$	Time Set	
$ZC[_{nn}]$	Zero position Counters or set position	(scaled)

Sequence commands

AS_{nn}	set AutoStart sequence	(restricted)
CE_{nn}	Execute sequence continuously as timed event	
ED_{nn}	Execute sequence event after time delay	
EE_{nn}	Execute sequence on error condition	
ER	End Repeat	
ES_{nn}	Enter Sequence	(restricted)
GE_{nn}	Execute sequence after position Gain event. (local)	
GL_{n}	Go to line in sequence	
HR_{nn}	Set Hard Reset sequence or initiate Hard Reset	
HS_{nn}	Display history of sequences executed	
$LS[_{nn}]$	List Sequence	(restricted)
MF	display Free Memory	
NE_{nn}	Execute sequence after snapshot event. (local)	
PE_{nn}	Set parameter for ED event	
PL_{nn}	Set parameter for looping continuous event	
RE_{nn}	Execute sequence after reference event. (local)	
$RP[_{nn}]$	RePeat command line	
$SK(_{A-F})_{nn}$	Execute sequence on keypad entry event	
XS_{nn}	eXecute Sequence	
XT	Exit from current sequence	
WS_{nn}	Wait until stopped before executing sequence	(restricted)

Profile commands

EO_{nn}	Enter Offset correction table $_n$	(restricted)
EP_{nn}	Enter Profile	(restricted)
LO_{nn}	List Offset correction table $_n$	
$LP[_{nn}]$	List Profile	

PV$_n$	set Profile Velocity
XP$_{nn}$[-]	eXecute Profile
XO$_{nn}$	Execute offset correction table $_{nn}$

Logic commands

CY$_{nn}$	Clear node $_{nn}$
EJ$_{nn}$	Enter logic definition $_{nn}$
FC$_{n1}$:$_v$:I/O/N$_{n2}$±	Fill logic counter value
FT$_{n1}$:$_v$:I/O/N$_{n2}$±	Fill logic timer value
LJ$_{nn}$	List logic definition
IY$_{nn}$±	If node true do command line
RY$_{nn}$	Read node state(s)
SY$_{nn}$	Set node $_{nn}$
FT$_{n1}$:$_v$:I/O/N$_{n2}$±	Fill timer value

Wait commands

WA±$_{nn}$	Wait for Absolute position	(scaled)
WB	Wait for Bound position	
WC±$_{nn}$	Wait for bound overflow counter	
WE	Wait End	
WF	Wait for reFerence signal	
WI$_n$±	Wait for Input	
WK($_{nn}$)	Wait for keypad entry	
WN	Wait for network acknowledge response	
WR±$_{nn}$	Wait for Relative position	(scaled)
WS$_{nn}$	Wait until stopped before executing sequence.	(restricted)
WT$_{nn}$	Wait for Time	

Error handling

EG$_{bb}$	Set global error options word.	(restricted)
EW$_{bb}$	set Error options Word	(restricted)
LE	display Last Error	
LH±$_{nn}$	set High position Limit	(restricted)(scaled)
LL±$_{nn}$	set Low position Limit	(restricted)(scaled)
RT$_{nn}$	set Reference Timeout	(restricted)
SE$_{nn}$	Set maximum position Error	(restricted)(scaled)
TO$_{nn}$	set TimeOut	(restricted)

Gain commands

AO$_{n}$:$_{xx}$	Set analogue output to global value.	
CW$_{bb}$	set control word	(restricted)
IT$_{n}$	set Integration Time constant	(restricted)
KD$_{nn}$	set differential gain constant	(restricted)
KF$_{nn}$	set velocity feed-forward gain constant	(restricted)
KI$_{nn}$	set integral gain constant	(restricted)
KP$_{nn}$	set proportional gain constant	(restricted)
KV$_{nn}$	set velocity feedback gain constant	(restricted)
KM$_{nn}$	set Monitor output gain constant	(restricted)
LV$_{nn}$	Set lower stepper resonance velocity bound.	(restricted)
OM$_{nn}$	set Offset on Monitor output	(restricted)
SF$_{n}$	Set monitor Function	(restricted)
UV$_{nn}$	Set upper stepper resonance velocity bound.	(restricted)
WD$_{nn}$	Set stepper direction delay.	(restricted)

Reference commands

BS$_{nn}$	Set no of bounds for deferred snapshot correction.	
DF	Display reFerence error	
PF±$_{nn}$	Set snapshot reference offset	(restricted)
PJ±$_{nn}$	Set snapshot deferred adjustment position	(restricted)
PR$_{nn}$	Set maximum snapshot reference correction	(restricted)
PT$_{n}$	Set position snapshot averaging constant	
RA$_{n}$	Set reference averaging constant	
RV$_{nn}$	set Reference correction Velocity	(restricted)
RL$_{nn}$	Set Reference repeat Length	(restricted)
RF±$_{nn}$	Set Reference oFfset	(restricted)
RW$_{bb}$	set Reference options Word	(restricted)
SG$_{nn}$	Set snapshot goal value	(restricted)
SR$_{nn}$	Set maximum Reference correction	(restricted)
SJ±$_{nn}$	Set deferred adJustment position	(restricted)(scaled)
VP$_{nn}$	Set snapshot reference correction velocity	(restricted)

Input/output commands

CM$_{nn}$	Complement output line $_{nn}$.
CO[$_{nn}$]	Clear Output line(s)
CY$_{nn}$	Clear node $_{nn}$
DIL$_{nn}$±	Define input line to be latched. (restricted)
EI[$_{nn}$]	Enable Input(s)
EID	Enable input line latch
EK$_{nn}$	Enable keypad input.
FO$_{nn}$	Flash output line $_{nn}$
FR$_{nn}$	Set flash rate $_{nn}$ for output lines
II$_{nn}$±	If Input true, do command line
IO$_{nn}$±	If Output true do command line
IY$_{nn}$±	If node true do command line
MID	Mask input line latch.
MI[$_{nn}$]	Mask Input(s)

MK$_{nn}$	Mask keypad input
OC$_{nn}$	Output Code value
RI[$_{nn}$]	Read Input line(s)
RIL$_{nn}$±	Read input line latch
RO[$_{nn}$]	Read Output line state(s)
RY$_{nn}$	Read node state(s).
SO[$_{nn}$]	Set Output line(s)
SY$_{nn}$	Set node $_{nn}$
WK($_{nn}$)	Wait for keypad entry

Configuration commands

AA$_{a}$:$_{nn}$	Analogue input averaging term.	
AG$_{nn}$	Set analogue input gain constant. (restricted)	
AI$_{nn}$	Read analogue input	
AL	Set analogue input to link to motion	
AU	Unlink analogue input from motion.	
AW$_{bb}$	Set analogue input link word.	(restricted)
BD$_{n}$	Set baud rate	
BO$_{n}$±	define Bound Overflow Output	(restricted)
DB$_{nn}$	set input DeBounce time	(restricted)
DE±	Define Error output sense	(restricted)
DH$_{nn}$±±/$_{s1}$/$_{s2}$	Define inching input.	(restricted)
DI±$_{nn}$	Define function Input	(restricted)
DL±$_{nn}$	Define Limit switch input	(restricted)
DQ$_{n}$±	Define multi-axis stop input.	(restricted)
DR±	Define Reference input sense	(restricted)
DX$_{n}$±	Define eXpanded input group	(restricted)
DZ$_{n}$±	Define zero marker input on/off.	(restricted)
EC$_{nn}$±	Define external counter input.	(restricted)
HW$_{bbbb}$	Set hardware setup word.	(restricted)
LI	List I/O line definitions	
OX$_{n}$±	define eXpanded Output group	(restricted)
PO$_{n}$±	define Position trigger Output	(restricted)(scaled)
PS$_{n}$±	Define position snapshot input.	(restricted)
RX	Read external counter input	
SI$_{n}$±	Set input to stop current motor channel.	(restricted)
WX$_{nn}$	Wrap for external counter input	
ZX[$_{nn}$]	Zero external counter input or set counter input	

Display commands

BE_n	Set echo mode on for serial port number $_n$	
BN_n	Set echo mode off for serial port number $_n$	
CB_{nn}	Character delay, auxiliary serial port	
CD_{nn}	set Character Delay	
CN_n	Stop continuous display of position data to serial port number $_n$.	
DC	Display bound overflow Count	
DD	Display Demand position	(scaled)
DF	Display reFerence error	
DK	Display system constants	
DM	set continuous Display Mode on	
DN	use Decimal Numbers	(restricted)
DO	Display mode Off	
DP	Display current Position	(scaled)
DS	Display snapshot position data.	
DV	Display current Velocity	(scaled)
DT	Display time	
DW_{bb}	set Display Word	(restricted)
EO	Echo mode off	
EM	Echo mode on	
HE	print HElp display	
HC_{nn}	Display history of commands executed	
HI	Display history of PC3/100 control board	
HS_{nn}	Display history of sequences executed	
HN	use Hexadecimal Numbers	(restricted)
LD_n	Set L.E.D. display number	
LE	Display Last Error	
MD_{v1}"ccc"	Send a character string to the LCD/VFD display	
MS_n	Map position output to serial port number $_n$	
NR_n	Set number of retries for RB command	
OB_{nn}	Output brightness value for numeric display	
OD_n	Set number of digits for L.E.D. display unit	
OE	Clear output of L.E.D. display unit	
ON_{nn}	Output to numeric display	
OP_{nn}	Set decimal point position for L.E.D. display unit	
OS(M/C/R)($_{"s"}$)	Output a string.	
OW_{bb}	Output options word.	(restricted)
RA_n	Set reference averaging constant	
RB_n	Receive a variable from serial port number $_n$.	
$SM_n/_v$	Send a variable $_v$ to serial port number $_n$.	
SN_n	Send continuous display of position data to serial port number $_n$.	
TB_n	Set timeout period for RB command.	
TD	Display measured tension	
TR	Display required tension ("set point")	
VT_n	Set velocity averaging time constant	
XW_{bbbb}	auXiliary output word.	(restricted)
XU_n	Set update rate (ticks) for continuous auxiliary output display.	

Loop commands

FN$_{v1n1n2n3}$	For variable $_{v1}$ = $_{n1}$ to $_{n2}$ step $_{n3}$ do command line.
GL$_{n}$	Go to line in sequence

Conditional commands

GX$_{v1}$	If the external counter is greater than $_{v1}$, then do the rest of the command line
IA	If network acknowledge received, do command line
IC$_{n1:v1}$	If bit $_{n1}$ of variable $_{v2}$ is clear, then do the rest of the command line.
IF$_{v1v2}$	If $_{v1}$ is not equal to $_{v2}$, then do the rest of the command line
II$_{nn}$±	If Input line true, do command line
IO$_{nn}$±	If Output true do command line
IE$_{v1v2}$	If variable v1=v2, do command line
IG$_{v1v2}$	If variable v1>v2, do command line
IS$_{n1:v1}$	If bit $_{n1}$ of variable $_{v2}$ is set, then do the rest of the command line
TE$_{v1}$	If variable $_{v1}$ is equal to the time counter, then do the rest of the command line
TG$_{v1}$	If the time counter is greater than $_{v1}$, then do the rest of the command line
XE$_{v1}$	If variable $_{v1}$ is equal to the external counter, then do the rest of the command line
XG$_{v1}$	If variable $_{v1}$ is greater than the external counter, then do the rest of the command line

Variable commands

AV$_{v1v2}$	Absolute value of variable (ABS($_{v1}$) -> $_{v2}$)
FN$_{v1n1n2n3}$	For variable $_{v1}$ = $_{n1}$ to $_{n2}$ step $_{n3}$ do command line
GT$_{v1}$	If variable $_{v1}$ is greater than the time counter, then do the rest of the command line
IC$_{n1:v1}$	If bit $_{n1}$ of variable $_{v2}$ is clear, then do the rest of the command line
IE$_{v1v2}$	If variable v1=v2, do command line
IF$_{v1v2}$	If $_{v1}$ is not equal to $_{v2}$, then do the rest of the command line.
IG$_{v1v2}$	If variable v1>v2, do command line
IS$_{n1:v1}$	If bit $_{n1}$ of variable $_{v2}$ is set, then do the rest of the command line
IV$_{v1}$±$_{nn}$	Input a Variable
MV$_{v1v2v3}$	Move a block of numeric variables
NV$_{v1}$:$_{v2}$	Display range of numeric variables.
OT	Output the time counter
OV$_{v1}$	Output a Variable
RB$_{n}$	Receive a variable from serial port number $_{n}$.
SM$_{n}$/$_{v}$	Send a variable $_{v}$ to serial port number $_{n}$.
VA$_{v1v2v3}$	Add variables v1 + v2 ‣ v3
VBC/S$_{n1v1}$	Set or Clear a bit in a variable
VM$_{v1v2v3}$	Multiply variables v1 × v2 ‣ v3
VS$_{v1v2v3}$	Subtract variables v1 - v2 ‣ v3
VZ$_{v1v2v3v4v5}$	Multiply and Divide a variable ($_{v1}$ ×$_{v2}$ ÷ $_{v3}$ -> $_{v4}$).
TE$_{v1}$	If variable $_{v1}$ is equal to the time counter, then do the rest of the command line.
TG$_{v1}$	If the time counter is greater than $_{v1}$, then do the rest of the command line.
TVS/C/T$_{v1v2}$	Trigonometric functions(Sine/Cosine/Tangent($_{v1}$) -> $_{v2}$).
TW$_{nn}$	Wrap for time counter input.
UR$_{n}$	Set update rate (ticks) for continuous variable display.
VD$_{v1v2v3v4}$	Divide a variable ($_{v1}$ ÷ $_{v2}$ -> $_{v3}$).
VI$_{n1}$/$_{n2}$±	Set system to input variable from input lines.
VK$_{v1}$	Set a variable from the keypad.
VO$_{n1}$	Sends a variable continuously to display.
VR$_{v1v2v3}$	Square root of variable (√$_{v1}$ -> $_{v2}$).

XE_{v1}	If variable $_{v1}$ is equal to the external counter, then do the rest of the command line.	
XF	Set display offset value.	
XG_{v1}	If variable $_{v1}$ is greater than the external counter, then do the rest of the command line.	
XD_n	Set display division factor.	
XM_n	Set display multiplication factor.	
ZM_{v1v2v3}	Set a block of numeric variables [to zero].	
$ZT[_{nn}]$	Zero time counter or set time counter.	

Data logging commands

AR	Abort Recording of data	
$DG_n\pm$	Define logging output line. (restricted)	
IR	Initialise Recording of data	
LF_n	Set logging frequency	
$LG_n:_o:_p$	Define logging channel.	(restricted)
MN_n	Set Maximum Number of stored samples.	
NL_n	Set number of channels to log.	
UL	Unload Logged data.	

Boot programme commands

EN	Erase non-volatile memory.
UP	Upload new main programme (low level code).
XF	Execute final running code.

Network commands

IA	If network acknowledge received, do command line.	
NA_n	Set a serial network communication address	
$NC_{v1\ c}/_a$	Send a network command on the serial network	
NS	Display status of the serial communication network.	
NW_{bbbb}	Serial network control word.	(restricted)
TN_n	Set timeout period for response time from network	
WN	Wait for network acknowledge response	

10.2 Status Messages

>	normal prompt character
:	motor off prompt character
?	parameter value prompt character
C	executing slow Creep
I	Initialising to reference position
M	Moving to new position
P	executing a Profile
S	Stopping under normal deceleration
V	Velocity control mode
W	Waiting

10.3 Error Messages

B	Binary number required
D	Decimal number required
E	Error - unrecognised or invalid command
F	Failed parameter save or checksum test
G	position error Greater than maximum
H	Hexadecimal number required
L	Limit switch detected or position limit exceeded
N	No room in memory
O	parameter Out of range
R	Restricted command
T	Timeout abort
U	line in Use

11 PANTERM communications programme for personal computer

11.1 Introduction

The program can be run on an IBM Personal Computer or compatible MS-DOS computer (PC) to make that PC act as a terminal to a PAN digital control system. Once installed and running, most commands entered on the PC are transmitted directly to the control system and any responses from the control system are displayed on the PC. In addition, some special commands are provided which are not transmitted to the control system but are handled locally by the PC. These commands allow text files to be edited locally on the PC and then transmitted to the control system. This allows complex command strings for the control system to be created and amended using the PC and then transmitted to the control system together.

11.2 Setting up the serial link

Some PAN control systems have more than one serial port. The port marked Terminal should be used. Most PCs have up to four serial ports referred to as COM1, COM2, COM3 & COM4. PANterm assumes that COM1 will be used but this can be changed. RS-232 levels should be used and it is assumed that the PC has a 9 way or 25 way D connector plug. If the PC has a different connector type then the PC and control system should be linked with a four wire lead having the following connections. The connections between a PC and the Control Board are as follows:

Connections for cable for IBM compatible Personal Computer:

	9-pin plug(Controller)		25-pin socket(PC)	9-pin socket (PC)	
	2	->	3	2	RX
	3	<-	2	3	TX
	5	<->	7	5	GND
Hardware)	7	<-	4	7	RTS
Handshake)	8	->	5	8	CTS
Connect these 4	)	->	8	1	DCD
lines together	)	->	6	6	DSR
when using hardware	)	->	22	9	RI
handshaking	)	<-	20	4	DTR

RS-232 serial connections

PAN control system's terminal ports are configured to run at 9600 baud and to use hardware handshaking by default. This can be changed by means of the "BD" baud rate command. PANterm defaults to the same settings but can be changed by means of the configuration menu.

11.3 Using PANTERM as a simple terminal

PANterm is started from the MS-DOS command prompt (A> or C>) by typing:

```
PANTERM
```

There are no command line options. The PC will then display its copyright message for a few seconds, before going into terminal mode. When you receive your copy of PANterm it should already be configured to work with your control system. Typing RETURN at this point should produce a control system prompt on the screen:-

```
1:
```

If this does not happen, you will need to configure PANterm to work with your control system which is explained in the Configuration section below.

At this stage, control system commands such as CH1, PC, MR1000 etc can be typed and will be transmitted directly to the control system. The responses from the control system such as 1:, 1> and 2M will be displayed on the PC. Your PC is now working as a terminal on the control system.

To leave the PANterm terminal program and return to MS-DOS type Alt-X (Exit).

11.4 Preparing system command files

PANterm has been designed not just to send single commands to a control system, but also for the preparation and issuing of files of command to the systems. PANterm can also be used to receive a set of commands which actually exists on a control system, and place these commands in a file. You are then able to edit this file as required, and issue the new commands to the control system in a similar way.

You may edit a file of commands with any text editor. PANterm also allows you to set up a specific editor for use with the program, and this is explained in the configuration section. Note that your editor must be able to prepare a file which contains pure ASCII test, without hidden control codes. Most editors will do this in "non document" or "ASCII" mode. The resulting file should contain a list of control system commands in exactly the same format as you would type them in direct terminal mode. The file could look like this:

```
# move forward and back
CH1
PC
SA500
SV1000
MR5000
SV200
MR-5000
# move complete
```

Note that comment lines can be put in the file by starting the line with a "#" (pound or hash) symbol. In due course we will see what happens to these comment lines.

When the above example is sent to a control system it would cause the system to execute the move forward and back immediately. This assumes that the control system is ready to move and the velocity and accelerations are sensible. If an error occurs, perhaps because the motor cannot achieve this acceleration, then the system will respond in the normal way with an error message. If the move fails early on then all the subsequent move commands will generate errors and a whole batch of error messages may be received by the PC. Error messages can be recorded to a file disk (using the Alt-R PANTERM command). In addition the last error on the control system can be interrogated using the LE last error command.

If you do not want the sequence of commands to be executed immediately but to be stored on the control system for future use we could edit the file to look like:-

```
# enter sequence to move forward and back
ES1
CH1
PC
SA500
SV1000
MR5000
SV200
MR-5000

# end of sequence
# now define input line 1
DI1-/XS1
```

When this is sent to the control system, the sequence will be stored and input line 1 will be set up so that when it is activated (perhaps by an operator pushing a button) then the sequence will be executed. Note that the file contains a blank line at the end of the sequence. This tells the control system that the end of the sequence has been reached. This blank line must also be included when entering profiles and maps. Note that this command file could be written on fewer lines by putting several commands on one line with a delimiter between them:-

```
# enter sequence to move forward and back
ES1
CH1/PC/SA500/SV1000/MR5000
SV20/MR-50000

# end of sequence
# now define input line 1
DI1-/XS1
```

Putting several commands on one line has no effect on the way the control system will respond. The only exception to this rule is the repeat (RP) command. This is used as the last command on a line and causes all the previous commands on that one line to be repeated, in order, the specified number of times.

Spreadsheet programs can also be used instead of editors to prepare control system command files. Any spreadsheet which allows the user to output columns to a disk file in an ASCII format can be used. The following example shows how a spreadsheet might be set up to create a move profile:-

```
  ___________________________________________________________
 |  |<        A        ><   B   ><   C   ><   D   ><   E   >
 | 1|      JOLT FREE PROFILE CALCULATION                 EP1
 | 2|                                           1          0
 | 3|Parameters:-                               2          1
 | 4|                                           3          3
 | 5|Move distance          500 Counts          4          8
 | 6|Move time               30 Ticks           5         14
 | 7|                                           6         24
 | 8|w = 2*3.14159/B6 =   .2094393              7         38
 | 9|                                           8         54
 |10|Formula used in cells E2 to E31:-          9         74
 |11|                                          10         98
 |12|Profile = B5/B6*(Dn-SIN(B8*Dn)/B8)        11        124
 |13|                                          12        153
 |14|  (where n is the row number)             13        184
 |15|                                          14        217
 |16|                                          15        250
       ~~~                   ~~~                ~~~

 |28|                                          27        497
 |29|                                          28        499
 |30|                                          29        500
 |31|                                          30        500
 |32|
```

If the spreadsheet is used to print column E to a disk file then that disk file would be a suitable control system command file. Note that the printout should include row 32 containing a blank entry. Note also that column E should be formatted so that the numbers in it are integers.

It will be obvious to any computer programmer that programming languages such as "Basic", "C" and "Fortran" can also be used to produce control system command files. By using such a language or a spreadsheet it is possible to produce complicated profiles and maps and load them into a control system far more easily than can be done by hand. Also, this approach is far quicker when it comes to making amendments. For instance, in the above example a second profile move, numbered 2, of 600 counts could be created by amending just two cells - E1 and B5.

11.5 Loading command files

Once you have prepared a control system command file and set up PANterm in its terminal mode, you are ready to transmit command files to the control system. This is called loading. To do this type Alt-L (Load). You will then be asked the name of the file that you wish to load. Type its name and press return. PANterm will now start sending the file to the control system. The characters that you will see on your screen are all echoed back from the control system, so that you know they have been received correctly. The exception to this rule is the comment; comments are not sent, but printed directly to the screen, in reverse colours.

When your command file has been loaded, the program will revert to the terminal mode. Also, pressing any key while loading will cause the load to be paused; a second keypress will continue the load.

PANterm incorporates a delay mechanism so that you can check the response of the control system to your command file. You can type Alt-S (Slow) or Alt-F (Fast) to slow down or speed up the loading. A very slow speed combined with pausing can be useful for spotting errors.

While in terminal mode, you may want to record what some of the commands and definitions currently on the control system are. You can start this recording by typing Alt-R. You will be prompted to enter a name for the recording file. Both the commands that you type, and the control system's response will be recorded. You can stop the record at any time by pressing Alt-R again.

While loading a command file, you may wish to abort the load and return to the terminal mode. Press Alt-T (terminal) if you wish to do this. Pressing Control-C, which stops many programs, will not affect PANterm - this gets sent straight to the control system.

You might have aborted because you spotted an error in your command file. If so you will want to edit your file. PANterm allows you to type Alt-E which will start up your editor without leaving PANterm, provided that PANterm has been told which editor to use in the configuration section. You may use your editor as normal. On exiting your editor you will return immediately to PANterm in its terminal mode. By leaving PANterm in your PC's memory in this way, restarting PANterm is considerably speeded up.

11.6 Loading system programme files

The controller system programme can be changed by using the UP command from the boot programme. A system programme in Motorola S-record format must have bee prepared in advance, and this can be uploaded using the Alt-U option. Otherwise this option is similar in operation to the Alt-L load file option.

11.7 Configuring PANTERM

PANterm will generally be configured when you receive it. This facility allows you to set up differences required for your system once, and then to use that version in future without further difficulty.

To reconfigure PANterm, type Alt-C when you are in terminal mode. A new window will appear, with the header 'Configuration'. Each entry, together with its default value, is explained below.

Communications Port

Normally, PCs have up to 4 serial ports, one of which will be used to talk to the control system. PANterm is set up to use Port 1 by default. Some PCs have more than four ports; because this is not standard, and methods of implementation between different computers vary, it has not been possible to write PANterm to use other ports.

Baud Rate

This item indicated the speed at which PANterm will attempt to talk to the control system. This is normally set at 9600 baud, because this is the default speed at which the control system will communicate. You may want to change this value if you are talking to a different device (other than a control system).

Serial mode; RS232, 422 or 485

PANterm is designed to operate in conjunction with the standard 16450 or 16550 UART device. This option allows the use of an RS-422 or RS-485 converter (as manufactured by KK systems) to be connected to the standard RS-232 serial port of a personal computer. The hardware handshake mode (see below) can only be selected if the RS-232 option is selected, since the RS-422 and RS-485 adaptors do not have hardware handshake facilities. The RS-485 option uses the RTS line to control the direction of data flow.

File Loading delay

The rate of loading a file to the control system will be controlled by this value, which is in milliseconds. You may also speed up file loading by pressing Alt-F, which has the effect of halving the delay time on each press, or pressing Alt-S, which has the reverse effect. When you start to download a new file, the initial delay time will revert to this value. The default value is 0 milliseconds.

Communication mode; Transparent or Checksum

This determines how PANterm communicates with the control system. Under transparent mode, PANterm will send characters exactly as they are entered on the keyboard, or extracted from a file. Under checksum mode, PANterm always transmits a checksum after each <CR> character. It then waits for a <ACK> or <NAK> character. If <ACK> is received, then the next line is transmitted. If <NAK> is received, then line is re-transmitted.

Software or Hardware Handshaking

If set to software handshaking, PANTERM will use the *XON* and *XOFF* characters for flow control. If set to hardware handshaking, PANTERM will use to dedicated hardware lines, *CTS* and *RTS* for flow control. Hardware handshaking is only allowed when the RS-232 serial mode has been selected.

Colour or Mono display

You should set this to colour if you have a colour screen, for better presentation. The default is mono.

Even parity or none

This should normally be set to Even. This performs parity checking on incoming data and generates a parity bit for outgoing data.

Editor name

This item is used to set the editor that you wish to use to edit the control system command files. By default, it is set to 'ed'.

11.8 Automatic baud rate configuration

PANterm has the facility to set its baud rate automatically to that of the controller to which it is connected. Having established the baud rate of the controller, PANterm sets the baud rate of the PC to match, and updates the PANterm configuration file accordingly. This feature is invoked by typing Alt-B.

11.9 PANTERM command summary

Alt-B	(Baud rate set) Sets PANTERM to the baud rate of the controller automatically.
Alt-C	(Configure) Allows configuration PANTERM.
Alt-D	(Dos) Starts an MS-DOS shell.
Alt-E	(Editor) Temporarily leaves terminal mode and loads your editor for editing control system command files.
Alt-F	(Faster) Increases the speed at which command files are sent to the control system.
Alt-H	(Help) Display a list of possible commands and other help information on screen.
Alt-L	(Load) Starts loading of a control system command file. When your file has been loaded, the program will revert to the terminal mode.
Alt-R	(Record) Allows recording of control system commands to a file. You will continue to record all commands until you repeat the Alt-R command.
Alt-S	(Slower) Slows the speed at which command files are sent to the control system.
Alt-T	(Terminate) Terminates loading of a control system command file and returns to terminal mode.
Alt-U	(Upload) Starts loading of a control system S-record file. This is used for changing the system low level programme, and can only be used from the boot controller programme using the UP command. When the file has been loaded, the program will revert to the terminal mode.
Alt-X	(eXit) Exits the PANterm terminal program and returns to MS-DOS.

12 APPENDIX

12.1 Sample programme listing

```
#
#  PANMC/V154.3  Rev 6th July 1999
#  Pan Controls PC3/100 board, Issue F - No of wait states = 0
#  AMD 1048576 bytes flash memory
#  Operator Interface V106.5 OI 19980530 PC3/120 Issue C
#
#
#          Hardware; PC3/100 Host control board
#
#
#
# The machine is a 3 axis flexographic printer with axes defined,
#
#     Desc.          CH3           CH2           CH1
# Master/Slave     Master        Slave 1       Slave 2

# Axis             Web          Colour 1(7)   Colour 2(8)
# Encoder [ppr]    1000
# Counts/rev       4000      6.75*4*       6.75*4*
# Distance/rev  444.5mm (17.5") 1 repeat      1 repeat
# Counts/mm
#
# The type roll is driven via a 140t 1/4" pitch gear mounted on the
# impression roll whose circumference is 889mm (35").
# Set slave motor to 280ppr gives 1120 cts/rev so 35" is approx 7560cts
# Master is 4000 cts for 17 1/2" so gives 8000 cts per 35" an approximate
# ratio of 1:1. This is a fixed nominal ratio.
# Store web/slave counts/inch *1000 (cts/0.001") multiply by repeat (in 0.01")
# then divide by 100000
# G = 15483, H = 15483  Ratio starting values for FM FD = 14
# G = 30966, H = 30966  Ratio starting values for FM FD = 15
#
# Modification History;
#
# Mod [28/07/99]
#           [Link ratio resolution increased to achive finer correction
#             using FD15 (32768) and FM 30966]
# Mod [06/08/99]
#           [Added F4 to allow operation with no registration]
# Mod [10/08/99]
#           [Made fine adv/ret 3 was 10]
#
# Mod [11/08/99]
#           [Added a jog mode on F5 for type roll clean up operation
#             making use of the advance/retard buttons with the colour
#              select switch
#                conditions; ..Machine must be in machine mode
#                            ..Press must not be inched during this mode
#                 When ESC is used to quit jog rolls will realign to last
#                  position before jog was used]
#           [also modified S122 to pass the reg. error to both colours
#             simultaneously using pointer system >, < ]
```

```
#
# Variable used list,
#  A.."Repeat Length    mm "
#  B..Ch2 DS colour 1 error
#  C..
#  D..Scaled colour 1 Ch2 FM
#  E..
#  F..Slave bounds used to calculate adv/ret shift
#  G..Nominal FM for Col 1
#  H..
#  I..CH2 running average total
#  J..
#  K..Desired offset, ch 2
#  L..
#  M..Error (signed), ch 2
#  N..
#  O..
#  P..Cumulative CH3 DF 5 samples
#  Q..Correction for one shot PGC
#  R..Repeat for display use
#  S..Speed for display use
#  T..Registration error for display use
#  U..Colour 1 error multiplier for FM
#  V..Colour 1 error divisor for FM
#  W..Web bounds used to monitor actual repeat
#  X..Display actual repeat integer
#  Y..Display actual repeat decimal
#  Z..Number of samples for reg correction
#
#  a..Constant -1
#  b..Constant  0
#  c..Constant  1
#  d..Constant  2
#  e..Constant  3
#  f..Constant  4
#  g..Constant  5
#  h..Constant  60
#  i..Constant  216000  (Counts per 1000")
#  j..Constant  100
#  k..Number of samples before correction
#  l..Constant 100000
#  m..Web modified bounds
#  n..Slave modified bounds
#  o..Correction limit value [S122, S123]
#  p..Counts per foot for speed conversion S200
#  q..calculated reg error to load into Q
#  r..Reset initialisation average total
#  s..Set RE sequence trigger to initialisation sequence
#  t..Testing variable
#  u..
#  v..
#  w..Initialisation offset value for registration
#  x..Reset initialisation value for x
#  y..run mode flag
#  z..Division remainder
#
#
# INPUTS to Panax from machine devices,
#    1..Web registration detector
#    2..Colour 1 registration detector
#    3..Colour 2 registration detector
#    4..
#    5..Coarse advance/retard inc HS
#    6..Fine advance/retard inc HS
#    7..Phase advance HS
#    8..Phase retard HS
#    9..Select colour 2 for advance/retard HS
#   10..E.Stop has been reset
#   11...
#   12..
#   13..
#   14..
#   15..
#   16..
#   18..
#   19..
#   20..
#   21..
#   22..
#   23..
#   24..
#   25..
#   26..
#   27..
#   28..
#   29..Stepper drives colour 2 ready SA2R X1.15 [144]
#   30..Stepper drives colour 1 ready SA1R X1.15 [145]
#   31..Drive 2 ready TDA2 X2.4 [146]
#   32..Drive 1 ready TDA1 X2.4 [147]
#
# OUTPUTS from Panax to machine interface
#    1..
#    2..
#    3..
#    4..
#    5..
#    6..Control OK monitor relay R1 [123]
#    7..
#    8..
#    9..
#   10..
#   11..
#   12..
#   13..
#   14..
#   15..
#   16..
#   17..
#   18..
#   19..
#   20..
#   21..
#   22..
#   23..
#   24..
#   25..
#   26..
#   27..
#   28..
#   29..
#   30..
#   31..
#   32..
#
# SEQUENCE LIST
#     1..
#     20...Escape key sequence
#     30...Start screen menu
#     36...Display screen with registration
#     38...Display screen with registration disabled
#     40...Edit sequence
#     41...Reset all working values sequence
#     50...Starting up screen  Ms
#     51...Colour 1 drive ready  Ms
#     52...Colour 2 drive ready  Ms
#     53...Colour 1 steppers ready  Ms
#     54...Colour 2 steppers ready  Ms
#     55...Startup complete Ms
#     56...Reset E.Stop prompt
#     59...Power up header
#     60...Diagnostic/service screen
#     70...Phase advance print setup
#     71...Phase advance print
#     72...Phase retard print setup
#     73...Phase retard print
#     76...Jog positive off Advance button
#     77...Jog negative off Retard button
#     120..Initialisation of registration seq called via RE
#     122..Fine reg correction col 1
#     123..Fine reg correction col 2
#     124..Adjust the repeat length
#     140..One shot correction sequence
#     145..One shot correction sequence complete colour 1
#     146..One shot correction sequence complete colour 2
#     200..Tick driven sequence
#     201..Machine settings update
#     202..Set new ratio
#     203..Change direction mode
#     204..E.stop relink
#     250..Motor off sequence
#     252..E.stop Auto restart sequence
#     253..Default system parameters called from S255
#     254..Default variable values called from S255
#     255..Autostart system setup
#
#
#
EV"BEMIS54.PAN"
DW100010

#**************************************************************************
#*                            MAIN MENU                                   *
#**************************************************************************

ES20                    # Escape key sequence
CE0
ABK/XS30
```

```
ES30                            # Start screen
IVy5                            #Load y with run value
ABK                             #Abort keypad activity
EK/WT128                        #Enable keys + wait to clear pending tick
MD                              #Clear display & address each line
MD1"*** SYNPRO M. I. ***"
MD2" Use function keys  "
MD3" to select required "
MD4" option             "
SKA40                           #F1 key executes seq 40 [Edit]
SKB36                           #F2      ditto       36 [Run]
SKC60                           #F3      ditto       60 [Diag]
SKD38                           #F4      ditto       38 [Plain run]
SKE32                           #F5      ditto       32 [Jog mode]
SKK20                           #ESC     ditto       20
EK/MKE                          #Enable 'F' keys mask jog mode
SK0

#**************************************************************************
#*                            RUN SCREEN                                  *
#**************************************************************************

ES32                    # Display screen with jog facility F5
IF$43:b/XT              #If press is running go back to last menu
IVy-1/CE0               #Load y with jog value turn off update tick
MK/EKK/MI3/SKK34        #Mask F keys & register input
CH2/DPOu/ST             #Record colour 1 position
CH1/DPOv/ST             #Record colour 2 position
MD/CO7                  #Clear display & set output 7
MD1"      JOG MODE      "
MD2" Use Colour, +, -,  "
MD3" to jog type rolls  "
MD4"  Esc to Quit jog   "

ES34                    #Quit jog mode sequence
IVy0                    #Clear jog mode flag
MD                      #Clear display
MD1"      JOG MODE      "
MD2"Resetting type rolls"
MD3"   Please wait....  "
MD4"                    "
CH2/ST/MAVu/LM3         #Reset colour 1 type roll
CH1/ST/MAVv/LM3         #Reset colour 2 type roll
EI/SO7/SKK20/XS20       #Clear output 7 & go to menu header screen


ES36                    # Display screen with registration F2
IF$43:b/XS20/XT         #If press is running go back to menu
MK/EKK/EKE/EI           #Mask F Keys enable ESC & F5 [jog]
IVy2                    #Load y with run value
MD                      #Clear display
CE200/PL256             #Tick sequence per second
MD1"    PRINT MODE      "
MD2"Print Repeat  "A:"."o
MD3"Press speed   "n
MD4"Actual Repeat "X:"."Y

ES38                    # Display screen with registration disabled F4
IF$43:b/XS20/XT         #If press is running go back to menu
```

```
IVy3                    #Load y with run value
MK/EKK/MI1/MI2/MI3      #Mask F keys & register inputs
CH2/ST/FD15/FMVG/LM3    #Set up starting link
CH1/ST/FD15/FMVG/LM3    #Setup starting link
MD                      #Clear display
MD1"    PLAIN MODE      "
MD2"  Please note that  "
MD3"   Registration is  "
MD4"     DISABLED !!!   "

#***************************************************************************
#*                            EDIT MENU                                    *
#***************************************************************************

ES40                            #Edit sequence F1
IF$43:b/XS20/XT                 #1If press is running go back to menu
EK/MKA/MKB/MKC/MKD/MKE/MKK      #2 Mask F keys
IVy1                            #3 Load y with run value
MD                              #4
MD1"     EDIT MODE      "       #5
MD2"Enter Repeat length "       #6
MD3"1700 - 4700 inch*100"       #7
GL14                            #8 Go to line 13
MD1"     EDIT MODE      "       #9
MD2"Length out of range "       #10 Wrong length input message
MD3"1700 - 4700 inch*100"       #11
GL14                            #12 Go to line 13
MD3"xx00 xx25 xx50 xx75 "       #13 Wrong resolution entered [1/4 "]
VK4:%1                          #14
WK                              #15
IVt1700/IGt%1/GL9               #16 If length out of range go to line 8
IVt4700/IG%1:t/GL9              #17 If length out of range go to line 8
IVt25/VD%1:ttz/IFzb/GL13        #18 If resolution is not .25" go to line 12
VM%1:iF                         #Repeat *228343 (cts/1000")
VDFlFz                          #Divide by 1000 to get cts/repeat
VDFj%9:z                        #Bounds/100 = cts/0.01inch
VD%1:jAo                        #Scale repeat to integer & decimal
IVt228571/VM%1:tW               #Web repeat *228571 (cts/1000")
VDWlWz                          #Web divide by 1000 to get cts/repeat
CH1/SBVF                        #Set slave bounds to current repeat
CH2/SBVF                        #Set slave bounds to current repeat
CH3/SBVW                        #Set web bounds to current repeat
MD2
MD3
MD4
MD3"Saving settings...  "
SPV                             #Save values
XS30

ES41                    #Reset working values
IV%1:1700
IV%2:1
IV%3:2
IV%4:0

#***************************************************************************
#*                     STARTING UP MESSAGE SCREENS                         *
#***************************************************************************
```

```
ES50                     #Starting up screen
MD1"   JODE SYSTEMS     "
MD2"  Machine starting  "
MD3"   waiting for...   "

ES51                     #Colour 1 drive ready
MD
MD1"   JODE SYSTEMS     "
MD2"  Machine starting  "
MD3"   waiting for...   "
MD4"Colour 1 drive ready"

ES52                     #Colour 2 drive ready
MD
MD1"   JODE SYSTEMS     "
MD2"  Machine starting  "
MD3"   waiting for...   "
MD4"Colour 2 drive ready"

ES53                     #Colour 1 steppers ready
MD
MD1"   JODE SYSTEMS     "
MD2"  Machine starting  "
MD3"   waiting for...   "
MD4" Col 1 stepper ready"

ES54                     #Colour 2 steppers ready
MD
MD1"   JODE SYSTEMS     "
MD2"  Machine starting  "
MD3"   waiting for...   "
MD4" Col 2 stepper ready"

ES55                     #Startup complete
MD
MD1"   JODE SYSTEMS     "
MD2"  Machine starting  "
MD3"   waiting for...   "
MD4" Startup Completion "

ES56                     #Reset E.Stop prompt
MD1"   JODE SYSTEMS     "
MD2"  Machine starting  "
MD3"   waiting for...   "
MD4"E.Stop to be reset "

ES59                            #Power up header
MD                              #Clear display & address each line
MD1" Control by         "
MD2"     JODE SYSTEMS   "
MD3"  for               "
MD4"    SYN             "
WT200
MD4"    SYNPRO          "
WT200
MD4"    SYNPRO M.       "
WT200
MD4"    SYNPRO M.I.     "
WT500
```

```
#**************************************************************************
#*                        DIAGNOSTIC SCREEN                               *
#**************************************************************************

ES60                    #Diagnostic/service screen F3
MK/EKK                          #Mask F2 & F3
IVy4                            #Load y with run value
MD                              #Clear display
MD2"   PANMC/V154.6     "
MD3" Rev 15th Sep 1999  "
MD4"    BEMIS54.PAN     "

#**************************************************************************
#*         START OF PUSH BUTTON ADVANCE & RETARD CONTROL                  *
#**************************************************************************

ES70                    #Advance setup sequence
MI7/MI8/EK/MKA/MKB/MKC/MKD/MKE/CE0/RE0  #Mask F keys
IEay/XS76/GA/EI/EK/XT           #If in jog mode do Seq76
II5-/II6+/IVQ230                #Set coarse adv/ret increment
II5+/II6+/IVQ100                #Set medium adv/ret increment
II5+/II6-/IVQ3                  #Set fine adv/ret increment
XS140                           #Set advance/retard tables if req.
XS71

ES71                            #Phase advance print
CH3/CE0/RE0                    #Mask advance & retard push buttons
IE%5:%17/CH2/PGP               #Positive shift on colour 1
IE%5:%18/CH1/PGP               #Positive shift on colour 2
IEyd/IGQj/MD1"  ADVANCING..coarse "
IEyd/IEQj/MD1"  ADVANCING..medium "
IEyd/IGjQ/MD1"  ADVANCING..fine   "

ES72                    #Retard setup sequence
MI7/MI8/EK/MKA/MKB/MKC/MKD/MKE/CE0/RE0  #Mask F keys
IEay/XS77/GA/EI/EK/XT           #If in jog mode do Seq77
II5-/II6+/IVQ230                #Set coarse adv/ret increment
II5+/II6+/IVQ100                #Set medium adv/ret increment
II5+/II6-/IVQ3                  #Set coarse adv/ret increment
XS140                           #Set advance/retard tables if req.
XS73

ES73                            #Phase retard print
CH3/CE0/RE0                    #Mask advance & retard push buttons
IE%5:%17/CH2/PGN               #Negative shift on colour 1
IE%5:%18/CH1/PGN               #Negative shift on colour 2
IEyd/IGQj/MD1"  RETARDING..coarse "
IEyd/IEQj/MD1"  RETARDING..medium "
IEyd/IGjQ/MD1"  RETARDING..fine   "

ES76                            #Jog positive off Advance button
IE%5:%17/CH2/SV1750/VC+         #Set jog speed & direction col 1
IE%5:%17/WT64/II7-/RP           #Test advance button still pressed
IE%5:%17/CH2/ST/XT              #Stop colour 1 on button release
IE%5:%18/CH1/SV1750/VC+         #Set jog speed & direction col 2
IE%5:%18/WT64/II7-/RP           #Test advance button still pressed
IE%5:%18/CH1/ST/XT              #Stop colour 2 on button release
CH2/ST/CH1/ST/GA/EI             #Stop trap if colour changed during jog
```

```
ES77                               #Jog negative off Retard button
IE%5:%17/CH2/SV1750/VC-            #Set jog speed & direction col 1
IE%5:%17/WT64/II8-/RP              #Test retard button still pressed
IE%5:%17/CH2/ST/XT                 #Stop colour 1 on button release
IE%5:%18/CH1/SV1750/VC-            #Set jog speed & direction col 2
IE%5:%18/WT64/II8-/RP              #Test retard button still pressed
IE%5:%18/CH1/ST/XT                 #Stop colour 2 on button release
CH2/ST/CH1/ST/GA/EI                #Stop trap if colour changed during jog

#***************************************************************************
#*         START OF INITIALISATION OF REGISTER POSITION                    *
#***************************************************************************

ES120                      #Initialisation of registration seq called via RE
IE$43:b/CH3/RE120/EI/XT #If press is stopped reset reg. and eXiT
MI3/VA%12:c%12:            #Mask reg input & inc counter (%12 )
CH2/DSOB/VSKBM/VAMII       #Get current reference error into variable B
IG%12:Z/XS122              #If average total complete do correction col 2
CH3/DFOR/VAPRP             #Collect DF samples
VAkck/IGk%10:/XS124        #Increment k to average error
CH3/RE120/EI               #Rearm reg seq trigger

ES122                      #Fine reg correction col 1
VDI%12:Mz                  #Divide total error by number of samples
IGM%16:/IVM1500            #Limit correction to + value approx 6" coarse
IG%15:M/IVM-1500           #Limit correction to - value approx 6" coarse
IGM%15:/IG%16:M/VMMUM/VDMVMz  #Fine correction scaling
VAMGD/AP>2:VD/AP>1:VD      #Scale FM load AP for col 1 & col 2
FM</IVIO/IV%12:0           #Action FM on both channels using AP value

ES124                     #Adjust the repeat length
VDPkPz/VSWPm/VMm%1:%11:/VD%11:W%11:z  #Calc. act. rpt. on web in %11
IVk0/IVP0/VD%11:jXY   #Reset accumulator totals & calc. display
IG%10:Y/MD4:15:"     "  #Refresh only if Y < 10
IEyd/MD4"Actual Repeat "X:"."Y

#***************************************************************************
#*                START OF ADV/RET POSITION CONTROL                        *
#***************************************************************************

ES140                     #One shot correction sequence
CH1
CUC3000/CLC600/PGCVQ  #Load correction with variable Q
CH2
CUC3000/CLC600/PGCVQ  #Load correction with variable Q

ES145                      #One shot correction sequence complete colour 1
EI/EKE                     #Enable all inputs
CH3/RE120
IEyd/MD1"    PRINT MODE      "   #Clear advance/retard message

ES146                      #One shot correction sequence complete colour 2
EI/EKE                     #Enable all inputs
CH3/RE120
IEyd/MD1"    PRINT MODE      "   #Clear advance/retard message

#***************************************************************************
#*                     START OF TICK DRIVEN ACTIVITY                       *
#***************************************************************************
```

```
ES200                   #Tick driven sequence
VM$43:hn/VDnpnz         #System var.$43 to cts/min then per foot
IG%99:n/MD3:15:"     "  #Refresh only if n < 99
IEyd/MD3"Press speed   "n    #Update display if in run page
IG%10:Y/MD4:15:"     "  #Refresh only if Y < 10
IEyd/MD4"Actual Repeat "X:"."Y
IEnb/XS201              #When press speed is 0 update settings

ES201                   #Machine settings update
IF%1:%14:/XS202         #If repeat length has changed set new ratio
IE%250:c/XS204          #Relink after E.stop
CE0                     #Disable tick

ES202                   #Set new ratio
EI1/EI2/EI3
VA%1:b%14:/IVk0         #Copy repeat length into %14 reset offset correction
CH2/ST/FD15/FMVG/LM3    #Set up starting link
CH1/ST/FD15/FMVG/LM3    #Setup starting link
VAGbD/VAGbE             #Copy starting ratio to working reg.
CE0/RE120               #Turn off tick arm reg

ES204                   #E.stop relink
IV%250:0
CH2/ST/PC/LM3           #Relink colour 1
CH1/ST/PC/LM3           #Relink colour 2

#**************************************************************************
#*                      ERROR REACTION ACTIVITY                          *
#**************************************************************************

ES250                   #Motor off sequence
CE0/MI3
CH2/MO
CH1/MO
IV%250:1                #Set an E.stop marker
IVk0                    #Reset average counter
XS252                   #E.stop Auto restart sequence

#**************************************************************************
#*                START OF POWER UP AUTOSTART ACTIVITY                   *
#**************************************************************************

AS255

ES252                   #E.stop Auto restart sequence
MI1
MD                      #Clear display
XS50                    #Start screen
XS56                    #Reset E.Stop prompt
II10+/RP                #Wait for reset
XS51                    #Drive ready prompt
MI                      #Mask inputs
WI32-                   #Drive 1 ready TDA1 X2.4 [147]
XS52
WI31-                   #Drive 2 ready TDA2 X2.4 [146]
XS53
WI30-                   #Stepper drives colour 1 ready SA1R X1.15 [145]
XS54
WI29-                   #Stepper drives colour 2 ready SA2R X1.15 [144]
```

```
WT512                   #Delay the enable signal
CH3/RE120               #Set reference trigger sequence to initialise
XS30                    #Start screen
EI                      #Enable inputs

ES253                   #Default system parameters for testing
CH2/GE145/FD15          #Set reference complete trigger sequence colour 1
LW111110/RA0            #+/- tables, 16 bit num., DF/DS average,use demand
SA60000/SZ80000         #Acc/Dec
KP100/KF970             #Gain terms
SE8000/SW100            #Error
CW1011011               #Setup control word
PC/WT64/ID              #Enable drive and initialise demand RF [X4.2]
DB25/GW110              #Debounce inputs
SBVF                    #Set slave bounds to current repeat
CH1/GE146/FD15          #Set reference complete trigger sequence colour 2
LW111110/RA0            #+/- tables, 16 bit num., DF/DS average,use demand
SA60000/SZ80000         #Acc/Dec
KP100/KF970             #Gain terms
SE8000/SW100            #Error
CW1011011               #Setup control word
PC/WT64/ID              #Enable drive and initialise demand RF [X4.2]
SBVF                    #Set slave bounds to current repeat
CH3/SBVW                #Set web bounds to current repeat
AP>3:1                  #Set a default value for CH3 FM
VT5                     #Average the web velocity to smooth out

ES254                   #Default variable values
IE%1:b/IV%1:2100        #If the saved repeat is 0 then default to 21"
VA%1:bA                 #Copy %1 to A (add 0 to %1)
VD%1:jAo                #Scale repeat to integer & decimal
IV%10:10                #Lower one shot value also counter 'k' target
IV%500:500              #Upper one shot value
IV%99:99                #Upper limit to refresh display
VM%9:%4:Q               #Shift value
IV%11:0                 #Starting actual repeat value
IV%12:0                 #Reset reg average counter 1
IV%13:0                 #Reset reg average counter 2
IV%14:0                 #Reset first time stored length comparison
IV%15:-1500             #Lower coarse band
IV%16:1500              #Upper coarse band
IV%17:1                 #Constant
IV%18:2                 #Constant
IVG30966                #Ratio starting value
IVI0                    #Starting error average values
IVK2000              #Offset, ch 2
IVP0                    #Starting DF accumulator
IVU10/IVV15             #Fine error gain values
IVX0/IVY0               #Starting actual display
IVZ3                    #Number of samples for reg correction
IVa-1                   #Constant
IVb0                    #Constant
IVc1                    #Constant
IVd2                    #Constant
IVe3                    #Constant
IVf4                    #Constant
IVg5                    #Constant
IVh60                   #Constant seconds to min.
IVi216000               #Constant slave counts per 1000"
```

```
IVj100                  #Constant
IVk0                    #Number of samples before correction counter reset
IVl100000               #Constant for slave repeat correction
IVp2473                 #Constant for counts per foot on the web
IVr0                    #Reset initialisation average total
IVs120                  #Set RE sequence trigger to initialisation sequence
IVx0                    #Reset initialisation value for x
IVw0                    #Reset increment total value
VM%1:iF                 #Repeat *228343 (cts/1000")
VDFlFz                  #Divide by 1000 to get cts/repeat
VD%1:jAo                #Scale repeat to integer & decimal
IVt228571/VM%1:tW       #Web repeat *228571 (cts/1000")
VDWlWz                  #Web divide by 1000 to get cts/repeat

ES255                   #Autostart sequence
MI1
XS59                    #Startup Synpro banner
MD                      #Clear display
XS50                    #Start screen
XS56                    #Reset E.Stop prompt
CO8                     #Set control OK
II10+/WT64/RP           #Wait for reset
WT350                   #Delay to clear stepper startup
XS51                    #Drive ready prompt
MI                      #Mask inputs
WI32-                   #Drive 1 ready TDA1 X2.4 [147]
XS52
WI31-                   #Drive 2 ready TDA2 X2.4 [146]
XS53
WI30-                   #Stepper drives colour 1 ready SA1R X1.15 [145]
XS54
WI29-                   #Stepper drives colour 2 ready SA2R X1.15 [144]
XS55                    #Waiting startup complete message
XS254                   #Default variable values
WT512                   #Delay the enable signal
II9+/IV%5:1             #Set start up shift colour 1 selection
II9-/IV%5:2             #Set start up shift colour 2 selection
XS253                   #Default system settings
XS140                   #Set default shift
XS30                    #Start screen
EI                      #Enable inputs
CH3/RE120/RW1           #Set reference trigger sequence to initialise

#DEFINED INPUTS
CH3/DR3-                #Reference input on master axis
CH2/PS2-                #Slave axis position snapshot [linked to input 1]
DI7-/XS70               #Advance Pb.
DI8-/XS72               #Retard Pb.
DI9-/IV%5:2             #Set colour 2 for shift
DI9!/IV%5:1             #Set colour 1 for shift
DI10+/XS250             #E.stop monitor to MO motors
#CH1/PS1-                #Slave axis position snapshot [linked to input 1]

#POSITION TRIGGERED OUTPUT
#CH1/POn-/m/p

# System diagnostic sequences if required

#ES101                            #Channel 1 system test
```

```
#OV$1                           #Channel 1 status
#OV$9                           #Channel 1 mode
#OV$17                          #Channel 1 error
#OV$25                          #Channel 1 bound count
#OV$33                          #Channel 1 position
#OV$41                          #Channel 1 velocity
#OV$49                          #Channel 1 position error

#ES102                          #Channel 2 system test
#OV$2                           #Channel 2 status
#OV$10                          #Channel 2 mode
#OV$18                          #Channel 2 error
#OV$26                          #Channel 2 bound count
#OV$34                          #Channel 2 position
#OV$42                          #Channel 2 velocity
#OV$50                          #Channel 2 position error
```

12.2 Operator interface keypad codes

Backspace (X)	08
Enter	13
ESC	108
·	46
-	45
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
Left arrow	109
Right arrow	110
Up arrow	111
Down arrow	112
Secret key	113
Centre dot key	114
F1	97
F2	98
F3	99
F4	100
F5	101
F6	102
F7	103
F8	104
Left display	105
Centre display	106
Right display	107
F9	115
F10	116
F11	117
F12	118
F13	119
F14	120
F15	121
F16	122

12.3 Operator interface mask and sequence parameters

Parameter	Operator interface Mk2		Operator interface Mk1	16-key Keypad
A	F1		F1	A
B	F2		F2	B
C	F3		F3	C
D	F4		F4	D
E	F5		F5	E
F	F6			F
G	F7			
H	F8			
I	Left display			
J	Centre display			
K	Right display			
L	ESC		ESC	
M	Left arrow			
N	Right arrow			
O	Up arrow			
P	Down arrow			
Q	Secret key			
R	Dot key			
S	F9	(top left)		
T	F10	(second top left)		
U	F11	(third top left)		
V	F12	(fourth top left)		
W	F13	(top right)		
X	F14	(second top right)		
Y	F15	(third top right)		
Z	F16	(fourth top right)		

12.4 Error codes

0	None

General errors 1-7

1	Unknown command
2	Invalid command entry
3	Command not allowed at this time
4	Cannot change parameter at this time
5	Parameter out-of-range
6	Restricted command
7	Restricted parameter

Reference errors

11	No reference input defined

Memory errors

12	No more room in memory for sequences/profiles
13	No more room for DI strings
14	Stack overflow

Nvm and checksum errors

15	Nvm write timed out
16	Nvm verify failed
17	Data overflowed save space
18	Calculated checksum differs from saved
19	Cannot calculate checksum

Move errors

20	Move target outside limits
21	Failed to reach final position window
22	Moving away from specified wait position (WR)

Sequence/profile errors

23	Undefined sequence
24	Undefined profile
25	Profile step data too large
26	Cannot enter a sequence while running any sequence
27	Cannot enter profile while it is in use
28	Cannot execute command unless in sequence
29	Cannot goto sequence line no

I/O line errors

30	Line in use (busy)
31	Line not yet defined
32	Input line too noisy (II)
33	2nd posn must be > 1st (PO)
34	No output group defined (OC)
35	Not an output (DSC-2 RO)
36	Cannot use this input while initialising (DSC-2)

Numeric errors

37	Binary no. expected
38	Decimal no. expected
39	Hex no. expected

40	Numeric overflow

Other errors

41	Password incorrect
42	No commands before ...
43	No commands after ...
44	Only 1 repeat per line
45	Cannot execute string while busy
46	Command decode error
47	Variable name out of range
48	Time out for auxiliary serial port
49	Undefined bounds table
50	Already data logging
51	Logging frequency of zero
52	Timeout waiting for new code
53	Illegal circle parameters
54	System variable, write only
55	System variable, read only
56	Reading default params (LK 29)
57	Calculated checksum for variables differs from saved
58	Already doing VK
59	Pos gain table parameters inconsistent
60	Undefined offset table
61	Calculated sequence checksum differs from saved checksum
62	Time out for camera

Motor off errors

107	Limit switch errors from 1 to here
108	Reference timeout error
109	Reference error outside limits
110	Reference error correction overrun
111	Position error
112	Timeout error
113	High position limit error
114	Low position limit error

Motor off errors unique to multi-axis version

115	Map update timeout error
116	Map position overflow (out of range)
115	Bounds limit error

12.5 ASCII Table

(American Standard Code for Information Interchange)

ASCII	EQUIVALENT FORMS			
Char.	Binary	Oct	Hex	Dec
NUL	00000000	000	00	0
SOH	00000001	001	01	1
STX	00000010	002	02	2
ETX	00000011	003	03	3
EOT	00000100	004	04	4
ENQ	00000101	005	05	5
ACK	00000110	006	06	6
BEL	00000111	007	07	7
BS	00001000	010	08	8
HT	00001001	011	09	9
LF	00001010	012	0A	10
VT	00001011	013	0B	11
FF	00001100	014	0C	12
CR	00001101	015	0D	13
SO	00001110	016	0E	14
SI	00001111	017	0F	15
DLE	00010000	020	10	16
DC1	00010001	021	11	17
DC2	00010010	022	12	18
DC3	00010011	023	13	19
DC4	00010100	024	14	20
NAK	00010101	025	15	21
SYN	00010110	026	16	22
ETB	00010111	027	17	23
CAN	00011000	030	18	24
EM	00011001	031	19	25
SUB	00011010	032	1A	26
ESC	00011011	033	1B	27
FS	00011100	034	1C	28
GS	00011101	035	1D	29
RS	00011110	036	1E	30
US	00011111	037	1F	31
space	00100000	040	20	32
!	00100001	041	21	33
"	00100010	042	22	34
#	00100011	043	23	35
$	00100100	044	24	36
%	00100101	045	25	37
&	00100110	046	26	38
'	00100111	047	27	39
(	00101000	050	28	40
)	00101001	051	29	41
*	00101010	052	2A	42
+	00101011	053	2B	43
,	00101100	054	2C	44
-	00101101	055	2D	45
.	00101110	056	2E	46
/	00101111	057	2F	47
0	00110000	060	30	48
1	00110001	061	31	49
2	00110010	062	32	50
3	00110011	063	33	51
4	00110100	064	34	52
5	00110101	065	35	53
6	00110110	066	36	54
7	00110111	067	37	55
8	00111000	070	38	56
9	00111001	071	39	57
:	00111010	072	3A	58
;	00111011	073	3B	59
<	00111100	074	3C	60
=	00111101	075	3D	61
>	00111110	076	3E	62
?	00111111	077	3F	63

ASCII	EQUIVALENT FORMS			
Char.	Binary	Oct	Hex	Dec
@	01000000	100	40	64
A	01000001	101	41	65
B	01000010	102	42	66
C	01000011	103	43	67
D	01000100	104	44	68
E	01000101	105	45	69
F	01000110	106	46	70
G	01000111	107	47	71
H	01001000	110	48	72
I	01001001	111	49	73
J	01001010	112	4A	74
K	01001011	113	4B	75
L	01001100	114	4C	76
M	01001101	115	4D	77
N	01001110	116	4E	78
O	01001111	117	4F	79
P	01010000	120	50	80
Q	01010001	121	51	81
R	01010010	122	52	82
S	01010011	123	53	83
T	01010100	124	54	84
U	01010101	125	55	85
V	01010110	126	56	86
W	01010111	127	57	87
X	01011000	130	58	88
Y	01011001	131	59	89
Z	01011010	132	5A	90
[	01011011	133	5B	91
\	01011100	134	5C	92
]	01011101	135	5D	93
^	01011110	136	5E	94
_	01011111	137	5F	95

ASCII	EQUIVALENT FORMS			
Char.	Binary	Oct	Hex	Dec
`	01100000	140	60	96
a	01100001	141	61	97
b	01100010	142	62	98
c	01100011	143	63	99
d	01100100	144	64	100
e	01100101	145	65	101
f	01100110	146	66	102
g	01100111	147	67	103
h	01101000	150	68	104
i	01101001	151	69	105
j	01101010	152	6A	106
k	01101011	153	6B	107
l	01101100	154	6C	108
m	01101101	155	6D	109
n	01101110	156	6E	110
o	01101111	157	6F	111
p	01110000	160	70	112
q	01110001	161	71	113
r	01110010	162	72	114
s	01110011	163	73	115
t	01110100	164	74	116
u	01110101	165	75	117
v	01110110	166	76	118
w	01110111	167	77	119
x	01111000	170	78	120
y	01111001	171	79	121
z	01111010	172	7A	122
{	01111011	173	7B	123
|	01111100	174	7C	124
}	01111101	175	7D	125
~	01111110	176	7E	126
DEL	01111111	177	7F	127

12.6 Operator interface template

The following template can be copied and used as a template for creating new screens.

DISPLAY TEMPLATE

13 INDEX