Mclennan Servo Supplies Ltd.

PM304 Mk.II Servo Controller
User Manual

USER'S MANUAL FOR PM304 Digiloop ® MkII

SAFETY NOTICE !

Position control systems are inherently hazardous. Even a small motor, if coupled to a leadscrew, gearbox, or any other form of mechanism that provides a mechanical advantage, can generate considerable force and could cause serious injury. Incorrect operation can also lead to damage to the motor or associated machinery. It is essential that the purpose of the various safety features built into the PM304 be fully understood and used correctly.

Caution

STATIC SENSITIVE DEVICES

This unit has static sensitive devices. Observe handling precautions: Hold card by edges only. Do not touch connector pins. Ship only in anti-static packaging.

Mclennan Servo Supplies Ltd.
Unit 1, The Royston Centre,
Lynchford Road,
Ash Vale,
GU12 5PQ UK

Telephone: +44 (0)8707 700 700
Fax: +44 (0)8707 700 699

This manual is written for ROM version 6.15

The manufacturer reserves the right to update the data used in this manual in line with product development without prior notice.

CONTENTS

1. OVERVIEW

The PM304 *Digiloop* controller is an RS232 interfaced, Eurocard format, intelligent motor controller for DC brushed or brushless construction servo motors. It utilises encoder feedback to continually monitor position, speed and direction; in most cases no tachometer is necessary.

Features

The PM304 allows the user to control motor position, velocity, acceleration and deceleration, using commands in ASCII format, sent down the RS232 data path.

The PM304 may be used to slave the speed of its motor to another quadrature signal at user defined ratios. This is known as *electronic gearbox* mode. These quadrature signals can come from an encoder on another motor or a manually driven device such as a trackerball.

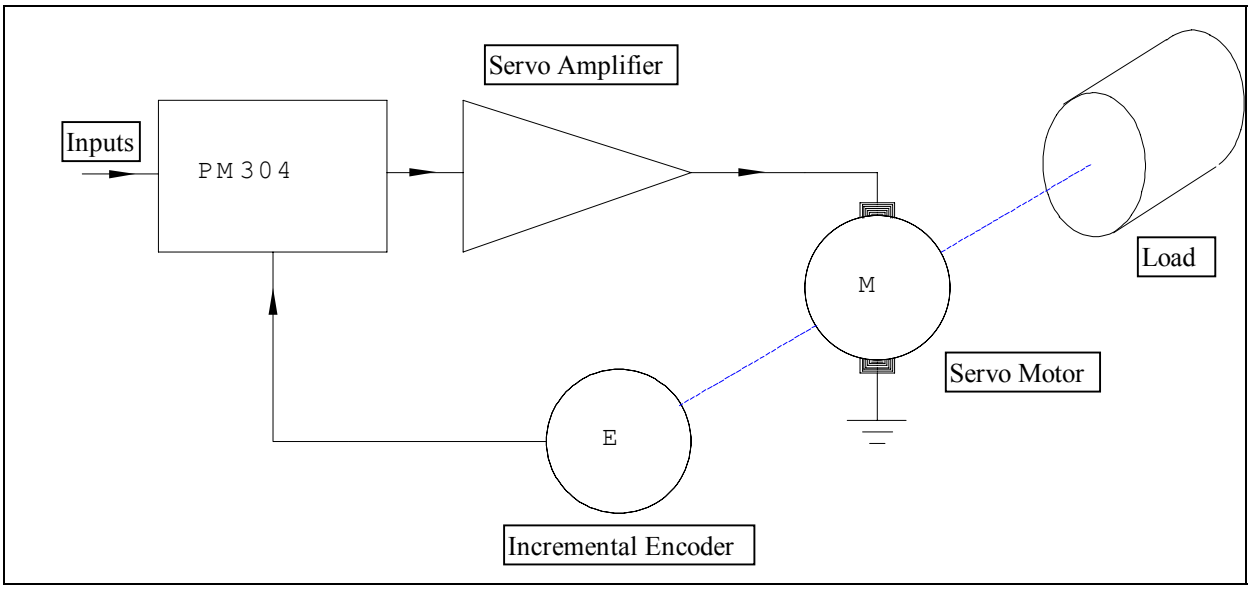
A number of PM300 series controllers (PM301, PM304, PM341, PM381, PM368) may be daisy chained along the data bus. Each command is prefixed by a number defining the axis to be addressed. A PM304 will pass on all commands and only act upon a command prefixed by its own address.

Strings of such commands may be sent directly from a host computer or processor in immediate *real time* mode, or loaded into the on-board memory of the PM304 and executed in sequence. In either mode the controller may be interfaced to external devices via its I/O facilities to take account of various contingencies, and provide a handshake with other machine functions.

An on board switched-mode regulated power supply allows the PM304 to be supplied from an unregulated DC source of between +10V and +32V.

Two types of limits are provided; *hard* limits and *soft* limits as well as an emergency stop input.

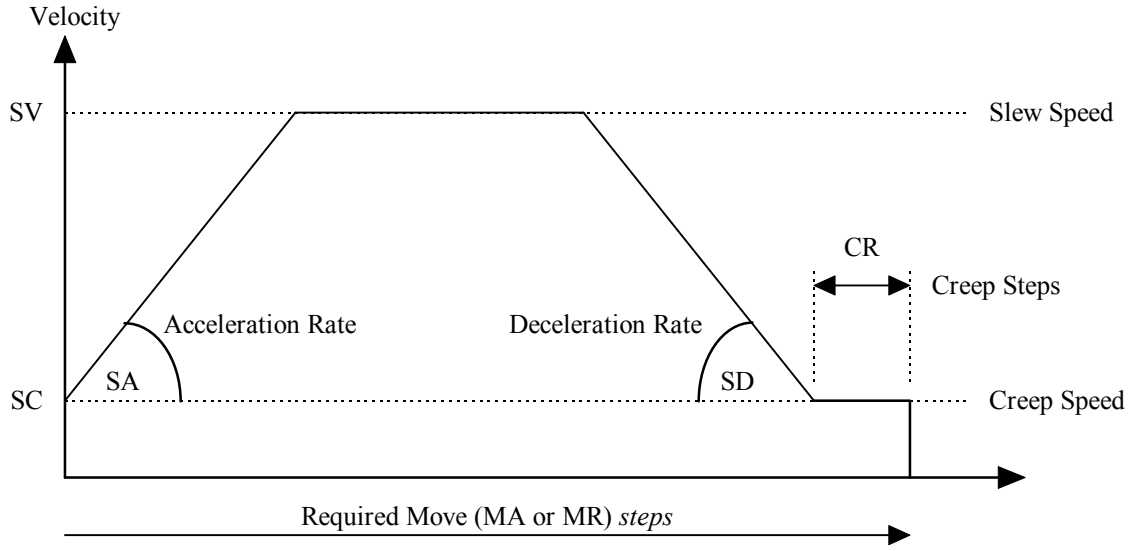
System Block Diagram



The minimum arrangement for a PM304 system consists of a servo motor fitted with an incremental encoder, a servo amplifier, a PM304 servo controller and suitable system power supplies (not shown). The PM304 generates a *command* position and compares it with the *actual* position read from the incremental encoder. Any error between these two is used by the PM304 to calculate an error voltage that is feed to a servo amplifier. The servo amplifier in turn drives the servo motor that drives the load to the correct position.

Typical Movement Profile

A typical move of a PM304 controller is made by either the **MA** (*move absolute*) or **MR** (*move relative*) command. The speed will the ramp up linearly at the rate defined by the **SA** (set acceleration) command, until the *slew speed* is reached (programmable by the **SV** command). It will continue at this speed until it decelerates at the **SD** rate, and then finish the last steps at the *Creep Speed* defined by the **SC** (set creep speed) command. The number of these last *Creep steps* is defined by the **CR** command.



The end of a move is defined as having occurred when the actual position has settled close to the required position. The distance at which the motor starts to *settle* is set by the **WI** (end of move window) command. For the move to be defined as being complete, the motor must *settle* within the end of move window for the time set by the **SE** (settle time) command.

2. INSTALLATION

2.1 Physical Installation:

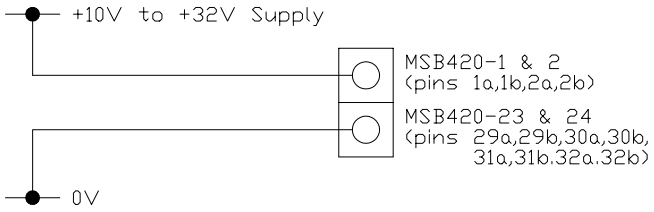
The controller is constructed on a single EXTENDED EUROCARD standard printed circuit board. The dimensions of the PCB are 100mm x 220mm. It is designed for mounting in a 3U high 19" rack and is fitted with front panel that is 7HP (35.2mm) wide.

Connections are made via a 64 pin a & b DIN41612 type B connector. A mating half connector may be fitted in the 19" rack or preferably use a Mclennan PCB motherboard. The MSB420 motherboard has been specifically designed for installation of the PM304 controller. It has the DIN41612 socket on one side and plug-in screw terminals on the other for external connections.

2.2 Connections:

POWER SUPPLY

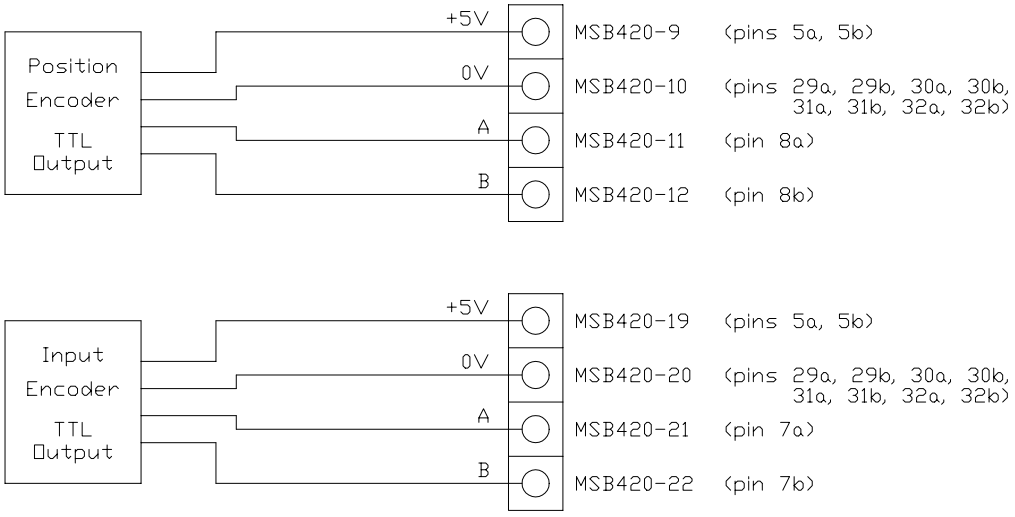
The unit requires a single unregulated DC supply of between +10V and +32V. THIS UNIT MUST NOT BE REVERSE POLARISED!



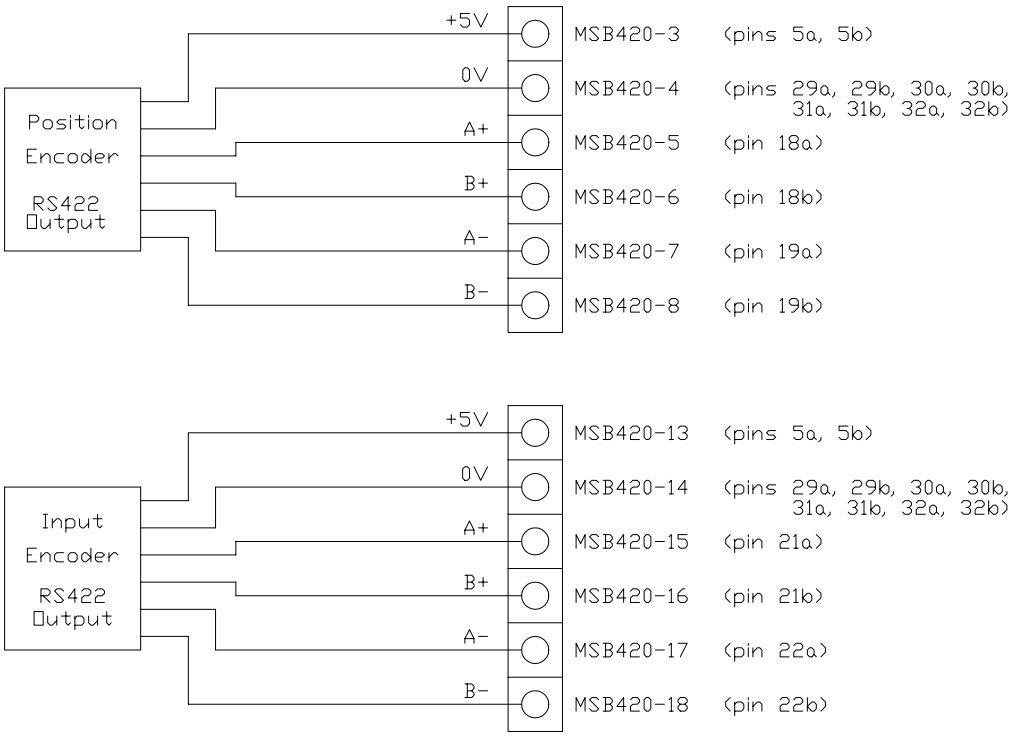
ENCODER

The PM304 has inputs for either TTL output (or *sinking* open collector output) or RS422 complementary output type. The regulated +5 volt output may be used to power the encoders. The leads to the encoders should be screened, with the screen grounded. If RS422 connections are used, *twisted pair* cable should be used.

TTL Encoder Connections



Complementary Encoder Connections



EMERGENCY STOP

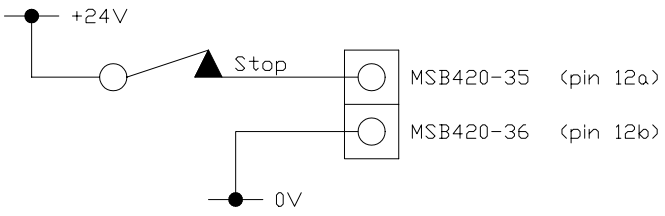
The Emergency Stop on the PM304 is activated by an opto-coupled input.

The switches used should be *normally-closed*. If the input is open-circuited, movement is stopped by setting the output of the PM304 to 0V. All subsequent move commands are not acted upon. An emergency stop may be reset using the **RSES** (reset emergency stop) command. The unit may also be reset by powering-down.

The response to a move command is **`! EMERGENCY STOP`**. The response to a **CO** (current operation) command is **`Emergency Stop`**.

If the Emergency Stop is not used the input connection must be **made** for normal operation. The emergency stop input should be connected to the +ve supply, and its isolated 0v to the supply 0v.

IMPORTANT - The Emergency Stop input on the PM304 must not be used in isolation as a safety stop and as such should only be treated as a monitoring device or a move interrupt facility. For a correct safety stop the power must be removed from the motor.

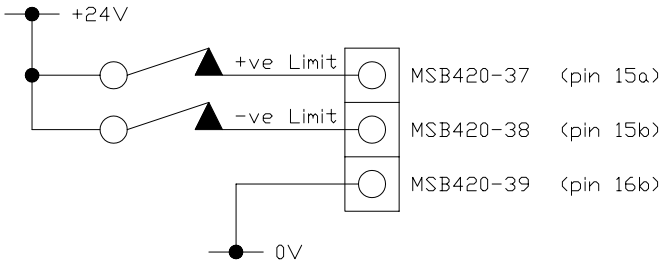


HARD LIMITS

The + and - hard limits are inputs to the PM304 which if activated prevent further movement of the motor in the appropriate direction. They are derived from normally-closed limit switches. In the open state, movement is disabled.

The inputs are opto-coupled, so the respective inputs should be connected to the +ve supply, and their isolated 0V to the supply 0V. Normally-closed limit switches should be used.

If no limit switches are to be used, the limit input connections should be **made** to allow moves.
The **QA** and **OS** commands may be used to check the action of the limit switches before a servo motor is connected to the system.



JOG CONTROLS

Jog switches are opto-coupled and operate when closed. Momentary closure of the Jog+ or Jog- switches produces a single step. If the Jog+ or Jog- switch is closed for more than 0.5 seconds the motor accelerates to the slow jog speed.

If the fast jog switch is then closed, the motor will accelerate to the fast jog speed. Opening the fast jog switch decelerates the motor to the slow rate until the slow jog switch is opened.

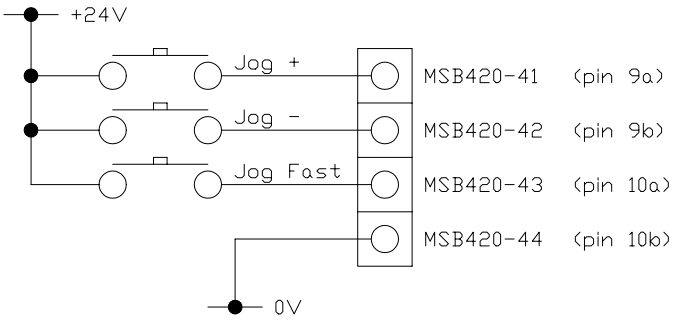
If the fast jog switch is closed first, nothing will happen until the Jog+ or Jog- switch is closed momentarily, whereupon the motor will be jogged 10 steps each time.

The jog speeds are set using the commands:

SJ set jog speed.
SF set fast jog speed.

The operation of the jog (remote) control switches can be enabled and disabled by software commands:

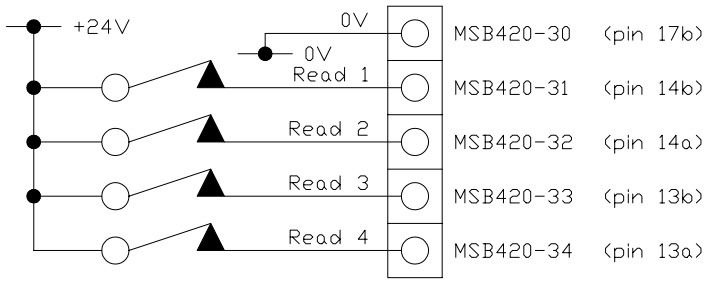
IR Ignore remote.
AR Allow remote.



READ PORTS

The PM304 controller has four user input ports, known as *read ports* 1 to 4. These inputs are opto-coupled and are activated by an input voltage of between +16V and +24V (**0**). If a *read port* is unconnected it will read as **1**.

The input may be connected to either a PNP signal output, a switch, or another controller's output *write port*.



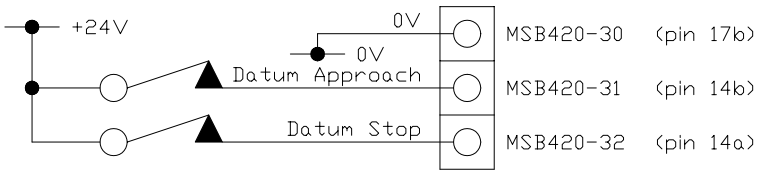
The **RP** Read input Port command is used to check the operation of the *read ports*. This instruction returns a four digit binary number of either **0** or **1** characters to represent the current state of the read port. These start with read port 4, through to 1. Referring to the diagram above, a **0** represents a switch closed and a **1** represents a switch open.

INDEX TO DATUM INPUTS

With an incremental system, it is often necessary to find some datum point, so that moves are then relative to that physical position of the mechanism. The *index* (**IX**) command is used to search for datum.

One or two switches or sensors may be used. These are connected to Read Ports 1 & 2. The first (optional) switch is used as a *Datum Approach* marker to tell the controller to slow down the datum search to the creep speed, and a second to mark the precise *datum stop* position.

Datum Approach and *Datum Stop* signals should be normally closed. If low speed only is utilised, Read Port 1 (Datum Approach) should be left open.

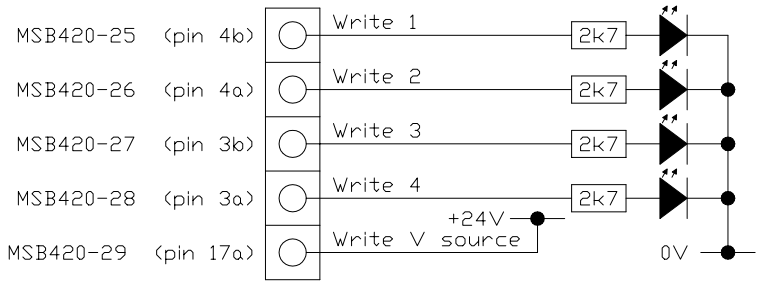


The switches should be positioned so that they are activated as the load passes them and will therefore not be damaged.

WRITE PORTS

The PM304 controller has four output ports, known as *write ports* 1 to 4. These outputs are driven by opto-couplers. The outputs may be connected to an indicator (LED) opto-isolator or another controller's input *read port*. The **WP(bit pattern)** command is used to write to the output port.

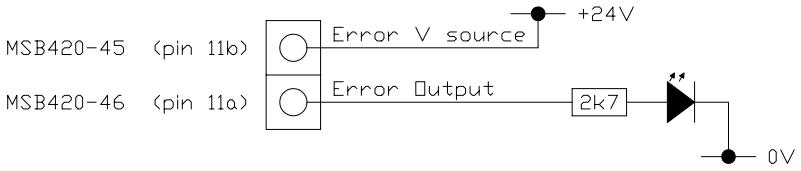
The bit pattern is specified as a four digit binary number. The digits will be either characters **0, 1** or **2** starting with *write port* 4, through to 1. A **0** defines that the output will be **on**, a **1** defines that the output will be **off** and a **2** defines that the output will not change from its current state.



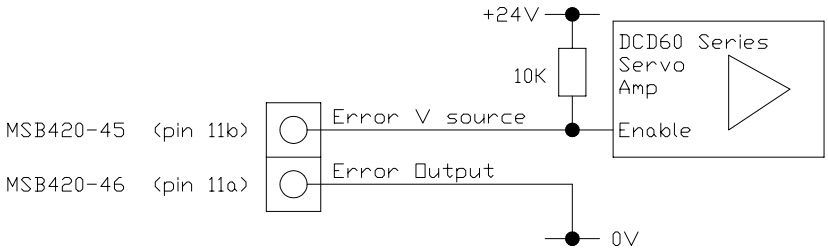
The power-on states of the *write ports* are **1111** - i.e. all **off**.

ERROR OUTPUT

The Error output is activated at the same time as the front panel **ERROR** LED.



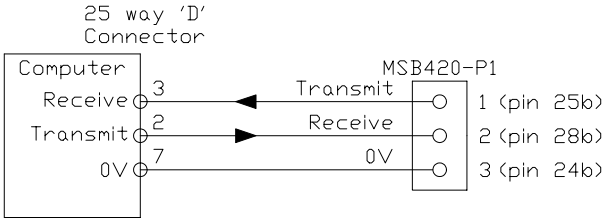
The Error output is typically used in a system to disable the Servo amplifier such as the DCD60 series.



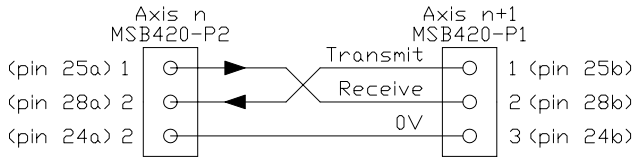
RS232. INTERFACE CONNECTIONS

The PM304 commands are received by the RS232 port as strings of characters.
Transmitted characters are not echoed back to the terminal.
The RS232 is configured in a three wire format. No handshaking is used.

Connections to three-wire DTE (Data Terminal Equipment):



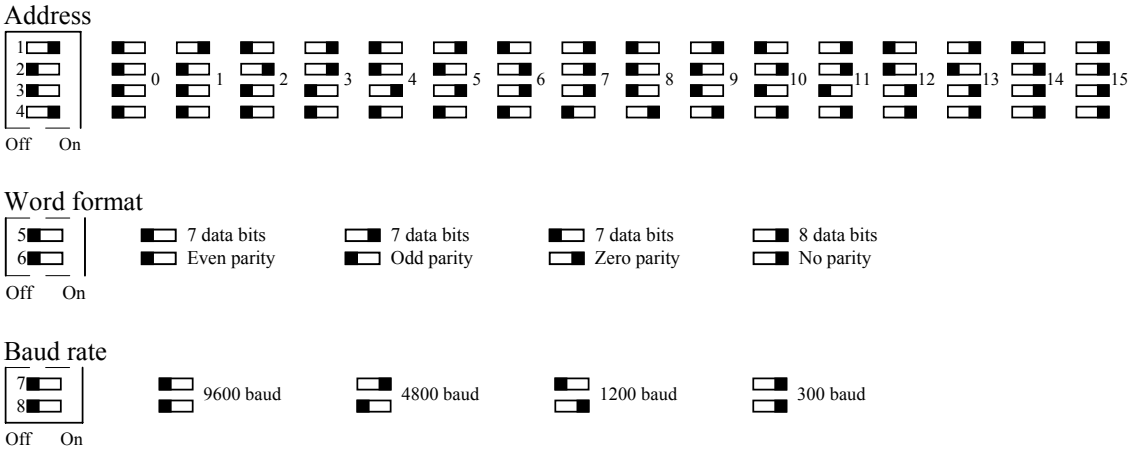
Connections for PM304s in multi-axis applications:
Port 1 connections are as shown above, further PM304s are then *daisy-chained*:



As the RS232 signals are a true *daisy-chain*, if a card is removed in a multi-axis system, the connection to subsequent axes will be lost.

2.3 Switch Settings

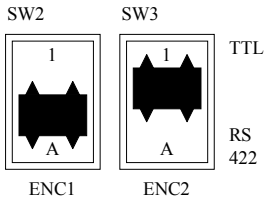
SW1 - designates the axis address number of the PM304, the word format, and the Baud rate.



The above example shows a controller set to axis 9 with communications at 9600 baud, 7 data bits and an even parity bit. There is always one stop bit.

The default RS232 setting of the PM304 is 9600 baud, 7 data bits, even parity and one stop bit (9600,7,E,1).
All of the controllers in a serial *daisy chain* <u>must</u> be set to the same RS232 baud rate and word format.

SW2 & SW3 - switches select the type of encoder input used. SW2 for Encoder 1 (Input Encoder) and SW3 for Encoder 2 (position or motor Encoder).



3. COMMANDS - HOW TO TALK TO THE PM304

COMMAND WORD SYNTAX

The PM304 has a wide range of command options extending beyond the main move functions. The aim is to provide a flexible and comprehensive control device for integration of motion control into larger systems.

Commands

Most commands are two letters, the function of each, being described below. Each command is preceded by the appropriate address to identify the axis for which the command is intended.

Where applicable (e.g. move commands, setting of system parameters, etc.) the command should then be followed by the desired value:

aXXnnn<cr>

a = address
XX = command
nnn = value (if required)
<cr> = carriage return.

Command strings should be terminated with a carriage return character (ASCII 0D hex).
Upper or lower case characters may be used for the command. Spaces within the command line are ignored.

All commands except for Control C and ESC are buffered. Commands are executed in consecutive order. Commands will be acted on sequentially, as they have been entered. If any command cannot be executed immediately (because it may need to wait for some condition or a previous command to finish) then the command and any that follow it will be buffered internally (up to 256 characters).

NOTE: Delete, backspace and cursor movement characters are NOT detected by the controller. With some terminals or emulators these keystrokes will be translated as an escape sequence, i.e. a sequence of characters beginning with an escape character (ASCII 1B hex). The controller will detect the escape character and act on it accordingly.

Replies

Response to a command, once it has been accepted, is either an **OK** string or an alpha-numeric string. Responses terminate in a carriage return character (0D) and a line feed character (0A). An appropriate message is sent if a mistake or conflicting instruction creates an error. The first character of an error message is **!**

Any reply can be prefixed with the axis address and a colon, i.e. **1:OK** (useful in a multi-axis system). The address prefix is toggled on and off by the **AD** command.

Battery-backed Memory

All set-up parameters (control coefficients, acceleration, deceleration, velocities, jog speeds, creep speed, etc.) sequences and profiles will be held in the battery-backed RAM on power-down. They will remain at the previously set values until altered via the RS232 bus.

4. SETTING UP - GETTING STARTED

4.1 Checking RS232 Connections

The PM304 should be connected to an RS232 terminal or a PC running a terminal emulation such as Procomm™. The default RS232 setting of the PM304 is 9600 baud, 7 data bits, even parity and one stop bit (9600,7,E,1). Assuming axis one is being communicated with, send a **1ID** (identify) command. A response of typically: **`Mclennan Servo Supplies Ltd. PM304 V6.15`** should be received.

4.2 Checking Emergency Stop

The state of the emergency stop input may be checked by sending a **CO** (current operation) command. If the Emergency Stop input is active, a response of **`Emergency Stop`** will be received.

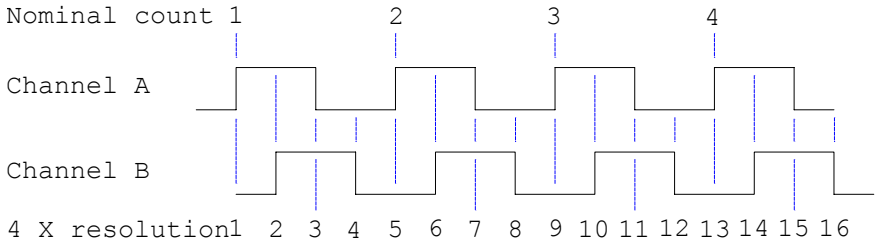
4.3 Checking Hard Limits

The state of the hard limits may be checked using the **QA** page. If a limit is active (open-circuit) the **QA** page will show it as being **ON**.

4.4 Checking Encoder Feedback

This test should be done with the motor disconnected or disabled.

The resolution of the encoder is multiplied by four since all transitions of the quadrature signals are counted:



The correct operation of the feedback encoder must be established before any parameters are set. If possible move the encoder a known number of steps (say one revolution) then send a **1OA** (axis one - output actual position) command. A response of the number of steps moved should be received. The test should be repeated in the opposite direction. Remove power from the PM304.

The next stage should be attempted with caution.

The motor may now be reconnected or enabled. A relative move of 2000 steps should be sent using a **1MR2000** command. If the motor drives continuously until the Error indicator lights, then the encoder rotation sense is incorrect. If the motor moves towards its position then the encoder rotation sense is correct. If the motor does not move, increase the value of **KP** in the order 20, 50, 100, 200, etc. A **1OD** (axis one - output position difference) command may be sent to find the position error remaining.

WARNING:

A serious situation can arise if the feedback from the position encoder fails. A common reason for this is the accidental disconnection of a plug carrying the encoder signals. When this happens the PM304 believes the motor to be stationary and applies full drive voltage to the motor in order to correct a supposed position error. A stalled motor (or encoder failure) is detected by looking for changes in the position encoder signals (or equivalently the changes in observed motor position). If the position encoder does not move, and the voltage output from the PM304 exceeds the value set by the **TH** (threshold) command for a time of 256ms, then the PM304 will set its output to zero and set a *Motor Stalled* condition.

4.5 Setting The Servo Loop Coefficients

The values for **KP**, **KS**, **KV KF** and **KX** determine the characteristic of the servo loop, and will need to be established (even if only roughly) before proceeding further. See section 5. The coefficient **KX** is **only** used in the *Double Encoder Mode*.

An approximate set of coefficients can usually be derived by invoking the **TUNE** command.

The tuning algorithm may fail if there is excessive backlash, or if the low frequency loop gain is very small or very large. Further optimisation of system response will be required to achieve the desired performance.

4.6 Setting Speeds

The slew speed is set using the **SV** command. The initial value for **SV** is 100 steps per second, which with a servo motor fitted with a 1000 line encoder equates to 1.5 r.p.m. This speed is usually too slow for the intended use. If the value of **SV** is set greater than maximum speed attainable by the motor, then an error between the command position and the actual position will build up and a tracking error will occur.

The acceleration setting **SA** and the deceleration setting **SD** can initially be set to give say a 0.5 Sec. ramp time. For example if the value of **SV** is 10000 steps/sec then the values of **SA** and **SD** should be set to 20000 steps/sec^2.

4.7 Setting Tracking Abort

The value of **TR** (tracking window) should be set so that if an error between the command position and the actual position exceeds an allowable value, a tracking abort occurs. The error between actual and command positions may be found using the **OD** (output difference) command. The value of **TR** should be set high enough to avoid nuisance triggering, but low enough to detect a system failure. If a long move or a constant velocity move (**CV**) is executed, the *following error* can be found by repeatedly sending an **OD** command and examining the replies. The magnitude of the *following error* will increase during the acceleration and deceleration phases of a move, so this should be taken into account when setting **TR**. The default value of **TR** is 4000 steps, this equates to one revolution of the motor when using a motor fitted with a 1000 line encoder.

The controller may be set to ignore an abort condition by sending an **IA** (ignore abort) command. The Error LED will still light and the external Error signal will still activate, but the controller will not abort and the control loop will remain active. The tracking abort function may be reinstated using the **AA** (allow abort) command.

A tracking abort may be reset by sending an **RS** (reset abort) command.

4.8 Setting Stall Threshold

The value of **TH** will set the motor stalled threshold. Failure of an encoder is indistinguishable from a stalled motor, and messages from the PM304 refer to *motor stalled* rather than encoder failure.

A stalled motor (or encoder failure) is detected by looking for changes in the position encoder signals. If the motor does not move, the voltage output value from the PM304 will increase until exceeds the value set by the **TH** command for a time of 256ms. The PM304 will then set its output to zero and set a Motor Stalled condition.

The servo system will have coulomb friction and the voltage required to overcome this friction, varies from system to system. The value of **TH** must therefore be large enough not to nuisance trigger but small enough to detect any failure. A motor stall may be detected if the Error output signal is used to inhibit the servo amplifier and the tracking window is exceeded while the tracking abort function is disabled.

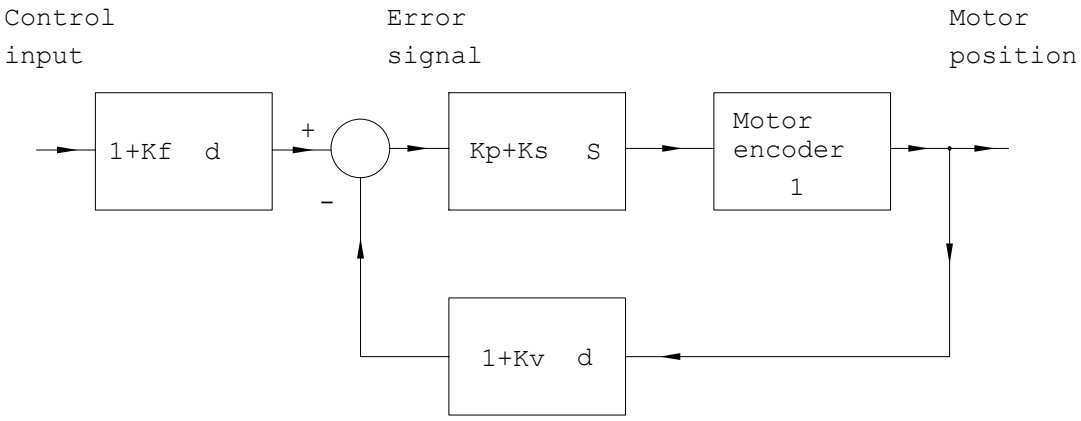
4.9 Setting Soft Limits

The values of upper (**UL**) and lower (**LL**) soft limits may be set to restrict the range of movement of a mechanism. Soft limits are checked when *jogging* or in response to **MA** (move absolute) or **MR** (move relative) commands. As the PM304 assumes its position to zero on power-up, the soft limits are relative to this position. If subsequently the zero position is changed then the soft limits will also move.

The controller may be set to ignore soft limits by sending an **IL** (ignore soft limits) command. The soft limits may be reinstated using the **AL** (allow soft limits) command.

5. OPTIMISATION

5.1 PM304 Control Coefficient Model



The above diagram illustrates the relationship between the control coefficients in a PM304 servo loop. The *control input* is a number generated by a move command.

The PM304 then generates a signal to drive the motor via a servo amplifier. The encoder (usually mounted on the rear of the motor) produces a feedback signal of the motor's position. This enables the PM304 to calculate a position error signal and continuously update the command signal to the amplifier.

The PM304 may be considered as a discrete-time *P.I.D.F.* controller. (Proportional, Integral, Derivative, Feedforward). Coefficients may be varied to change the system characteristics or to optimise the response of the motor in a particular application.

5.2 Optimisation Of System Response.

In a control system the following characteristics should be considered in the process of optimisation:

1. Command response - steady-state accuracy, rise time, overshoot, settling time.
2. Disturbance response - steady-state, transient.
3. Sensitivity to parameter changes.

Some order of priority of the system behaviour should be established to give an objective to the optimisation process. The choice of these parameters is to some extent interactive, and compromises are sometimes necessary in order to achieve a satisfactory solution. The values of **KP**, **KS**, **KV** and **KF** will depend on the characteristics of the system.

KP Proportional gain coefficient.
This coefficient controls the proportional gain in the loop. The controller multiplies the position error by the value of the **KP** coefficient. This influences the amount that the system will react to the error.
System stability will limit the maximum value of this coefficient. Increasing the value of **KP** increase the position accuracy and dynamic response.

KV Velocity feedback coefficient.
The value of this coefficient defines the magnitude of the position encoder derived velocity feedback signal. This velocity signal is combined with the position feedback signal, and produces a damping effect.
This coefficient influences the transient response. It has the effect on the system of reducing overshoot and enhancing stability, but too high a value can create a *buzzy* system, and ultimately an unstable system.

KS Sum coefficient.
The controller sums the value of the position error every millisecond. This sum is then multiplied by the coefficient **KS**. A non-zero value of **KS** will result in a final position at the end of a move that has zero error. The larger the value of **KS** the faster the system will reach its final position. Too large a value of **KS** will cause overshoot and system oscillation. The **KV** coefficient is used to damp overshoots and oscillation.

KF Velocity feedforward coefficient.
This coefficient compensates for the servo lag created by the **KV** coefficient.
The value of **KF** should be zero in positioning moves. In a profile or a move where the motor must follow the acceleration/deceleration profile closely, the value of **KF** should be equal to the value of **KV**.

HINTS:

Initially increase **KP** in the sequence 20, 50, 100, 200, 500, etc. This should produce a *stiffer* feel to the motor; try *testing* the motor shaft each time by moving it from its static position and releasing it suddenly.

Observe the settling behaviour, and when **KP** is at a level that causes motor *ringing* without actual oscillation, begin adding a little **KV**. This should damp out the ringing, but too much will probably cause oscillation. This oscillation will be at a higher frequency than that caused by too much **KP** - a sort of *grittiness*.

Now try adding **KS** to enhance the disturbance response and final position accuracy. **KV** will have to be increased to reduce overshoot.

Remember that an unstable loop might cause damage to a mechanism, so the process should be done with care.

The **IN** (initialise) function will revert the coefficients to default levels of KP=10, KS=0, KV=0 and KF=0.

What To Do If You Don't Want To Optimise It Yourself.

As an alternative to the above there is available some software designed to perform the optimisation task automatically, which works on the great majority of systems. Please contact Mclennan for advice if this is required.

6. APPLICATIONS

6.1 Finding Datum

A datum point search is used to accurately reference a fixed point of a mechanism. The absolute position reference signal in a system is either generated by a precision switch or by an index pulse on a high resolution position encoder. This reference signal (index pulse) occurs typically once per revolution or once per linear stroke. To allow the datum operation to take less time, a datum approach switch may be used to slow down the mechanism close to the datum point. The PM304's index to datum command **IX** is then used to drive to the datum point where the index pulse occurs. The Index pulse from a precision encoder such as a ROD800 is one step wide, therefore the **IX** command will execute a move that stops at a position that is repeatable to within one step. The **IX** command will *Index to datum* in a +ve direction (or **IX-1** for a -ve direction). The system accelerates at **SA** rate, then runs at the set **SV** rate until a *datum approach* signal is received on *read port 1*. It will then decelerate at **SD** rate to **SC** creep speed. It will continue until a Stop signal from the position encoder Index pulse (on *read port 2*) of 1mS or greater occurs.

To allow the PM304 to detect the index position accurately the value of **SC** should be set well below the PM304's sampling frequency of 1KHz. A typical value of **SC** is 500 (steps/sec). If a datum approach switch is not connected the Index move will take place only at the creep speed that will take a long time. This is usually not important in high accuracy system.

Note - the datum stop input always take precedence over the datum approach input.

As the index to datum operation uses two of the read port inputs, the state of the index inputs way be read using the **RP** read port command.

Assuming no connections to read ports 3 or 4, the reply to an **RP** command under various conditions will be:

`1100` indexing at slew speed.
`1101` *datum approach* active - indexing at creep speed.
`1111` index complete.

6.2 Sequences

Commands:

DS<n> Start definition of sequence n.
ES End definition of sequence.
XS<n> Execute defined sequence n.
VS<n> Verify sequence definition (list sequence n).
US<n> Undefine sequence n (clear sequence n from memory).
AE<n> Automatic execution of sequence n on power-up.

The PM304 may be programmed to execute a sequence of commands upon a prompt via the RS 232 data bus or from an external signal using a **WA** command.

The sequence is stored on the on-board RAM that is battery- backed so that the sequence is retained after power-down.

Sequence Example:

Command	Function
1DS3	Define start of sequence 3.
1SV2000	Set max. speed.
1AP0	Define present position as zero.
1MA45000	First move (absolute).
1MR5000	Next move (relative).
1MR3000	Next move (relative).
1MA70000	Next move (absolute).
1SV200000	Set new max. speed.
1MA0	Next move,(return to zero).
1ES	Define end of sequence.

The sequence may then be executed by the command: **1XS3**

If an **XS**<n> command forms part of the sequence, then that sequence will repeat from the beginning. To escape from a sequence before completion the Control **C** or **ESC** should be used. This will arrest the sequence and reset it to the start point but will not remove it from memory. It will however reset the auto-execute flag.

The PM304 may be operated without the further presence of the host computer (i.e. *stand alone*). The **AE** command should be issued to set the auto-execute flag. This flag instructs the controller to execute the programmed sequence immediately on power-up.

The controller may be signalled to execute its sequence on receipt of an external signal. Use may be made of the *wait for input event* function **WA**. This command would be placed at the start of the program.

The **SN** skip next and **WA** commands may be used to create *smart* sequences.

6.3 Multiple Sequences and Memory Allocation

Up to eight *sequences* may be defined, these are numbered 0 to 7. They are executed using the **XS** command, as before but including an argument for the *sequence* number.

For example:

1XS3 Axis 1 - Execute sequence no. 3

If an **XS<n>** command is contained within a *sequence*, commands will execute from the beginning of that *sequence* until an **ES** command is encountered.

Any defined sequence may be set to execute automatically on power-up. The **AE** command can include an argument that is the number of the sequence to execute.

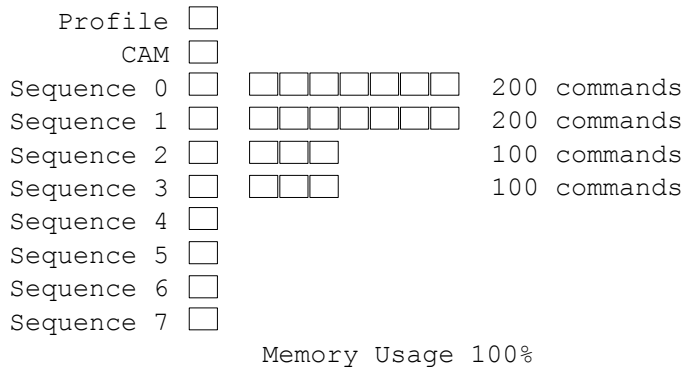
For example:

1AE3 auto-execute sequence no. 3

The sizes of the *sequences* are not pre-defined, and memory is allocated as the *sequences* are defined. Similarly the allowable size of *profile* or *cam* is dependent on the amount of memory available.

The memory is allocated in blocks. Each block represents 25 command lines in a *cam* or *sequence* definition, or 50 command lines in a *profile*. There are ten blocks permanently allocated, one to each of the *sequences*, one to the *profile* and one to the *cam*. This condition is defined as a *memory usage* of 0%. Twenty blocks are available for use with any *define* command. Each time a new memory block is allocated, the *memory usage* increases by 5%.

For example:



The **QA** command displays as one of its items the **Memory Usage** in **%**.

If the memory limit is exceeded during definition of a sequence, profile or cam, a **! MEMORY OFLO** error message is returned.

To remove unwanted data from memory the following commands are used.

US<n> undefine sequence
UP undefine profile
UC undefine cam

6.4 Profiling

The PM304 may be programmed to execute a series of **MR** (move relative) commands without stopping between each move. Each section of the profile is performed at a constant velocity, so the more points defined in the move, the smoother the motion. A constant time interval between the points is set as part of the execute command. The profile does not have to be reloaded to run it at a different speed.

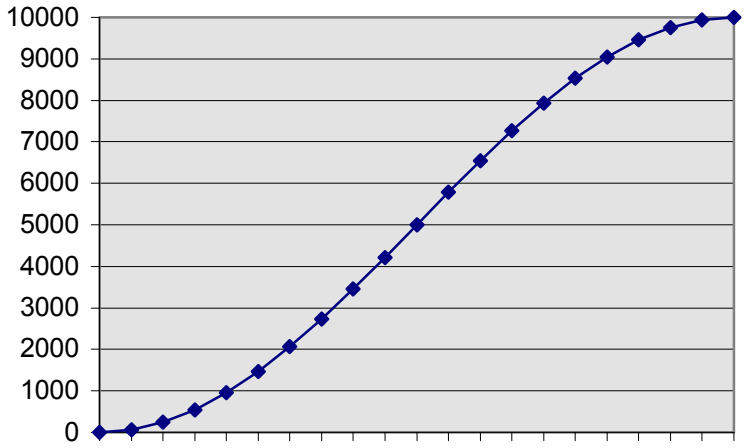
If required, a profile may be executed as part of a sequence. Profiles, like sequences, are retained in the battery backed RAM of the PM304 until overwritten. For optimum profiling response, the value of **KF** should be set equal the value of **KV**.

Commands:

DP	Start definition of profile.
MR<VALUE>	*Move relative* command to define the number of steps to be moved in successive time periods.
EP	End definition of profile.
VP	Verify definition of profile (list profile moves).
XP<VALUE>	Execute defined profile at a rate defined by the *value* in milliseconds as the time taken to complete each **MR** command.
UP	Undefine profile (clear profile memory).

Example:

The following profile was calculated using a cosine speed profile of a move of 10,000 steps split into 20 segments. During execution of the profile, each segments is completed in a time defined by the **XP** (execute profile) command.



This profile would be defined using the following commands:

1DP	Open profile definition.
1MR61	First move relative.
1MR183	Next move relative.
1MR300	Next move relative.
1MR410	Next move relative.
1MR510	Next move relative.
1MR597	Next move relative.
1MR669	Next move relative.
1MR724	Next move relative.
1MR763	Next move relative.
1MR783	Next move relative.
1MR782	Next move relative.
1MR763	Next move relative.
1MR724	Next move relative.
1MR669	Next move relative.
1MR597	Next move relative.
1MR510	Next move relative.
1MR410	Next move relative.
1MR300	Next move relative.
1MR183	Next move relative.
1MR61	Last move relative.
1EP	End profile definition.

6.5 Electronic Gearbox

The PM304 controller may be used to slave the speed of its motor to that of another quadrature signal, usually from a *master* encoder (Input Encoder) mounted on another motor. When not in *gearbox* mode, the controller will still read the position of the input encoder.

Alternatively, handwheels or trackerballs have been used to provide fine manual control of motors in X-Y tables.

The ratio of slaved velocity to that of the input encoder is variable *on the fly*.

The electronic gearbox mode is selected by sending a **GB** command. The controller will then immediately begin slaving its motor to the signal from the input encoder at a ratio set by the **GR** command.

Using the **GB** command the controller will enter gearbox mode and move relative to the position of the input encoder. If the **GA** command is used (absolute gearbox mode) the motor will only move when the input encoder is equal to the position (motor) encoder. This facility makes implementation of an electronic clutch possible.

The method for inputting the gearbox ratio is **GR**.

`<address>`**GR**`<numerator>`**/**`<denominator>`

For example:

2GR22/7 Set gearbox ratio to 22:7. For every 7 steps of the input encoder the motor will move 22 steps.

When *gearbox* mode is entered the value of **KX** is set to zero. This is to avoid unwanted feedforward signals effecting the control loop.

To allow the synchronisation position to be varied the commands **DA** (difference actual position), **DC** (difference command position) and **DI** (difference input position) are available to offset the positions.

Most applications require the slave motor to be *geared-down* with respect to the master. However, if the application requires a large *gear-up* ratio care should be taken in the selection of line counts on both encoders. If master and slave encoders have the same line count a small gear up ratio of 5:1 is normally OK. Very high ratios of 1000:1 and above leads to problems since the transition of a single encoder line on the master produces a large stepped error at the slave motor. For applications requiring a large *gear-up* ratio it is recommended that the slave encoder has fewer lines than the master.

COMMANDS:

Whilst in gearbox mode, the PM304 will ignore **MA**, **MR** and **CV** commands. All other dynamically-related commands will be accepted, but may have little bearing on motor behaviour whilst still in gearbox mode. These are the gearbox mode commands:

GR Gearbox ratio

GB Enter gearbox mode.

GA Enter gearbox mode when the value of the input encoder is equal to the value of the position encoder. Operation is then the same as above.

ST Stops motor and return to normal mode.

Example: **1GR1/4**
1GB

In this case, the PM304 will drive the slave motor at a rate 1/4 of the speed of the master encoder.

To reverse the direction of the motor with respect to the input encoder a negative value of gear ratio should be used.

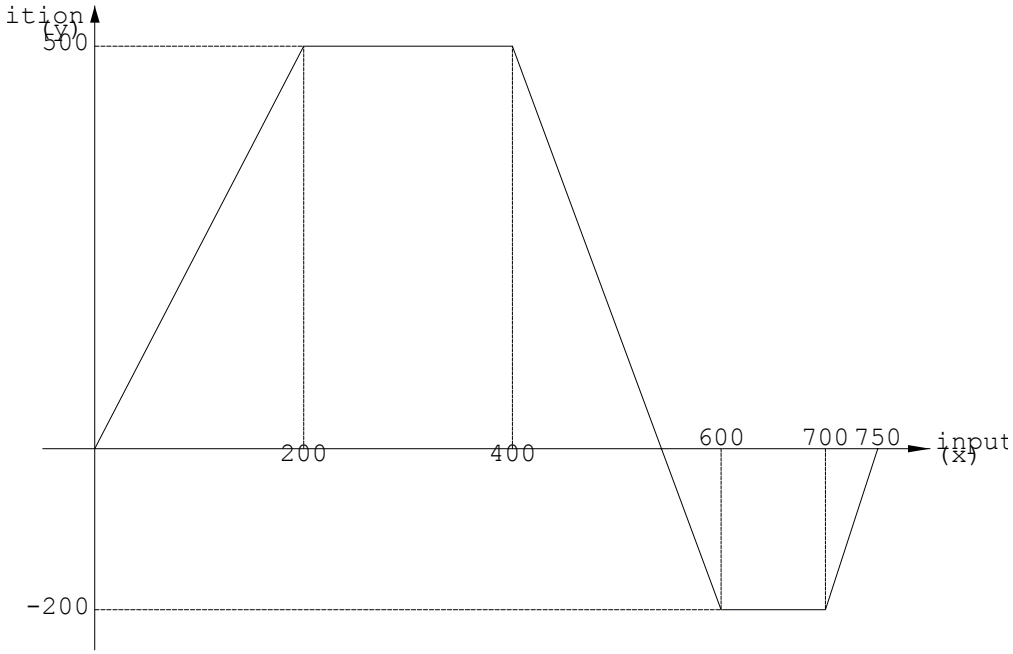
Example: **1GR-1/4**

6.6 Software Cam

The start and finish commands for defining a cam are **DM** and **EM**. Cams are executed using **XM**. Cam co-ordinates are input using the **XY** command:

```
<address>XY<x co-ord.>/<y co-ord.>
```

Cam profiles are *piecewise linear*, with the first co-ordinate implicitly (x=0, y=0).
For example:



This cam profile would be defined using the following commands:

1DM
1XY200/500
1XY400/500
1XY600/-200
1XY700/-200
1XY750/0
1EM

Cam Operation:

The first point is always x=0, y=0. Co-ordinate pairs must be defined in order of increasing x co-ordinate.

The x co-ordinate is limited to the range 0 to 32767 and the y co-ordinate is limited to the range -32768 to 32767. The x co-ordinate of the last pair defines the *modulo*, that is the repeat distance. In the example given above the modulo is 750, so that the y values for x=200, x=950, x=1700, etc. are the same. The **QA** command displays as one of its items **Cam Modulo =**

While in *cam mode* commands such as **QA** (query all), **OA** (output actual position), etc. can still be performed. Exit from *cam mode* can be achieved by either **AB (**user abort) or **ST** (soft stop) commands.

To obtain the most accurate cam action the feedforward coefficient should be made equal to the velocity coefficient. **KF**=**KV**.

Cam positions are absolute, not relative, so that the motor position should be around zero before starting cam. Alternatively a *cam index* can be used to set the *cam* reference position to the current position.

The motor will only start to move when the *input* position divided by the *cam modulo* is equal to the equivalent motor position defined by the *cam*.

Cam Commands:

DM	start cam definition.
EM	end cam definition.
VM	verify cam definition (list cam points).
UM	undefine cam (clear cam from memory).
XM<n>	execute cam n times (n=0 continuous).
IM	set cam index (offset) to current command position.

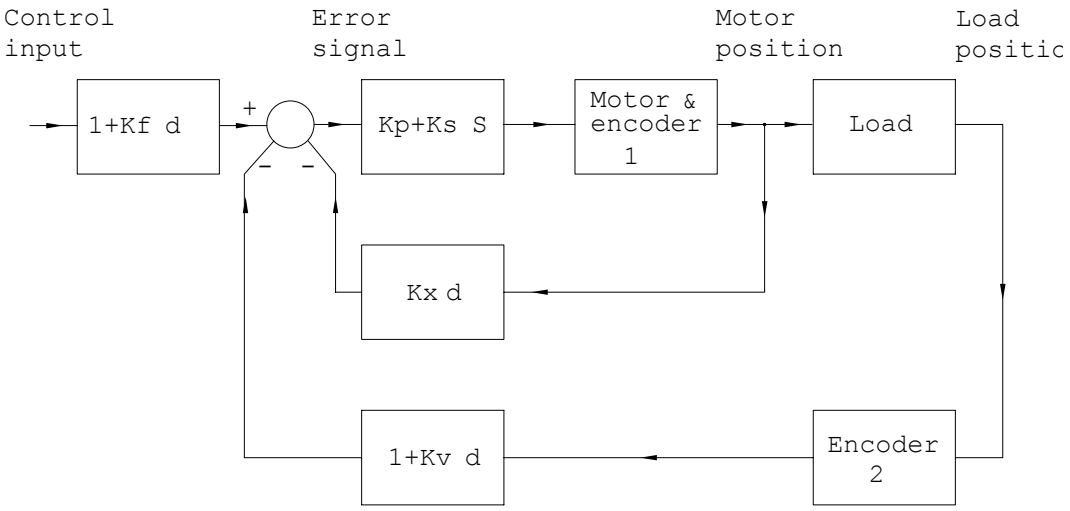
CO returns **Synchronising** or **Execute Cam**

6.7 Double Encoder Mode

In high resolution systems where a remote encoder with a large number of counts per revolution of the motor is used, the amount of damping available from the **KV** coefficient may be insufficient.

An extra encoder on the rear shaft of the motor may be used to give the required damping factor. The level of the feedback signal from this encoder is controlled by the coefficient **KX**. The position (remote) encoder is connected to ENC 2 input and the motor encoder is connected to the ENC 1 input.

Coefficient Model



The increased resilience of the coupling between Encoder 1 and the motor compared with that of Encoder 2 means that the value of **KX** may be much larger than **KV**

Setting Up

The correct phasing must be established for both encoders. This is dependent on the number of mechanical inversions in the system.

1. If the motor has an associated tacho loop this should be set-up first (set for unity gain).
2. With the motor disconnected from the load, and Encoder 2 disconnected, increase the value of **KX** and disturb the output shaft of the motor. If the motor races off then Encoder 1's phasing must be reversed (Swap A+ & A- with B+ & B-). If it is not possible to get to the motor shaft, then set-up the speed **SV**, acceleration **SA** and deceleration **SD**, and attempt a short move.
3. Connect Encoder 2 and increase the value of **KP**. Attempt a short move. Again if the motor races off then Encoder 2's phasing must be reversed (Swap A+ & A- with B+ & B-).

Monitoring the encoder positions using the **OA** and **OI** commands should show that a positive move gives a negative value of input encoder counts.

In practice the process of optimising the coefficients is similar to that of a single encoder system but with **KX** replacing **KV**. Note however that the **TUNE** command only affects **KP, KV, KS and KV** therefore its use in a double encoder system is inappropriate and may produce a **! TUNE FAILURE** error.

7. PROGRAMMERS REFERENCE

CONTROL **C** (ASCII *03*) **Hard Stop**.

Moves, sequences and profiles halted immediately.
Command buffer cleared.
Error signal cleared.
Sets status to **Idle**.
Auto-execute flag cleared.
Sequences and profiles retained in memory.
Operates on all axes.

ESC (ASCII *27*) **Soft Stop**

Command buffer cleared.
Motion stopped at the **SD** rate.
Status returns to **Idle**.
Auto-execute flag cleared.
Sequences and profiles retained in memory.
Operates on all axes.

AA Allow Abort

Allow abort mode. If the error between **CP** and **AP** exceeds the value of **TR** (tracking window) the controller latches the error signal, sets the analogue output signal to the amplifier to 0V, turns the Error LED on and aborts a move. The PM304 remains aborted until reset by sending **RS** or powering-down. The controller will in the meantime respond to queries only. If a move command is attempted the controller will respond with an **`! TRACKING ABORT`** error message. This mode is the default if the controller is re-initialised.

In this mode, selected values for the tracking window will have to take into account the normal position error or *lag* that occurs during rapid acceleration.

Syntax	**Units**	**Range**	**to**	**Initial State**
<ad>AA	N/A	N/A		Enabled

Condition Requirements
None.

Notes:
Value retained on power-up.

Response:

`OK`	Command has been accepted.

Example:

1AA	Sets axis 1 to abort on a tracking error.

AB Abort

The control of the motor may be aborted by sending **AB**. When aborted, the Error LED will illuminate the Error output will be activated, the servo loop is disabled and the motor shaft will be free to rotate. A user abort may be reset with the **RS** command. The position encoder is still read while aborted.

Syntax	Units	Range	to	Initial State
<ad>AB	N/A	N/A		N/A

Condition Requirements
None.

Notes:
The response to a **CO** command will be `User Abort`.

Responses:

OK Command has been accepted.

Example:

1AB Abort axis 1.

AD Toggle address prefix.

Toggle address prefix. Turns on or off (depending on the previous state) the axis address number attached to the beginning of a reply string. A colon **:** is added between the address and the reply.

Syntax	Units	Range	to	Initial State
<ad>AD	N/A	N/A		Prefix off

Condition Requirements
None

Notes:
Value retained on power-up.

Response:

OK Command has been accepted.

Example:

1AD Toggle address prefix on axis 1.
Reply `1:OK`
1AD Toggle address prefix on axis 1.
Reply `OK`

AE SET AUTO-EXECUTE SEQUENCE

Set sequence *n* to run on power-up (auto-execute) of the controller. This may be used in stand alone systems where there is no permanent host computer or terminal.

Syntax	Units	Range	to	Initial State
<ad>AEn	Seq. No.	0	7	Disabled

Condition Requirements
None

Notes:
Value retained on power-up.
Cleared by Control-C and ESC.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.
! NO SEQUENCE	Sequence specified has not been defined yet.

Example:

1AE5 Sets auto execute of axis 1 to run sequence 5 on power-up.

AL ALLOW SOFT LIMITS

Set the soft limit protection enable to ON. Further movement is bounded by the upper and lower soft limits. Soft limits may be turned OFF by the **IL** command.

Syntax	Units	Range	to	Initial State
<ad>AL	N/A	N/A		Enabled

Condition Requirements
None.

Notes:
Value retained on power-up.

Response:

OK	Command has been accepted.

Example:

1AL Sets the soft limits ON for controller axis 1.

AP SET ACTUAL POSITION

Set the actual position value to that given in the argument.

Syntax	Units	Range	to	Initial Value
<ad>APnnn	Steps	-2147483647	2147483647($\pm 2^{32}$)	N/A

Condition Requirements
Idle or Constant velocity

Notes:
Value zero on power-up.

Response:

OK	Command has been accepted.

Examples:

	1AP5000	Set the axis 1 Actual Position to 5000.
or	1AP0	Set the axis 1 Actual Position to zero.

AR ALLOW REMOTE (JOG) CONTROLS

Set the manual JOG control enable to ON. This enables movement by the JOG inputs. The JOG enable may be turned OFF by the **IR** command.

Syntax	Units	Range	to	Initial State
<ad>AR	N/A	N/A		Enabled.

Condition Requirements
None

Notes:
Value retained on power-up.

Response:

OK	Command has been accepted.

Example:

1AR	Sets the jog control enable ON for controller axis 1.

CO	Display the Current Operation

Output the current operation that the controller is executing, i.e. its status.

Syntax	Units	Range	to	Initial State
<ad>CO	N/A	N/A		N/A

Condition Requirements
None

Notes:

Responses:

`Constant Velocity`	Constant velocity move is executing.
`Creep`	Creep steps section of move is executing.
`Delay`	Time delay counter running.
`Emergency Stop`	Emergency Stop input active. No moves executing.
`Execute Cam`	Cam profile move executing.
`Execute Profile`	Timed profile move executing.
`Idle`	No moves executing.
`Index`	Index to datum executing.
`Jog`	Jog move executing.
`Motor Stalled`	**TH** (threshold) value exceeded due to position encoder failure or stalled motor. No moves executing.
`Move`	Move Absolute or Move Relative executing.
`Settle`	End of move settle time counter running.
`Soft Stop`	Decelerating to stop.
`Synchronising`	Waiting for position parity in absolute gearbox mode.
`Tracking Abort`	Controller is aborted due to exceeding the tracking window value **TR**. No moves executing.
`User Abort`	Controller is aborted due to the use of an **AB** (user abort) command. No moves executing.

Example:

1MR10000	Axis 1 move relative by 10,000 steps.
1CO	Query current operation for controller axis 1.

Response:

`Move`	Move Absolute or Move Relative executing.

CP SET COMMAND POSITION

Set the command position value to that given in the argument. The command position is the position generated by a move command.

Syntax	Units	Range	to	Initial Value
<ad>CPnnn	Steps	-2147483647	2147483647($\pm 2^{32}$)	N/A

Condition Requirements
Idle or Constant velocity

Notes:
Value zero on power-up.

Response:

OK	Command has been accepted.

Examples:

	1CP5000	Set the axis 1 Command Position to 5000.
or	1CP0	Set the axis 1 Command Position to zero.

CR SET CREEP DISTANCE

Set number of creep steps at the end of a move. The motor will decelerate and execute this number of steps at the creep speed.

Syntax	Units	Range	to	Initial Value
<ad>CRnnn	Steps	0	2147483647(2^{32})	0

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up.

Response:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Examples:

1CR50	Set the creep distance to 50 steps on axis 1.

CV Constant Velocity Move

A Constant velocity move ramps up at **SA** acceleration rate, then moves the motor at the set **SV** speed. The **SV** command is used to change the speed whilst motion is in progress. The **SA** and **SD** rates define the rate at which the change of speed will be made. Constant velocity mode is exited by an **ST** command, **ESC** or Control **C**.

Syntax	Units	Range	to	Initial Value
<ad>CVn	Direction	-ve	+ve	N/A

Condition Requirements
Idle.

Notes:
During a constant velocity move, **SA**, **SD** and **SV** commands are acted upon immediately.

Responses:

OK	Command has been accepted.
! EMERGENCY STOP	The Emergency Stop has been activated.
! TRACKING ABORT	Controller has aborted due to a Tracking error.
! USER ABORT	Controller is aborted due to a user command.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.

Examples:

1CV	Start constant velocity move in positive direction on axis 1.
1CV-1	Start constant velocity move in negative direction on axis 1.

DA Difference Actual Position

Add value to actual (position encoder's) position.

Syntax	Units	Range	to	Initial Value
<ad>DAnnn	Steps	-2147483648	2147483647 ($\pm 2^{32}$)	N/A

Condition Requirements
None.

Notes:

Response:

OK	Command has been accepted.

Examples:

1OA	Get the axis 1 Actual Position.
Response:	`15000`
1DA5000	Difference axis 1 actual position by 5000.
1OA	Get the axis 1 Actual Position.
Response:	`20000`

DB Set Deadband

Set *Deadband* in number of steps either side of command position. The output from the controller will not increase or decrease for position errors less than the value of the *deadband*.

Syntax	Units	Range	to	Initial Value
<ad>DBnnn	Steps	0	4000	0

Condition Requirements
None.

Notes:
Value retained on power-up. The value of *deadband* is set to zero by the **TUNE** command.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1DB5	Set deadband on axis 1 to 5 steps.

DC Difference Command Position

Add value to Command Position.

Syntax	Units	Range	to	Initial Value
<ad>DCnnn	Steps	-2147483648	2147483647 (± 2^{32})	N/A

Condition Requirements
None.

Notes:

Response:

OK	Command has been accepted.

Examples:

1OC	Get the axis 1 Command Position.
Response:	`15000`
1DC5000	Difference axis 1 Command position by 5000.
1OC	Get the axis 1 Command Position.
Response:	`20000`

DE DELAY

This command will start a delay timer for the time given in the argument. After completion of this time, the controller will return to the idle mode.

If the next command should not execute until the end of this delay time, and is not a *wait for idle* command, then the **DE** command must be followed with a Wait for End (**WE**) command. This will make the controller wait until it returns to the *idle* state before executing the next command.

Syntax	Units	Range	to	Initial Value
<ad>DEnnn	milliseconds	0	60000	N/A

Condition Requirements **Notes:**
Idle and not Constant velocity or Cam.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.
! TRACKING ABORT	Controller has aborted due to a Tracking error.
! USER ABORT	Controller is aborted due to a user command.
! EMERGENCY STOP	The Emergency Stop has been activated.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.
! CONTEXT	Not available in CV, gearbox or Cam modes.

Examples:

1MR400	Move 400 steps positive.
1DE2000	Delay for 2 seconds then....
1MR-400	Move 400 steps negative.
1WP2220	Turn LED on (write port 1).
1DE1000	Delay for 1 second.
1WE	Wait for End of Delay
1WP2221	Turn LED off (write port 1).

DI Difference Input Position

Add value to Input position.

Syntax	Units	Range	to	Initial Value
<ad>DInnn	Steps	-2147483648	2147483647 ($\pm 2^{32}$)	N/A

Condition Requirements **Notes:**
Idle or Constant velocity.

Response:

OK	Command has been accepted.

Examples:

1OI	Get the axis 1 Input Position.
Response:	15000
1DI5000	Difference axis 1 Input position by 5000.
1OI	Get the axis 1 Input Position.
Response:	20000

DM DEFINE CAM

This command will start a Cam profile definition. The only command that is used during a Cam profile definition is **XM.** Any other commands except for **EM** will cause a ! DM SYNTAX error.

The commands that follow this **DM** command will not be executed, but will be stored in the on board non-volatile memory until the End Cam definition (**EM**) command is received. If a Control-C or ESCAPE command is received or the controller runs out of memory, the Cam definition will cease, the Cam will not be stored and the controller will return to the idle state. The Cam Modulo (profile length) is calculated automatically and displayed on the QA page.

Syntax	**Units**	**Range**	**to**	**Initial Values**
<ad>DM	N/A	N/A		N/A

Condition Requirements	**Notes:**
Idle	If a Cam is defined, that fact is shown on the **QA** page.

Responses:

OK	Command has been accepted.
! RECURSIVE DM	DM attempted when already defining a Cam.
! MEMORY OFLO	The available memory has overflowed.
! DM SYNTAX	Command not **XY** or **EM**.

Example:

1DM	Start Cam definition.
1XY200/500	Second Cam profile point. (First Cam profile co-ordinates 0,0.)
1XY400/500	Next Cam Profile point.
1XY600/-200	"
1XY700/-200	"
1XY750/0	"
1EM	End of Cam profile definition.

DP DEFINE PROFILE

This command will start a Profile definition. The only command that is used during a Profile definition is **MR** any other commands except for **EP** will cause a ! SYNTAX error.

The commands that follow this **DP** command will not be executed, but will be stored in the on board non-volatile memory until the End Profile definition (**EP**) command is received. If a Control-C or ESCAPE command is received or the controller runs out of memory, the Profile definition will cease, the Profile will not be stored and the controller will return to the idle state.

Syntax	Units	Range	to	Initial Values
<ad>DP	N/A	N/A		N/A

Condition Requirements
Idle.

Notes:
If a Profile is defined, that fact is shown on the **QA** page.

Responses:

OK	Command has been accepted.
! RECURSIVE DP	DP attempted when already defining a Profile.
! MEMORY OFLO	The available memory has overflowed.
! DP SYNTAX	Command is not **MR** or **EP**.

Example:

1DP	Start Profile definition.
1MR200	First Profile move.
1MR500	Next Profile move.
1MR-500	"
1MR-200	"
1MR50	"
1EP	End of Profile definition.

DS DEFINE SEQUENCE

This command will start a sequence definition. There are eight sequences that may be defined and the argument selects which sequence is to be defined (0 to 7).

All valid commands that follow this **DS** command will not be executed, but will be stored in the on board non-volatile memory until the End Sequence definition (**ES**) command is received. If a Control-C or ESCAPE command is received or the controller runs out of memory, the sequence definition will cease, the sequence will not be stored and the controller will return to the idle state.

Syntax	Units	Range	to	Initial Values
<ad>DSn	Seq. No.	0	7	N/A

Condition Requirements
Idle

Notes:
Auto-execute flag is cleared. The sequences defined are shown on the **QA** page.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument (sequence number) is out of valid range.
! RECURSIVE DS	DS attempted when already defining a sequence.
! MEMORY OFLO	The available memory has overflowed.

Example:

1DS4	Start definition of sequence 4.
1SV2000	Set slew speed.
1MA8000	First move (absolute).
1MR5000	Next move (relative).
1MR3000	Next move (relative).
1SV20000	Set new slew speed.
1MA0	Next move (return to start position).
1XS4	Execute sequence 4 (loop to start of this sequence).
1ES	End of sequence definition.

EM END CAM DEFINITION

This command will end a Cam profile definition. The Cam definition must have been started by the Define Cam (**DM**) command.

Syntax	Units	Range	to	Initial Value
<ad>EMN/A	N/A		N/A	

Condition Requirements
Define Cam

Notes:

Responses

OK	Command has been accepted.
! EM WITHOUT DM	EM attempted when NOT already defining a Cam.

Example:

1DM	Start definition of Cam.
1XY200/500	Second Cam profile point.
1XY400/500	Next Cam Profile point.
|	
1EM	End of Cam definition.

EP END PROFILE DEFINITION

This command will end a Profile definition. The Profile definition must have been started by the Define Profile (**DP**) command.

Syntax	**Units**	**Range**	**to**	**Initial Value**
<ad>EP	N/A	N/A		N/A

Condition Requirements **Notes:**
Define Profile

Responses:

OK	Command has been accepted.
! EP WITHOUT DP	EP attempted when NOT already defining a Profile.

Example:

1DP	Start Profile definition.
1MR200	First Profile move.
1MR500	Next Profile move.
|	
1EP	End of Profile definition.

ES END SEQUENCE DEFINITION

This command will end a sequence definition. The sequence definition must have been started by the Define Sequence (**DS**) command. No argument is necessary as the sequence number is specified with the Define Sequence (**DS**) command.

Syntax	**Units**	**Range**	**to**	**Initial Value**
<ad>ES	N/A	N/A		N/A

Condition Requirements **Notes:**
Define Sequence

Responses

OK	Command has been accepted.
! ES WITHOUT DS	ES attempted when NOT already defining a sequence.

Example:

1DS2	Start definition of sequence 2.
1MR400	First move (relative).
1MR-400	Next move (relative).
1XS5	Execute sequence 5 (transfer control to start of this sequence 5).
1ES	End of sequence definition.

GA ABSOLUTE GEARBOX (RATIO) MODE

Enter gearbox mode when the value of the Input (master) encoder is equal to the value of the Position (slave) encoder. The slave motor will then be driven at a ratio of the Input encoder speed. The ratio is specified by the gear ratio command **GR**. Gearbox mode is exited by an **ST** command, **ESC** or Control **C**. If a **CO** command is received while in absolute gearbox mode, and the PM304 is waiting for the Input and Position encoder values to become equal, a response of **`Synchronising`** will be returned.

Syntax	Units	Range	to	Initial Value
<ad>GA	N/A			N/A

Condition Requirements
Idle.

Notes:
In absolute gearbox mode, **SA**, **SD** and **SV** values are not active.

Responses:

`OK`	Command has been accepted.
`! EMERGENCY STOP`	The Emergency Stop has been activated.
`! TRACKING ABORT`	Controller has aborted due to a Tracking error.
`! USER ABORT`	Controller is aborted due to a user command.
`! MOTOR STALLED`	Controller is aborted due to stalled motor or encoder loss.

Example:

1GA Axis 1 enter absolute gearbox ratio mode.

GB GEARBOX (RATIO) MODE

Enter gearbox mode. The slave motor is now driven at a ratio of the Input encoder speed. The ratio is specified by the gear ratio command **GR**. Gearbox mode is exited by an **ST** command, **ESC** or Control **C**.

Syntax	Units	Range	to	Initial Value
<ad>GB	N/A			N/A

Condition Requirements
Idle.

Notes:
In gearbox mode, **SA**, **SD** and **SV** values are not active.

Responses:

`OK`	Command has been accepted.
`! EMERGENCY STOP`	The Emergency Stop has been activated.
`! TRACKING ABORT`	Controller has aborted due to a Tracking error.
`! USER ABORT`	Controller is aborted due to a user command.
`! MOTOR STALLED`	Controller is aborted due to stalled motor or encoder loss.

Example:

1GB Axis 1 enter gearbox ratio mode.

GR GEARBOX RATIO

Set gearbox ratio. In gearbox modes the ratio is specified by two arguments separated by a / character. The ratio is therefore specified as a fraction with the format: numerator/denominator.

Syntax
<ad>GRnnn/nnn

	Units	Range	to	Initial Value
Numerator	N/A	-32768	32767 ($\pm 2^{15}$)	1
Denominator	N/A	1	32767 (2^{15})	1

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1GR2/5 Axis 1 Set electronic gearbox ratio to 2:5 - i.e. for every 5 steps of the input encoder the command position will change by 2 steps.

HE HELP

Help pages. The commands **HE1** and **HE2** return pages showing Digiloop commands. These help pages give a concise list of the commands available and their function.

Syntax	Units	Range	to	Initial Value
<ad>HEn	page	0	2	N/A

Condition Requirements
None.

Notes:
The command HE0 is the same as HE1. Values of HE greater than 1 show HE2.

Responses:

HE1

AA	allow abort	AB	abort move
AD	toggle addressing	AE<seq no.>	autoexecute sequence
AL	allow limits	AP<position>	set actual position
AR	allow jog	CO	current operation
CP<position>	set command position	CR<distance>	set creep steps
CV<direction>	constant velocity	DA<difference>	actual position
DB<distance>	set deadband	DC<difference>	command position
DE<time in ms>	delay	DI<difference>	input position
DM .. EM	define cam	DP .. EP	define profile
DS<seq no.>.. ES	define sequence	GA	absolute ratio mode
GB	ratio mode	GR<num>/<den>	gearbox ratio
HE1, HE2	help	ID	identify
IN	initialise	IA	ignore abort
IL	ignore limits	IM	index cam
IP<position>	set input position	IR	inhibit jog
IX<direction>	index	KF<value>	set feedforward co.
KP<value>	set proportional co.	KS<value>	set sum co.
KV<value>	set velocity co.	KX<value>	set input vel. co.
LL<position>	set lower soft limit		

HE2

MA<position>	move absolute	MR<distance>	move relative
OA	output actual pos	OC	output command pos
OD	output difference	OI	output input pos
OS	output status string	QA	query all
QK	query coefficients	QS	query speeds
RP	read port	RS	reset from abort
RSES	reset from emg. stop	RSST	reset from stalled
SA<acceleration>	set acceleration	SD<deceleration>	set deceleration
SE<time in ms>	set settling time	SC<speed>	set creep speed
SF<speed>	set fast jog speed	SJ<speed>	set jog speed
SN<bit pattern>	skip on condition	ST	stop
SV<speed>	set velocity	TH<value>	set stall threshold
TR<distance>	set tracking window	TUNE	tune coefficients
UL<position>	set upper soft limit	UM, UP, US	undefine cam etc.
VM, VP, VS	verify cam etc.	WA<bit pattern>	wait for condition
WE	wait for end of move	WI<distance>	set window
WP<bit pattern>	write to port	WS	wait for sync
XM<count>	execute cam	XY<xval>/<yval>	cam coordinates
XP<time in ms>	execute profile	XS<seq no.>	execute sequence

Example:

1HE1 Show the first help page of the controller of axis 1.

IA IGNORE ABORT

Ignore Tracking abort. If the error between the command position and actual position exceeds the **TR** Tracking window value, the controller does not abort but continues to control the motor in the normal way. The Error signal and LED are still activated for the duration of the error. If the size of the tracking window is altered when in this mode, the error message **`! TRACKING DISABLED`** will be returned. The setting of the Tracking Abort, either Enabled or Disabled is shown in the **QA** page.

Syntax	Units	Range	to	Initial Value
<ad>IA	N/A	N/A		Enabled

Condition Requirements
None.

Notes:
Value retained on power-up.

Response:

`OK` Command has been accepted.

Example:

1IA Tracking errors on axis 1 will not cause an abort.

ID IDENTIFY

This command is used to give the type of controller and its internal software revision.

Syntax	Units	Range	to	Initial Value
<ad>ID	N/A	N/A		N/A

Condition Requirements
None

Notes:

Response:

`Mclennan Servo Supplies Ltd. PM304 V6.15`

Example:

1AD Toggle address prefix on or off.

1ID Identify controller of axis 1. If this was a PM304 and the address prefix was previously off, it would respond:
`1:Mclennan Servo Supplies Ltd. PM304 V6.15`

IL IGNORE SOFT LIMITS

Set the soft limit protection enable to OFF. Further movement is NOT bounded by the upper and lower soft limits. Soft limits may be turned ON by the **AL** (allow limits) command. Hard limits will still be active and cannot be disabled.

Syntax	Units	Range	to	Initial Value
<ad>IL	N/A	N/A		Enabled.

Condition Requirements
None.

Notes:
Value retained on power-up.

Response:

OK Command has been accepted.

Example:

1IL Sets the soft limits OFF for controller axis 1.

IM INDEX CAM

This command will index the start position of the Cam profile to the present command position.

Syntax	Units	Range	to	Initial Value
<ad>IM	N/A	N/A		N/A

Condition Requirements
Idle

Notes:

Response:

OK Command has been accepted.

Example:

1IM Set offset the Cam start to the present position.
1XM5 Execute Cam profile 5 times.

IN INITIALISE

This command will set all the programmable parameters back to their initial values, clear sequences and profiles. This is used to re-initialise all the non-volatile memory values to 'safe' values - e.g. if the controller was to be used in a new application.

Syntax	Units	Range	to	Initial Value
<ad>IN	N/A	N/A		N/A

Condition Requirements
None.

Notes:

Response:

OK	Command has been accepted.

Example:

1IN	Set all parameters on axis 1 back to their initial values.

IP SET INPUT POSITION

Set the Input Encoder position value to that given in the argument.

Syntax	Units	Range	to	Initial Value
<ad>IPnnn	Steps	-2147483647	2147483647($\pm 2^{32}$)	N/A

Condition Requirements
Idle or Constant velocity

Notes:
Value zero on power-up.

Response:

OK	Command has been accepted.

Examples:

1IP5000	Set the axis 1 Input Encoder Position to 5000.

IR INHIBIT REMOTE (JOGS) CONTROLS

Disable movement by the JOG inputs. The JOG inputs may be re-enable by the **AR** (allow remote) command.

Syntax	Units	Range	to	Initial Value
<ad>IR	N/A	N/A		Enabled.

Condition Requirements
None

Notes:
Value retained on power-up.

Response:

OK	Command has been accepted.

Example:

1IR	Disable the Jog control inputs for controller on axis 1.

IX FIND INDEX MARKER (DATUM)

This command is used to find a datum point of a mechanism.

Index to datum ramps the motor up at **SA** rate, then moves at the set **SV** rate until a *Slow down* (datum approach) signal is received on read port 1. It will then ramp down at **SA** rate to **SC** creep speed until receipt of a *Stop* (datum stop) signal on Read Port 2. User may then define this position as required.

Important. The respective **SV** and **SA** rates and index sensor positions should be set such that the creep speed is reached before the final stop signal is received. For accurate location of the *datum* position the creep speed **SC** should be set at below 1000 steps per second. After an index operation the positions are not reset.

If no Datum Approach switch is used then the search is performed only at the creep speed.

The **IX-1** command will perform the same in the negative direction.

Soft limits are **not** used during an Index mark search.

Syntax	Units	Range	to	Initial Value
<ad>IXnnn	N/A	-ve	+ve	N/A

Condition Requirements
Idle.

Notes:

Responses:

OK	Command has been accepted.
! HARD LIMIT	Move attempted when already on hard limit.
! TRACKING ABORT	Controller has aborted due to a Tracking error.
! USER ABORT	Controller is aborted due to a user command.
! EMERGENCY STOP	The Emergency Stop has been activated.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.

Examples:

1IX	Search for datum point of axis 1 in positive direction.
1IX-1	Search for datum point of axis 1 in negative direction.

KF Set Feedforward Coefficient

Set velocity feedforward servo coefficient. This compensates for the position offset caused by the velocity lag introduced by **KV**. For positioning moves **KF** is normally set at zero, but for Profiles and Cam moves where the actual position should not lag behind the command position, **KF** should be set equal to **KV**.

In dual encoder feedback systems **KX** also causes a velocity lag. The value of complete **KF** compensation needed is equal to **KX** multiplied by the ratio of Input encoder pulses to Position Encoder pulses plus the value of **KV**. It is not usually necessary for complete compensation of the velocity lag as this adversely effects the settling time of the system.

Syntax	Units	Range	to	Initial Value
<ad>KFnnn	Number	0	32767	0

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Examples:

1KF500	Set velocity feedforward on axis 1 to 500.

KP Set Proportional Gain Coefficient

Set proportional gain servo coefficient. The stiffness and accuracy of the servo loop are controlled by the magnitude of the proportional gain.

Syntax	Units	Range	to	Initial Value
<ad>KPnnn	Number	0	32767	10

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Examples:

1KP100	Set the proportional gain on axis 1 to 100.

KS	Set Sum Gain Coefficient

The Sum servo coefficient is the sum of the integral and proportional components of the servo control loop. The accuracy of the servo loop depends on having a non-zero value of **KS** at the expense of transient response.

Syntax	Units	Range	to	Initial Value
<ad>KSnnn	Number	0	32767	0

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Examples:

1KS50	Set the Sum gain on axis 1 to 50.

KV	Set Velocity Feedback Coefficient

The value of this coefficient defines the magnitude of the velocity feedback signal derived from the position encoder. This coefficient influences the transient response by producing a damping effect. It effects the system by reducing overshoot and enhancing stability, but too high a value can create a *buzzy* system, and ultimately an unstable system.

Syntax	Units	Range	to	Initial Value
<ad>KVnnn	Number	0	32767	0

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Examples:

1KV500	Set the Velocity feedback on axis 1 to 500.

KX Set Extra Velocity Feedback Coefficient

The Extra Velocity Feedback coefficient. It is used in Dual Encoder feedback mode. The value of this coefficient defines the magnitude of the velocity feedback signal derived from the input encoder. This coefficient influences the system transient response by producing a damping effect.

Syntax	Units	Range	to	Initial Value
<ad>KXnnn	Number	0	32767	0

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Examples:

1KX5000 Set the Extra Velocity feedback on axis 1 to 5000.

LL SET LOWER SOFT LIMIT POSITION

This command will set the Lower Soft Limit Position to the value given in the argument. Subsequent moves by the Move Absolute (**MA**) or Move Relative (**MR**) and manual Jog moves will not be allowed below this Lower Limit if the Soft Limits are enabled.

Syntax	Units	Range	to	Initial Value
<ad>LLnnn	Steps	-2147483648	2147483647 ($\pm 2^{32}$)	-2147483648 (-2^{32})

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! LIMITS DISABLED	A warning that the soft limits are currently disabled.
! LIMIT CONFLICT	Attempting to set lower limit above or equal to upper limit

Example:

1LL-4000 Set the axis 1 Lower Soft Limit Position to -4000.

MA MOVE TO ABSOLUTE POSITION

This command will move the motor to the position given in the argument. This position is relative to the Command Position of zero.

Syntax	Units	Range	to	Initial Value
<ad>MAnnn	Steps	-2147483647	2147483647 ($\pm 2^{32}$)	N/A

Condition Requirements **Notes:**
Idle

Responses

OK	Command has been accepted.
! EMERGENCY STOP	The Emergency Stop has been activated.
! TRACKING ABORT	Controller has aborted due to a Tracking error.
! USER ABORT	Controller is aborted due to a user command.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.
! HARD LIMIT	Move attempted when already on a hard limit.
! SOFT LIMIT	Move attempted beyond a soft limit.

Example:
If axis 1 has a current Command Position of 5000 then the command:
1MA4000 Will move 1000 steps in the negative direction to arrive at a Command position of 4000.

MR MOVE TO RELATIVE POSITION

This command will move the motor to the position given in the argument relative to the current Command Position.

Syntax	Units	Range	to	Initial Value
<ad>MRnnn	Steps	-2147483647	2147483647 ($\pm 2^{32}$)	N/A

Condition Requirements **Notes:**
Idle

Responses

OK	Command has been accepted.
! EMERGENCY STOP	The Emergency Stop has been activated.
! TRACKING ABORT	Controller has aborted due to a Tracking error.
! USER ABORT	Controller is aborted due to a user command.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.
! HARD LIMIT	Move attempted when already on a hard limit.
! SOFT LIMIT	Move attempted beyond a soft limit.

Example:
If axis 1 has a current Command Position of 5000 then the command:
1MR4000 Will move 4000 steps in the positive direction to arrive at a Command position of 9000.

OA OUTPUT ACTUAL POSITION

This command will give the current encoder read Actual Position. This position is derived from the incoming position encoder pulses.

Syntax	Units	Range	to	Initial Value
<ad>OA	N/A	N/A		N/A

Condition Requirements **Notes:**
None

Response:
The response is a string of numeric characters.

Example:
If the controller of axis 1 currently has an Actual Position of 20501 then the command:
1OA will respond: **`AP=20501`**

OC OUTPUT COMMAND POSITION

This command will give the current Command Position.

Syntax	Units	Range	to	Initial Value
<ad>OC	N/A	N/A		N/A

Condition Requirements **Notes:**
None.

Response:
The response is a string of numeric characters.

Example:
If the controller of axis 1 currently has a Command Position of 45280 then the command:
1OC will respond: **`CP=45280`**

OD OUTPUT DIFFERENCE BETWEEN COMMAND AND ACTUAL POSITIONS

This command will give the difference between the current Command Position and the current encoder read Actual Position. Numerically it is the Command Position (**CP**) - Actual Position (**AP**).

Syntax	Units	Range	to	Initial Value
<ad>OD	N/A	N/A		N/A

Condition Requirements
None.

Notes:

Response:
The response is a string of numeric characters.

Example:
If the controller of axis 1 currently has a Current position of 1000 and an Actual Position of 1050 then the command:
1OD will respond: **DP=-50**

OI OUTPUT INPUT POSITION

This command will give the current encoder read Input Position. This position is derived from the incoming Input encoder pulses.

Syntax	Units	Range	to	Initial Value
<ad>OI	N/A	N/A		N/A

Condition Requirements
None

Notes:
Used in *electronic gearbox*, *Cam profiles* and *Dual Encoder feedback* applications.

Responses:
The response is a string of numeric characters.

Example:
If the controller of axis 1 currently has an Input Position of 30401 then the command:
1OI will respond: **IP=30401**

OS	OUTPUT STATUS STRING

This command will return an eight bit string that indicates the status of the controller in a format that is more easily interpreted by a host computer.

Syntax	Units	Range	to	Initial Value
<ad>OS	N/A	N/A		N/A

Condition Requirements **Notes:**
None.

Responses:

The response is a string of four numeric characters. The characters are either '0' for not active or '1' for active. The **1:** part would only appear if the **AD** address toggle had been set to prefix replies with the axis address of the replying controller.

```
1:00000000
  ||||||└─ Emergency Stop: 1 = active.
  |||||└── Motor Stalled: 1 = active.
  ||||└─── Tracking Abort: 1 = active.
  |||└──── User Abort: 1 = active.
  ||└───── Controller Idle, i.e., awaiting next command: 1 = idle.
  |└────── Not Error: 1 = not stopped, not stalled nor aborted.
  └─────── +ve Hard Limit: 1 = activated.
           -ve Hard Limit: 1 = activated.
```

Example:

If the PM304 on axis 1 currently is idle, not stopped, stalled, aborted nor on either hard limit, then the command:

1OS will respond: **00110000**

QA QUERY ALL PARAMETERS

Query All. Returns all of the current settings and modes of the controller along with the current positions in a single page format.

Syntax	Units	Range	to	Initial Value
<ad>QA	N/A	N/A		N/A

Condition Requirements **Notes:**
None.

Response:
The response is alpha-numeric strings of characters. Each line gives the parameter name and its value. See example for the format.

Example:
1QA Will generate a response of the form:

```
Mclennan Servo Supplies Ltd  PM304  V6.15
Address: 1                      Address Echo: Enabled
Status: Idle
KP=2132         KS=2304         KV=370          KF=370          KX=0
Slew Speed =    200000
Acceleration =  10000           Deceleration =  10000
Creep Speed =   100             Creep Steps =   0
Jog Speed =     100             Fast Jog =      500
Settling =      10              Deadband =      0
Window =        4               Threshold =     200
Tracking Abort: Enabled         Tracking =      4000
Soft Limits: Enabled            Jog: Enabled
Lower Limit =   -2147483648     Upper Limit =   2147483647
Lower hard limit: Off           Upper hard limit: Off
Gbox Num =      1               Gbox Den =      1
Command Pos =   98789           Actual Pos =    98789
Pos Error =     0               Input Pos =     -189
Autoexec: Sequence #6
Sequences: 0,1,2,3,4,5,6        No Profile
Cam Defined                     Cam modulo =    8000
Memory Usage    95%
Read Port:      1111            Last Write:     1111
```

QK QUERY K COEFFICIENTS

Query servo loop coefficients. Returns the current settings of the KP, KS, KV, KF and KX coefficients.

Syntax	Units	Range	to	Initial Value
<ad>QK	N/A	N/A		N/A

Condition Requirements **Notes:**
None

Response:
The response is an alpha-numeric string of characters showing the parameter name and its value. See example for the format.

Example:

1IN	Set to initial values.
1KP2909	Set proportional gain to 2909.
1KV357	Set velocity feedback to 357.
1KS3258	Set Sum coefficient to 3258.
1QK	Will generate a response of the form: **KP=2909,KS=3258,KV=357,KF=0,KX=0**

QS QUERY SPEEDS

Query the current settings for the speeds and accelerations. Returns the current settings of SV, SC, SA and SD.

Syntax	Units	Range	to	Initial Value
<ad>QS	N/A	N/A		N/A

Condition Requirements **Notes:**
None

Response:
The response is an alpha-numeric string of characters showing the parameter name and its value. See example for the format.

Example:

1SC1000	Set creep speed to 1000 steps/sec.
1SV16200	Set slew speed to 16200 steps/sec.
1SA100000	Set deceleration to 100,000 steps/sec^2.
1SD100000	Set deceleration to 100,000 steps/sec^2.
1QK	Will generate a response of the form: **SV=16200,SC=1000,SA=100000,SD=100000**

RP READ INPUT PORT

This command will examine the read port inputs and return their current state as a four digit numeric string of either **0** or **1** characters. The string starts with read port 4. A **1** indicates that the input is low (0V or open-circuit) and a **0** indicates that the input is high (+24V).

Syntax	Units	Range	Initial Value
<ad>RP	N/A	N/A	N/A

Condition Requirements
None

Notes:
If an **RP** command is executed with the read ports open circuit, a reply of **`1111`** will be returned

Responses
A four digit numeric string.

Example:
If the following states are present on the inputs:

PORT :	4	3	2	1
STATE :	Low	Low	Low	High

then the command 1RP will reply: **`1110`**

RS RESET FROM ABORT

This command will reset the *tracking abort* or *user abort* conditions and re-enable the servo control loop. It will also set the Command position to be equal to the Actual position.

Syntax	Units	Range	Initial Value
<ad>RS	N/A	N/A	N/A

Condition Requirements
None.

Notes:

Responses

`OK`	Command has been accepted.
`! EMERGENCY STOP`	The Emergency Stop has been activated.
`! MOTOR STALLED`	Controller is aborted due to stalled motor or encoder loss.
`! NOT ABORTED`.	The unit has not aborted.

Example:

1RS	Reset abort on axis 1 controller.

RSES RESET FROM EMERGENCY STOP

This command will, if the emergency stop input is not active, resets the *Emergency Stop* condition and re-enables the servo control loop. It will also set the Command position to be equal to the Actual position.

Syntax	Units	Range	Initial Value
<ad>RSES	N/A	N/A	N/A

Condition Requirements **Notes:**
None.

Response

OK	Command has been accepted.
! NOT STOPPED	Stop Input has not been activated.

Example:

1RSES	Reset stopped condition on axis 1 controller.

RSST RESET FROM MOTOR STALLED

This command will reset the *Motor Stalled* condition and re-enable the servo control loop. It will also set the Command position to be equal to the Actual position.

Syntax	Units	Range	Initial Value
<ad>RSST	N/A	N/A	N/A

Condition Requirements **Notes:**
None.

Response

OK	Command has been accepted.
! NOT STALLED	Motor Stalled threshold has not been exceeded.

Example:

1RSST	Reset motor stalled condition on axis 1 controller.

SA SET ACCELERATION

Set the acceleration rate for changes of velocity for all following moves. This may also be used during a constant velocity move.

Syntax	Units	Range	to	Initial Value
<ad>SAnnn	Steps/sec^2	1	20000000	10000

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1SA10000 Sets acceleration of axis 1 controller to 10000 Steps/sec^2.

SC SET CREEP SPEED

Set the creep speed for all following moves. This is the speed that at which moves with a non-zero creep distance will stop.

It is also the speed that slow datum search will be moved at (**IX** command).

Syntax	Units	Range	to	Initial Value
<ad>SCnnn	Steps/sec	1	400000	100

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1SC700 Sets creep speed of axis 1 controller to 700 Steps/sec.

SD SET DECELERATION

Set the deceleration rate for changes of velocity for all following moves. This may also be used during a constant velocity move.

Syntax	Units	Range	to	Initial Value
<ad>SDnnn	Steps/sec^2	1	20000000	10000

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1SD100000 Sets acceleration of axis 1 controller to 100000 Steps/sec^2.

SE SET SETTLING TIME

Set the settling time for all following moves. This time elapses at the end of each move to allow the motor to settle. The end of a move is defined by the **OD** (position difference) value being less than the **WI** (end of move window) value for the **SE** (settling) time.

Syntax	Units	Range	to	Initial Value
<ad>SEnnn	milliseconds	0	20000	10

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up. While the settling time is elapsing the **CO** command will give a reply of `Settle`.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1SE1000 Sets settling time of axis 1 controller to 1 second.

SF SET FAST JOG SPEED

Set the fast speed for all following manual *jog* moves. The jog movement will accelerate up to this speed when a jog input and the jog fast inputs are active.

Syntax	Units	Range	to	Initial Value
<ad>SFnnn	Steps/sec	1	400000	500

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1SF1000 Sets fast jog speed of axis 1 controller to 1000 Steps/sec.

SJ SET JOG SPEED

Set the normal speed for all following manual *jog* moves. The jog movement will be at this speed when a jog input is active, but not the jog fast input.

Syntax	Units	Range	to	Initial Value
<ad>SJnnn	Steps/sec	1	4000	100

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1SJ50 Sets jog speed of axis 1 controller to 50 Steps/sec.

SN	SKIP NEXT

Skip next command if true. The controller will *skip over* (ignore) the following command if the read ports correspond to the bit pattern specified. This command will examine the read port inputs and compare them with the specified bit pattern argument. If the inputs are equal to the specified bit pattern, then the controller will skip over, i.e. not execute the next command. If no commands are in the command buffer or in a sequence the *next* command will be the next received command. If the *skip* condition is not met, then the next command will be executed as normal. If the next command is skipped, the controller will give the response **SKIPPED** instead of **OK** or any other response for that command.

The bit pattern is specified as a four digit binary number of either **0, 1** or **2** characters starting with read port 4, through to 1. A **0** defines that the input must be high (+24V), a **1** defines that the input must be low (0V) and a **2** defines that the input is not relevant or *don't care*. If less that four digits are specified in the argument, then the preceding ones are assumed as high (**0**).

This command may be used to introduce a conditional response to some machine functions, and may be used to create *smart* sequences.

Syntax	Units	Range	Initial Value
<ad>SNbbbb	Bit pattern	4 digits of 0, 1 or 2	N/A

Condition Requirements
None.

Notes:

Responses

OK	Command has been accepted.
! SN SYNTAX	Invalid argument i.e. bit specified was not 0, 1 or 2 OR the Number of bits was greater than 4.

Example:

1DS3	Define the start of the sequence 3.
1WA2221	Wait here until read port 1 goes low.
1SN2212	Skip next command if read port 2 is low, state of ports 1, 3 & 4 not important.
1IX	Search for datum (only executed if port 2 was high, above).
1MR1000	Move motor 1000 steps.
1XS	Loop back to start of sequence.
1ES	End definition of sequence.

The sequence shown is a repeated incremental indexing motion, but allows an operator to search for datum only when required, usually just after power-up. If the **AE** (Auto execute) flag is set, the system could be operated without the presence of a host computer.

ST STOP

This command will stop any current move, decelerate the motor speed down at the **SD** rate, then stop and return to *idle* mode.

This command is buffered and is only responded to when it reached in the command queue. Care must therefore be taken that there are no commands that hold up the queue between the move command and the **ST** command.

Syntax	Units	Range	to	Initial Value
<ad>ST	N/A	N/A		N/A

Condition Requirements
None

Notes:
Will exit constant velocity mode or gearbox mode.

Responses

OK	Command has been accepted.
! EMERGENCY STOP	The Emergency Stop has been activated.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.

Example:

1CV	Will start axis 1 moving in constant velocity mode.
1ST	This will then stop the current move of axis 1.

SV SET VELOCITY

Set the Slew (maximum) velocity for all following moves. This may also be used during a constant velocity move.

Syntax	Units	Range	to	Initial Value
<ad>SVnnn	Steps/sec	1	400000	100

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1SV5000	Sets slew speed of axis 1 controller to 5000 Steps/sec.

TH SET THRESHOLD

This command will set the motor stalled threshold. Failure of an encoder is indistinguishable from a stalled motor, and messages from the PM304 refer to *motor stalled* rather than encoder failure.

A stalled motor (or encoder failure) is detected by looking for changes in the position encoder signals (or equivalently the changes in observed motor position). If the motor does not move, and the voltage output value from the PM304 exceeds the value set by the **TH** command for a time of 256ms, then the PM304 will set its output to zero and set a Motor Stalled condition..

The servo system will have coulomb friction and the voltage required to overcome this friction, varies from system to system, so the value of **TH** must be large enough not to nuisance trigger but small enough to detect any failure.

If a *stalled motor* condition occurs, the error signal and front panel LED are both activated, and movement is stopped. Subsequent moves will not function but will return the response **`! MOTOR STALLED`** until reset by either a Reset Stall (**RSST**) command or by powering off.

The response to a **CO** command is **Motor Stalled**.

Syntax	Units	Range	to	Initial Value
<ad>THnnn	Steps	0	2047	200

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses:

`OK`	Command has been accepted.
`! OUT OF RANGE`	Argument is out of valid range.

Example:

1TH400 Set the Threshold before *motor stalled* condition for axis 1 to 400.

TR SET TRACKING WINDOW

This command will set the tracking window. The *Tracking window* is the allowable difference between the *Command Position* and the *Actual Position*. When the motor is stationary this is the allowable static error. During a move, a changing *command position* is generated. The *Tracking Window* operates on the difference between the *actual position* and this moving *command position*. The servo system will have a *following error,* so the value of **TR** must be large enough not to nuisance trigger but small enough to detect any failure.

If the *tracking* window is exceeded the Error output signal and front panel LED are activated and (if abort is enabled) the controller *aborts*.

The abort function may be inhibited by using the **IA** (ignore abort) command, or enabled using the **AA** (allow abort) command.

Subsequent moves will not function but will return the response **! TRACKING ABORT** until reset by either a Reset (**RS**) command or by powering off.

Syntax	Units	Range	to	Initial Value
<ad>TRnnn	Steps	0	2147483647 (2^{32})	4000

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.
! TRACKING DISABLED	Warning that tracking abort is inhibited. Value will be accepted.

Example:

1TR400 Set the Tracking Window for axis 1 to 400 steps.

TUNE TUNE COEFFICIENTS

An approximate set of servo coefficients can usually be derived by invoking the **TUNE** command. The controller will *exercise* the motor over a small displacement for a few seconds and obtain a set of values for the *K* coefficients that should be stable and provide a reasonable disturbance rejection.

The tuning algorithm may fail if there is excessive backlash, if the low frequency loop gain is either very small or very large or the feedback encoder phasing is wrong. Further optimisation of system response will be required in almost all cases to achieve the desired performance.

The **TUNE** command only affects **KP, KV, KS and KV** therefore its use in a double encoder system is inappropriate and may produce a **`! TUNE FAILURE`** error.

Syntax	Units	Range	to	Initial Value
<ad>TUNE	N/A	N/A	N/A	N/A

Condition Requirements
Idle.

Notes:
Value retained on power-up.
Deadband value set to zero.

Responses:

`nn,mm`	n and m are parameters relating to the system response. These parameters are used by optimisation programs.
`! TUNE FAILURE`	A stable set of coefficients could not be found.
`! EMERGENCY STOP`	The Emergency Stop has been activated.
`! TRACKING ABORT`	Controller has aborted due to a Tracking error.
`! MOTOR STALLED`	Controller is aborted due to stalled motor or encoder loss.
`! USER ABORT`	Controller is aborted due to a user command.
`! HARD LIMIT`	Tune attempted when already on hard limit.

Example:

1TUNE Tune coefficients on axis 1 controller.

UM UNDEFINE CAM

This command will undefine or cancel a Cam definition. This will then free the memory used by the Cam for use in other sequences or profile definitions.

Syntax	Units	Range	to	Initial Value
<ad>UM	N/A	N/A		N/A

Condition Requirements
Idle.

Notes:

Responses

`OK`	Command has been accepted.

Example:

1UM Delete Cam from axis 1 controller.

UL SET UPPER SOFT LIMIT POSITION

This command will set the Upper Soft Limit Position to the value given in the argument. Subsequent moves by the Move Absolute (**MA**) or Move Relative (**MR**) and manual Jog moves will not be allowed above this Upper Limit if the Soft Limits are enabled.

Syntax	Units	Range	to	Initial Value
<ad>ULnnn	Steps	-2147483647	2147483647 ($\pm 2^{32}$).	2147483647 (2^{32})

Condition Requirements
Idle or Constant velocity

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! LIMITS DISABLED	A warning that the soft limits are currently disabled.
! LIMIT CONFLICT	Attempting to set upper limit below or equal to lower limit

Example:

1UL8000 Set the axis 1 Upper Soft Limit Position to 8000.

UP UNDEFINE PROFILE

This command will undefine or cancel a Profile definition. This will then free the memory used by the Profile for use in other sequences or profile definitions.

Syntax	Units	Range	to	Initial Value
<ad>UP	N/A	N/A		N/A

Condition Requirements
Idle.

Notes:

Responses:

OK	Command has been accepted.

Example:

1UP Delete Profile from axis 1 controller.

US UNDEFINE SEQUENCE

This command will undefine or cancel a sequence definition. This will then free the memory used by the sequence for use in other sequences or profile definitions.

Syntax	Units	Range	to	Initial Value
<ad>USn	Seq. No.	0	7	N/A

Condition Requirements
Idle.

Notes:
If the sequence specified has been set to auto-execute, then the auto-execute flag will be reset.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument (sequence number) is out of valid range.

Example:

1US6	Delete sequence 6 from axis 1 controller.

VM VERIFY CAM

This command will list a previously defined Cam profile.

Syntax	Units	Range	to	Initial Value
<ad>VM	N/A	N/A		N/A

Condition Requirements
None.

Notes:

Responses
The command will either respond with the axis address identifier (if address prefix enabled) followed by each line of the Profile, or an error message:

! NO CAM	Cam has not been defined yet.

Example:

A controller that had previously been programmed with:

1DM	Start Cam definition.
1XY200/500	First Cam profile point.
1XY400/500	Next Cam Profile point.
1XY600/-200	"
1XY700/-200	"
1XY750/0	"
1EM	End of Cam profile definition.

The command 1VM would give (address prefix enabled):

```
1:
XY 200/500
XY 400/500
XY 600/-200
XY 700/-200
XY 750/0
OK
```

VP	VERIFY PROFILE

This command will list a previously defined Profile.

Syntax	Units	Range	to	Initial Value
<ad>VP	N/A	N/A	N/A	N/A

Condition Requirements
None.

Notes:
Arguments with a value of zero are not shown.

Responses:
The command will either respond with the axis address identifier (if address prefix enabled) followed by each line of the Profile, or an error message:

! NO PROFILE	Profile has not been defined yet.

Example:
A controller that had previously been programmed with:

1DP	Start definition of Profile.
1MR2000	First move.
1MR7000	Next move.
1MR1000	"
1MR0	"
1EP	End of Profile definition.

The command 1VP would give:

```
1:
MR 2000
MR 7000
MR 1000
MR
OK
```

VS	VERIFY SEQUENCE

This command will list a previously defined Sequence.

Syntax	Units	Range	to	Initial Value
<ad>VSn	Seq. No.	0	7	N/A

Condition Requirements
None.

Notes:
Arguments with a value of zero are not shown.

Responses

The command will either respond with the axis address identifier (if address prefix enabled) followed by each line of the sequence, or an error message:

! OUT OF RANGE	Argument (sequence number) is out of valid range.
! NO SEQUENCE	Sequence specified has not been defined yet.

Example:

A controller that had previously been programmed with:

1DS2	Start definition of sequence 2.
1MA2000	First move (absolute).
1MR7000	Next move (relative).
1DE1000	Delay for 1 second.
1MA0	Next move (return to start position).
1XS2	Execute sequence 2 (loop to start of this sequence).
1ES	End of sequence definition.

The command 1VS2 would give:

```
1:
MA 2000
MR 7000
DE 1000
MA
XS 2
OK
```

WA WAIT FOR INPUT PORT CONDITION

This command will examine the read port inputs and compare them with the specified bit pattern argument. It will wait until the inputs are equal to the specified bit pattern before issuing its **'OK'** response and moving on to the next command.

The bit pattern is specified as a four digit binary number of either **0, 1** or **2** characters starting with read port 4, through to 1. A **0** defines that the input must be high (+24V), a **1** defines that the input must be low (0V or open-circuit) and a **2** defines that the input is not relevant or 'don't care'. If less that four digits are specified in the argument, then the preceding ones are assumed as low (**0**).

Syntax	Units	Range	Initial Value
<ad>WAbbbb	Bit pattern	4 digits of 0, 1 or 2	N/A

Condition Requirements **Notes:**
None.

Responses

OK	Command has been accepted.
! WA SYNTAX	Invalid argument i.e. bit specified was not 0, 1 or 2 OR the Number of bits was greater than 4.

Example:

1WA2210 Will wait until the following condition is on the read input port before continuing:

PORT:	4	3	2	1
STATE:	(Ignored)	(Ignored)	Low	High

WE WAIT FOR END

This command will wait for the end of a move or delay. It will wait until any current move or delay has finished and detects the return to the *idle* state. The 'OK' response will not be issued until the move or delay has been completed. Therefore **WE** can be used to execute I/O commands after a move is complete.

Syntax	Units	Range	to	Initial Value
<ad>WE	N/A	N/A		N/A

Condition Requirements **Notes:**
None.

Response:

OK	Command has been completed.

Examples:

1MR4000	Move 4000 steps positive.
1WE	Wait for End of above move
1WP2220	Turn LED on (write port 1) when move has finished.
1DE1000	Delay for 1 second.
1WE	Wait for End of Delay
1WP2221	Turn LED off (write port 1).

WI SET WINDOW

This command will set the window for end of move checking. At the end of a move, when the motor comes within the **WI** range of this final target, the **SE** (settling time) counter counts down. When the settling time reaches zero the controller will either accept the next command or go to the *idle* condition.

If the motor overshoots the window before to the settling time reaches zero, the settling time counter is reset and started again.

Syntax	Units	Range	to	Initial Value
<ad>WInnn	Steps	0	2147483647 (2^{32})	4

Condition Requirements
None.

Notes:
Value retained on power-up.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.

Example:

1WI2 Set the Window for axis 1 to 2 steps.

WP WRITE TO OUTPUT PORT

Write to output port. The PM304 controller has four user output ports, known as write ports 1 to 4. This command will set the write port outputs to a state defined by the specified bit pattern argument. The bit pattern is specified as a four digit binary number. The digits will be either **0, 1** or **2** characters starting with write port 4 through to 1

Format: Four digit binary string
consisting of 0s, 1s or 2s.
0 = *On* +24V (depending on the voltage of Write Port V_{source})
1 = *Off* 0V or open-circuit
2 = *Don't change*

Syntax	Units	Range	Initial Value
<ad>WPbbbb	Bit pattern	4 digits of 0, 1 or 2	N/A

Condition Requirements
None.

Notes:
Initial state on power-up: all **1** = *Off*
The last *write* is shown on the **QA** page.

Responses:

OK	Command has been accepted.
! WP SYNTAX	Invalid argument i.e. bit specified was either not 0, 1 or 2 or the Number of bits was greater than four.

Example:

If a PM304 on axis 1 currently has the following states on its output write ports:

	PORT:	4	3	2	1
	STATE:	on	on	off	off
1WP1200	Will set the outputs to:				
	PORT:	4	3	2	1
	STATE:	off	on	on	on
		1	2 (unchanged)	0	0

WS WAIT FOR SYNCHRONISATION

This command will make the PM304 wait and not execute any more commands until the Input position equals the Motor Position. This command is used in Absolute gearbox mode.

Syntax	Units	Range	to	Initial Value
<ad>WSN/A	N/A		N/A	

Condition Requirements
Synchronised in absolute gearbox.

Notes:

Response:

OK	Command has been completed.

Example:

1GA	Axis 1 enter absolute gearbox mode.
1WS	Axis 1 wait for synchronisation.

XM EXECUTE CAM

This command will execute the defined Cam profile. The argument sets the number of times that the Cam repeats. A zero value will cause the Cam to repeat continuously.

The **CO** command returns **Synchronising** or **Execute Cam.**

Syntax	Units	Range	to	Initial Value
<ad>XMnnn	N/A	0	32767	N/A

Condition Requirements
Idle.

Notes:
This command sets the value of **KX** to zero.

Responses

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.
! NO CAM	Cam has not been defined yet.
! EMERGENCY STOP	The Emergency Stop has been activated.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.
! TRACKING ABORT	Controller has aborted due to a Tracking error.
! USER ABORT	Controller is aborted due to a user command.

Example:

1XM10	Axis 1, execute Cam 10 times.

XP EXECUTE PROFILE

This command will execute the defined Profile. The move occurs at a rate, defined in milliseconds, for each **MR** segment to be completed.

Syntax	Units	Range	to	Initial Value
<ad>XPnnn	millisecs.	1	65535	N/A

Condition Requirements **Notes:**
Idle.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument is out of valid range.
! NO PROFILE	Profile has not been defined yet.
! EMERGENCY STOP	The Emergency Stop has been activated.
! MOTOR STALLED	Controller is aborted due to stalled motor or encoder loss.
! TRACKING ABORT	Controller has aborted due to a Tracking error.
! USER ABORT	Controller is aborted due to a user command.

Example:

1XP100 Axis 1, execute Profile. Each segment takes 100 mS.

XS EXECUTE SEQUENCE

This command will start execution of a sequence. The argument selects which sequence is to be executed (0 to 7). The sequence must have already been defined with a Define Sequence **DS** command.

If the Execute Sequence (**XS**) command is encountered during a sequence, it will explicitly transfer control to the beginning of the sequence specified, whether it is the sequence already running or another sequence. It may therefore be used to make a loop type sequence or jump to any other sequence. Please note that it should not be considered as a subroutine. It is like a GOTO rather than a GOSUB.

A sequence execution may be stopped before completion, or if in a continuous loop, by a Control-C or ESCAPE command.

Control-C will stop any movement immediately, exit the sequence and return to idle.

ESCAPE will decelerate any move to a stop, exit the sequence and return to idle.

Syntax	Units	Range	to	Initial Value
<ad>XSn	Seq. No.	0	7	N/A

Condition Requirements **Notes:**
None.

Responses:

! OUT OF RANGE	Argument (sequence number) is out of valid range.
! NO SEQUENCE	Sequence specified has not been defined yet.

Other responses may be generated by commands within the sequence. At the completion of the sequence, the response to the last command is sent.

Example:

1XS1 Execute sequence 1

XY CAM CO-ORDINATES

Set Cam co-ordinates. In Cam mode the slave motor is driven at a ratio of the Input encoder speed. This Cam profile is specified by two arguments separated by a / character.

The first point is always x=0, y=0. Co-ordinate pairs must be defined in order of increasing x co-ordinate.

The x co-ordinate of the last pair defines the *modulo*, that is the repeat distance. In the example given below the modulo is 750, so that the y values for x=200, x=950, x=1700, etc. are the same. The **QA** command displays as one of its items **Cam Modulo =**

Exit from *cam mode* can be achieved by either AB or ST commands.

To obtain the most accurate cam action the feedforward coefficient should be made equal to the velocity coefficient. KF=KV.

Cam positions are absolute, not relative, so that the motor position should be around zero before starting cam. Alternatively a *cam index* command **IM** can be used to set the *cam* reference position to the current command position.

The motor will only start to move when the *input* position divided by the *cam modulo* is equal to the equivalent motor position defined by the *cam*.

Syntax

<ad>XYnnn/nnn

	Units	Range	to	Initial Value
x-value	N/A	0	32767 (2^{15})	N/A
y-value	N/A	-32768	32767 ($\pm 2^{15}$)	N/A

Condition Requirements **Notes:**

Define Cam.

Responses:

OK	Command has been accepted.
! OUT OF RANGE	Argument, either X or Y is out of valid range.
! CONTEXT	Command may only be used in cam definition
! CAM ORDER	Cam co-ordinate non-monotonic.

Example:

Cam profiles are *piecewise linear*, with the first co-ordinate implicitly (x=0, y=0). A cam profile would be defined using the following commands:

1DM	Open Cam definition.
1XY200/500	Second Cam co-ordinate.
1XY400/500	Next Cam co-ordinate.
1XY600/-200	"
1XY700/-200	"
1XY750/0	Last Cam co-ordinate.
1EM	End Cam Definition.

8. FAULT FINDING

Here is a list of some common problems with PM304 controllers:

Problem	Possible Cause
Unable to communicate with controller.	All controllers and computer or terminal are not at the same *baud rate* or *word mode*. Incorrect transmit and receive wiring. Handshaking of computer or terminal not disabled. Incorrect commands sent - no axis address or *return* character.
Motor races off in one direction.	Phasing of the position encoder feedback is reversed. In this case either swap encoder connections A+ and A- with B+ and B-, or swap both the motor and the tacho (if used) lead polarities.
Motor does not move.	**K** coefficients not set - system initialised. Motor stalled. Controller aborted. Hard limits activated. Emergency Stop activated.
Motor Stalled.	No encoder feedback - encoder connected to the wrong input. Encoder type switches SW2 & SW3 set incorrectly. Insufficient motor or amplifier current.
Tune failure.	Encoder connections incorrect. Servo amplifier settings incorrect - motor tacho reversed (if used) amplifier not set for *flat* gain, current limit too low. Unsuitable load - offset or high inertia, backlash. Large gear ratio between motor and position encoder.
Unstable motor.	Incorrect **K** coefficients Servo amplifier settings incorrect - motor tacho reversed (if used) amplifier not set for *flat* gain, current limit too low.
Position seems to drift.	Noise on encoder signals - unsuitable cabling. Encoder fixing failure - loose grub screws.
Index Mark not seen.	Index pulse occurred for less than 1mS.

9. ERROR MESSAGES

! CAM ORDER	Cam co-ordinate non-monotonic
! CONTEXT	Command may not be used in current mode.
! DM SYNTAX	Command in Cam definition is not **XY** or **EM**.
! DP SYNTAX	Command in Profile definition is not **MR** or **EP**.
! EMERGENCY STOP	A move command was received when the Emergency Stop input has been activated.
! EM WITHOUT DM	An **EM** (end Cam) command was received without first using a **DM** command to define the start of the Cam profile.
! EP WITHOUT DP	An **EP** (end Profile) command was received without first using a **DP** command to define the start of the profile.
! ES WITHOUT DS	An **ES** (end sequence) command was received without first using a **DS** command to define the start of the sequence.
! HARD LIMIT	Hard limit input activated.
! ILLEGAL COMMAND	Command not recognised by the PM304.
! LIMIT CONFLICT	Soft limit values would *overlap* - i.e. the upper limit is lower than the lower limit.
! LIMITS DISABLED	An attempt to redefine the soft limit positions with soft limits disabled. The soft limit position is changed, but the message is reminder.
! MEMORY OFLO	The memory limit was exceeded during definition of a sequence, profile or cam.
! MOTOR STALLED	A command was received when the controller has aborted. The PM304 output has exceeded the **TH** value without the position encoder moving.
! NO CAM	An attempt to execute a non-existent Cam.
! NO PROFILE	An attempt to execute a non-existent Profile.
! NO SEQUENCE	An attempt either to execute a non-existent sequence or set the autoexecute flag on a non-existent sequence.
! NOT ABORTED	An **RS** (reset abort) has been sent unnecessarily.
! NOT STALLED	An **RSST** (reset motor stalled) has been sent unnecessarily.
! NOT STOPPED	An **RSES** (reset emergency stop) has been sent unnecessarily.
! OUT OF RANGE	Value is higher or lower than the allowed range.
! RECURSIVE DS	An attempt has been made to define the start of a sequence whilst already defining a sequence.
SKIPPED	Last command sent has been skipped if instructed to do so with a **SN** (skip next) command. Note; not actually an error message.
! SN SYNTAX	Number in **SN** command not appropriate.

! SOFT LIMIT	The soft limit range would be violated if the move command were executed.
! TRACKING ABORT	A move command was received when the controller has aborted due to a tracking error.
! TRACKING DISABLED	An attempt to redefine the size of the tracking window with that function disabled. The appropriate changes are made, but the message is a reminder.
! TUNE FAILURE	A stable set of servo coefficients could not be derived with the current system set-up.
! USER ABORT	A move command was received when the controller has been aborted by using an **AB** user abort command.
! WA SYNTAX	Number in **WA** command not appropriate.
! WP SYNTAX	Number in **WP** command not appropriate.

10. ELECTRICAL SPECIFICATION

Signal	Pin no.	Characteristics	
+24 V Supply	1a,1b,2a,2b	Supply voltage	+10.0 V to +32 V DC.
		Supply current (With 500mA from +5V output)	300 mA @ 24 V 600 mA @ 10 V 200 mA @ 32 V
+5 V Output	5a,5b	Output current	500 mA max.
Analogue Output	23a,23b	Output voltage (load 2.2kΩ)	-10V to +10V nom.
Analogue 0v	20a,20b		
Hard Limits	15a,15b	Opto-coupler anodes	
Isolated 0v	16b	Fed to opto-coupler cathodes via 4.7kΩ resistors	
		High level input current	3mA min.
		Input voltage	+10 to +32 V
Emergency Stop	12a	Opto-coupler anode	
Isolated 0v	12b	Fed to opto-coupler cathode via 4.7kΩ resistor	
		High level input current	3mA min.
		Input voltage	+10 to +32 V
Jog +, Jog -	9a,9b	Opto-coupler anodes	
Fast Jog	10a	Opto-coupler anode	
Isolated 0v	10b	Fed to opto-coupler cathodes via 4.7kΩ resistors	
		High level input current	3mA min.
		Input voltage	+10 to +32 V
Read Ports	13a,13b,14a,14b	opto-coupler anodes	
Isolated 0v	17b	Fed to opto-coupler cathodes via 4.7kΩ resistors	
		High level input current	3mA min.
		Input voltage	+10 to +32 V
Write Ports	3a,3b,4a,4b	Opto-coupler emitters	
Voltage source	17a	Opto-coupler collectors	
		Output current	10mA max.
		Source Voltage	+10 to +32 V
Error	11a	Opto-coupler emitter	
Voltage source	11b	Opto-coupler collector	
		Output current	10mA max.
		Source Voltage	+10 to +32 V
Encoder Inputs:			
TTL		Pulled up to +5V by 1KΩ resistor	
Enc. 1	7a,7b	Low level input current	-5.5 mA max.
Enc. 2	8a,8b	High level input voltage	5.0 V max.
		+ve going threshold voltage	1.6 V typ.
			1.4 V min.
			1.9 V max.
		-ve going threshold voltage	0.8 V typ.
			0.5 V min.
			1.0 V max.
RS422			
Enc. 1	21a,21b,22a,22b	RS422 Line receiver - 26LS32	
Enc. 2	18a,18b,19a,19b	Terminating resistor 180Ω	

11. Rear Connector Pin Assignments - Pin View.

Connector Type: DIN 41612, 64 WAY, A & B

POWER SUPPLY INPUT	1b	1a	POWER SUPPLY INPUT
POWER SUPPLY INPUT	2b	2a	POWER SUPPLY INPUT
▲ WRITE PORT 3	3b	3a	WRITE PORT 4 ▲
▲ WRITE PORT 1	4b	4a	WRITE PORT 2 ▲
+5V AUXILIARY OUTPUT	5b	5a	+5V AUXILIARY OUTPUT
	6b	6a	
ENC 1 TTL INPUT TRACK B	7b	7a	ENC 1 TTL INPUT TRACK A
ENC 2 TTL INPUT TRACK B	8b	8a	ENC 2 TTL INPUT TRACK A
※ JOG -VE	9b	9a	JOG +VE ※
JOG ISOLATED 0V	10b	10a	FAST JOG ※
ERROR V SOURCE	11b	11a	ERROR SIGNAL ▲
STOP ISOLATED 0V	12b	12a	STOP ※
※ READ PORT 3	13b	13a	READ PORT 4 ※
※ READ PORT 1	14b	14a	READ PORT 2 ※
※ HARD LIMIT -VE	15b	15a	HARD LIMIT +VE ※
HARD LIMIT ISOLATED 0V	16b	16a	
READ ISOLATED 0V	17b	17a	WRITE V SOURCE
ENC 2 RS422 INPUT B+	18b	18a	ENC 2 RS422 INPUT A+
ENC 2 RS422 INPUT B-	19b	19a	ENC 2 RS422 INPUT A-
ANALOGUE 0V	20b	20a	ANALOGUE 0V
ENC 1 RS422 INPUT B+	21b	21a	ENC 1 RS422 INPUT A+
ENC 1 RS422 INPUT B-	22b	22a	ENC 1 RS422 INPUT A-
ANALOGUE SIGNAL OUTPUT	23b	23a	ANALOGUE SIGNAL OUTPUT
RS232 0V	24b	24a	0V RS232
RS232 PORT 1 Tx	25b	25a	Tx RS232 PORT 2
	26b	26a	
	27b	27a	
RS232 PORT 1 Rx	28b	28a	Rx RS232 PORT 2
SUPPLY 0V	29b	29a	SUPPLY 0V
SUPPLY 0V	30b	30a	SUPPLY 0V
SUPPLY 0V	31b	31a	SUPPLY 0V
SUPPLY 0V	32b	32a	SUPPLY 0V
※ Opto-coupled inputs			
▲ Opto-coupled outputs			

12. APPENDICES

12.1 Command Table

GETTING STARTED COMMANDS			
HE<n>	***HE***lp pages	**IN**	***IN***itialise
TUNE	Auto ***TUNE***	**QA**	***Q***uery ***A***ll
QK	***Q***uery constants (***K***)	**QS**	***Q***uery ***S***peeds
ABORT, STOP & RESET COMMANDS			
CONTROL **C**	Hard Stop	**ESC**	Soft Stop
AA	***A***llow Tracking ***A***bort	**AB**	User ***AB***ort
IA	***I***gnore Tracking ***A***bort	**RS**	***R***e***S***et abort
RSES	***R***e***S***et ***E***mergency ***S***top	**RSST**	***R***e***S***et Motor ***ST***alled
ST	Soft ***ST***op	**TH**<VALUE>	Set Motor Stalled ***TH***reshold
TR<VALUE>	Set ***TR***acking window		
INFORMATION			
AD	Toggle ***AD***dress prefix	**CO**	Display the ***C***urrent ***O***peration
HE<n>	***HE***lp pages	**ID**	***ID***entify
OA	***O***utput ***A***ctual position	**OC**	***O***utput ***C***ommand position
OD	***O***utput ***D***ifference	**OI**	***O***utput ***I***ncremented position
OS	***O***utput ***S***tatus	**QA**	***Q***uery ***A***ll
QK	***Q***uery constants (***K***)	**QS**	***Q***uery ***S***peeds
COEFFICIENTS			
KF<VALUE>	Set ***F***eedforward coefficient	**KP**<VALUE>	Set ***P***roportional gain coefficient
KS<VALUE>	Set ***S***um coefficient	**KV**<VALUE>	Set ***V***elocity damping coefficient
KX<VALUE>	Set e***X***tra velocity feedback coefficient		
POSITION COMMANDS			
AP<VALUE>	Set ***A***ctual ***P***osition	**CP**<VALUE>	Set ***C***ommand ***P***osition
DA<VALUE>	***D***ecrement ***A***ctual position	**DC**<VALUE>	***D***ecrement ***C***ommand position
DI<VALUE>	***D***ecrement ***I***nput encoder's position	**IP**<VALUE>	Set ***I***nput encoder's ***P***osition
VELOCITY COMMANDS			
CV<DIR>	***C***onstant ***V***elocity mode	**SC**<VALUE>	***S***et ***C***reep speed
SF<VALUE>	***S***et ***F***ast jog speed	**SJ**<VALUE>	***S***et slow ***J***og speed
SV<VALUE>	***S***et ***V***elocity	**SA**<VALUE>	***S***et ***A***cceleration
SD<VALUE>	***S***et ***D***eceleration		
MOVE COMMANDS.			
CR<VALUE>	Set ***CR***eep steps	**DE**<VALUE>	Set ***DE***lay time
IX<DIR>	***I***nde***X*** to datum	**MA**<VALUE>	***M***ove ***A***bsolute
MR<VALUE>	***M***ove ***R***elative	**SE**<VALUE>	Set ***SE***ttling time
SOFT LIMIT COMMANDS			
AL	***A***llow ***L***imits	**IL**	***I***gnore ***L***imits
LL<VALUE>	Set ***L***ower *soft* ***L***imit	**UL**<VALUE>	Set ***U***pper *soft* ***L***imit
GEARBOX COMMANDS.			
GA	***G***earbox ***A***bsolute mode	**GB**	***G***ear***B***ox mode
GR<numerator>/<denominator>	***G***earbox ***R***atio		
			continued;

END OF MOVE COMMANDS			
DB<VALUE>	Set ***D***ead***B***and	**SE**<VALUE>	Set ***SE***ttling time
WE	***W***ait for ***E***nd of current move	**WI**<VALUE>	Set settling ***WI***ndow
READ & WRITE PORTS			
RP	***R***ead ***P***ort	**SN**<BIT PATTERN>	***S***kip ***N***ext
WA<BIT PATTERN>	***WA***it for input event	**WP**<BIT PATTERN>	***W***rite ***P***ort
JOG COMMANDS			
AR	***A***llow ***R***emote	**IR**	***I***gnore ***R***emote
SF<VALUE>	***S***et ***F***ast jog speed	**SJ**<VALUE>	***S***et slow ***J***og speed
SEQUENCE COMMANDS			
AE<n>	***A***uto-***E***xecute sequence	**DS<n>**	***D***efine ***S***equence
ES	***E***nd ***S***equence definition	**US<n>**	***U***ndefine ***S***equence
VS<n>	***V***erify ***S***equence	**XS<n>**	E***X***ecute ***S***equence
PROFILE COMMANDS			
DP	***D***efine ***P***rofile	**EP**	***E***nd ***P***rofile definition
UP	***U***ndefine ***P***rofile	**VP**	***V***erify ***P***rofile
XP<VALUE>	E***X***ecute ***P***rofile		
CAM COMMANDS			
DM	***D***efine ca***M***	**EM**	***E***nd ca***M*** definition
IM	Set cam index	**UC**	***U***ndefine ***C***am
VC	***V***erify ***C***am	**XM<n>**	E***X***ecute Ca***M***
XY	Cam co-ordinates		

12.2 System Variables

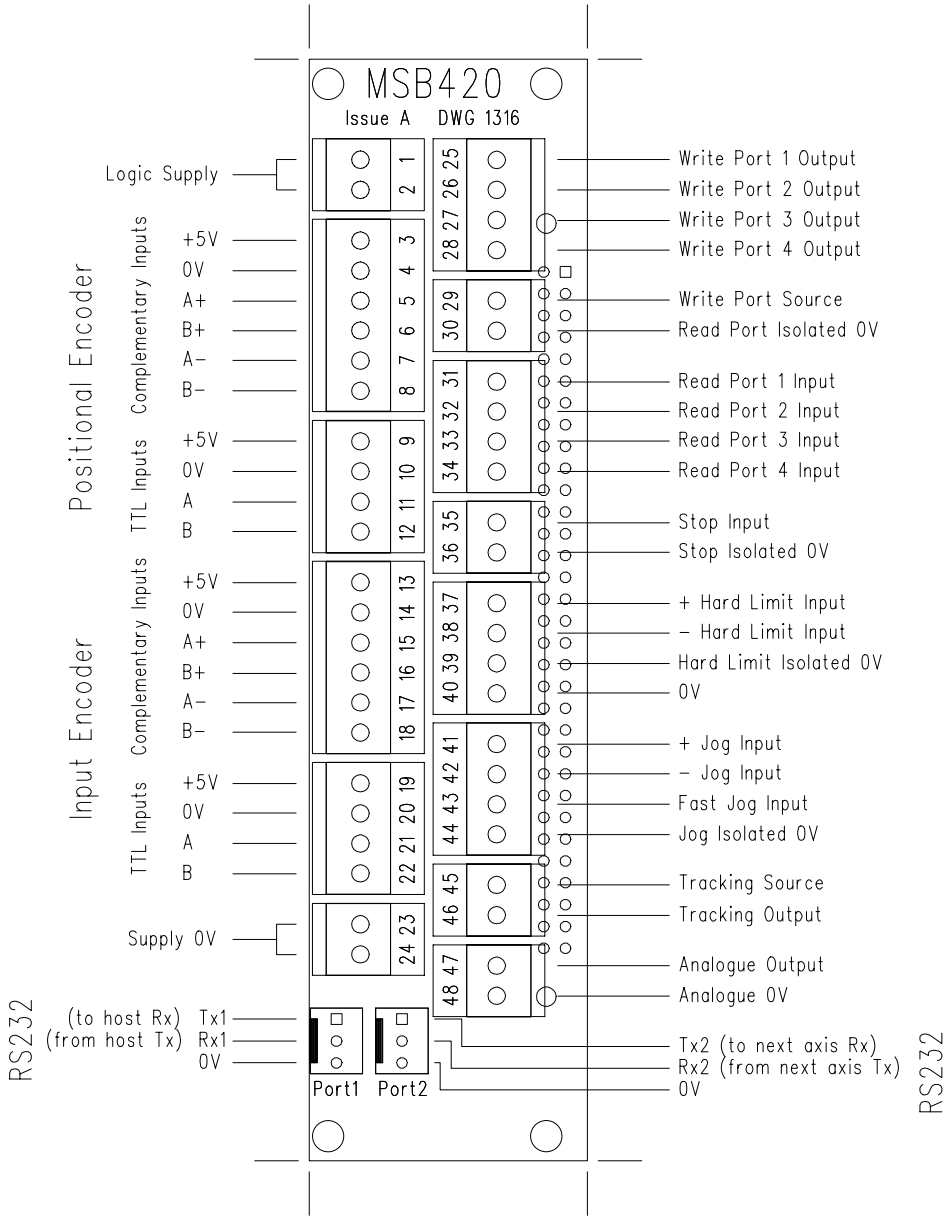
SYSTEM NUMBER/NAME	
AXIS NUMBER/NAME:	
OPERATOR	
DATE	
ENCODER SCALING	
Motor Encoder steps/rev	
Position Encoder steps at load	
Position Encoder steps/rev at motor	
SERVO LOOP COEFFICIENTS	
Proportional gain coefficient KP	
Sum coefficient KS	
Velocity feedback coefficient KV	
Velocity feedforward coefficient KF	
Extra velocity feedback coefficient KX	
SPEEDS AND ACCELERATIONS	
Maximum velocity SV	
Acceleration SA	
Deceleration SD	
Slow jog speed SJ	
Fast jog speed SF	
DYNAMIC CONSTANTS	
Creep Speed SC	
Creep Distance CR	
Deadband DB	
Settling Time SE	
Settling Window WI	
Tracking Window TR	
Tracking abort active? YES/NO	
Motor Stalled Threshold TH	
SOFTWARE LIMITS	
Software limits active? YES/NO	
Upper soft limit if active UL	
Lower soft limit if active LL	
ELECTRONIC GEARBOX	
Gearbox ratio GR	
SEQUENCES	
Sequences programmed?	
Auto execute sequence no.	
PROFILING	
Profile programmed? YES/NO	
CAM	
CAM Profile programmed? YES/NO	

12.3 QA Page Data

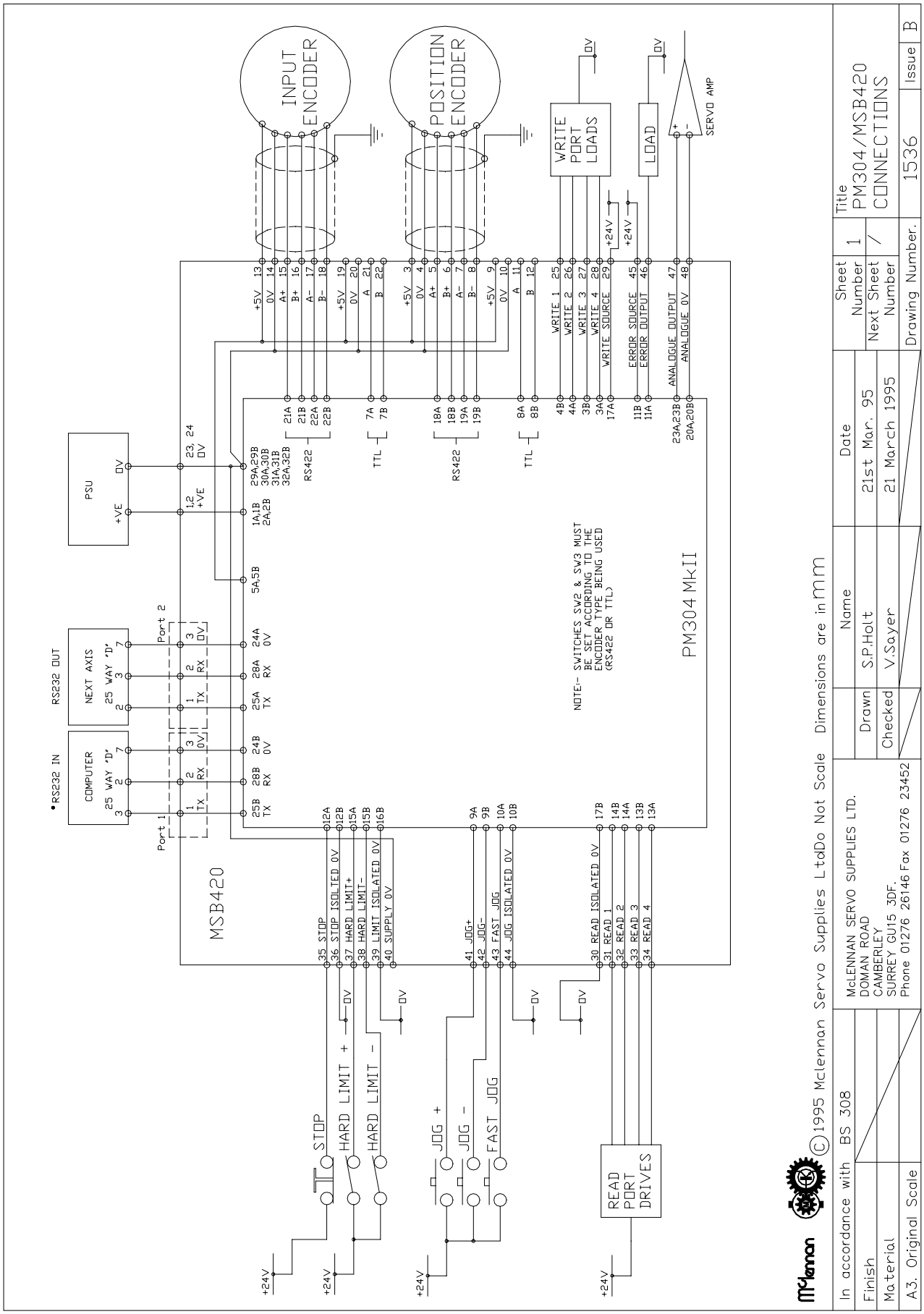
Version:			
Address:		Address Echo Enabled?	
KP		KS	
KV		KF	
KX			
Slew Speed =			
Acceleration =		Deceleration =	
Creep Speed =		Creep Steps =	
Jog Speed =		Fast Jog =	
Settling =		Deadband =	
Window =		Threshold =	
Tracking Abort Enabled?		Tracking =	
Soft Limits Enabled?		Jog Enabled?	
Lower Limit =		Upper Limit =	
Gbox Num =		Gbox Den =	
Autoexec Sequence no.			
Sequences:		Profile Defined?	
Cam Defined?		Cam modulo =	

12.4 MSB420 - Motherboard

The MSB420 motherboard is designed to mount in a Euro-rack to facilitate easier wiring to the PM304 via screw terminals and *Molex* connectors.



12.5 PM304/MSB420 Connections



12.6 Switch & Link Settings