

---

## Chapter 3. BASIC MOTION CONTROL CONCEPTS

---

**Chapter Objectives**

This chapter describes some of the basic concepts of motion control systems.

---

**Motion Profiles**

In any motion control application the most important requirement is precise shaft rotation, whether it be with respect to position, time or velocity. The type of motion profile needed will depend upon the motion control requirement. The following sections describe the basic types of motion profiles.

---

**Preset Moves**

A preset move referred to in this manual is a move of a specified distance (in user steps). Preset moves allow the user to position in relation to the motor's previous stopped position (incremental moves) or in relation to a defined zero reference position (absolute moves). Preset moves are selected by putting the positioner into incremental mode using the MPI command or the MN command and absolute moves are made using the MPA command.

---

**Incremental Preset Moves**

If the positioner is in the incremental mode (MPI command), a preset move will move the shaft of the motor the specified distance (in user steps) from its starting position. For example, to move the motor shaft 1.5 revolutions, a preset move with a distance of +6000 steps would be specified, assuming a 4000 step per rev encoder resolution setup. Every time this move is executed, the motor shaft will move 1.5 revolutions positive from its current position. The direction of the move can be specified at the same time as the distance by using the optional sign (D+6000 or D-6000), or it can be defined separately with the H command (H+ or H-).

---

**Absolute Preset Moves**

A preset move in absolute mode (MPA command) will move the shaft of the motor the specified distance (in user steps) from the absolute zero position. The absolute position can be set to zero with the PZ or SP commands, for instance at the end of a GO HOME move (GH command). The absolute zero position is initially the power-up position, and will remain that way until changed with a PZ command. Any preset move performed while in the absolute mode will position the motor shaft the defined distance (in user steps) from the absolute zero position. For example, with the positioner at the absolute zero position a move with a distance of +4000 will cause the motor shaft to turn 1 revolution in the positive direction. If a move with the same defined distance is executed immediately after this move, the motor shaft will not turn, since it is already +4000 steps from the absolute zero position.

The direction of an absolute preset move will depend upon the shaft position at the beginning of the move and the position that it is being commanded to move to. For example, if the motor shaft is at absolute position +12,800, and position commanded is +5000, the motor shaft will move in the negative direction a distance of 7800 steps to absolute position +5000.

The positioner saves the mode that it was in at power down and powers up again in the same mode. Issuing the MPA command will set the mode to absolute. Issuing the MPI command or MN command will switch the mode from absolute to incremental. The positioner retains the absolute position referenced, even while in incremental mode. However, the counter does have an upper limit just slightly more than 268,400,000 counts.

---

### **Continuous Moves**

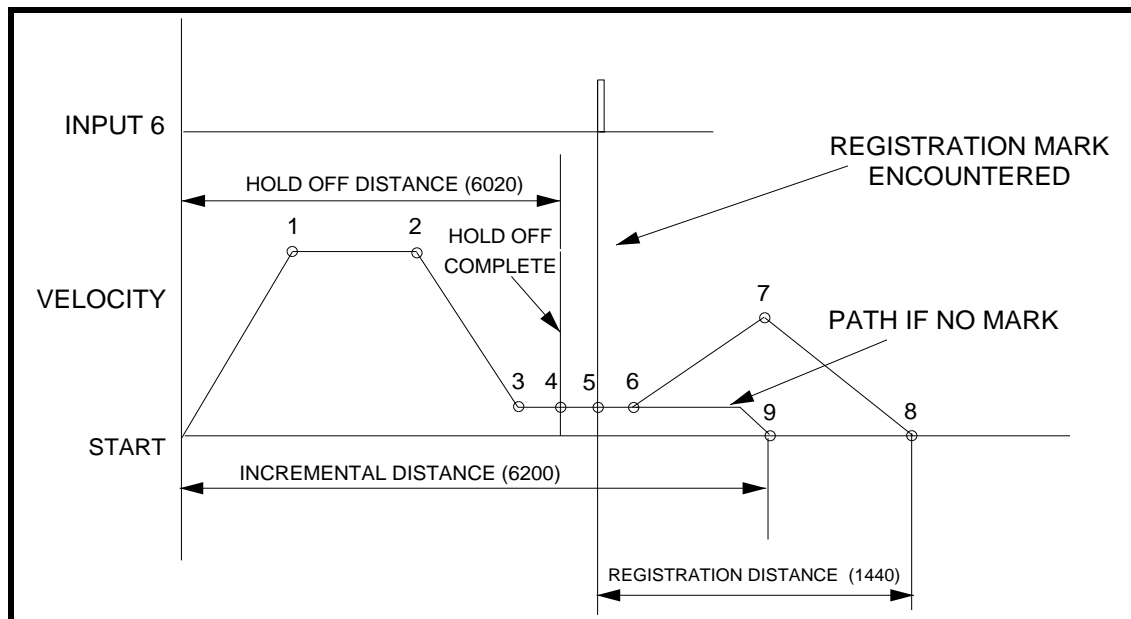
Continuous moves (MC command) cause the motor to accelerate and attain the specified velocity and then continue to move. To change velocity while the motor is moving use the V command. Issuing a stop (the S command) will cause the motor to decelerate to a stop at the last defined acceleration rate. The distance parameter is not used, although it is saved in case the mode is changed back to preset.

This mode is useful for applications which require constant spinning of the load, when the motor must stop after a period of time has elapsed rather than after a fixed distance, or when the motor must be synchronised to external events such as trigger input signals.

---

### **Registration Moves**

A move may be programmed to end a specified distance after a registration pulse appears at Input 6. The mode is selected using the TRR command with the required registration distance in the MQ or MC modes. Its use is easiest to understand in the context of an example registration move:



**Figure 3-1. Example Registration Move**

<b>Example Program</b>	MQ	Select MQ mode
	D6200	Set move distance to 6200 steps
	A100	Set all accelerations to 100 steps/rev <sup>2</sup>
	V10	Set velocity to 10 revs/sec
	G	Go
	TRD4000	At distance = 4000 steps
	V1	Change velocity to 1 rev/sec
	TRD6020	At distance 6020 look for registration pulse
	TRR1440	Travel 1440 steps from registration pulse
	V50	At a velocity of 50 revs/sec
	TRIP	Set Output 2 when in position

**Program Operation**

When executing this program, the axis will first accelerate at 100 steps/rev<sup>2</sup> to 10 revs/sec (1) and when it reaches a distance of 4000 steps (2), its speed will be reduced to 1 rev/sec (3) whilst it is waiting for the registration pulse. Shortly after the pulse is received (4), the axis will accelerate towards 50 revs/sec, but will not reach the speed before starting to decelerate (5) to stop at 1440 steps from where the pulse occurred (6).

If the pulse was not received after 6020 steps (the hold off distance) and before 6200 steps (the incremental distance), the axis will stop at 6200 steps. This sets a window of 180 steps during which the pulse is expected.

**Program Criteria**

The distance command (D6200)	This command sets the distance the axis will travel if no registration pulse is received and the maximum distance at which the pulse will be recognised.
Trigger distance command (TRD4000)	The distance at which the axis will start to decelerate to the slower speed of 1 rev/sec (V1) in readiness for the registration pulse is set by this command.
Trigger distance command (TRD6020)	This command sets the minimum distance at which the system will recognise the registration pulse.
The TRR command (TRR1440)	This command sets up the registration mode and the registration distance (the distance the axis will travel after the pulse is received)
Registration move velocity (V50)	To save time in travelling the registration distance, the axis then accelerates towards 50 revs/sec (V50), but it does not reach that speed before starting to decelerate to stop at 1440 steps from where the pulse occurred.

---

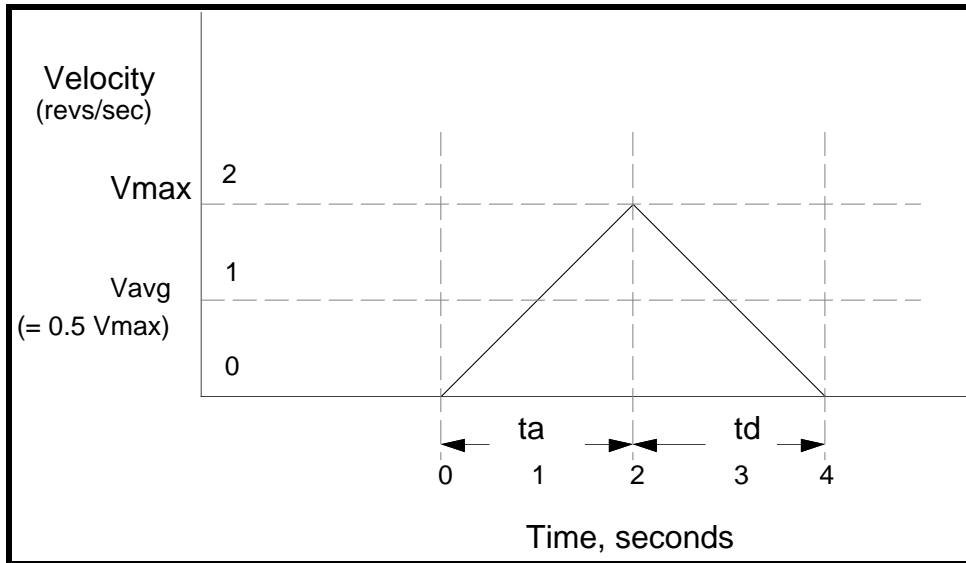
**Motion Profiles**

Velocity, acceleration and distance must be defined before any preset move can be executed. The value of these parameters determines the type of motion profile as either triangular or trapezoidal.

---

**Triangular Profile**

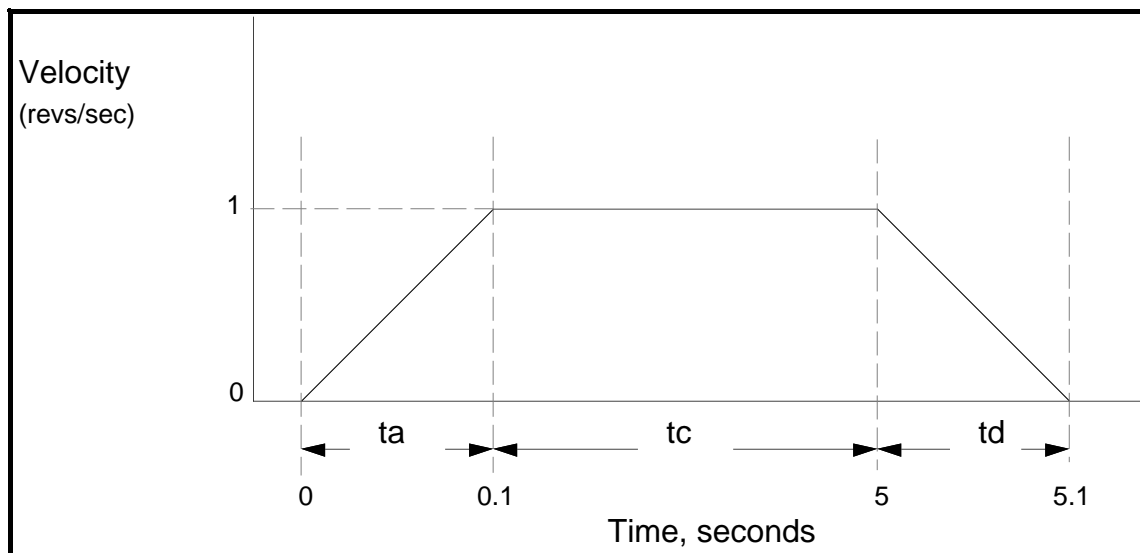
A triangular profile will result when the velocity and acceleration are set such that the designed velocity is not attained before half of the specified distance has been travelled. This results from either a very low acceleration or a very high velocity or both over a relatively short distance. For example, if the acceleration is set to 1 rev/sec/sec, velocity is set to 5 revs/sec and distance is set to 16000 steps (2 revs), a triangular motion profile will result. This is because by the time the motor shaft has reached a velocity of 2 revs/sec, it will also have travelled half of the defined distance due to the acceleration setting of 1 rev/sec/sec. The motion profile for this move would look like this.



**Figure 3-2. Triangular Profile**

**Trapezoidal Profile**

A trapezoidal move profile results when the defined velocity is attained before the motor shaft has moved half of the specified distance. This is due to a defined velocity that is low, a defined acceleration that is high, a move distance that is long, or a combination of all three. For example, if the acceleration is set to 10 revs/sec/sec, velocity is set to 1 rev/sec, and distance is specified as 20000 steps (5 revs), the resulting motion profile would look like this:



ta = Accelerate  
 tc = Constant velocity  
 td = Decelerate

**Figure 3-3. Trapezoidal Profile**

**User Profiles**

The user may define a move profile using the MC command to

establish the continuous mode. Velocity can be programmed on the fly by the V command. Sending commands to the positioner in rapid succession allows smooth profiling which will allow circles and arcs to be traced using a two-axis system, or allow smooth (low jerk) motion by virtue of S-curve acceleration rather than straight ramping as with triangular and trapezoidal moves. Once a sequence of commands is found to trace the correct profile, the profile will be very repeatable. This may require some trial and error to establish the correct sequence of V commands. The V commands can be combined with time delays in a sequence buffer to create a very complex move profile.

The positioner also has a mode MQ, which is like the preset move mode in that the move distance is pre-defined, but it is possible to change speed in the middle of the move as required based on a distance, input or time delay trigger.

---

### Encoder Following

The commands SIM and CCS may be used to allow the motor of one axis to follow the encoder of another axis or an externally-generated clock and direction signal. When the control module is to be used in following mode, the input from the encoder to be followed should be connected to the terminals marked CLK+, CLK-, DIR+ and DIR- on the motherboard. These inputs can be configured using the CCS command as an encoder input (x1, x2 or x4) or as a clock and direction input. For encoder input, connect:

CHA+	to	CLK+
CHA-	to	CLK-
CHB+	to	DIR+
CHB-	to	DIR-

The SIM command may be used to select:

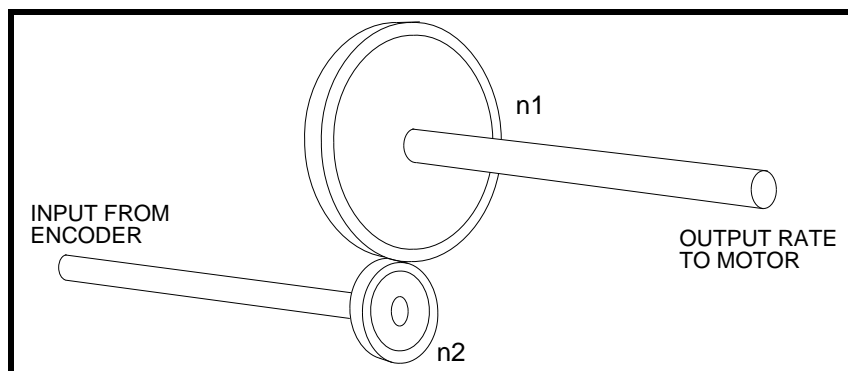
- a) Normal control module operation (SIM0), used for reverting to normal operation by overriding previous SIM commands
- b) Encoder following with control module motion commands inoperative (SIM1).
- c) Encoder following with motion control commands operative (SIM 2), allowing the superimposing of indexer moves.
- d) Software scaled encoder following (SIM3).

- e) Software scaled encoder following with direction reversal (SIM4).
- f) Preset following index mode (SIM 5).

For SIM1 and SIM2 operation the motor output rate follows the encoder input, using hardware scaling, at a ratio of 1 or less.

#### Hardware Scaling

This is the scaling of the second encoder input when using the SIM1 and SIM2 commands to achieve following at a ratio of 1 or less relative to the following input. Scaling is achieved using a hardware rate multiplier, the division ratio of which is set by the RAT command, at a resolution of 16 bit or 24 bit operation determined by the OSJ command.



**Figure 3-4. Simple Gearbox**

The operation of the rate multiplier can be compared to that of a simple gearbox (see Figure 3-4), where the ratio of output revolutions to the input revolutions (always less than or equal to 1) is determined by the number of teeth on n2 divided by the number of teeth on n1. If the number of teeth on n1 is fixed, but the number of teeth on n2 is allowed to vary (up to a maximum of n1 teeth), the gearbox will provide a variable output rate which is always a fraction of the input rate. The resolution of the gearbox, that is how fine a gear ratio can be set, will be determined by the number of fixed teeth on n1. In the rate multiplier the RAT value sets the number of teeth on n2, and the resolution (n1 teeth) can have one of two fixed values determined by the OSJ command.

Using hardware scaling the output motor rate is determined by :

$$\text{Output Motor Rate} = \text{input rate} \times \frac{n}{65536} \text{ for OSJ} = 0$$

or

$$\text{Output Motor Rate} = \text{input rate} \times \frac{n}{16777216} \quad \text{for OSJ} = 1$$

Where n = the RAT value

**NOTE:** RAT now controls the division ratio, and the direction will be determined by the sign of the RAT value.

As the resolution (set by OSJ) is a large number, very fine adjustment of the ratio can be set by choosing a suitable value for the RAT command.

A limitation of binary hardware scaling is the lack of certain exact following ratios that can be obtained. Since the division ratio can only be a binary fraction it is impossible to choose an exact ratio such as 1:3, although it could be closely approximated if you were to choose a RAT value of 5592405 with 24-bit resolution selected.

Summary of hardware scaling :

The scaling ratio is determined by the RAT value.

Resolution is fixed at 16-bit or 24-bit , determined by the OSJ command.

Non binary fraction exact division ratios cannot be obtained.

---

#### Software Scaling

This is the scaling of the encoder input when using the SIM3 or SIM4 commands to achieve following at a ratio greater or less than 1. Unlike hardware scaling, exact following ratios can be achieved by controlling both the numerator and denominator parts of the fraction used to set the scaling ratio, thus ratios such as 3:1 can be obtained.

The scaling ratio is set using the CMR command value divided by the CUR command value to give :

$$\text{Motor Output Rate} = \text{2nd input rate} \times \frac{\text{CMR}}{\text{CUR}}$$

Software scaling works by calculation, allowing the motor to run 255 times faster than the encoder or 255 times slower. But because the the processor is always busy calculating pulse rates, it is not able to superimpose indexer moves on the motion.

Both CMR and CUR can take any value up to 32,767 but you must be able to write the CMR/CUR ratio as a fraction using numbers less than 255. If you cannot achieve this, the indexer will not accept it as a valid ratio and the following error message will be output:

#70 "255 reduced ratio exceeded"

For example, a motor has 4000 steps/rev, and the following source is a 500-line encoder giving 2000 counts/rev. We want the motor to move one rev for every 5 revs of the encoder, in other words 10,000 encoder counts should produce 4000 motor steps. The speed ratio must be 4000/10,000.

So CMR = 4000, CUR = 10,000. We can reduce the CMR/CUR fraction to 4/10, so both numbers are within the 255 limit. This ratio would be OK.

Consider a second example where the motor is still 4000 steps/rev, but this time the encoder has a binary output with 4096 counts/rev. We want one rev of the encoder to produce 20 revs of the motor, so the ratio is 80,000/4096. Dividing each by 4 will give CMR = 20,000 and CUR = 1024, so both values are less than 32,767.

The ratio 20,000/1024 can be further reduced to 625/32, but no further using whole numbers. So this ratio will not work because 625 is outside the 255 limit.

In general it's better to try and make the following encoder resolution the same as the motor resolution, or at least a simple ratio of it. This will give the widest choice of valid following ratios.

When using SIM3 or SIM 4 operation for short moves it is possible to predict the number of steps the motor will take using the formula :

Number of motor steps to move =

$$\text{INT} \left( \text{CMR} \times \text{no. of pulses received} + \frac{\text{prev. remainder}}{\text{CUR}} \right)$$

where INT means "take the integer part of", with the value rounded towards zero whether positive or negative. The controller can repeatedly apply this formula to establish an iterative calculation. In situations where short moves have been programmed the ratio of CMR/CUR may only approximate the number of steps the motor is required to move, but since the remainder is carried forward no steps are lost. This allows the CMR/CUR value to better approximate the following ratio in subsequent moves. In summary, short moves may only approximate the defined following ratio, but no positioning accuracy is lost in later moves.

**Mode** controller to buffer an unprofiled following-input pulse stream until the output velocity equals the following input velocity. The controller accelerates the output velocity to match the input velocity using an exponential acceleration profile, the time constant of which is set using the CAG command. The default time constant value of CAG is 1.00ms to maintain compatibility with earlier software issues, and to accept already profiled follower inputs.

The input pulses are buffered (or stored) at the input resolution, the actual number being stored at any particular instant being termed the following error. This can lead to input pulses being lost above a maximum speed, due to excessive following error.

The maximum number of input pulses that can be buffered is

+/-32767 which requires CAG to be less than  $\frac{32767}{\text{input pulse rate}}$  to prevent following error overflow.

**Comparison of following features**

<b>Hardware Scaling SIM1, SIM2</b>	<b>Software Scaling SIM3, SIM4</b>
The motor always moves by the same or fewer pulses than received. The pulse stream is effectively divided.	The motor can move by more or less pulses than received. The input pulse stream can be multiplied or divided.
The ratio is programmed by the RAT command with 16 or 24 bit resolution as a binary fraction.	The RAT command has no effect.
CMR and CUR have no effect.	The ratio is programmed by CMR and CUR.
Superposition onto an additional internally generated indexer motion is available.	Superposition is not available.
The pulse stream is followed directly without additional profiling.	The pulse stream is profiled (filtered) by a programmable exponential characteristic (CAG command)

**Preset Following Index Mode**

The following mode SIM5 selects indexing at a speed determined by the external input. In this mode the controller sets the motor velocity to a speed in rps determined as a percentage of the following input speed in rps. The percentage following factor is set by the FOL command which can be varied between 0.0 and 5000.0%.

When using this mode the acceleration is fixed to whatever is defined by the A command, and the V command value has no effect since the FOL command percentage value will now control the velocity.

The velocity is dependent on the user resolution (CUR value) and the motor resolution (CMR value). CUR and CMR set the encoder / motor resolution ratio so that the input shaft speed will match the output shaft speed with FOL set to 100%.

---

**Program Storage**

The program memory is battery backed up RAM with memory retention of 1000 hours. The RAM has 6K characters available for sequence storage and a write protect facility is provided. Programs are stored in variable length buffers and the total length, including servo parameters, cannot exceed 6K.

**Write Protection**

A write protection facility is incorporated for the protection of stored sequences and parameters stored by the SV command. If LK1 on the positioner card (see Figure 4-3) is fitted, then the memory is not write protected. An attempted SV command when the jumper link is not made will result in an error message being displayed.

---

**Motion Program Selection**

The system may be set up so that no sequence is executed at power-up. In this case sequence execution would be initiated from the controller over the RS232C.

Alternatively, a single sequence to be automatically executed at power-up can be programmed. After that sequence is executed, control can pass to another sequence or to the RS232C interface.

Sequence selection may also be initiated via inputs. Up to 63 user defined sequences can be selected for execution in this way.

---

**Parameter Ranges**

Table 1-3 gives the internal arithmetic positioner limits based on 4000 encoder steps/rev. These parameters apply only to the positioner and the motor/drive combination will not necessarily be able to respond to the full ranges stated:

	<b>Position</b>	<b>Velocity</b>	<b>Acceleration</b>
<b>Smallest</b>	1 step	0.488 steps/sec 0.000122 RPS 0.00732 RPM	244 steps/sec/sec 0.061 RPS <sup>2</sup> 3.66 RPM/sec
<b>Largest</b>	±268,435,455 steps ±500 revs at 4000 steps/rev	4 x 10 <sup>6</sup> steps/sec 999 RPS	4 x 10 <sup>9</sup> steps/sec/sec 999,999 RPS <sup>2</sup>

**Table 3-2. Parameter Ranges**

**NOTE:** Longer distances than 268,435,455 steps can be achieved using a sequence of incremental moves or continuous motion mode.

Additional flexibility is possible using false motor resolutions (e.g. 2,000 steps/rev instead of 200).

---

## Chapter 4. COMMUNICATING WITH THE POSITIONER

---

### Chapter Objectives

The information contained in this chapter will enable you to set up communications with the positioner. If more than one positioner is present in the system, details are provided on the connection of multiple positioners using the Zero Delay Daisy Chain.

---

### Command Interface

The interface is a three wire implementation (Tx, Rx, Ground) of RS232C. The Tx and Rx lines requires a minimum voltage swing of  $\pm 3$  volts.

Hardware handshaking is not supported in any form. The computer or terminal sending characters to the positioner should have its handshaking disabled by either hardware or software.

---

### Communication Parameters

The positioner communications protocol is fixed and as follows:-

9600 baud, 8 data bits, no parity, 1 start bit, 1 stop bit

Device address: 1-8 (set with jumper links on the positioner)

Echo function: all characters received are immediately re-transmitted by the last device to be addressed. This function can be enabled using the SSA0 command or disabled using the SSA1 command.

Once entered, a parameter will be remembered. It is not necessary to keep re-stating the same parameter value.

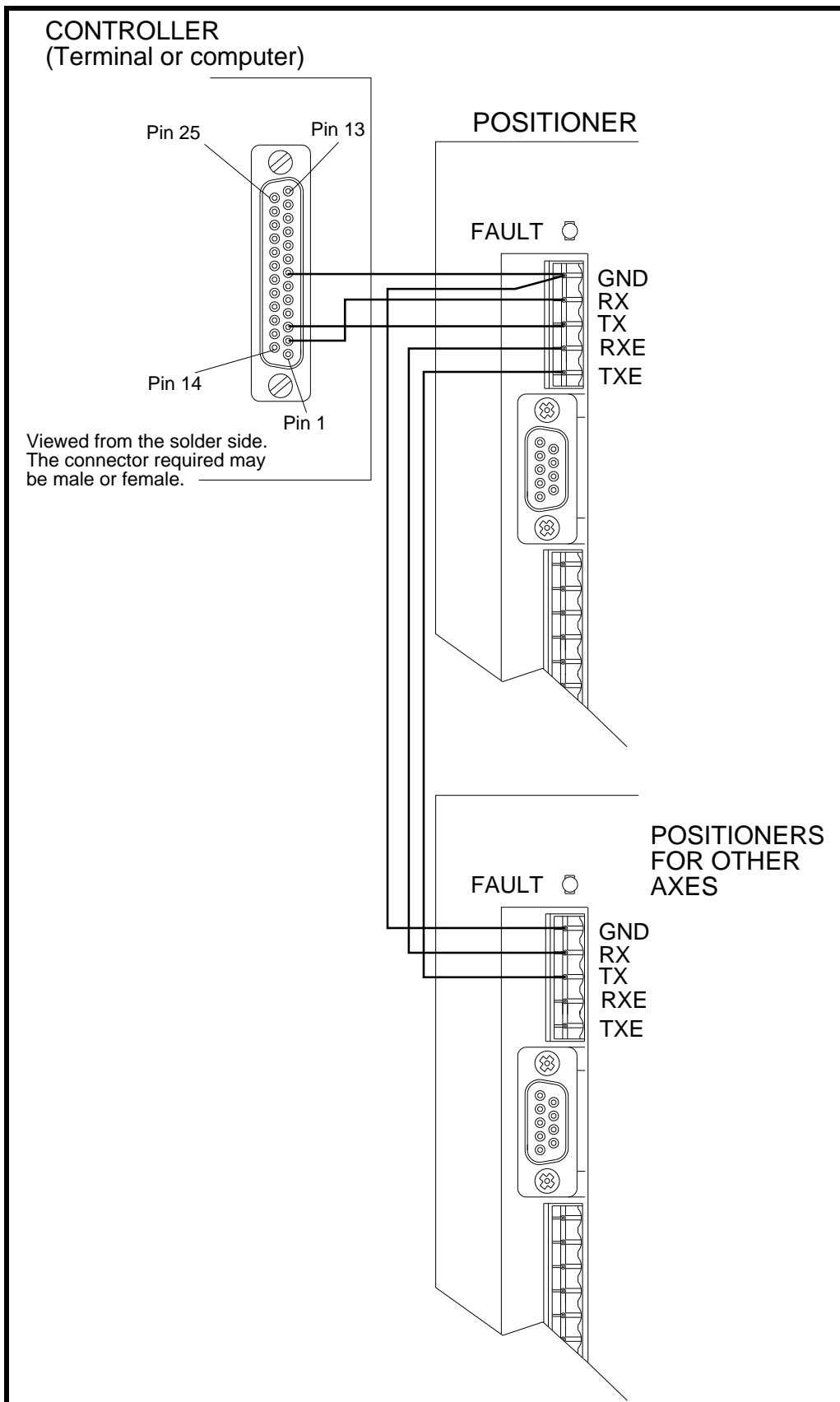
When you power up the positioner all parameters are assigned default values which the 1DR command will show you. You can change any of these and then make them the new default value by typing the SV (Save command). (see also commands RFS, RIFS).

XON, XOFF software handshaking is supported.

---

### Installing the RS232C

RS232C connections from the controller to the positioner are as shown in Figure 4-1. Note that the Tx and Rx lines are cross-connected so that transmit output is connected to receive input.



**Figure 4-1. Controller to Positioner Connections**

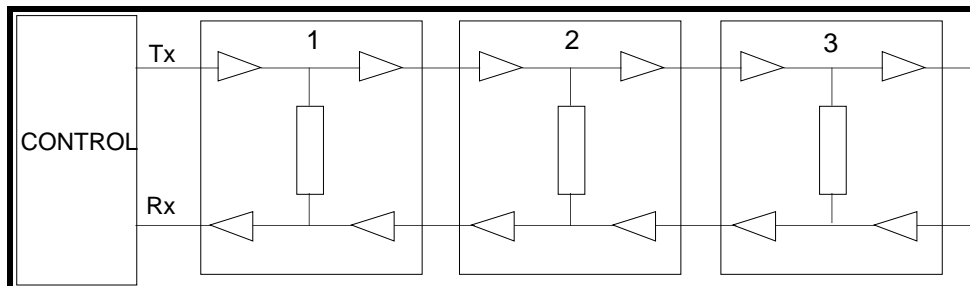
A cable suitable for this connection is available from Digiplan. Its part number is 7967.100. The connections RXE and TXE are for connection to other positioners in the system using the ZDDC as shown.

### ZDDC (Zero Delay Daisy Chain)

The Zero Delay Daisy Chain is used to connect up to eight positioners to the controller and keep propagation delays to a minimum. Characters sent by the controller to the positioners are echoed back to the controller to verify that the character has been correctly received. In the ZDDC, for practical purposes, all devices are connected in parallel. To avoid confusion of data on the return line, only the addressed positioner performs the echo back.

Universal commands will be echoed back by the last positioner to be addressed; this function defaults to device no.1 at power-up.

Each positioner is equipped with an additional RS232C receiver and transmitter to enable successive positioners to be linked together.

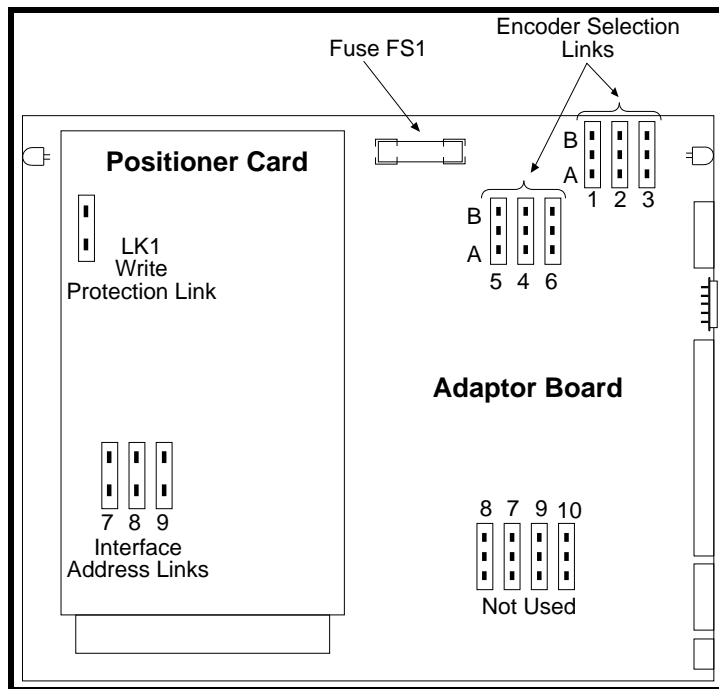


**Figure 4-2. Zero Delay Daisy Chain**

### Positioner Address Jumper Linking

A unique address must be assigned to each positioner in the chain. Normally positioner 1 will be the first in the serial communication link. The positioner address is assigned by configuring jumper link 6, 7 and 8 on each positioner in a binary pattern as shown in Table 4-1.

The device address should ideally be set before the system is installed. Changing the address setting will cause the positioner to lose the drive definition, so it will need to be re-defined (see RFS command). This is done to minimise the risk of unexpected movement following an address change.



With jumper links 1 to 6 on the adaptor board in position B, the positioner accepts signals from the encoder wired to the Motor Feedback socket on the drive. With these jumper links in position A, the positioner accepts signals from a separate encoder connected to the 9-way Encoder socket on the adaptor board.

Links 7 to 10 on the adaptor board are not used and must be left in position A. Links 2 to 6 on the positioner card are not used.

**Figure 4-3. Positioner Jumper Link Locations**

	Jumper Link 7	Jumper Link 8	Jumper Link 9
<b>Interface 1</b>	0	0	1
<b>Interface 2</b>	0	1	0
<b>Interface 3</b>	0	1	1
<b>Interface 4</b>	1	0	0
<b>Interface 5</b>	1	0	1
<b>Interface 6</b>	1	1	0
<b>Interface 7</b>	1	1	1
<b>Interface 8</b>	0	0	0

1 = link fitted 0 = no link

**Table 4-1. Interface Address Jumper Links**

### Drive Configuration

The BL or BR drive should normally be configured as a torque amplifier when the positioner option is included. Please refer to the drive User Guide for information on setting up as a torque amplifier.

Configure the positioner to suit the drive by typing RFS4 (for a BL 16 or 23-size motor), followed by the SV command to save. Refer to the Command Listing for further information on these commands.

**Using  
Windows™**

Microsoft Windows™ comes with a terminal emulation program which can be used in place of XWARE. The terminal emulation program is configured as follows:

Run the terminal program.

From the FILE menu select NEW.

From the SETTINGS menu, setup the following options:

Terminal emulation sub-menu	DEC VT-100 (ANSI)
Terminal preferences sub-menu	Terminal modes Line wrap
ON	
	Local echo OFF
	CR_>CR/LF Inbound ON
Communications sub-menu	Baud rate 9600
	Data bits 8
	Stop bits 1
	Parity NONE
	Flow control NONE

Also choose the appropriate connector

These basic configurations will allow operation with the terminal only.

