

**PVP152**  
**Single Axis**  
**Stepper Motor Controller**  
**User Manual**

## **Copyright Notice**

This document is the Copyright of Alzanti Limited. Reproduction or storage by any means without the express written permission of Alzanti Limited is prohibited.

All rights reserved  
Copyright © Alzanti Limited 1995..2001

## **Disclaimer**

All information in this manual is given in good faith; however we cannot be held liable for any errors or omissions or for any consequential loss or damage arising therefrom.

Alzanti Limited reserves the right to make changes without further notice to any products or information mentioned herein. Alzanti Limited makes no warranty, representation or guarantee regarding the suitability of this product for any particular purpose, nor does Alzanti Limited assume any liability arising out of the application or use of this product.

Alzanti Limited equipment is not designed, intended, or authorized for use in systems intended to support or sustain life, or for any application in which the failure of the product could create a situation where personal injury or death could occur.

The responsibility of determining the suitability of this product for a particular application lies with the purchaser.

## **How to contact Alzanti**

If you have any queries regarding this or any other of our products then please contact us weekdays from 9:00am to 5.00pm (GMT or BST) at the address below:

Alzanti Limited  
The Warren,  
Darby Green Lane,  
Blackwater,  
Camberley  
Surrey, GU17 0DN.  
United Kingdom

Tel: +44 (0)1252 861113  
Fax: +44 (0)1252 861103  
Email: <mailto:sales@alzanti.com>  
Web: <http://www.alzanti.com/>

## Table Of Contents

1 Introduction .....	5
2 Electrical connections .....	6
2.1 Input Details .....	6
2.2 Output Details .....	6
2.3 Serial Communication .....	6
2.4 Pin Out Details .....	7
2.4.1 Power Inputs .....	7
2.4.2 Motor Outputs.....	7
2.4.3 Serial Communications.....	7
2.4.4 Control Inputs .....	8
2.4.5 Control Outputs .....	8
2.4.6 General Purpose Inputs.....	8
2.4.7 General Purpose Outputs.....	9
3 Getting Started .....	10
4 Using Terminal Mode .....	11
4.1 Starting Terminal Mode.....	11
4.2 Creating a stored program .....	12
5 Computer Mode .....	13
5.1 Starting Computer Mode .....	13
5.2 Using Computer Mode .....	14
6 Command Reference .....	15
6.1 STOP .....	15
6.2 STATUS.....	15
6.3 ?INPUTS .....	16
6.4 ?MODE .....	17
6.5 ?POS .....	18
6.6 ?SETUP .....	19
6.7 INSERT .....	20
6.8 DELETE .....	20
6.9 ACCEL .....	21
6.10 BASE .....	21
6.11 BACKLASH.....	22
6.12 DELAY .....	23
6.13 ERASE.....	24
6.14 END MOVE.....	25
6.15 FULLSTEP & HALFSTEP .....	26
6.16 GO .....	26
6.17 GOSUB .....	27
6.18 HELP.....	28
6.19 HIGH .....	29

6.20 HOME FORWARDS & HOME REVERSE .....	30
6.21 IF INPUTS.....	30
6.22 NOJOG & JOG.....	31
6.23 JUMP TO .....	32
6.24 LIST .....	33
6.25 LOOP TO .....	33
6.26 MOVE ABSOLUTE .....	34
6.27 MOVE RELATIVE .....	34
6.28 OUTPUT .....	35
6.29 POS= .....	35
6.30 RANGE .....	36
6.31 READ .....	36
6.32 RETURN .....	37
6.33 START FWD, START REV .....	37
6.34 WRITE .....	38
7 Speed Control .....	41
8 Utility Programs.....	44

## Introduction

The PVP152 is a standalone microprocessor based motion control system. The unit incorporates the following facilities:

- \* On board bipolar stepper drive (1 Amp per phase) for use with small motors.
- \* Pulse and direction outputs compatible with Alzanti's range of stepper drives.
- \* Eight general purpose inputs.
- \* Eight general purpose outputs.
- \* Emergency stop input.
- \* Upper and Lower limit inputs.
- \* Jog forward and reverse inputs.
- \* Run stored program input.
- \* Datum input.
- \* EEPROM storage of motion control programs.
- \* Three operational modes: Terminal/Computer/Standalone.
- \* RS232 interface allowing 15 controllers to be connected to one serial port.
- \* Comprehensive, simple to use command language.
- \* Full and Half step modes provided.
- \* Position range +/- 16,777,216 pulses

## Electrical connections

### Input Details

All inputs are TTL & CMOS compatible up to 30Vdc input maximum.

Emergency Stop - Normally Closed to 0V

Upper (Forward) Limit - Normally Closed to 0V

Lower (Reverse) Limit - Normally Closed to 0V

Datum (Home) - Normally Open to 0V

8 General Purpose Inputs - Normally Open to 0V

**(Note:** These inputs can be configured for 'sourcing' inputs, see section 2.4.6)

Jog Forward - Normally Open to 0V

Jog Reverse - Normally Open to 0V

Run Program - Normally Open to 0V

Reset Controller - Normally Open to 0V

Motor Standby Current - Normally Open to 0V

### Output Details

8 General Purpose Outputs - Current sinking 100mA each up to 30Vdc

Commutation Diode Clamp input for drivers (30V dc Max)

**(Note:** These outputs are also available as 'sourcing' outputs, contact Alzanti for details)

Step Pulse Output - TTL Open Collector Output (20ma 30V max)

Direction Output - TTL Open Collector Output (20ma 30V max)

Fullstep / Halfstep Output - TTL Open Collector Output (20ma 30V max)

Motor Stopped Output - TTL Open Collector Output (20ma 30V max)

### Serial Communication

RS 232C Running at 9600 baud **ONLY**

1 x RxD & 1 x TxD - Input from source

1 x RxD & 1 x TxD - Output for Daisy Chaining



**Control Inputs** (Sinking/NPN)

Pin 10a	Emergency Stop	)	Active High Inputs (Open Circuit)
		)	
Pin 9b	Upper (Forward) Limit	)	
		)	
Pin 9a	Lower (Reverse) Limit	)	
Pin 8b	Datum (Home) Switch	)	Active Low Inputs (Close to 0V)
		)	
Pin 8a	Jog Forward Button	)	
		)	
Pin 7b	Jog Reverse Button	)	
		)	
Pin 10b	Standby Motor Current	)	
		)	
Pin 11a	Reset Controller	)	
		)	
Pin 7a	Run Stored Program	)	

**Control Outputs** (Sinking/NPN)

Pin 22b	<u>Step Pulse</u>	)	For use with an external drive or control system
		)	
Pin 22a	Direction Control	)	
		)	
Pin 21b	<u>Half / Full Step Mode</u>	)	
		)	
Pin 21a	<u>Motor Stopped (Busy)</u>	)	

**General Purpose Inputs**

Pin 19a	Input 8 (MSB)	)	Active Low Inputs (Close to 0V)
		)	
Pin 18a	Input 7	)	<b>Note:</b> These inputs are configured as sinking/NPN as standard (LK2 = '+'), for sourcing/PNP inputs move LK2 to position '0'. The logic interpretation of these inputs is unaffected, see section 6.3
		)	
Pin 17a	Input 6	)	
		)	
Pin 16a	Input 5	)	
		)	
Pin 15a	Input 4	)	
		)	
Pin 14a	Input 3	)	
		)	
Pin 13a	Input 2	)	
		)	
Pin 12a	Input 1 (LSB))	)	

**General Purpose Outputs**

Pins 20 a&b Output Commutation (Clamp) Diodes

Pin 19b	Output 8 (MSB)	)	Active low Output
		)	(Current Sinking)
Pin 18b	Output 7	)	
		)	<b>Note:</b> Contact Alzanti for 'sourcing' output
Pin 17b	Output 6	)	option
		)	
Pin 16b	Output 5	)	
		)	
Pin 15b	Output 4	)	
		)	
Pin 14b	Output 3	)	
		)	
Pin 13b	Output 2	)	
		)	
Pin 12b	Output 1 (LSB)	)	

## Getting Started

The PVP152 interfaces to the outside world using a three wire RS232 interface running at 9600 baud, 8 Data bits, 1 stop bit with no parity. A simple terminal emulator (TERM.EXE) for use with the controller is provided on the demonstration disk.

The operational mode of the controller is defined at power up by the settings on the 8 DIP switches as follows:

8	7	6	5	4	3	2	1
On = Computer mode  Off = Terminal mode	On = Execute stored program at power up	Reserved	Reserved	Address 4	Address 3	Address 2	Address 1

When the unit is shipped from the factory all DIP switches are off allowing the unit to be directly used from an RS232 terminal.

## Using Terminal Mode

### Starting Terminal Mode

To use terminal mode, connect a suitable terminal to the controller and set the DIP switches as shown below.

#### **DIP Switch Settings**

8 Off	7 Off	6 Off	5 Off	4 Off	3 Off	2 Off	1 Off
----------	----------	----------	----------	----------	----------	----------	----------

When the controller is powered up, a message similar to the one shown below will be displayed:

```
PVP152-02-03 Alzanti Ltd © 1998  
  
Type 'H' for help  
  
>
```

Each command has a one or two letter abbreviation which is used to uniquely identify it. For example **MA** is the abbreviation for the **M**ove **A**bsolute command. During command entry, the controller will echo characters as you type them (full duplex). Abbreviations will automatically be expanded and any parameters validated.

To send commands to the controller, type the abbreviation of the command and press carriage return (<CR>). If you use a command that requires parameters then press the space bar after you have entered the command and enter any parameters required separated by commas or spaces. When all parameters have been entered, press <CR>.

If you make a mistake during command entry then press BACKSPACE to abort the command and start again.

A list of all command abbreviations can be obtained by typing:

**H<CR>**

See the command reference section of this manual for detailed information on each command.

## **Creating a stored program**

The controller will normally execute each command as soon as carriage return is pressed. It is however possible to store up to 99 commands in the controller's memory. These commands will be executed sequentially after power up (if switch 7 is on) or when the 'Go' command is issued or when the external 'run program' input is activated.

The command language supports loop commands and allows branching based on the status of the general purpose inputs. This allows sophisticated internal programs to be developed using relatively small programs.

Commands are entered in to the stored program by typing a line number (1 to 99) followed by a space and then the command abbreviation with any parameters required. The parameters of the command will be checked but the command will not be executed until the stored program is run.

Program steps can be overwritten by entering a new command for the required line number. If just a line number is entered and then <CR> is pressed the command at the specified position will be erased.

Several commands are specifically used to help create a stored program:

### ***ERASE***

This command erases the entire stored program memory ready for a new program.

**Note:** This command always has to be typed in full to help prevent inadvertent program erasure.

### ***L(ist) startline,endline***

This command lists a section of the internal program.

### ***+ (Insert) lineno***

This command inserts an empty program step at the specified line number.

### ***- (Delete) lineno***

This command deletes the program step specified by the line number and then moves all program steps up a line to remove the gap.

### ***G(o)***

This command executes the stored program from step 1. During execution each command will be displayed on the terminal as it is run. The stored program will stop after line 99 has been executed or a program step containing the stop command is found.

Program execution may also be aborted by sending a **S(top)** command to the controller.

**Note:** *The information shown in brackets above does not need to be entered.*

Computer Mode

**Starting Computer Mode**

To set up the controller for running computer mode set the DIP switches as shown below:

**DIP Switch Settings**

<b>Prefix</b>	<b>SW 8</b>	<b>SW 7</b>	<b>SW 6</b>	<b>SW 5</b>	<b>SW 4</b>	<b>SW 3</b>	<b>SW 2</b>	<b>SW 1</b>
<b>None</b>	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
<b>A</b>	ON	OFF	OFF	OFF	OFF	OFF	OFF	ON
<b>B</b>	ON	OFF	OFF	OFF	OFF	OFF	ON	OFF
<b>C</b>	ON	OFF	OFF	OFF	OFF	OFF	ON	ON
<b>D</b>	ON	OFF	OFF	OFF	OFF	ON	OFF	OFF
<b>E</b>	ON	OFF	OFF	OFF	OFF	ON	OFF	ON
<b>F</b>	ON	OFF	OFF	OFF	OFF	ON	ON	OFF
<b>G</b>	ON	OFF	OFF	OFF	OFF	ON	ON	ON
<b>H</b>	ON	OFF	OFF	OFF	ON	OFF	OFF	OFF
<b>I</b>	ON	OFF	OFF	OFF	ON	OFF	OFF	ON
<b>J</b>	ON	OFF	OFF	OFF	ON	OFF	ON	OFF
<b>K</b>	ON	OFF	OFF	OFF	ON	OFF	ON	ON
<b>L</b>	ON	OFF	OFF	OFF	ON	ON	OFF	OFF
<b>M</b>	ON	OFF	OFF	OFF	ON	ON	OFF	ON
<b>N</b>	ON	OFF	OFF	OFF	ON	ON	ON	OFF
<b>O</b>	ON	OFF	OFF	OFF	ON	ON	ON	ON

## **Using Computer Mode**

In computer mode, up to 15 controllers can be used from one RS232 port. The controller will work as a command response system.

Commands sent to the controller should consist of a command abbreviation followed by a space and any parameters. The command should be terminated by carriage return followed by an optional line feed character.

If the command is valid the controller will execute it and return the message:

**OK<CR><LF>**.

If the command is unrecognised or cannot be executed the controller will return.

**ERROR<CR><LF>**.

If the DIP switches have been set up so that an address prefix is required the controller will only execute commands that are prefixed by the relevant address code.

For example:

**MA 100<CR><LF>** Tell a controller set up to have no address prefix to move to absolute position 100.

**NMR -100<CR><LF>** Tell a controller set up to have a prefix of **N** to move 100 pulses backwards from its current position.

**AG<CR><LF>** Tell a controller set up to have a prefix of **A** to run its internal program. Note: When an internal program is started from computer mode it will not display commands as they are being executed. This means that programs run much quicker than when they are run in terminal mode.

**Note:** A controller without an address prefix should not be used on the same serial port as a controller with an address prefix.

## Command Reference

### STOP

Syntax: S <CR>

#### Terminal & Computer Mode

All motion is immediately stopped. If a program is running then it is stopped and the controller will return to terminal or computer mode depending on the setting of DIP switch 8.

#### As part of a program

This command should be used as the last line in any program; it stops all motion in progress and returns the controller to terminal or computer mode.

#### Example Use

Suppose the controller was being used by a computer to perform simple moves, a short internal program could be created to set up the required operational speeds so that the computer would only need to send move commands.

```
1 BASE 784  
2 HIGH 10000  
3 ACCEL 1000  
4 HOME FWD  
5 STOP
```

The computer could now use the controller by issuing move relative and move absolute commands without the need to define any speeds.

### STATUS

Syntax: ?<CR>

#### Terminal Mode

This command allows the current status of the controller to be examined. The controller will respond with one of the following status messages:

**BUSY<CR><LF>** The controller is executing the last command or the internal program is running.

**ULIMIT<CR><LF>** The upper limit is active.

**LLIMIT<CR><LF>** The lower limit is active.

**ESTOP<CR><LF>** Emergency stop is active.  
**JOGGING<CR><LF>** The jog inputs are being used to move the motor.  
**IDLE<CR><LF>** The controller is ready for another command.

### Computer Mode

This command allows the current status of the controller to be examined. The controller will respond with one of the following status messages:

**BUSY<CR><LF>**  
**OK<CR><LF>** The controller is executing the last command or the internal program is running.

**ULIMIT<CR><LF>**  
**OK<CR><LF>** The upper limit is active.

**LLIMIT<CR><LF>**  
**OK<CR><LF>** The lower limit is active.

**ESTOP<CR><LF>**  
**OK<CR><LF>** Emergency stop is active.

**JOGGING<CR><LF>**  
**OK<CR><LF>** The jog inputs are being used to move the motor.

**IDLE<CR><LF>**  
**OK<CR><LF>** The controller is ready for another command.

### As part of a program

This command cannot be used as part of a stored program.

### Example Use

A computer interfacing to the controller should use this command to check that the controller has finished the last movement before issuing a new command.

### ?INPUTS

Syntax: ?I<CR>

### Terminal Mode

This command allows the eight general purpose inputs to be read. A typical response would be as follows:

*01011010<CR><LF>*

^ = MSB (Input 8)

The reply represents the binary value read from the general purpose inputs. A one means that the corresponding input is on (active low / 0 Volts) and a zero means that the input is off (active high / open circuit).

### Computer Mode

This command allows the eight general purpose inputs to be read. A typical response would be as follows:

*01011010<CR><LF>*  
*OK<CR><LF>*

The reply represents the binary value read from the general purpose inputs. A one means that the corresponding input is on (active low / 0 Volts) and a zero means that the input is off (active high / open circuit).

### As part of a program

This command is not allowed as part of an internal program.

### Example Use

The controller can execute this command even when the status indicates that it is busy performing a move or running an internal program. A computer interfacing to the controller could use this command to stop a move in progress when a particular condition on the input port is met.

### ?MODE

Syntax:        ?M<CR>

### Terminal Mode

This command allows the current mode of operation to be examined. Typical responses are shown below:

<i>Terminal&lt;CR&gt;&lt;LF&gt;</i>	The controller is running in terminal mode.
<i>Running&lt;CR&gt;&lt;LF&gt;</i>	The internal program is running.

### Computer Mode

This command allows the current mode of operation to be examined. Typical responses are shown below:

*Computer<CR><LF>*

**OK<CR><LF>**                      The controller is running in computer mode.

**Running<CR><LF>**

**OK<CR><LF>**                      The internal program is running.

#### Example Use

This command would normally be used in computer mode to determine if the internal program is running.

#### ?POS

Syntax:        ?P<CR>

#### Terminal Mode

This command allows the current position of the controller to be read. A typical response would be:

**-10234<CR><LF>**

#### Computer Mode

This command allows the current position of the controller to be read. A typical response would be:

**-10234<CR><LF>**

**OK<CR><LF>**

#### As part of a stored program

This command is not allowed as part of a stored program.

#### Example Uses

The motion control logic within the controller operates on a 24 bit system. This allows for a longest individual move of +/- 16,777,216 pulses. The software however keeps track of position using a 32 bit counter and hence the position value returned will be in the range of +/- 2,147,483,647.

**?SETUP**

Syntax:        ?S<CR>

Terminal Mode

This command allows various parameters within the controller to be examined. A typical response would be:

```
BASE            = 100<CR><LF>
HIGH            = 200<CR><LF>
ACCEL           = 123<CR><LF>
BACKLASH        = 350<CR><LF>
RANGE           = 4<CR><LF>
STEPSIZE = FULL<CR><LF>
JOGGING        = ENABLED<CR><LF>
```

Computer Mode

This command allows various parameters within the controller to be examined. A typical response would be:

```
BASE            = 100<CR><LF>
HIGH            = 200<CR><LF>
ACCEL           = 123<CR><LF>
BACKLASH        = 350<CR><LF>
RANGE           = 4<CR><LF>
STEPSIZE = FULL<CR><LF>
JOGGING        = ENABLED<CR><LF>
OK<CR><LF>
```

As part of a program

This command is not allowed as part of a program.

Example Use

This command allows you to verify that the controller has correctly interpreted the speed settings that you have sent to it.

## **INSERT**

Syntax:        + lineno <CR>

### Terminal Mode

This command allows a blank line to be inserted at the specified point in the stored program. This command may take several seconds to execute since it is necessary to erase and reprogram all EEPROM locations.

### Computer Mode

This command is not allowed.

### As part of a program

This command is not allowed.

### Example Use

Use this command to make space in programs for extra commands that are needed. After using this command, check all jump and loop commands to ensure that they are pointing to the correct line numbers.

## **DELETE**

Syntax:        - lineno <CR>

### Terminal Mode

This command deletes the specified line number in the stored program and moves up all of the program steps to close the gap. This command may take several seconds to execute since it is necessary to erase and reprogram all EEPROM locations.

### Computer Mode

This command is not allowed.

### As part of a stored program

This command is not allowed.

### Example Use

Use this command to remove unwanted lines in the program to free up more space at the end of the program storage area.

## **ACCEL**

Syntax: A acceleration\_distance <CR>

### All modes

This command sets the distance over which the controller will accelerate and decelerate. Values in the range 4 to 11,220 are allowed. This command should only be used when the controller is idle. ( See section 6.2).

### Example Use

Assuming we are controlling a mechanical arm that can pick up heavy items by turning on output 1, then a typical program to move items would be as follows:

1	BASE 30	
2	HIGH 2000	
3	<b>ACCEL 10</b>	<i>Use quick acceleration to find home position</i>
4	HOME FWD	
5	<b>ACCEL 11220</b>	<i>Use slow acceleration when we are moving items</i>
6	IF INPUTS ???????0,6	<i>Wait for a product sensor on input 1 to activate</i>
7	OUTPUT ???????1	<i>Pick up item and move it</i>
8	MOVE REL 1000	
9	OUTPUT ???????0	
10	<b>ACCEL 10</b>	<i>Go back to start point quickly</i>
11	MOVE REL -1000	
12	JUMP TO 5	<i>Get ready for next item</i>
13	STOP	

## **BASE**

Syntax: B base\_speed <CR>

### All Modes

This command defines the speed in Hz at which the motor can start without stalling. All moves will start at base speed accelerate to high speed and then decelerate to base speed at the end of the move. This command should only be used when the controller is idle.

The controller uses a divider to generate pulses and as such all speed settings are only approximate. For detailed information on suitable ranges of speed please refer to the speed control section of this manual.

### Example Use

Assuming the controller is indexing a conveyor belt with varying load and that the general purpose inputs are connected to an eight bit load cell then a typical program would be:

1	ACCEL 1000	
2	HIGH 1000	
3	IF INPUTS 00000000,15	<i>Jump if we have zero load</i>
4	IF INPUTS 000000??,17	<i>Jump if load lower than 4</i>
5	IF INPUTS 00000???,19	<i>Jump if load lower than 8</i>
6	IF INPUTS 000?????,21	<i>Jump if load lower than 32</i>
7	IF INPUTS 0????????,23	<i>Jump if load lower than 128</i>
8	<b>BASE 1</b>	<i>Heavy load use slowest base speed</i>
9	MOVE REL 100	
10	DELAY 1000	
11	JUMP TO 3	
15	<b>BASE 500</b>	
16	JUMP TO 9	
17	<b>BASE 400</b>	
18	JUMP TO 9	
19	<b>BASE 200</b>	
20	JUMP TO 9	
21	<b>BASE 100</b>	
22	JUMP TO 9	
23	<b>BASE 20</b>	
24	JUMP TO 9	

### **BACKLASH**

Syntax: BK correction\_pulses<CR>

#### All Modes

The single axis controller supports automatic backlash compensation which guarantees that the controller goes to each position from the same direction. If a positive number is specified then all moves in the positive direction will overshoot by the specified number of pulses and then the controller will reverse back to the required position. If a negative number is specified then the controller will overshoot in the negative direction in a similar manner. This command should only be issued when the controller is idle.

#### Example Use

**BK 100 <CR>** Set positive backlash compensation

**MR 10 <CR>** The controller will move forwards 110 pulses and then reverse 100 pulses to end up 10 pulses in front of its current position.

**MR -10<CR>** The controller will move backwards 10 pulses.

**DELAY**

Syntax: D time\_in\_millisecs <CR>

Terminal & Computer Mode

Not allowed.

As part of a program

This command delays for the number of milliseconds specified before allowing the next program step to execute. The controller operates to a resolution of 4 milliseconds. If a time that is not a multiple of 4 is entered the controller will wait to the nearest multiple of 4.

Example Use

Assuming that the general purpose inputs and outputs are connect to a hydrogen-oxygen burner as follows:

Output 1	Nitrogen Purge
Output 2	Hydrogen Pilot
Output 3	Oxygen Pilot
Output 4	Ignite Pilot
Output 5	Hydrogen Main
Output 6	Oxygen Main
Input 1	Start Burner Push button
Input 2	Stop Burner Push button
Input 3	Flame Detect

A typical program to control the burner would be as follows:

```
1   IF INPUTS ?????1??,3   Jump past next line if flame is ok
2   OUTPUT 00000000        Flame is out here so make sure gas flows are off
3   IF INPUTS ???????1,10  Jump if start burner is pressed
4   IF INPUTS ???????1?,30 Jump if stop burner is pressed
5   JUMP TO 1              Loop waiting for an input

10  OUTPUT 00000001        Turn on N2 purge
11  DELAY 5000           Wait for 5 seconds
12  OUTPUT 00000010        Turn on H2 pilot
13  OUTPUT 00001010        Try to light pilot
14  IF INPUTS ??????1??,20 Jump if we have a flame
15  DELAY 500            Wait half a sec
16  LOOP TO 13,10         Keep trying to light it for 10 tries
17  JUMP TO 1              Give up, pilot won't light
20  DELAY 2000           Wait before turning on O2 pilot
21  OUTPUT 00000110        Turn on oxygen pilot
22  DELAY 1000           Wait for one second
23  OUTPUT 00010110        Turn on H2 Main
```

```
24  DELAY 1000      Wait for one second
25  OUTPUT 00110110 Turn on O2 Main
26  JUMP TO 1

30  OUTPUT ??0????? Turn off O2 Main
31  DELAY 1000      Wait for one second
32  OUTPUT ??00???? Turn off H2 Main
33  DELAY 1000      Wait for one second
34  OUTPUT ??00??0? Turn off O2 pilot
35  DELAY 1000
36  OUTPUT 00000000 Turn off H2 pilot
37  JUMP TO 1
```

## **ERASE**

Syntax:       ERASE<CR>

### Terminal Mode & Computer Mode

This command erases the contents of the program memory area. The command may take several seconds to execute since a large number of EEPROM locations need to be erased and reprogrammed. This is the only command that has no abbreviation and must always be typed in full.

### As part of a program

This command is not allowed as part of a stored program.

### Example Use

Use this command to delete the entire program memory before you enter a new program.

**END MOVE**

Syntax: EM<CR>

Terminal Mode & Computer Mode

This command is not allowed.

As part of a program

This command stops a move started using the *START FWD* or *START REV* commands. The motor stops instantly even when it is running at high speed. To decelerate the motor to a stop use a *MOVE REL* command which will decelerate the motor to a stop before performing the relative move.

Example Use

1	BASE 250	
2	HIGH 250	<i>Set a fairly slow high speed</i>
3	ACCEL 1000	<i>Take 1000 pulses to accelerate</i>
4	START FWD	<i>Start moving forwards</i>
5	IF INPUTS ????????,5	<i>Loop until input 1 goes active</i>
6	<b>END MOVE</b>	<i>Stop the motor instantly</i>
7	HIGH 3000	<i>Increase the high speed</i>
8	START FWD	<i>Start moving forwards</i>
9	IF INPUTS ????????,9	<i>Loop until input 1 goes inactive</i>
10	MOVE REL -1000	<i>Decelerate to a stop and move backwards by the deceleration distance</i>
11	STOP	

## FULLSTEP & HALFSTEP

Syntax:      FS <CR>  
              HS <CR>

### All modes

These command switches the controller between full and half step modes of operation. The commands should only be used when the controller is idle.

### Example Use

Move the motor forwards in full step and backwards in halfstep.

```
1      BASE 100
2      HIGH 200
3      ACCEL 1000
4      FULLSTEP
5      MOVE REL 100
6      HALFSTEP
7      MOVE REL -200
8      DELAY 1000
9      LOOP TO 4,9
10     STOP
```

## GO

Syntax:      G<CR>

### Terminal & Computer Mode

This command begins execution of the stored program. During program execution the controller will only respond to commands starting with ? or the stop command. The stored program will automatically stop if any of the following events occur:

- a) All program steps have been executed.
- b) A program step containing the stop command is encountered.
- c) A stop command is sent over the RS232 link.
- d) Emergency stop is activated.
- e) Active limit switch (except during a HOME command)
- f) An invalid command is found in the program.

After the program stops, the controller will return to mode defined by DIP switch 8.

Stored programs can be set up to run automatically by setting DIP switch 7 to the on position.

### As part of a stored program

This command is not allowed as part of a stored program.

Example Use

Suppose the controller was required to perform a sequence of moves at a certain time of the day. A program to perform the required sequence could be set up in the controller and a computer used to send the go command to the controller at the required time.

```
1    BASE 100
2    ACCEL 1000
3    HIGH 200
4    MOVE REL 100
5    OUTPUT 00000001
6    MOVE REL 200
7    OUTPUT 00000000
8    MOVE REL -300
9    DELAY 1000
10   LOOP TO 9,59
11   LOOP TO 9,59
12   LOOP TO 9,24
13   JUMP TO 4
```

With the above example program, the computer should send the STOP command to the controller to stop the current program and then issue the GO command to start it. If the computer does not start the controller at the same time the next day, then the controller will do the move itself after an extra hour has passed.

**GOSUB**

Syntax: GS<CR>

Terminal Mode & Computer Mode

This command is not allowed.

As part of a stored program

This command allows a common section of the stored program to be called and executed. After the section of stored program has been executed control is returned to the next program step after the GOSUB command. GOSUBS can be nested to a maximum level of 9.

Example Use

Assuming the general purpose inputs and outputs are connected to a drill as follows:

Output 1	Move Drill down
Output 2	Move Drill Up
Input 1	Drill is down
Input 2	Drill is up

A typical program to perform a drilling sequence would be as follows:

1	<b>GOSUB 30</b>	<i>Make sure drill is up</i>
2	HOME FWD	<i>Establish home position</i>
3	<b>GOSUB 10</b>	<i>Drill a hole</i>
4	MOVE REL 100	<i>Move to next position</i>
5	LOOP TO 3,9	<i>Drill another 9 holes</i>
6	STOP	
10	<b>GOSUB 20</b>	<i>Move drill down</i>
11	DELAY 5000	<i>Wait for 5 seconds</i>
12	<b>GOSUB 30</b>	<i>Move the drill up</i>
13	RETURN	
20	OUTPUT 00000001	<i>Move drill down</i>
21	IF INPUTS ????????,21	<i>Wait for drill to reach position</i>
22	OUTPUT 00000000	<i>Turn off motion</i>
23	RETURN	
30	OUTPUT 00000010	<i>Move drill up</i>
31	IF INPUTS ????????,31	<i>Wait for drill to reach position</i>
32	OUTPUT 00000000	<i>Turn off motion</i>
33	RETURN	

## HELP

Syntax: H<CR>

### Terminal Mode

When this command is sent the controller will display some basic instructions on how to enter commands. A list of all the commands together with an indication of the number of parameters required for each command will be displayed.

### Computer Mode

This command is not allowed in computer mode.

### As part of a program

This command is not allowed as part of a program.

### Example Use

Use this command to get a list of all command abbreviations.

## **HIGH**

Syntax: HI high\_speed <CR>

### **All Modes**

This command defines the speed in Hz at which the motor will travel at. All moves will start at base speed, accelerate to this speed and then decelerate to base speed at the end of the move. This command should only be used when the controller is idle.

The controller uses a divider to generate pulses and as such all speed settings are only approximate. For detailed information on suitable ranges of speed please refer to the speed control section of this manual.

### **Example Use**

Move forwards quickly and backwards slowly.

```
1   BASE 100
2   ACCEL 400
3   HIGH 1000
4   MOVE REL 100
5   HIGH 10
6   MOVE REL -100
7   LOOP TO 3,5
8   STOP
```

## **HOME FORWARDS & HOME REVERSE**

Syntax:      HF<CR>  
              HR<CR>

### All Modes

When this command is executed, the controller will search for a datum switch in either the forward or reverse direction. If a limit switch is hit during the search the controller will automatically reverse direction and continue looking for the switch.

Once the datum switch has been found the controller will stop the motor and reverse off the switch at base speed in the opposite direction to the start direction. The position count will then be set to zero.

### NOTE:

The controller will always creep off the switch in the opposite direction to its start direction and hence the HOME FWD command will always find the reverse edge of the datum switch even if the controller hits a limit switch and approaches the switch from the opposite direction.

### Example Use

1	ACCEL 1000	
2	BASE 100	
3	HIGH 1000	
4	<b>HOME FWD</b>	<i>Find the datum switch looking in the forwards direction first</i>
5	MOVE ABS 100	<i>Move 100 pulses past the datum switch</i>
6	MOVE REL 10	<i>Do some relative moves</i>
7	LOOP TO 6,5	
8	MOVE ABS 0	<i>Go back to the datum switch</i>
9	STOP	

## **IF INPUTS**

Syntax:      I pattern,line\_number<CR>

### Terminal Mode

This command is not allowed in terminal mode.

### Computer Mode

This command is not allowed in computer mode.

### As part of a program

This command allows control to be transferred to a different program step if the general purpose inputs match the pattern specified. If the pattern is not matched then the next program step will be executed.

Each pattern represents the binary data on the general purpose input port. For each position one of the following characters should be entered:

- 1 = Input should be ON
- 0 = Input should be OFF
- ? = Don't care

Example Use

- 1 IF INPUTS 00000000,50 *Go to step 50 if all inputs are off*
- 2 IF INPUTS 11111111,51 *Go to step 51 if all inputs are on*
- 3 IF INPUTS 00000001,52 *Go to step 52 if all inputs are off apart from input 1*
- 4 IF INPUTS ????????1,53 *Go to step 53 if input 1 is on*
- 5 IF INPUTS 0???????1,54 *Go to step 54 if input 1 is on and input 8 is off*
- 6 IF INPUTS ????????55 *Go to step 55*
- 7 IF INPUTS 00000000,7 *Loop until an input goes active*
- 8 IF INPUTS 0???????8 *Loop until input 8 is active*
- 9 IF INPUTS ?1???????9 *Loop until input 7 is inactive*
- 10 IF INPUTS 1???????0,10 *Loop until input 8 is inactive unless input 1 is on*

**NOJOG & JOG**

Syntax: NJ<CR>  
J pulses\_count<CR>

All Modes

When a jog input is activated the motor will output single pulses until the jog button is released. If the jog input is held active for more pulses than the pulse count specified in the jog command then the motor will accelerate to high speed until the jog input is released. Values in the range 0 (default) to 200 may be specified for the pulse count.

When the NOJOG command is executed, the controller will ignore the external jog inputs. The JOG command can be used to enable the inputs again. If a move command is executed while the controller is jogging then jogging will automatically stop while the move is taking place.

Example Use

Assume the controller is drilling holes and an operator needs to set up the position of the first hole using the jog inputs. A typical program would be as follows:

- 1 BASE 100
- 2 HIGH 100
- 3 ACCEL 4
- 4 IF INPUTS 00000000,4 *Wait for operator to press an input to start the drilling sequence. While this command is looping the operator can use the jog buttons to move the motor*

5	<b>NOJOG</b>	<i>Disable jogging</i>
6	OUTPUT 00000001	<i>Turn on the drill for a while</i>
7	DELAY 5000	<i>NOTE: The operator cannot accidentally jog the motor when drilling is taking place</i>
8	OUTPUT 00000000	<i>Turn off the drill</i>
9	MOVE REL 100	
10	LOOP TO 6,4	<i>Drill another 4 holes</i>
11	MOVE REL -500	<i>Go back to the start point</i>
12	<b>JOG 0</b>	<i>Allow jogging again</i>
13	JUMP TO 4	

### **JUMP TO**

Syntax: JU<CR>

#### Terminal Mode

This command is not allowed in terminal mode.

#### Computer Mode

This command is not allowed in computer mode.

#### As part of a stored program

This command transfers control to a different section of the stored program.

#### Example Use

```
1 OUTPUT 00000001
2 OUTPUT 00000010
3 OUTPUT 00000100
4 OUTPUT 00001000
5 JUMP TO 1
```

## **LIST**

Syntax: L start\_line,end\_line<CR>

### Terminal Mode

This command displays the specified section of the internal program.

### Computer Mode

This command is not allowed in computer mode.

### As part of a program

This command is not allowed as part of a stored program.

### Example Use

Use this command to view the internal program.

## **LOOP TO**

Syntax: LP line\_no,repeat\_count<CR>

### Terminal Mode

This command is not allowed in terminal mode.

### Computer Mode

This command is not allowed in computer mode.

### As part of a stored program

This command allows a section of the stored program to be repeated for the specified number of times.

### Example

1	BASE 10	
2	HIGH 100	
3	ACCEL 100	
4	MOVE REL 100	
5	DELAY 1000	<i>Wait for 1 second</i>
6	<b>LOOP TO 5,59</b>	<i>Wait for another 59 seconds</i>
7	OUTPUT 00000001	<i>Output a pulse every minute</i>
8	OUTPUT 00000000	
9	<b>LOOP TO 5,59</b>	<i>Do another 59 minute loops</i>
10	JUMP TO 4	<i>Do it all again</i>

## **MOVE ABSOLUTE**

Syntax:      MA absolute\_position<CR>

### All Modes

This command moves the motor to an absolute position referenced from the position at start up or from the datum switch if a datum sequence has been run. The controller must be within 16,777,216 pulses of the target position for this command to execute.

The controller should be idle before this command is issued.

### Example Use

```
1     BASE 100
2     HIGH 200
3     ACCEL 4
4     POS=0                     Define where we are now as zero
5     MOVE REL 100             Do some relative moves
6     LOOP TO 5,10
7     IF INPUTS 00000000,9
8     MOVE REL 10
9     MOVE ABS 0             Move back to the start position regardless of which
                               path we took to get to this program step
10    STOP
```

## **MOVE RELATIVE**

Syntax:      MR relative\_distance<CR>

### All Modes

This command moves the motor a relative distance from the current position. The distance must be within the range +/- 16,777,216 pulses.

The controller should be idle before this command is issued.

### Example Use

```
1     BASE 100
2     HIGH 200
3     ACCEL 4
4     HOME FWD
5     MOVE REL 16777216         Move a long way
6     MOVE REL 16777216
7     MOVE REL -16777216        Move back to start position.
8     MOVE REL -16777216
9     STOP
```

## **OUTPUT**

Syntax:        O pattern <CR>

### **All Modes**

This command sends the eight bit binary pattern to the general purpose outputs. At each bit position the following characters can be used.

1 = Turn output on  
0 = Turn output off  
? = Leave output in its previous state

### **Example Uses**

1	OUTPUT 00000000	<i>Turn all outputs off</i>
2	OUTPUT 11111111	<i>Turn all outputs on</i>
3	OUTPUT ???????0	<i>Turn output 1 off, leave others unchanged</i>
4	OUTPUT 1???????	<i>Turn output 8 on, leave others unchanged</i>

## **POS=**

Syntax:        P position <CR>

### **All Modes**

This command overrides the current position with a new one.

### **Example Use**

1	BASE 10	
2	HIGH 100	
3	ACCEL 1024	
4	HOME REV	
5	<b>POS=100</b>	<i>Define the current position as 100</i>
8	MOVE ABS 0	<i>Move to point zero</i>
9	IF INPUTS 00000000,9	<i>Allow the operator to fine tune zero point by using the jog switches</i>
10	<b>POS=0</b>	<i>Call this position zero</i>
11	STOP	

## **RANGE**

Syntax: R speed\_range

### All Modes

This command sets the pre-divider for the speed control system. Please refer to the speed control section of this manual for detailed information.

### Example Use

1	<b>RANGE 483</b>	<i>Select a slow speed range</i>
2	BASE 6	
3	HIGH 300	
4	ACCEL 10	
5	MOVE REL 100	
6	<b>RANGE 4</b>	<i>Select a quick speed range</i>
7	BASE 800	
8	HIGH 10000	
9	MOVE REL -100	
10	STOP	

## **READ**

Syntax: RD line\_no < CR>

### Terminal & Computer Modes

This command reads the raw EEPROM data for the specified line number. A typical response is of the form:

*1,0,100,0,0<CR>*

The first byte is a command identifier and the other bytes are parameters for the commands. Refer to the write command for further details.

### As part of a stored program

This command is not allowed as part of a stored program.

### Example Use

The utility program *LOADER.EXE* uses this command together with the write command to upload programs from one controller and download them into another one.

**RETURN**

Syntax: RT<CR>

Terminal Mode & Computer Mode

This command is not allowed.

As part of a stored program

This command is used to terminate a section of stored program called with a *GOSUB* command.

Example Use

```
1   GOSUB 10           Call stored program step 10
2   STOP
10  OUTPUT 00000000
11  OUTPUT 11111111
12  OUTPUT 11111111
13  RETURN           Go back to program step 2
```

**START FWD, START REV**

Syntax: SF<CR>  
SR<CR>

Terminal Mode & Computer Mode

This command is not allowed.

As part of a stored program

This command starts the motor moving at high speed in the relevant direction. The motor can be stopped by using the *END MOVE* command or by issuing a *MOVE ABS* or *MOVE REL* command. The motor will also stop after 16,777,216 pulses have been output.

Example Use

```
1   START FWD           Start a move
2   OUTPUT 00001111
3   OUTPUT 11110000
4   IF INPUTS ????????,2   Loop until input 1 goes active
5   END MOVE             Stop the motor
6   STOP
```

**WRITE**

Syntax: W line\_no,cmd\_id,data1,data2,data3,data4

**Terminal & Computer Modes**

This command allows data to be written in to the EEPROM for the specified line number. Each program step consists of 5 bytes of data. The first byte is a command ID and the use of the following four bytes depends on the command. This command should be used with great care since it bypasses the internal range checking mechanism.

The following table shows the use of the data parameters for each command.

**NOTE: Alzanti reserve the right to change command ID's in future versions of the controller software without prior notice.**

<b>Name</b>	<b>Cmd ID</b>	<b>Data 1</b>	<b>Data 2</b>	<b>Data 3</b>	<b>Data 4</b>
Stop	0				
Status	1				
?Inputs	2				
?Mode	3				
?Pos	4				
?Setup	5				
Insert	6				
Delete	7				
Accel	8	MSB Accel	LSB Accel		
Base	9	MSB Base	LSB Base		
Backlash	10	MSB Lash	LSB Lash		
Delay	11	MSB Delay	LSB Delay		
Erase	12				
End Move	13				
Fullstep	14				
Go	15				
Gosub	16		Line Number		

<b>Name</b>	<b>Cmd ID</b>	<b>Data 1</b>	<b>Data 2</b>	<b>Data 3</b>	<b>Data 4</b>
Help	17				
High	18	MSB Speed	LSB Speed		
Home Fwd	19				
Home Rev	20				
Halfstep	21				
IF Inputs	22	Mask for input port	Comparator for input port		Line Number
Jog	23		Jog Pulses		
Jump To	24		Line No		
List	25				
Loop To	26		Line No		Count
Move Abs	27		24 Bit distance MSB to LSB		
Move Rel	28		24 Bit distance MSB to LSB		
No Jog	29				
Output	30	Mask	Value		
Pos =	31	32 bit position MSB to LSB			
Range	32	MSB range	LSB Range		
Read	33		Line No		
Return	34				
Start Forwards	35				
Start Reverse	36				
Write	37				

As part of a program

This command cannot be used as part of a stored program.

Example Use

The utility program *LOADER.EXE* uses this command to download programs to the controller. It is however possible to make other uses of this command providing care is taken. Suppose the controller was being used to cut labels of various lengths. A typical internal program to cut labels would be:

```
1    BASE 100
2    HIGH 1000
3    ACCEL 1000
4    MOVE REL 100
5    OUTPUT 00000001      Assume output 1 is a connected to a cutter
6    DELAY 100
7    OUTPUT 00000000
8    JUMP TO 4
```

A computer interfacing to the controller could adjust the program to deal with different length labels before issuing the go command by sending:

*W 4,28,<MSB LENGTH>,<LSB LENGTH>,0,0<CR>*

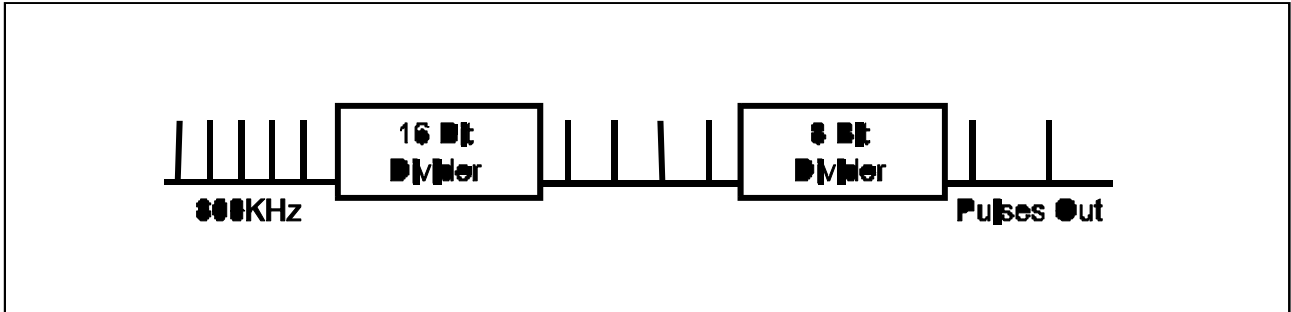
Another use of this command would be to modify a jump command at the start of a program to choose which part of an internal program to run:

```
1    JUMP TO 10
10   Task 1 steps
20   Task 2 steps
30   Task 3 steps
```

*W 1,24,0,<Line No Of Task>,0,0<CR>*

## Speed Control

The single axis controller generates pulses at a particular frequency by using two dividers as shown below:



The value of the 16 bit divider (referred to as range in this document) is set by the *RANGE* command which allows values between 4 and 65535 to be specified. This allows frequencies in the range 12Hz to 200KHz to be passed on to the 8 bit divider.

The value for the 8 bit divider is automatically calculated by the single axis controller to generate a frequency as close to demand as possible. For example, suppose the value for the 16 bit divider was 250 and you asked for a base speed of 90Hz, the controller would calculate the closest divider as follows:

$$\text{Clock} = 800,000/250 = 3200 \text{ Hz}$$

$$\text{Divider} = \text{Clock} / 90 = 36$$

$$\text{Actual Frequency} = \text{Clock}/\text{Divider} = 88.89\text{Hz}$$

The minimum frequency for a particular range is obtained by using a divider of 255. For example, suppose that you were using a motor with a base speed of around 100Hz then you should calculate the appropriate speed range as follows:

$$\text{Range} = 800,000 / (\text{Base Speed} * 255) = 31$$

$$\text{Minimum Frequency} = 800,000 / (31 * 255) = 101\text{Hz}$$

The maximum frequency for a particular range is obtained by using a divider of 2. For example if the range was 31 then the maximum speed that can be generated is:

$$800,000 / (31 * 2) = 12903\text{Hz}$$

Obviously during motor acceleration the controller would at some stage need to switch from a divider of 3 to a divider of 2 to reach the top speed. With the above settings, this would cause a step change of 4300Hz and almost certainly stall the motor. Practical experiments indicate that divider values less than 20 are impractical and the maximum frequency should be calculated as follows:

$$\text{Maximum Frequency} = 800,000 / (\text{range} * 20)$$

## **PVP152 Single Axis Controller - User Manual V2.23**

The following table summarises some typical speed ranges:

<b>Range</b>	<b>Clock</b>	<b>Min Freq.</b>	<b>Max. Freq.</b>	<b>Max Step Change</b>
4	200000.00	784.31	10000.00	476.19
5	160000.00	627.45	8000.00	380.95
6	133333.33	522.88	6666.67	317.46
7	114285.71	448.18	5714.29	272.11
8	100000.00	392.16	5000.00	238.10
9	88888.89	348.58	4444.44	211.64
10	80000.00	313.73	4000.00	190.48
11	72727.27	285.20	3636.36	173.16
12	66666.67	261.44	3333.33	158.73
13	61538.46	241.33	3076.92	146.52
14	57142.86	224.09	2857.14	136.05
15	53333.33	209.15	2666.67	126.98
16	50000.00	196.08	2500.00	119.05
17	47058.82	184.54	2352.94	112.04
18	44444.44	174.29	2222.22	105.82
19	42105.26	165.12	2105.26	100.25
20	40000.00	156.86	2000.00	95.24
21	38095.24	149.39	1904.76	90.70
22	36363.64	142.60	1818.18	86.58
23	34782.61	136.40	1739.13	82.82
24	33333.33	130.72	1666.67	79.37
25	32000.00	125.49	1600.00	76.19
26	30769.23	120.66	1538.46	73.26

<b>Range</b>	<b>Clock</b>	<b>Min Freq.</b>	<b>Max Freq.</b>	<b>Max Step Change</b>
27	29629.63	116.19	1481.48	70.55
28	28571.43	112.04	1428.57	68.03
29	27586.21	108.18	1379.31	65.68
30	26666.67	104.58	1333.33	63.49
35	22857.14	89.64	1142.86	54.42
40	20000.00	78.43	1000.00	47.62
45	17777.78	69.72	888.89	42.33
50	16000.00	62.75	800.00	38.10
100	8000.00	31.37	400.00	19.05
150	5333.33	20.92	266.67	12.70
200	4000.00	15.69	200.00	9.52
250	3200.00	12.55	160.00	7.62
300	2666.67	10.46	133.33	6.35
3000	266.67	1.05	13.33	0.63

The utility program *RANGE.EXE* can be used to help find the best setting of the range command for the type of motor you are using.

Utility Programs (ONLY FOR USE WITH MS DOS)

The following utility programs are provided on the demonstration disk:

**TERM.EXE**

This program is a simple terminal emulator for use with the single axis controller.

**LOADER.EXE**

This program allows stored programs to be read from a single axis controller and stored to disk. The program then provides the capability to download the disk file to other controllers so that batches of controllers with the same stored program are easily produced.

**RANGE.EXE**

This program helps determine the best speed range for the motor you are using.

**COMPU.EXE**

Example of how to use the computer mode interface of the controller.