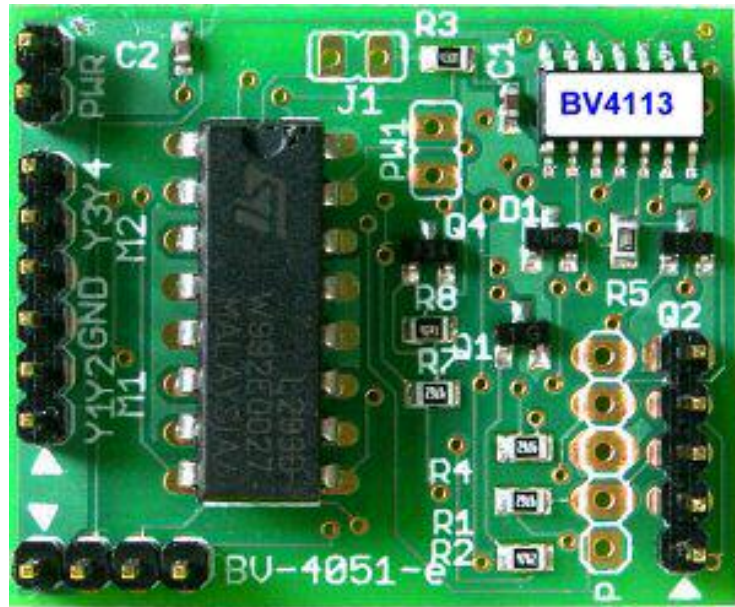


DC / Stepper Motor Controller

BV4113



BV4113 DC / Stepper Motor Controller

Product specification.

August 2006

DC / Stepper Motor Controller**BV4113****Contents**

1.	Document Versions.....	4
2.	Introduction.....	4
3.	Features.....	4
4.	Device Output.....	4
5.	DC and Stepper.....	4
6.	DC Motor Controller	4
6.1.	DC Motor Configurations	5
7.	Stepper Motor Controller	5
7.1.	Stepper Motor Configurations	5
8.	Digital and Analogue Channel.....	6
9.	Power Saving.....	6
10.	All Commands.....	6
10.1.	DA.....	7
10.2.	DB – as DA.....	7
10.3.	PA	7
11.	Digital I/O & Analogue	7
11.1.	DA.....	7
11.2.	DB see DA.....	7
11.3.	VA.....	7
11.4.	PA	8
12.	DC Motor Control.....	8
12.1.	EA.....	8
12.2.	EB.....	8
12.3.	YA.....	8
12.4.	YB – as YA.....	8
12.5.	YC.....	8
12.6.	YD – as YC	9
12.7.	MA	9
12.8.	MB as MA.....	9
12.9.	WA	9
12.10.	WB.....	9
13.	Stepper Motor Control.....	10
13.1.	SC.....	10
13.2.	SN.....	10
13.3.	SS.....	10
13.4.	SP.....	10
13.5.	SX.....	10
13.6.	SD.....	11
13.7.	SR.....	11
13.8.	SM	11
14.	Error codes Specific to this interface	11

DC / Stepper Motor Controller**BV4113**

15.	Revisions to the IASI Section	12
16.	Introduction to IASI.....	12
17.	IASI Electrical Interface.....	12
18.	Serial Connections	12
19.	Factory Configuration.....	13
20.	Non-Inverted Mode.....	13
21.	Commands	13
21.1.	Addressing	14
21.2.	Command Format.....	14
21.3.	Command Parameters.....	14
21.4.	Command Line (Buffer).....	14
22.	Command Set	14
22.1.	The Star *	14
23.	Common Command Set (Z)	15
23.1.	Summary	15
23.2.	ZA.....	15
23.3.	ZB.....	15
23.4.	ZC.....	16
23.5.	ZD.....	17
23.6.	ZF	17
23.7.	ZK.....	17
23.8.	ZL	17
23.9.	ZM	18
23.10.	ZR.....	18
23.11.	ZT.....	18
23.12.	ZV.....	18
23.13.	ZW.....	18
24.	Z Command Error Codes	19
25.	Connecting and Configuration	19
25.1.	Reconfiguring	19
26.	Microcontroller Use	20
26.1.	Simple	20
26.2.	With feedback.....	20
26.3.	Baud rate	20
26.4.	Multiple Devices	20
27.	Restoring Factory Defaults.....	21
27.1.	Software	21
27.2.	Hardware	21
28.	Watchdog.....	22
29.	Troubleshooting	22

DC / Stepper Motor Controller

BV4113

1. Document Versions

- 1.0 December 2006
- 2.0 May 2008 – Note added about ADC pull up resistor.

2. Introduction

The DC motor controller is a dual purpose device capable of controlling both DC motors and Stepper motors.

In addition to this there are two general propose digital input / output lines and 10 bit one analogue input.

As this is an IASI device all control is affected by simple two character text commands. Software is built in for Pulse Width Modulation (PWM) when controlling DC motors and also step rates and ramp up etc. when controlling stepper motors.

The BV4113 uses the L293D bridge driver which is capable of controlling 2 DC motors in forward and reverse or 4 DC motors in one direction.

When used for a stepper motor the BV4113 can control either a bi-polar (4 wire) or a uni-polar (5 & 6 wire) type stepper motor.

3. Features

- Easy to use asynchronous serial interface requiring only 4 connections.
- Command set based on text
- Only 2 data lines required, transmit and receive. The device will work with transmit only.
- Many devices can use the same two data lines
- Each device has it's own user configurable address
- No specialist hardware, can work from a PC Com port
- Automatic Baud rate detection up to 38.4K, non-standard baud rates can be detected making interfacing with microcontrollers easier
- Simple software requirement for interfacing with a microcontroller.
- Common protocol used throughout range, devices can be mixed on same data bus
- Works with RS232 standard voltages and +5V, no level translator needed for receiving data
- Drive current 600mA, up to 36V (total power 4W)
- Up to 4 DC motors or one stepper motor either bi-polar or uni-polar

- PWM control for each DC motor when used in differential mode (forward and reverse)
- Differential mode, forward and reverse
- Half step or full step patterns, user definable full step pattern
- Continuous stepping or pre-defined number of steps
- Step read capability to indicate number of steps to go.
- Step speed controllable from 128us delay between steps to 32ms in 256 increments.
- Ramp up available for slow start.
- Two general purpose digital input / output lines
- One 10 bit analogue input channel
- Low power saver mode.

4. Device Output

The BV4113 is basically a device that controls the L293D integrated circuit that is a 4 channel push-pull driver with built in diode protection.

It is capable of 1.2A per channel peak and 600mA continuous with a suitable heat sink.

The PCB provides some degree of heat dissipation but for higher power work a heat sink should be attached to the back of the IC, this is not part of the supplied product.

There are two power supplies to the board. The logic supply that should be within 4.5V to 6V, this is supplied through pins 3 of the IASI connector and a motor supply. The motor supply is on a separate 2 pin connector, this can be any voltage up to 35V to suit the motor attached.

5. DC and Stepper

The BV4113 is a dual purpose device capable of controlling DC motors OR a stepper motor. In addition it has two general purpose digital lines that can be configured as either input or output and also one 10 bit analogue channel. This can be used for sensing or other purposes.

The next sections will deal with these in turn.

6. DC Motor Controller

Four channels are available for controlling DC motors, YA, YB, YC and YD. Channels can also be paired up to form differential outputs to enable forward and reverse.

YA and YB form differential output MA

YC and YD form differential output MB

There is also two enable lines associated with these channels, ENA and ENB. There is a facility to pulse modulate the enable lines. See the command that deals with pulse modulation.

DC / Stepper Motor Controller

BV4113

6.1. DC Motor Configurations

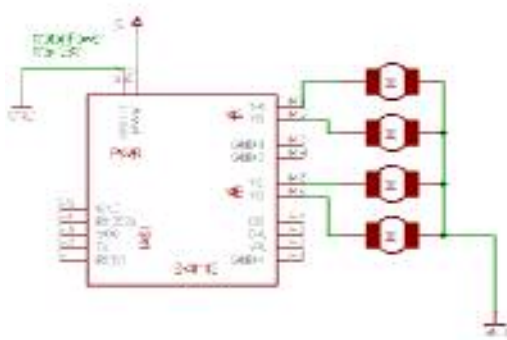


Figure 1 Single Direction Control

This configuration can control up to 4 DC motors but in one direction only.

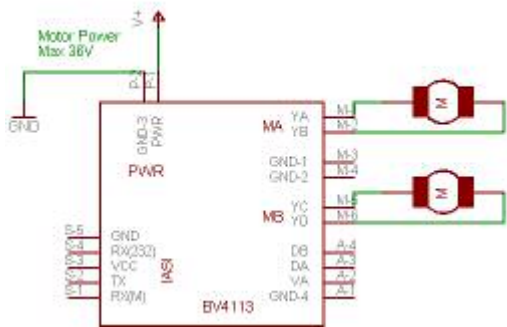


Figure 2 Differential Control

In this configuration two motors can be controlled but in both forward and reverse directions.

All of the configurations support Pulse Width Modulation. The PWM has a period of approximately 10us. The duty cycle can be adjusted from 0 to 100% of this by using the WA or WB commands.

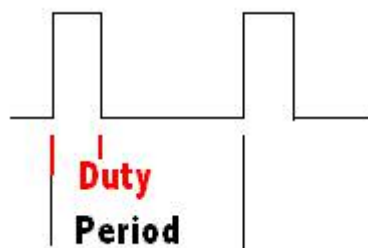


Figure 3 Duty Cycle & Period

The duty cycle is the active proportion of the period. This active period can be varied. The effect of this is to supply an average power output that can vary the speed of the motors. Depending on which command is used WA or WB this will activate the ENA or ENB lines for the duty period.

The PWM facility can be disabled so that the ENA and ENB commands can directly control the enabling of the YA to YD outputs

7. Stepper Motor Controller

The BV4113 can be used to control up to 4 DC motors OR a stepper motor, not both at the same time.

All of the channels YA to YD are used to control a stepper motor, the motor can be uni-polar or bi-polar.

7.1. Stepper Motor Configurations

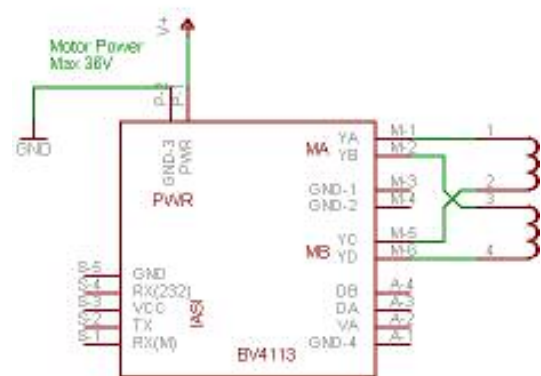


Figure 4 Bi-polar

A bi-polar stepper has 4 wires and they should be connected as shown.

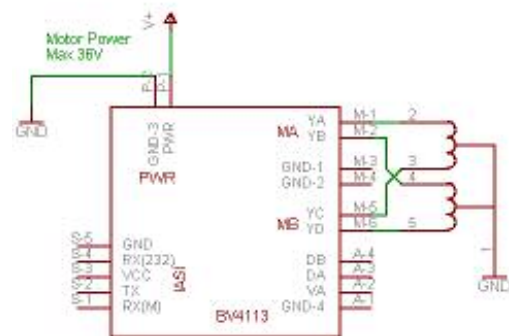


Figure 5 Uni-polar 5 wire

Uni-polar motors have either 5 or 6 wires. The common line shown here connected to ground can also be connected to the motor power supply instead of ground.

DC / Stepper Motor Controller

BV4113

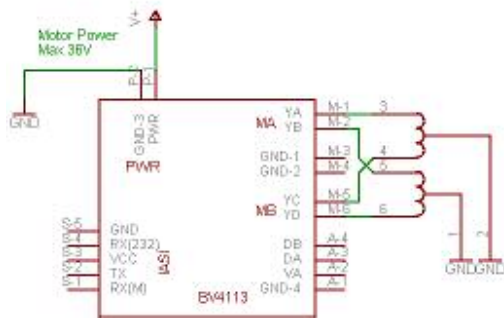


Figure 6 Uni-polar 6 Wire

6 Wire motors have two separate centre tap connections, these can be wired together to make an effective 5 wire motor.

8. Digital and Analogue Channel

There are two general purpose digital channels that can be either set to be input or output. There is also an 10 bit analogue input channel that can be read as an absolute value 0 – 1023 or as a percentage value 0 – 100. See the command for details of how there work.

The reference should be taken from pin 3 of the IASI connector as this is the logic supply.

NOTE also that the analogue input has a 100k pull up resistor connected to the +5V rail.



Figure 7 Connecting Analogue Example

As an example a potentiometer could be connected to an analogue channel. Note that one end of the potentiometer is connected to pin 3 (logic +5V) and the other to ground.

As usual entering an analogue command, say VA will return a value representing the voltage on pin 2 in text format. This is presented at the terminal.

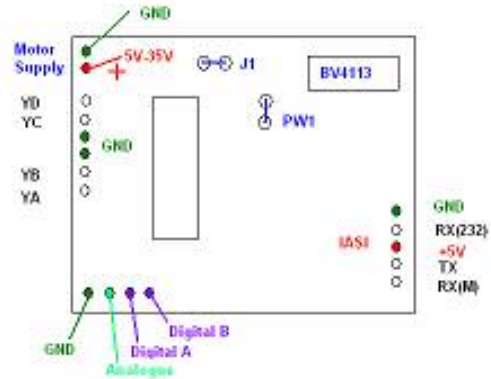
When the binary option is used, e.g. **VA*** the command will return **TWO** bytes. The first byte is the most significant byte. This also applies when command **PA*** is used, in this case the high byte will always be 0.

9. Power Saving

By default, when channel A (ENA) is deactivated (ENA 0) this also deactivates the L293 chip. The net effect is a considerable power saving. This however may not be important and may be even undesirable as ENA then effectively acts as a 'global' disable.

The action of this is controlled by a jumper PW1 on the circuit board which by default is shorted by a thin PCB track.

To disable the action of ENA controlling the power to the chip, the track between the jumper, this is on the underside of the board, should be cut.



Pin Layout

10. All Commands

Command	Name
Digital I/O & Analogue	
DA	Digital channel A
DB	Digital channel B
VA	Analogue value 0 – 1023
PA	Analogue Percent 0 – 100
Motor Control	
EA	Enable output YA,YB (MA)
EB	Enable output YC,YD (MB)
YA	Set output YA
YB	Set output YB
YC	Set output YC
YD	Set output YD
MA	Differential control Motor A
MB	Differential control Motor B
WA	PWM control A
WB	PWM control A
Stepper Control	
SC	Continuous Step
SD	Step Direction
SM	Step Mode
SN	Step number
SP	Step Pattern
SR	Step ramp up
SS	Step Speed

DC / Stepper Motor Controller

BV4113

SX	Stop
----	------

10.1. DA

Name: **Digital Channel A**

Command Parameters: [-o][-i][*][1][0]

Typical Use **DA -o**

Controls the digital channel. The channel can be set to either input or output. At reset the channel is set to input. Issuing the command without any parameters 'DA' will return the value on the port as text, either '0' or '1'. Using the command with a star (DA*) will return a single binary byte with a value of either 0 or 1.

The channel is configured to output by following the command with 'dash o', this is an o for 'output' not a zero. At reset all of the digital channels are configured for input.

When the channel is configured for output, DA 1 will set the channel high and DA 0 will set the channel low.

Parameter Table	
P	Meaning
-o	Sets the channel to output
-i	Sets the channel to input
*	Returns binary value instead of ASCII value
1	Sets the channel to high when in output mode, ignored when in input mode
0	Sets the channel to low when in output mode, ignored when in input mode

Example

DA -i

DA*

The above will set the channel to input and then read the value as binary, the value will be returned as a single byte with a value of either 0 or 1

10.2. DB – as DA

10.3. PA

Name: **Percentage value of Analogue A**

Command Parameters: [*]

Typical Use **PA**

The analogue channel is an input and has two options for the return values. Either as a percentage or as an absolute value.

The PA command will return a percentage, this is an internal integer calculation based on a

maximum input of 1023. Thus an absolute value of 511 will give a value of 50 when the PA command is used.

The voltage reference used for this is the +5V logic supply as connected to pin3 of the IASI connector.

By default, without the '*' the command will return a text value representing the value of the analogue voltage. In the above example if 511 was the value (approximately 2.38V), two bytes 53 (= ASCII 5) and 48 (= ASCII 0) will be returned. This will be shown on a terminal as 50. The number of bytes returned **without** the '*' will of course depend on the value. Single digit numbers will only return one byte.

The '*' parameter tells the command to return the value in binary and this will always return **two bytes** regardless of the value. The first byte is the most significant value. In the above example the first byte will be 0 and the second byte will be 50 (32h).

Using this particular format of the command can only return a maximum value of 100 so the first byte will always be a 0.

11. Digital I/O & Analogue

11.1. DA

Name: **Digital Channel A**

Command Parameters: [-o][-i][*][1][0]

Typical Use **DA -o**

Controls the digital channel. The channel can be set to either input or output. At reset the channel is set to input. Issuing the command without any parameters 'DA' will return the value on the port as text, wither '0' or '1'. Using the command with a star (DA*) will return a single binary byte with a value of either 0 or 1.

The channel is configured to output by following the command with 'dash o', this is an o for 'open' not a zero. At reset all of the digital channels are configured for output.

When the channel is configured for output, DA 1 will set the channel high and DA 0 will set the channel low.

Example

DA -i

DA*

The above will set the channel to input and then read the value as binary, the value will be returned as a single byte with a value of either 0 or 1

11.2. DB see DA

11.3. VA

Name: **Absolute value of Analogue A**

DC / Stepper Motor Controller

BV4113

Command Parameters: [*]

Typical Use **VA**

This is exactly the same command as PA except the output is an absolute 10 bit value from 0 to 1023.

The default is to convert this value to text for displaying on a terminal. If the '*' option is used then this value will be output as binary. The output is 2 bytes, the high value being output first.

Note that the maximum value is based on the logic supply voltage, any deviation from this will produce different results.

11.4. PA

Name: **Percentage value of Analogue A**

Command Parameters: [*]

Typical Use **PA**

The analogue channels are analogue inputs and have two options for the return values. Either as a percentage or as an absolute value.

The PA command will return a percentage, this is an internal integer calculation based on a maximum input of 1023. Thus an absolute value of 511 will give a value of 50.

The voltage reference used for this is 4.7V. This value can be obtained from pin 2 and cannot be altered.

By default, without the '*' the command will return a text value representing the value of the analogue voltage. In the above example if 511 was the value (approximately 2.38V), two bytes 53 (= ASCII 5) and 48 (= ASCII 0) will be returned. This will be shown on a terminal as 50. The number of bytes returned **without** the '*' will of course depend on the value. Single digit numbers will only return one byte.

The '*' parameter tells the command to return the value in binary and this will always return two bytes regardless of the value. The first byte is the most significant value. In the above example the first byte will be 0 and the second byte will be 50 (32h).

Using this particular command can only return a maximum value of 100 so the first byte will always be a 0.

12. DC Motor Control

12.1. EA

Name: **Enable Motor Channel A**

Command Parameters: [0][1]

Typical Use **EA0**

The primary function of this command is to control motor channel A. Motor channel A

consists of outputs YA and YB, setting this output to EA0 will disable the outputs

Setting EA to 1 (EA1) will enable the outputs.

PW Jumper

By default ENA has a dual purpose; it not only controls the selection of YA/B but is also controls a switch that switches off power to the L293. In most circumstances this is an advantage because of the power saving obtained. In standby mode the whole circuit only consumes about 2mA.

The action of this though is to disable all of the outputs effectively overriding ENB. If this action is not desired then there is an option to cut the track shorting the PW Jumper on the circuit board. With the track cut ENA no longer switches of power to the L293 chip and can be used as an independent controller for YA/B. The circuit will consume more power though.

12.2. EB

Name: **Enable Motor Channel B**

Command Parameters: [0][1]

Typical Use **EB0**

This command controls channel B. Channel B is outputs YC and YD.

EB0 disables the channel.

EB1 enables the channel.

12.3. YA

Name: **Motor Channel A output**

Command Parameters: [0][1]

Typical Use **YA1**

This will set output YA to either high or low. Primarily intended for controlling single motors or other high powered devices.

YA1 sets the output high

YA0 sets the output low

This output is enabled by ENA so ENA must be set to 1 for this to have any effect.

12.4. YB – as YA

12.5. YC

Name: **Motor Channel C output**

Command Parameters: [0][1]

Typical Use **YC1**

This will set output YC to either high or low. Primarily intended for controlling single motors or other high powered devices.

YC1 sets the output high

YC0 sets the output low

DC / Stepper Motor Controller

BV4113

This output is enabled by ENB so ENB must be set to 1 for this to have any effect.

12.6. YD – as YC

12.7. MA

Name: **Differential Output Control for motor channel A**

Command Parameters: **[0-4]**

Typical Use **MA0**

This command is designed for controlling motors as shown connected in Figure 2.

When controlling a single DC motor on outputs YA, and YB several things need to happen for the motor to go forward, reverse, stop etc. The MA command is a useful method of controlling this and effectively controls YA, YB and ENA according to the following table:

MA/B	ENA	Y1	Y2
0	0	0	0
1	1	1	0
2	1	0	1
3	1	1	1
4	1	0	0

As can be seen from the table MA0 disables the motor, MA1 drives the motor one way, MA2 drives it the other way and MA3 or 4 may be used for breaking or other effects.

Unlike the other commands MA must always be followed by a parameter (0-4), MA on its own will produce an error.

Example:

MA1 this will drive the motor one way
 MA2 this will reverse the motor
 MA0 this will stop the motor

12.8. MB as MA

12.9. WA

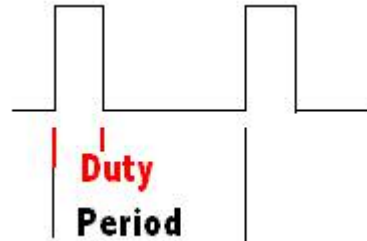
Name: **PWM Channel A**

Command Parameters: **[-A][-D][*][0-255]**

Typical Use **WA 50**

The pulse width modulator controls the enable lines ENA and ENB (WB). The effect of this is to control the speed of a DC motor.

The PWM has a period of approximately 10us. The duty cycle can be adjusted from 0 to 100% of this by using the WA command.



PWM Duty Cycle

The duty cycle is the active proportion of the period. This active period can be varied. The effect of this is to supply an average power output that can vary the speed of motors.

During the active period ENA is enabled.

Parameter Table

P	Meaning
-a	Activates PWM
-d	Deactivates PWM
*	Returns binary value instead of ASCII value
0-255	Sets the duty cycle within the period. The larger the number the more power delivered.

As the period for the modulator is approximately 10us. WA is a value from 0 to 256 that will vary the on duty cycle on this period. NOTE that this is used in conjunction with the **MA** command. MA1 for example will drive a motor connected to M1 in one direction. With WA-d full power will be applied to the motor, with WA-a power will be pulsed according to the value of WA.

WA can also be used with the YA and YB commands but not with the stepper motor commands.

An approximate calculation of the length of time ENA will be active is $(10/256) * WA_{us}$ or $0.039 * WA$. So if WA is 100 then the motor will be on for 3.9us and off for 6.1us.

The table shows the parameters that can be used with this command. **-a and -d** activates and deactivates the PWM mechanism giving digital (on/off) control back to the ENA and ENB lines.

Star * returns the binary value for the currently set PWM Value.

12.10. WB

This is as WA except that it will pulse line ENB.

DC / Stepper Motor Controller

BV4113

13. Stepper Motor Control

13.1. SC

Name: **Step Continuous**

Command Parameters: **[speed direction]**

Typical Use **SC**

If the command is used on its own the motor will continually step until a stop command is give (SX). The speed and direction are taken from stored parameters set by other commands.

As an option the speed and direction can be given in this command. If the speed is specified then direction must also be specified.

speed is a value form 0 to 256, the higher the number the less delay between steps and thus the faster the motor will go. The delay range is calculated by:

$(255 - \text{speed}) * 128\mu\text{s}$. This gives a minimum delay (max speed) of 128us when speed is 255 and a maximum delay (slowest speed) of 33ms when speed is 0.

direction is either 0 or 1

13.2. SN

Name: **Step Number of Steps**

Command Parameters: **[*][n][direction][active]**

Typical Use **SN 120**

Step number of steps specified by n, this is a number in the range of 1 to 32767. No checking is carried out so numbers outside this range may give unpredictable results.

A direction can be specified 0 or 1

Active: When the required number of steps have been reached the motor will stop and the L293 chip will be disabled by setting the ENA/B enables lines to low. This is the default action.

There may be a requirement for the motor to come to a 'firm' stop. This can be achieved by leaving the 293 chip energised at the expense of consuming current and possibly heating up the motor. This can be achieved by setting active to 1.

Examples

SN 120 Step the motor 120 steps

SN 120 1 Step the motor 120 steps in a particular direction

SN 120 0 Step the motor 120 steps in the opposite direction

SN 120 0 1 Step the motor 120 steps and leave energised when stopped

Specified on its own or with * it will return the number of steps to go. NOTE that if * is specified this is a 2 byte number, high byte first.

Using SN on its own to get the number of steps to go will give an unspecified value if the SC command has been used to energise the motor.

13.3. SS

Name: **Step Speed**

Command Parameters: **[*][0-255]**

Typical Use **SS 100**

Specifies the step speed for use with the SC or SN commands. The command on its own will display the speed setting. Using * will return a two byte binary value, high byte first.

speed is a value form 0 to 256, the higher the number the less delay between steps and thus the faster the motor will go. The delay range is calculated by:

$(255 - \text{speed}) * 128\mu\text{s}$. This gives a minimum delay (max speed) of 128us when speed is 255 and a maximum delay (slowest speed) of 33ms when speed is 0.

13.4. SP

Name: **Step Pattern**

Command Parameters: **[0-F] (hex)**

Typical Use **SP 3**

Step pattern applies only to full stepping mode and is specified as a hex value from 1 to F. The step patterns is a 4 bit value that the outputs step through in order to turn on and off the motor windings.

	y1	y3	y2	y4
s1	1	1	0	0
s2	0	1	1	0
s3	0	0	1	1
s4	1	0	0	1

If a step pattern of say 3 is specified using the SP command it will give the values in the above table, the pattern will be 3,9,c,6 and so on this will energise two coils at a time giving the maximum torque. As an alternative 1 could be specified for a step pattern:

	y1	y3	y2	y4
s1	0	0	0	1
s2	0	0	1	0
s3	0	1	0	0
s4	1	0	0	0

The above will not give as much torque but will use less current. Any pattern can be chosen including 0 that will do nothing at all.

13.5. SX

Name: **Stop Motor**

DC / Stepper Motor Controller

BV4113

Command Parameters: **[1]**

Typical Use **SX**

Stop the motor and disable 293 Chip

This is a general purpose stop command and by default, without the parameter will also pull the ENA/B lines low and switch of the logic on the chip.

Its primary purpose is to stop the stepper motor after using the SC, step continuous command. It can however be also used with DC motors as its action by disabling the 293 will effectively stop all functions.

Using the optional parameter will stop the stepping action of the stepper motor but leave the 293 enabled thus bringing the stepper to a firm halt and holding it there. This will consume current and may even overheat the motor if used for a prolonged periods.

13.6. SD

Name: **Step Direction**

Command Parameters: **[0][1]**

Typical Use **SD 0**

reports or sets the stepping direction. This can be used while the motor is turning. 0 and 1 are arbitrary directions and depend on how the motor is wired.

The direction can also be set using the SC and SN commands.

13.7. SR

Name: **Ramp Up**

Command Parameters: **[1-7]**

Typical Use **SR 4**

Most stepper motors are slow devices. To run at maximum speed it is useful to ramp up the speed over the period of a few steps. The ramp up and maximum possible speed is determined as much by the mechanical environment as well as the electrical.

The ramp up provided here is simple but effective. A number is specified between 1 and 7 and the step speed is simply divided by this value ^2. Thus if the step speed is 100 and the SR value is 4, the step values will be as follows:

Step 1 $100 / 4^2 = 6$

Step 2 $100 / 3^2 = 11$

Step 3 $100 / 2^2 = 25$

Step 4 $100 / 1^2 = 100$

Step 5 $100 / 1^2 = 100$

** There is no ramp down.

13.8. SM

Name: **Step Mode**

Command Parameters: **[H][F]**

Typical Use **SM H**

There is a choice of two step modes, half and full.

In full step mode there is full control over the step pattern as defined by the SP command. Half step mode uses a table to produce the following step pattern:

	y1	y3	y2	y4
s1	1	0	0	0
s2	1	1	0	0
s3	0	1	0	0
s4	0	1	1	0
s5	0	0	1	0
s6	0	0	1	1
s7	0	0	0	1
s8	1	0	0	1

The mode will give a smoother, slower performance from the motor. As this is half stepping mode, twice the number of steps will be required for a full revolution.

14. Error codes Specific to this interface

Code	Description
	none

DC / Stepper Motor Controller

BV4113

15. Revisions to the IASI Section

Rev	Change
May 2006	Additional information about connecting multiple devices and hardware factory reset.
June 2006	Troubleshooting guide added
V1.01	

16. Introduction to IASI

The Intelligent Asynchronous Serial Interface (IASI) is a common standard that makes it much easier to control and use hardware from either a standard communication interface (terminal) or a microcontroller.

It is based on a very simple text command set and a flexible hardware interface. The 'Intelligent' aspect is derived from the fact that each particular IASI knows about the connected hardware so a simple command can make the hardware perform a reasonably complex function. Scroll text on an LCD display for example.

when used in a microcontroller system this enables the controller and designer to concentrate on the important aspects of the design and control rather than the mundane job of controlling the hardware. It also means that the task of driving common peripherals is not being constantly re-invented.

17. IASI Electrical Interface

The device has very simple requirements. A power supply, transmit and receive lines as shown in table E1.

The interface is specifically designed so that it can be connected to either a standard com port (on a PC for example) or directly to a microcontroller UART or even a microcontroller port pin with a software generated UART (Universal Asynchronous Receiver and Transmitter). A five pin connector is used with normally only 3 or four pins being connected at any one time.

There are **TWO** receive lines, pin 1 receive line will accept normal 5V logic as presented by a microcontroller pin or UART and pin 4 will accept positive and negative voltages up to 15V that are normally present on a standard RS232 interface. Pin 4 will also invert the logic which is also normal for this interface.

The Baud rate is automatically detected at start up or it can be configured in software to a fixed, default baud rate. The device detects the baud rate on receiving a CR character (0Dh). Other received characters will be ignored until the Baud rate has been established.

The transmit pin has an open collector output that has a pull-up resistor on board connected through a jumper. Where more than one device is used on the same serial line, only one jumper should be shorted. See the section on multiple devices for further information.

18. Serial Connections

The device is designed to work in either of **two** modes: an INVETED mode for connecting directly to an RS232 port (factory default) or a NON-INVERTED mode for connecting to a microcontroller UART.

As previously described there are two inputs, one for each alternative interface. On the transmit side (output from the interface) there is only one pin that takes care of inverted and non-inverted logic, this is configured in software. The output is 0 to +5V only, rather than the RS232 specification requiring positive and negative signals.

On most RS232 specification interfaces this will work although it is not within the actual RS232 specification.

DC / Stepper Motor Controller

BV4113

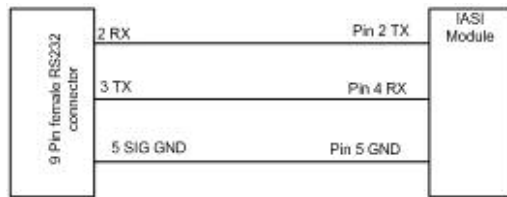


Figure 8 Connection to a PC

Figure E1 shows the connections to a 9 pin D type connector found on most PC's.

The Baud rate is automatically determined by detecting the first Carriage return it receives or it can be configured to a fixed baud rate in software. See the section on commands (**ZC**) for further details.

19. Factory Configuration

When an IASIM (Intelligent Asynchronous Serial Interface Module) leaves the factory it is configured to automatically detect the baud rate and interface with a standard PC com port. Regardless of the configuration the device will receive both forms of electrical interface (inverted and non-inverted) depending on which pin is used. So even if it may not be possible to see the output from the device, it will normally accept the input.

The transmit is configured by software and this is configured to invert. The interface does not need this pin to be connected to receive commands.

Factory settings can be restored both in software and hardware. The command ZF (see later) will restore the factory configuration. To restore factory defaults using hardware see Appendix A

20. Non-Inverted Mode

As previously mentioned the device is capable of operating with a standard RS232 communication port and this is the factory default and is useful for configuring and learning the device capabilities.

Most microcontrollers however operate on non-

inverted logic and output 0V for zero and +5V for logic 1. To operate in this mode issue the command:

ZC3 -r

See Section 25 or the command set for more information.

21. Commands

The interface is completely software driven, all commands and configuration are done through a serial interface. The only exception to this is the hardware factory default restore, see Section 27.

A very simple command protocol is used that essentially operates in two modes. An interactive mode and an addressing mode. The general format of the command is:

Interactive

<command><command-parameters><CR or ,>

e.g. **ZA**

Addressing

:<address><command><command-parameters><CR or ,>

e.g. **:00ZA**

The purpose of the interactive mode is to enable the user to learn the device capabilities and also to be used in a system where only one device is connected to the bus.

This is the default protocol the prompt for this is:

L>

The 'L' indicates that the device is in 'Listen' mode and will react to any command given and also produce error messages if non-existent commands are given.

An alternative protocol is available that allows multiple devices to be connected to the same serial bus. This is configured using the **ZC** command (see later).

In non-listen mode the prompt is:

Pin	Name	Description
1	RX	Receive data in non-inverted form at +5V logic levels. Use this pin for connecting to MAX232 devices or directly to microcontrollers.
2	TX	Transmit (output) data. This is 0V and +5V, RS232 levels are not used. Devices will work without this connected but no feedback can be received. This pin is configurable in software to transmit either normal or inverted logic. (see multiple devices section 26.4)
3	+5V	Standard 5V power to the device
4	RX-Invert	Receive data (input) this will accept -12V to +15V volts in inverted logic as is normally available on a PC Com port. The format is RS232 1 start bit 8 data bits and 2 stop bits.
5	GND	Ground

Table E1 Serial Connection Details

DC / Stepper Motor Controller

BV4113

:>

This indicates that a colon and address is required. In this mode the device will only accept commands with its own address and ignore everything else. In this mode the previous **ZA** command would be:

:00ZA

Assuming that the device address is 00.

Note that all addresses must be preceded by a colon. The prompt can be turned off by the **ZC** command.

21.1. Addressing

All devices have a two **character** address, this enables several devices to be connected to the same serial lines.

The address can be any two ASCII characters and it is case sensitive. Typical examples are **L1 AA 25**, etc. note that **aa** will be a different address to **AA**.

There is a special address **00** (zero, zero) that all devices will listen to regardless of their own address. There is nothing to stop more than one device having the same address, this can be useful if both devices always require the same commands.

As mentioned earlier if the device has been set to interactive mode and address is not required.

21.2. Command Format

The command is a two character string **ZA** for example and is **not** case sensitive, so **zc** would be the same as **ZC**. All commands have two characters and in general the first character is a class of command and the following letter is a subclass of that command. All commands beginning with **Z** are common to all devices and usually deal with configuration and communication.

Other two letter commands deal with the device direct and the same commands may mean different things for different devices.

21.3. Command Parameters

The command parameters can be anything and are specific to that command. In most circumstances spaces are ignored and can be used freely. Spaces do however serve to separate parameters (space is the delimiter) where more than one is required. For example given the command below that is used on the LCD IASM

L>WT3 "Hello"

The '3' requires a space after it so that the number can be distinguished from the text. In general though spaces are ignored and can be used freely.

21.4. Command Line (Buffer)

All commands strings are buffered in a 64 byte buffer and devices will store any text received on the bus. Any lines greater than 64 characters will be ignored and generate an error so it is important not to exceed this figure on an automatic system.

Providing the debug level is set to greater than zero (see **ZD** command) then this will be reported as an error.

When a Carriage Return (ASCII 0dh) is received (configurable) the text string is passed to the command processor. In non-listen mode only devices with a matching address will respond, two or more devices can have the same address if required.

The line end character is set at the factory to be 13 (0dh) and this can be changed using the **ZL** command. Some systems send only this others send a Line Feed (10, 0ah) and others send both, either way round. The IASI will only accept the configured value and ignore or skip over anything else.

A comma (,) can be used to allow more than one command in a single line.

This is especially useful in non-listen (address) mode as the device only needs addressing once:

:>:00ZAL2
:>:L2ZD1 or :>:00ZD1

can be sent as:

:>:00ZAL2,ZD1
(prompt)
(command)

The above will set the device address to L2 and the debug level to 1.

22. Command Set

Command sets consist of a two character command. The first character usually refers to the general type and the second later is specific to that command. The Z command set for example is the common command set for all devices. The Z commands configure the device interface and communications.

22.1. The Star *

As a general rule most commands issued on their own will return their value in printable format for example **ZD** will return the current debug level 0,1 or 2. This will be 'printed' on the terminal screen.

In absolute terms the values returned are in fact the values 30h, 31h or 32h which correspond to the printable values 1,2 or 3 (the ASCII codes for these characters).

When used with a microcontroller for example it may be more convenient to know the actual

DC / Stepper Motor Controller

BV4113

BINARY value. By using a * this will return the actual value thus.

ZD*

The above will return the actual binary value 1,2 or 3 which will not be printable on the terminal screen. This may be particularly important for numbers greater than 9 as the printed version of 10 is in fact 1 (31h) and 0 (30h), two bytes. Whereas the actual value 10 is only one byte and much easier to handle.

23. Common Command Set (Z)

A summary of the complete common command set is given in the table. All common commands begin with Z, as previously mentioned commands are grouped by the first letter so any command beginning with a Z is a common command.

A full description of each command is given after the summary.

23.1. Summary

Command	Description
ZA	Address
ZB	Baud Rate
ZC	Device configuration
ZD	Debug level
ZF	Factory reset
ZK	Ack character
ZL	Line end character
ZM	Record Macro
ZR	Reset
ZT	Test macro
ZV	Version
ZW	Wait

Note that any example given will assume that the device is in interactive mode (listen flag on) so no address is required to precede it.

23.2. ZA

Name: **Address**

Command Parameters: **[*address]**

Typical use **:00ZA or ZA (listen on)**

This command will show, confirm or set the address. To see the address of the current device simply use the command without any parameters:

ZA

This will return the actual address of the device, this will be 00 after factory reset.

ZAK1

This will set a new address for the device, K1. Addresses consist of two characters (including numbers) and are case sensitive so setting the address to AA will be different from aa.

:K1ZA*K1

This command is used for confirming that a device is on the bus and working okay. In the above if device K1 is on the bus it will respond with ACK (see the ZK command for ACK). The single ACK character is usually easier to detect by a microcontroller than the two byte address returned without the * parameter.

23.3. ZB

Name: **Baud rate**

Command Parameters: **[*][rate]**

Typical use **:00ZB or ZB (listen on)**

At factory default the Baud rate is detected by receiving a carriage return character (CR value 13, 0dh) from the sending device. A minimum of two are required, one for the detection and the other for confirmation. The quality and length of the serial bus may also effect the detection process.

The detection will chooses from a fixed set of baud rates:

0. 9600
1. 14400
2. 19200
3. 38400
4. 57600
5. 115200

This has been found to be very reliable depending on the quality of the electrical interface.

The command enables viewing and setting of any of the above values 0-5 that correspond to the Baud rate. If the Baud Detect flag (see the ZC command) is set to off (N) then the Baud rate will be determined by whatever this command is set to.

As an example if ZB is set to 3 and Baud detect is off, the device will communicate at 38400 Baud when switched on. If ZB is set to 3 and Baud detect is off then the Baud rate will be determined by the first acceptable Carriage Return character received.

ZB	Baud Detect	Baud Rate
3	Off	38400
3	On	Set by receiving CR character from host device.

DC / Stepper Motor Controller**BV4113**

Config	Byte	Ack	Macro	Baud-d	Invert	Listen	Prompt	Echo	
0	1E	n	n	y	y	y	y	n	Inv-interact
1	1A	n	n	y	y	n	y	n	Inv-colon
2	58	y	n	y	y	n	n	n	Inv-auto
3	16	n	n	y	n	y	y	n	Interact
4	12	n	n	y	n	n	y	n	colon
5	50	y	n	y	n	n	n	n	Auto

The Baud rate can also be reported in binary, in the above example:

BR

will return 3 (ASCII value 33h)

BR*

will return the value 3 which will not be a printable character, this is useful for microcontroller input, see the section on the star command extension.

23.4. ZC

Name: **Configure**

Command Parameters: **n [*][-w aaaaaa][-r]**

Typical Use **:00ZC3 -r**

This is probably the most complex command but will alter the way the device behaves and is only likely to be seldom used. There are 7 'flags' that will effect the device and these are:

1. ACK
2. Macro
3. Baud detect
4. Invert
5. Listen
6. Prompt
7. Echo

The above flags can either be on or off (Y or N) and they have the following meaning:

ACK

When the line end character (ZL command) is received by the device there is an option to send the ACK character (ZK command). If this is set to on (Y) then the ACK character will be sent. See the ZK command for details on why you would want to do this.

Macro

When the device is first powered on, optionally a set of commands can be run, see the ZM, ZT commands. If this flag is set to Y then the macro commands will be run at start up.

Baud detect

The device can operate at a fixed Baud rate or detect the host Baud rate at start up. When this is set to Y the device will wait for input from the controller Terminal to determine the Baud rate, see the ZB command)

Invert

This only applies to the transmit line of the device. When set to Y all transmitted data will be inverted, this is suitable for connection to normal RS232 type interfaces, a PC com port for example.

Listen

If this is set to Y a command will be accepted without addressing, this is useful for single devices connected to the bus and saves having to prefix all commands with colon, address.

Prompt

When set to Y will output on the transmit line a prompt. This is useful for experimenting with the device when using a standard terminal. There are two types of prompt depending on how the listen flag is set, see the section on addressing.

Echo

When this is set to Y all characters received will be transmitted to the transmit line. It is not advisable to have this on when used with automated systems. It may also slow down the command reception process.

The technique for setting these consists of storing a configuration in non-volatile memory and then restoring this configuration later.

There are eight configuration locations that can all be used. When restoring factory defaults the first 6 locations are overwritten with the values shown in the table.

To observe a configuration use:

ZCn or ZCn*

where n is a number between 0 and 8 using ZCn without a * will produce a sting of Y and N. As an example:

ZC0 will produce **NNYYYYN** given the factory defaults.

ZC0* will return 1Eh which will not be printable using a Terminal but will be more understandable to a microcontroller.

To change a configuration **-w** is used to indicate 'write'. As an example to set a new configuration in slot 6 the following command can be issued:

ZC6 -w NNYYYYN

This will write a new configuration to slot 6 having the same values as the factory default, configuration 0. Any slot (configuration) can be overwritten at any time but using the **ZF**

DC / Stepper Motor Controller

BV4113

command, factory defaults, will reset the first 6 configurations to that in the above table.

To set the device to a configuration **-r** is used (read), so:

ZC0 -r

will set the device to configuration 0.

NOTES

The purpose of the 8 configuration slots are to enable quick changing of device configuration. In most cases one of the default configurations will suffice and so the user may not be concerned with the individual aspects of each flag.

The first 3 (0-2) configurations are intended for using on PC terminals that have an inverted logic. The next three are intended for use with a microcontroller output that does not have an inverted logic.

There is nothing to stop the user from specifying illogical configurations or even using a slot that has not been previously configured. This can make the device very difficult to communicate with. If this should happen than a hard reset will probably be required, this is described in Section 27.

NOTE The use of this command with -r or -w will cause the device to reset, similar to using the ZR command.

23.5. ZD

Name: **Debug**

Command Parameters: **[*][n]**

Typical Use **:00ZD0**

There are three debug levels available. level 0 will not report any errors at all. This is used where multiple devices are on the same bus and the error reporting may interrupt normal command flow.

Level 1 will report to the normal transmit line and level 2 will report to the device if this is possible, LCD displays for example. Note if the device is not capable of displaying the error then level 2 will be equivalent to level 0.

As in the previous commands ZD on its own will report to the terminal in ASCII form and ZD* will report in binary form.

A new level can be set by following the command with the new level, e.g.

ZD1 set debug to level 1.

23.6. ZF

Name: **Reset to Factory Defaults**

Command Parameters: **none**

Typical Use **ZF**

This will set the device to the factory default configuration, see section 27 for what this is.

Providing there is communication with the device this command can be useful for establishing a known configuration automatically. For example:

```
ZF
CR (carriage return)
CR
:00ZF
CR
CR
```

Should set the factory defaults even if the device is in interactive mode or not. This can then be followed by a desired configuration, **ZC3 -r** for example.

23.7. ZK

Name: **ACK (acknowledge)**

Command Parameters: **[*][n]**

Typical Use **ZK6**

When a line end (CR) is sent to the device the contents of the buffer are interpreted for a command and then acted upon. This takes a finite time to complete. In an automated system a delay can be used to allow for this. However the delay must be set for the longest time to ensure that all commands can be dealt with.

Optionally (see the ZC command) an acknowledge (ACK) mechanism can be switched on. If this mechanism is switched on, when the command has completed an ACK will be sent to indicate to the controller that another line is ready to be accepted.

This forms a simple but reliable handshake method that can be used for sending text files to a device or for using automated devices.

The ACK character can be set or displayed using this command.

ZK displays the character represented as a decimal printable number, **ZK*** will return the ACK character as a binary value and **ZK6** will set the ACK character to the factory default value of 6

23.8. ZL

Name: **Line end character**

Command Parameters: **[*][n]**

Typical Use **ZL13**

The line end character (EOL) is important as it informs the IASI to interpret the contents of the buffer. Unfortunately this character is not in universal use, some systems send 13, other send 10 and others send a combination of both which can be either way round.

The ZL command allows the specification of the character that will mark the end of line. This also gives the opportunity for any special requirements, e.g. 0 can be specified.

Examples:

DC / Stepper Motor Controller

BV4113

ZL returns the value as a decimal number in text from.

ZL* returns the value in binary form (1 byte)

ZL2 sets the end of line character to 2. **WARNING** if you do this you will not be able to communicate with the device unless you end the command line with <CTRL>B (02)

NOTE:

The system will ignore or skip over any received values that are below a value of 32 (20h) except the following characters:

Escape	17, 1bh (interpreted as EOL)
Back space	8, 8h
ZL character	13, 0dh (by default)

23.9. ZM

Name: **Macro**

Command Parameters: **none**

Typical Use **ZM <see below>**

A macro is a common computer term to allow a sequence of commands to be recorded and played back at a later time. This command allows just that. The playback is normally at start up and allows programming of logos etc. on appropriate devices.

The command is used on its own; enter the command **ZM** and the following prompt appears:

175>

The number 175 (differing devices may have a different number) in front of the prompt shows the space available in characters. This will reduce as commands are entered.

Any valid commands can be used but the syntax is not checked until run time, the macro sequence can be checked with ZT. To exit this command use **Escape**.

Macros cannot be edited, the only way is to enter them from the beginning again. They can of course be sent as a text file from a terminal.

Note that if the macro flag is set, see the ZC command, the commands will be run at start up so care should be taken what commands are entered.

The factory defaults will not change the commands you have entered but it will set the macro flag to 0 so the macro will not run at start up. A brand new device has the macro flag set to 1 so it may be worth while setting this to 0 before experimenting. See the ZC command on how to do this.

The macro can be tested using the ZT command.

23.10. ZR

Name: **Reset**

Command Parameters: **none**.

Typical use **ZR**

This is the reset command and will reset the hardware and put the communication mode as if it had just been switched on.

23.11. ZT

Name: **Test Macro**

Command Parameters: **none**

Typical Use **:00ZT**

This will test any stored macro and should be used before setting the macro flag.

23.12. ZV

Name: **Version information**

Command Parameters: **none**

Typical Use **ZV**

This simply returns a string that contains the firmware and device version information.

Additional information is also displayed but this does not apply to all devices:

The ZV command is also used for detecting errors and indicating if the factory defaults have been set. The format is:

[WnFn] <version information>

Where n is either 0 or 1. Under normal circumstances this would read [W0F0]. For information about this see the separate headings under watchdog and factory reset.

NOTE that reading the data clears it so if the first read was:

[W1F0] <version information>

The second call of this command would read:

[W0F0] <version information>

23.13. ZW

Name: **Wait**

Command Parameters: **n (1-65534)**

Typical Use **ZW110**

This command will simply suspend the device for a length of time. The time is entered in approximately 10ms intervals, so 100 is 1000 ms which is 1 second.

A typical use is to 'hold back' a text file when downloading or in a macro so that animated displays or actions can be implemented. At any time during the delay if a character is received the delay will abort.

This command is not suitable for highly accurate delays as the 10ms interval may vary, particularly from device to device.

DC / Stepper Motor Controller

BV4113

Example:
ZW 200

The above example will delay approximately 2 seconds.

24.Z Command Error Codes

Error codes will be displayed if the debug level (ZD) is set to greater than 0.

Code	Description
1	End of input buffer reached, buffer full. If this happens the whole of the input line will be ignored.
2	Unknown command, the command issued is not in the command table for this device.
4	Non-valid hex, where a command is expecting a hex value and something is entered outside 0-9, a-f, A-F. NOTE This is rarely used as most numbers values are in decimal.
5	Incorrect parameter following command. This refers to commands that expect a particular value to follow. This error will indicate that a command has been entered that is not in the correct / expected format.
6	Bad decimal number. This is because the command is expecting a number and something else has been entered. Note that a space should always follow a number.

Start HyperTerminal or some other terminal software, BV Terminal is ideal and can be obtained from www.byvac.com **Error! Reference source not found.** The following settings should be used:

Baud rate 9600
Start bits 1
Stop bits 2
Handshake none
Local echo on

(The Baud rate is not that important as the IASI will adjust to the terminals Baud rate)

Power up the device and press 'return' a few times making sure that the terminal is active (mouse cursor in the terminal window). The device should respond with:

L>

when it is ready to accept commands. If this is not the case check the power supply and terminal settings.

The device is now ready to accept commands, try **za**. This will return 00 which is the default device address.

After this prompt the device is now ready to be configured but it may be worthwhile spending some time looking at the command options available.

25.1. Reconfiguring

It may be that you want to use the device through a line driver device (MAX232) or microprocessor UART without bothering with the PC com port cable. This is also possible.

25. Connecting and Configuration

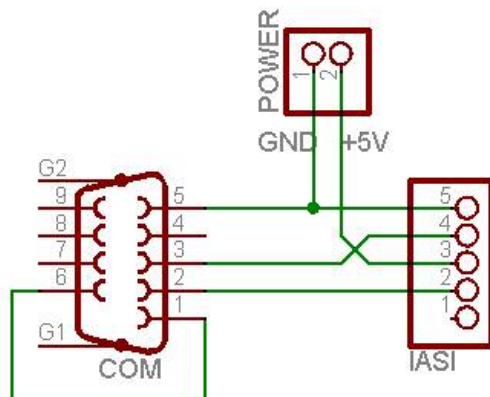


Figure 9 Connection wiring

The above wiring diagram shows the connections to a standard PC 9 way com port (RS232 connector). Pin 1 of the IASI has no connection as this is used to connect to a microcontroller UART.

The factory defaults will work with the above configuration.

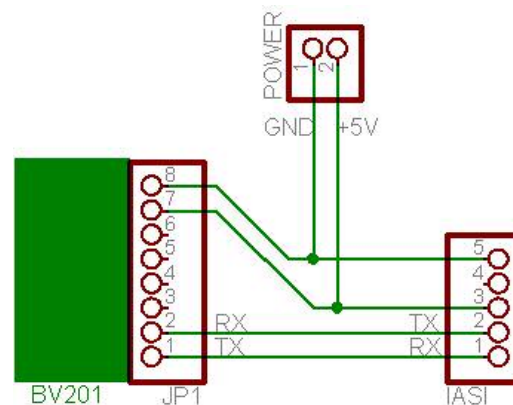


Figure 10 Using non-inverted

The above illustrates the connections used for, in this example a BV201 board that simply translates the PC com port to non-inverted 5V logic levels.

Using a BV101 a USB solution could be provided and there would be no need for a separate power supply.

See www.byvac.co.uk for these products.

Start up HyperTerminal or BV Terminal as before and press enter two or three times. In

DC / Stepper Motor Controller

BV4113

this instance the terminal will not show the prompt as the output is the wrong polarity, simply ignore any characters returned and type:

ZC3 -r

Now press return two or three times to establish the Baud rate and the **L>** prompt should appear. This illustrates that it is not necessary to receive anything from the device in order to reconfigure it.

The command ZC3 -r, loads configuration 'slot' 3 into the device. See the ZC command for standard configurations.

26. Microcontroller Use

26.1. Simple

There are several ways the device can be used with a microcontroller. The simplest way is to use it in the default mode:

ZC0 -r (inverted)

ZC3 -r (non-inverted - most likely)

For an output device, and LCD for example no return value is necessary but sufficient time should be given for the command and device to operate. The TX line (from the IASI) does not even need to be connected.

26.2. With feedback

A more sophisticated method would be to have some feedback from the device. There is also no need for a prompt.

The table shows a configuration that will do this for a single device. To place this is say, slot 6 requires the following command:

ZC6 -w YNYNYNN

To load this into the device type:

ZC6 -r

and issue two or three CR to establish the Baud rate.

ACK	Y
Macro	N
Baud-d	Y
Invert	N
Listen	Y
Prompt	N
Echo	N

As described in the ACK section of the ZC command the purpose of this is to tell the master controller (microcontroller) that another command can be issued. This can greatly speed up the transfer process.

26.3. Baud rate

It may be convenient to fix a Baud rate rather than having to establish it each time by sending CR. Two things need to be done for this:

ACK	Y
Macro	N
Baud-d	N
Invert	N
Listen	Y
Prompt	N
Echo	N

First set the desired baud rate according to the table illustrated in the ZB command. As an example, to set the baud rate to 38400 do this:

ZB3

Now configure the device so that the automatic Baud rate detection is switched off as in the table above.

The summary of commands would be as follows:

ZB3

ZC6 -w YNYNYNN

CR (carriage return)

CR

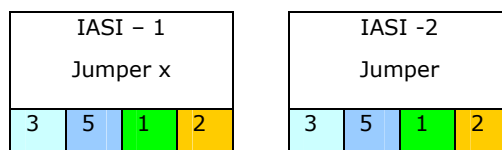
The device will retain the above configuration until either reconfigured or the factory defaults have been reset.

26.4. Multiple Devices

In the above examples the configuration assumed that only a single device was connected to the serial bus and so no addressing was required, commands such as ZA and ZL would be readily accepted by the device (called listen mode).

If more than one device is connected then, unless both are not required to respond to the same command, each device needs its own address.

Four (three if no feedback is required) connections are required. The diagram shows two devices but more devices can be added.



- 3 - Connect +5V together
- 5 - Connect Ground together
- 1 - Connect RX together

DC / Stepper Motor Controller

BV4113

2 – Connect TX together

Note: the electrical interface is designed to accept multiple devices on the RX side for either input from RS232 (inverted) or UART (non-inverted). The TX side however will only transmit to multiple UART devices in non-invert mode.

Jumper: remove all of the jumpers except one

ZAD1

Will set the device to address D1 (note that d1, lower case would be different). It is also a good idea to stop error messages:

ZD0

Finally configure the device to stop listening and no prompt.

ACK	Y
Macro	N
Baud-d	N
Invert	N
Listen	N
Prompt	N
Echo	N

To illustrate this as a series of commands the following would be required. Using slot 7 to hold the configuration data.

ZAD1
ZD0
ZC7 -w ynnnnnn
CR (carriage return)
CR

The device can now be commanded as follows:

:D1za

This will return 3 bytes 'D' '1' and 6 (if using the default ACK).

27. Restoring Factory Defaults

Factory defaults can be restored either by software or hardware. The factory default condition is:

ZA00 (address 00)
 ZC configuration 0 (see the ZC command)
 ZK6 (ack character)
 ZL13 (end of line character)
 ZB0 (baud rate 9600) **

** Note that the Baud rate will be ignored as the default Baud detect, set by configuration slot 0 is set to on.

The first 6 configuration slots will be set as the table given in the ZC command. Any user

information placed in those slots will be overwritten.

Applicable only to some devices:

Where a factory reset has occurred for whatever reason the ZV command will indicate this. A typical read out would be:

[W0F1] <version information>

The F1 in the square brackets indicates that a factory reset has taken place, this data is stored in EEPROM so even if the device is switched off the data still remains. The act of reading the data using the ZV command will clear the F1 back to F0, so this can only be seen once per factory reset.

27.1. Software

To set the above condition in software use the ZF command, once the ZF command has been issued the device will reset and wait for at least two CR (carriage returns) from the host device.

27.2. Hardware

It is unlikely that this will ever be used but has been provided in the unlikely event that a combination of configurations has rendered the device unable to communicate. It will require a short piece of wire and the holes will vary slightly from device to device.

The technique is as follows:

1. Power down the device.
2. Temporarily connect the two holes on the device together as shown. If the picture does not match exactly, then look for 5 holes in a row, at one end there will be a square hole, this is hole 1. Connect together holes 1 and 5.
3. Power up the device, this will restore the factory settings. On display devices this is shown as *F
4. Power down the device.
5. Remove the shorting link.

The device is now restored to the factory settings. NOTE that if a macro was programmed at the factory this will no longer show. On an LCD device for example it will not show the ByVac screen as it did when it left the factory, just the cursor will show.

DC / Stepper Motor Controller

BV4113

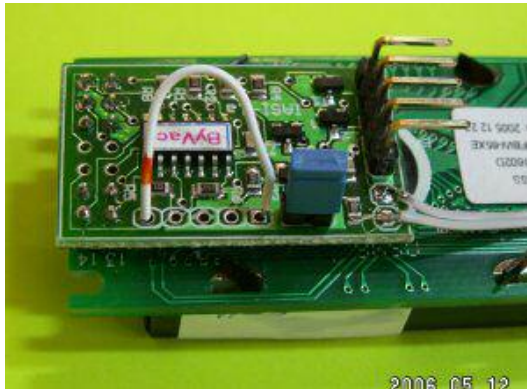


Figure 11 Shorting link

28. Watchdog

*** Not applicable to all devices ***

Some devices have additional error protection in the form of a watchdog timer. The timer is set internally and is constantly kept up to date by the firmware. Should anything unforeseen go wrong and the firmware is unable to update the timer a reset will occur.

The most likely scenario for this happening is if the serial interface is overwhelmed by input that the built in overrun protection cannot cope with.

If a watchdog time out occurs, a flag is written to the EEPROM. This can be read using the ZV command:

[W1F0] <version information>

The output from the ZV command will indicate that a watchdog time out has occurred by a '1'

following W in the example above. The act of reading the information will clear the flag, so the next read using ZV will show:

[W0F0] <version information>

Use this information to debug the external driving software.

29. Troubleshooting

The device does not show an L>prompt anymore and does not appear to respond to any commands.

The IASI device has a very comprehensive set of input options and it is quite possible to set a combination of input options that are not compatible with what is connected. It will appear that the device has stopped working when in fact it is simply operating in a different mode.

What's required is to get back to the factory default and take it from there. To do this follow the methods used in section **27 Restoring Factory Defaults**

Note that any factory macro which displays a start up sequence, the "ByVac" in the case of the LCD display for example will not be displayed. This is because the default setting is to turn the macro run off, this is a safety feature in case there is some erroneous code in the macro which prevents the display from starting.