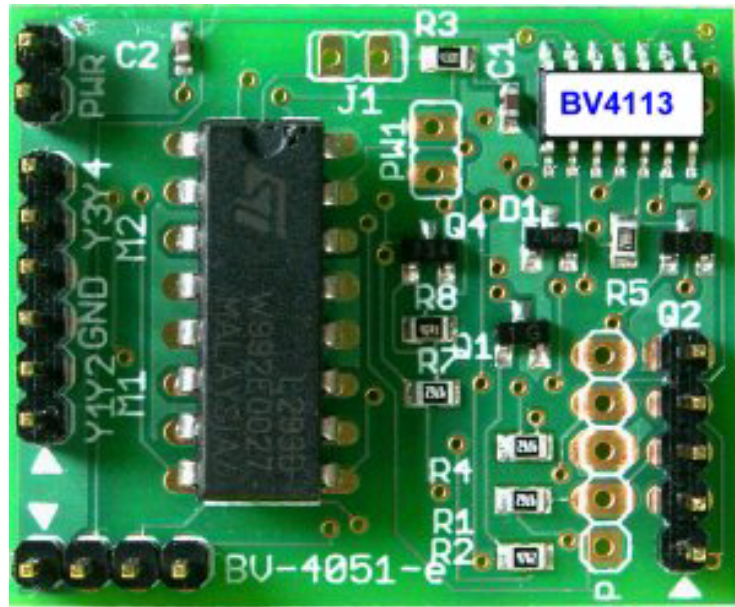


DC / Stepper Motor Controller

BV4113



BV4113 DC / Stepper Motor Controller

Product specification.

May 2009

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
11.	Digital I/O & Analogue.....	6
11.1.	d	6
11.2.	v	7
12.	DC Motor Control	7
12.1.	Power saving.....	7
12.2.	e	7
12.3.	y	7
12.4.	m	8
12.5.	w.....	8
13.	Stepper Motor Control	9
13.1.	c.....	9
13.2.	n	9
13.3.	s.....	9
13.4.	p	9
13.5.	x	10
13.6.	r.....	10
13.7.	u	10
13.8.	h	10
14.	Error codes Specific to this interface.....	10
15.	Revisions to the IASI-2 Section	11
16.	Introduction to IASI-2	11
17.	IASI-2 Electrical Interface	11
18.	Serial Connections	11
19.	Start Up	12
20.	Command Format	12
21.	Numbers.....	12
22.	Factory Configuration	12
23.	Non/Inverted Mode	12
24.	Commands.....	12

DC / Stepper Motor Controller**BV4113**

24.1.	Command 1	12
24.2.	Command 3	12
24.3.	Command 4	12
25.	Addressable Commands.....	13
25.1.	Summary	13
25.2.	A (0x41).....	13
25.3.	B (0x42).....	13
25.4.	C.....	13
25.5.	D.....	14
25.6.	E	14
25.7.	F	14
25.8.	G.....	14
25.9.	U.....	14
25.10.	M	14
25.11.	N.....	14
25.12.	P	15
25.13.	R.....	15
25.14.	V.....	15
25.15.	T.....	15
25.16.	Z.....	15
26.	Error Codes.....	15
27.	Connecting and Configuration.....	15
27.1.	Start Up	16
28.	Microcontroller Use	16
28.1.	Multiple Devices	16
29.	Restoring Factory Defaults	17
29.1.	Software.....	17
29.2.	Hardware.....	17

DC / Stepper Motor Controller

BV4113

1. Document Versions

1.0 November 2008 – For version 1.a

2.0 Additional notes for power save circuit

2.a Error in previous sheets, this device no longer outputs binary using the * option for any commands except the digital input.

2.b Version 2.a firmware now is in line with other IASI modules

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 IAS12 device all control is affected by simple single 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 from a selected set.
- 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 serial 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

DC / Stepper Motor Controller

BV4113

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.

6.1. DC Motor Configurations

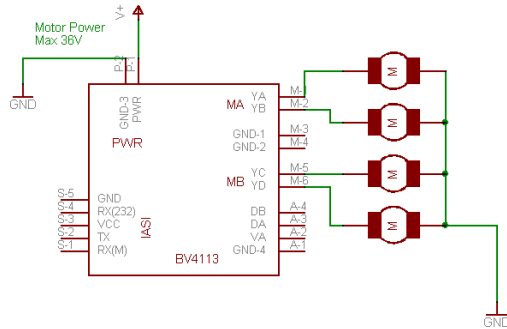


Figure 1 Single Direction Control

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

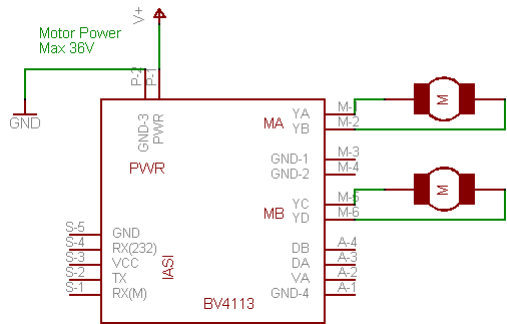


Figure 2 Differential Control

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.

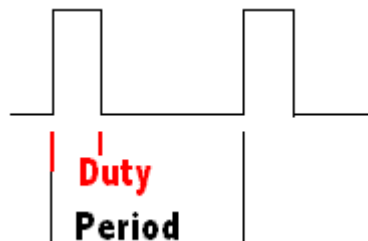


Figure 3 Duty Cycle & Period

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

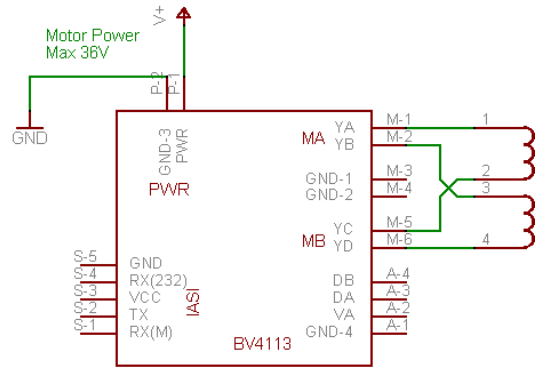


Figure 4 Bi-polar

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

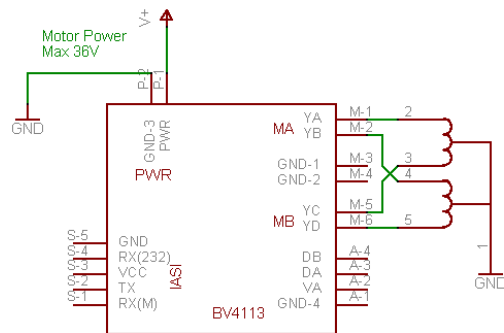


Figure 5 Uni-polar 5 wire

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

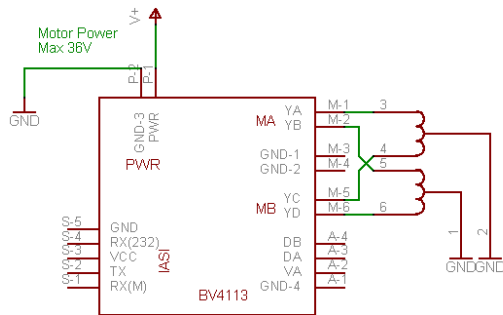


Figure 6 Uni-polar 6 Wire

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.

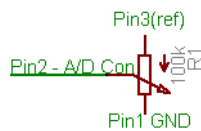


Figure 7 Connecting Analogue Example

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.

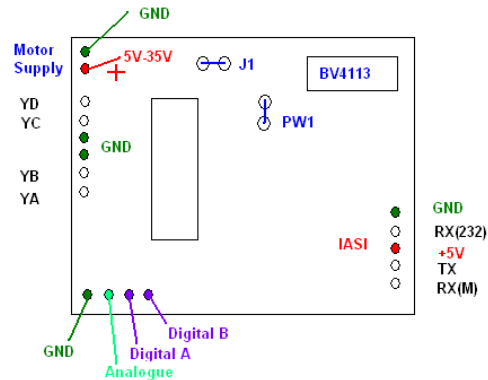
As usual entering an analogue command, say VA will return a value representing the voltage on pin 2 in text format. This is presented at the terminal.

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

All commands require to be prefixed with a device address in the range 'a' to 'z' and so any examples given use the default address of 'a'.

This device is set to **address "a"** on factory reset

Command	Name
Digital I/O & Analogue	
d	Digital channel A or B
v	Analogue value 0-1023 or as a percentage 0-100
Motor Control	
e	Enable output YA,YB,YC,YD
y	Set output YA,TB,YC,YD
m	Differential control Motor A,B
w	PWM control A or B
Stepper Control	
c	Continuous Step
r	Step Direction
m	Step Mode
n	Step number
p	Step Pattern
u	Step ramp up
s	Step Speed
x	Stop

11. Digital I/O & Analogue

11.1. d

Name: **Digital Channel A or B**

Command Parameters: **a or b[-o][-i][*][1][0]**

DC / Stepper Motor Controller

BV4113

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 or db' 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, da1 or db1 will set the channel high and da0 or db0 will set the channel low.

Parameter Table	
P	Meaning
-o	Sets the channel to output
-i	Sets the channel to input
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 (assuming device address is a)

ada Returns the value on channel A as a text '1' or '0'

ada-i Sets channel A to be an input (default value)

adb-o Sets channel B to be an output

adb0 Sets channel B to output 0 (low)

11.2. v

Name: **Percentage or absolute value for analogue channel**

Command Parameters: **[p]**

Typical Use **avp**

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

The 'v' command without the p specified will return the absolute value as a number between 0 and 1023 (10 bits)

The 'p' option 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 p option is used.

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

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), four bytes would be returned:

511>

53 (= ASCII 5) , 49 (= ASCII 1) , 49 (= ASCII 1) and ,62 (= ASCII >) will be returned.

Example:

av Returns the value as an absolute 0-1023

avp Returns the value as a percentage 0-100

12. DC Motor Control

12.1. Power saving

Most of the following commands are effected by the power saving feature. It is important to realise that the L293 is disabled by default. To enable it, ENA must be enabled. Please remember this when selecting functions for channel B only.

12.2. e

Name: **Enable Motor Channel A or B**

Command Parameters: **a or b [0][1]**

Typical Use **aea0**

The primary function of this command is to control motor channel A or B. Motor channel A consists of two outputs YA and YB, setting this output to **aea0** will disable the outputs, setting **aea0** will enable the outputs.

Likewise Channel B can be controlled by specifying 'b' after the command thus: **aeb0** and **aeb1**

PW Jumper

By default channel A 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 'eb'. 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 'ea' 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.

12.3. y

Name: **Motor Channel A,B,C or D output**

Command Parameters: **a,b,c or d [0][1]**

Typical Use **ayb0**

DC / Stepper Motor Controller

BV4113

This will set the individual outputs of channels a,b,c or d to either high or low state. Primarily intended for controlling single motors or other high powered devices.

As marked on the PCB

Y1 = b

Y2 = a

Y3 = d

Y4 = c

For these to work the channel also has to be enabled using the 'e' command.

12.4. m

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

Command Parameters: **a or b [0-4]**

Typical Use **ama3**

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 'm' command is a useful method of controlling this and effectively controls YA,YB and the enable 'e' command according to the following table:

M[a or b]x	'ea'	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 form the table 'ma0 or mb0' disables the motor, 'ma1 or mb1' drives the motor one way, 'ma2 or mb2' drives it the other way and 3 or 4 may be used for breaking or other effects.

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

Example:

ama1 this will drive the motor one way on channel A (Y1,Y2)

amb1 this will drive the motor one way on channel B (Y3,Y4)

ama2 this will reverse the motor on channel A

ama0 this will stop the motor on channel A

NOTE: If the default power saving mode jumper is not cut then disabling channel A (ama0) will also disable channel B

12.5. w

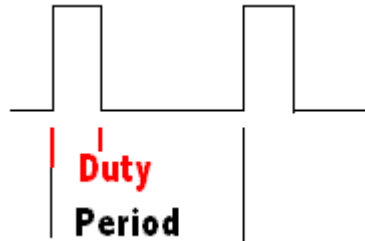
Name: **PWM Channel A or B**

Command Parameters: **a or b [-A][-D][0-255]**

Typical Use **awa 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 or ENB is enabled.

Parameter Table	
P	Meaning
-a	Activates PWM
-d	Deactivates PWM
0-255	Sets the duty cycle within the period. The larger then 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. This is used in conjunction with the 'm' command and the enable and Y commands.

An approximate calculation of the length of time ENA will be active is $(10/256) * WAus$ 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.

Example:

awa-a Activate PWM for channel A

DC / Stepper Motor Controller

BV4113

awa50 Set PWM to 50 for channel A
Note the PWM requires enabling before use with the -a switch

13. Stepper Motor Control

13.1. c

Name: **Step Continuous**

Command Parameters: **[speed direction]**

Typical Use **ac**

If the command is used on its own the motor will continually step until a stop command is given ('x'). 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

Examples

ac 120 1 Continuous step, speed 120, direction 1

ac Continuous step

13.2. n

Name: **Step Number of Steps**

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

Typical Use **an120**

Step number of steps specified by x, 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

an 120 Step the motor 120 steps

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

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

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

Specified on its own it will return the number of steps to go.

13.3. s

Name: **Step Speed**

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

Typical Use **as 100**

Specifies the step speed for use with the 'c' or 'n' commands. The command on its own will display the speed setting.

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. p

Name: **Step Pattern**

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

Typical Use **ap 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 'p' 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

DC / Stepper Motor Controller

BV4113

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.

Using the command on its own will return the current step value.

13.5. x

Name: **Stop Motor**

Command Parameters: **[1]**

Typical Use **ax**

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 'c', 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. r

Name: **Step Direction**

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

Typical Use **ar 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 'c' and 'n' commands.

13.7. u

Name: **Ramp Up**

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

Typical Use **au 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 $100 / 2 = 50$

Step 5 $100 = 100$

** There is no ramp down.

13.8. h

Name: **Step Mode**

Command Parameters: **[h][f]**

Typical Use **ah 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 'p' 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

Data sheet for the L293 can be found on the CD-ROM that came with this purchase or alternatively at www.byvac.co.uk

DC / Stepper Motor Controller

BV4113

15. Revisions to the IASI-2 Section

Rev	Change
Oct 2008	Preliminary

16. Introduction to IASI-2

The Intelligent Asynchronous Serial Interface (IASI-2) 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 and software interface. The 'Intelligent' aspect is derived from the fact that each particular IASI-2 knows about the connected hardware so a simple command can make the hardware perform a reasonably complex function.

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-2 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

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 28.1)
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

microcontroller pin or UART and pin 4 will accept positive or 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 on the first or second receipt of Carriage Return (#13). The detection is from a fixed set of Baud rates: 9600, 14,400, 19,200 and 38,400.

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 INVERTED 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.

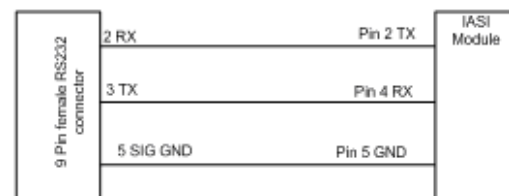


Figure 8 Connection to a PC

Figure shows the connections to a 9 pin D type

DC / Stepper Motor Controller

BV4113

connector found on most PC's.

19. Start Up

The interface will wait for a Carriage Return (#13) from either the inverted or non-inverted input in order for it to establish a Baud rate. The Baud rate is determined from a fixed range 9600, 14,400, 19,200 and 38,400.

No feedback is given and so it is possible wise to send more than one CR just in case. Once the Baud rate has been established the interface is ready to receive commands.

20. Command Format

All devices have an address which is one byte in the range 97 to 122 (0x61 to 0x7A), this corresponds to the printable ASCII characters 'a' to 'z'

The **default address is 'a'** and all devices must be addressed although there are some global commands that address all of the devices at once.

There are basically two sets of commands, those which are common to all devices, these are usually bytes that correspond to upper case characters 'A' - 'Z' and there are device specific commands using higher values that correspond to lower case characters 'a' - 'z'.

This section deals with the system commands.

21. Numbers

Some commands require an ASCII coded number and other commands require a byte, for example when specifying the brightness of the LED display the command is **aj4**.

'a' Is the address

'j' Is the brightness command and

'4' Is the value of the brightness.

This command is specified as an ASCII code so the actual bytes sent to the LED device is:

97 (a) 106 (j) 52 (4)

Note that the '4' is sent as byte 52 (0x34) and not the byte 4.

This is convenient when directly typing commands at a terminal but can cause confusion when using code. As a generalisation if a byte value is required then the code will be something like:

Send(#4)

But if an ASCII coded command is required as in the above example, it would probably be sent as text:

Send("aj4");

22. Factory Configuration

When an IASIM (Intelligent Asynchronous Serial Interface Module) leaves the factory it is usually configured to address 'a'

Factory settings can be restored normally by shorting two connections with a piece of wire and cycling the power.

23. Non/Inverted Mode

As previously mentioned the device is capable of operating with a standard RS232 communication port (inverted) and a microcontroller (non-inverted). The device will accept either signal but will output only one and at reset this is inverted

24. 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.

When a command has successfully completed it will return the byte value 62 (0x3e) (displayed value '>') This can be detected by software as an acknowledgement (ACK).

There are a few special commands that enable discovery of the devices and system wide defaults.

24.1. Command 1

This is the discovery command and it is a byte with a value of 1 that needs to be sent to the device, this can be done on a terminal by (Ctrl-A). On receiving this, the device will send back its address. This however is done in a timely fashion with address 'a' being sent first and 'z' last. Each device has 30ms to send its address and will wait its turn, therefore the device with address 'z' will wait 26x30=780ms to send its address.

The address is sent along with the ACK '>' As an example if 3 devices were connected to the bus 'a', 'f' and 'p' the response to command 1 would be:

a>f>p>

24.2. Command 3

This will reset all devices as if they had just been powered up. Following this command one or two CR is required to establish the Baud rate.

24.3. Command 4

At reset the output from the device will be inverted, this command will set all devices to non-inverted. This command should be used if the devices are connected to a microcontroller. Or if a USB-TTL (BV101) type device is being used. The start up sequence for example would be:

DC / Stepper Motor Controller

BV4113

CR
 CR ; to establish baud rate
 Command 4 ; to set non-invert

At this point a discovery command (1) could be sent to see if all of the expected devices are working.

25. Addressable Commands

The next block of commands are directed at a single device and so need an address before sending the command.

The default address of a device is 97 ('a') By convention these commands are in the range 65 to 90 giving a printable character of 'A' to 'Z', this makes it easier for text input if required.

25.1. Summary

Command	Description
A	Address
B	Write to EEPROM
C	Turn off ACK
D	Delay
E	Turn off error reporting
F	Factory reset
G	Read EEPROM
U	Unlock
M	Macro run at start up
N	Switch to non-inverted
P	Print contents of EEPROM
R	Reset device
V	Version
T	Test macro
Z	Create macro

Note that examples will use the default address of 'a' and the address and commands will be shown as their ASCII code because these can be entered directly from a terminal. The device however will only recognise the byte value so when 'aA' is entered the device will see two bytes 97 and 65

25.2. A (0x41)

Name: **Address**

Command Parameters: **byte 97-122**

Typical use **aAp**

This command is used to set the address of the device. The address is one byte with a value of between 97 and 122, giving 26 possible addresses. The range has been chosen because

it renders the values as printable characters in the range 'a' to 'z'.

To set a device from its default address to address 'p':

aU

aAp

If this is successful the device will return byte value 0x3e which is the ASCII code for '>' Note that this command requires an unlock (aU) before it can be issued, this is a safeguard to prevent the device from unexpectedly changing the address.

The address is stored at location 0 of the EEPROM – see command B.

When a device is used with other devices on the same bus the addresses must be set up individually before placing them on the same bus.

25.3. B (0x42)

Name: **Write to EEPROM as text**

Command Parameters: <address><space>'text'

Typical use **aB10 'Hello'**

This device has an internal EEPROM with an address range 0 to 255. Some of the addresses are used for system and macro storage so care must be taken where this text goes.

No check is made that the system or macro area is being overwritten, the first 16 bytes are reserved for system use so overwriting this may necessitate a hardware factory reset.

The macro area starts at 0xB0 so if there is a macro defined then this should be avoided.

The command format is
 aB<address><space>'text'

Where <address> is the starting address of the EEPROM between 0 and 256. There must be a space between the starting EEPROM address and the single text quote. Note that this is a single quote (0x27). The text is written and the command appends a 0 onto the end of it so it will occupy one extra EEPROM space. Hello for example would be stored as:

0x48,0x65,0x6c,0x6c,0x6f,0x0

This is 6 bytes not 5 as may be expected.

25.4. C

Name: **Turn off ACK**

Command Parameters: **None**

Typical use **aC**

Some devices may be adversely effected by the ACK command or the controlling software may not require the ACK #67 byte. This command will suppress the ACK.

The device must be reset to turn it back on.

DC / Stepper Motor Controller

BV4113

25.5. D

Name: **Delay**

Command Parameters: **1-255**

Typical use **aD50**

Pauses the device for a number of milliseconds. Some devices may require a small delay between commands particularly when used with the macro facility.

The delay is only approximate and should not be used for timing purposes.

As an example the LCD display required a delay after clearing and cursor home, so the macro would look like this:

```
aZac1;aD50;at'Hello';
```

25.6. E

Name: **Turn off error reporting**

Command Parameters: **none**

Typical Use **aE**

By default error reporting is enabled and this will be reported and an output prefixed by Error, for example **'Error 2'**. This may get in the way of the program trying to control the device and so it can be disabled with this command. The only way to enable it again is by resetting the device.

Example: aE.

25.7. F

Name: **Factory reset**

Command Parameters: **YeS**

Typical Use **aFYeS**

Sets the device back to the factory defaults, the command must be followed by bytes 0x59, 0x65 and 0x53 which is the ASCII codes for 'Y' 'e' 'S'

This will prompt on completion and will not require re-initialisation, the address of course will now be 'a'.

25.8. G

Name: **Read EEPROM**

Command Parameters: **aGss nn**

Typical Use **aG0 3**

The EEPROM values can be read with this command.

ss is the start address of the EEPROM in **hex**

nn is the number of bytes to read in **hex**

This command will accept ss and nn as number text values, this means that for the command:

```
aG10 3
```

The actual bytes sent to the device are:

```
0x61,0x47,0x31,0x30,0x20,0x33
```

Note how the 10 for the start address of the EEROM is specified as 0x31,0x30 which is the ASCII code for 10.

In a similar way the command returns the values as text.

Example:

```
aG0 3
```

Will typically return:

```
610DFF>
```

25.9. U

Name: **Unlock**

Command Parameters: **none**

Typical Use **aU**

The unlock command is required for certain other commands that may change the way the device works. It is a safeguard from accidentally issuing a command, change of address for example.

25.10. M

Name: **Run macro at start up**

Command Parameters: **1 [0 or nothing]**

Typical Use **aM1**

Macro commands are stored at 0xB0 onwards on the EEPROM. This command will set a flag in EEPROM that will be detected by the start up procedure and run the macro.

The macro will be run before the auto Baud detection. Once activated the command will always be run so care should be taken to test the macro (command T) before using this command otherwise a hardware factory reset may be required.

To activate macro at start up issue **aM1**, to turn off macro at start up issue **aM0** or just **aM**. Note the 1 and 0 are text numbers, i.e 0x31 or 0x30

25.11. N

Name: **Non-Inverted output**

Command Parameters: **none**

Typical Use **aN**

Pin 2 on the electrical interface that supplies the output information (Tx line), can be supplied inverted (at reset, start up) or non-inverted. Inverted is used if the device is connected directly to an RS232 PC Com. port and non-inverted is used when the device goes through a converter (BV201, BV101) or is connected to a microcontroller.

At reset the device is always in the inverted mode. To set the device to non-inverted use aN, reset is required to set the device back to inverted mode again.

DC / Stepper Motor Controller

BV4113

This command is similar to Command 4 except this acts on a device individually whereas command 4 will set all of the devices on the same bus to non-inverted.

25.12. P

Name: **Print contents of EEPROM**

Command Parameters: **<start address>**

Typical use **aP10**

This will take the contents of the EEPROM at the given starting address and output the data to Tx (pin 2) as raw data (bytes) unlike the G command that will output the data as text numbers.

The command will stop outputting either when it reaches a value of 0 in the EEPROM or when the end of the EEPROM (255) is reached.

This command is the opposite of the B command, the B command will write text to the EEPROM and this command will read and output it.

The start address of the message location within the EEPROM needs to be specified, e.g. **aPA0**.

25.13. R

Name: **Reset**

Command Parameters: **none**

Typical Use **aR**

Resets and individual device. The baud rate will need establishing again after this command is used.

This is similar to command 3 but works on a single device.

25.14. V

Name: **Version information**

Command Parameters: **none**

Typical Use **aV**

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

25.15. T

Name: **Test macro**

Command Parameters: **none**

Typical Use **aT**

Runs the macro. This is created by the Z command. It is wise to test macros with this command before using the M command.

25.16. Z

Name: **Create macro**

Command Parameters: **see text**

Typical Use **aZac1;at'Fred'**

A macro is created at 0xB0 in the EEPROM space and so if it is used, it is up to the user not to write over it. The whole macro must be specified on one line (maximum number of bytes 63) and ';' (semicolon) are used to separate commands, they are interpreted as EOL when the macro is running. Example

aZaN;aV;aP10;

The above example will change the mode to inverted, print out the version number and print a message stored in the EEPROM at address 0x10. Note that the macro also finishes with a ';'.

26. Error Codes

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

Code	Description
2	Unknown command, the command issued is not in the command table for this device.
3	Bad device address, the address specified is outside the address range.
4	Bad number usually caused by specifying a hex number (say D0) when a decimal number is required.
5	No terminating quote, for example: aB10 'Hello would give this error.
6	Command locked, the command used should be unlocked with the U command before using.

27. Connecting and Configuration

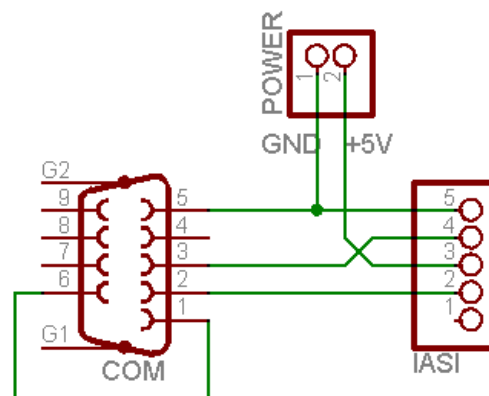


Figure 9 Connection wiring

The above wiring diagram shows the connections to a standard PC 9 way com port (RS232 connector). Pin 1 of the IASI-2 has no connection as this is used to connect to a microcontroller UART.

DC / Stepper Motor Controller

BV4113

The factory defaults will work with the above configuration.

Start HyperTerminal or some other terminal software, BV Terminal is ideal and can be obtained from www.byvac.com The following settings should be used:

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

(The Baud rate can be one of the selected rates, see earlier)

Power up the device and press 'return' a few times. The device should now be listening. Press **CTRL-A**, this will send command 1 to the device, the device should respond with **a>**.

At this point if you are going to use multiple devices than this is where you would set the address. To set the address to 'b' for example the following is required:

aU

aAb

The first command unlocks and the second command sets the device address to 'b'. This can be verified by issuing **bV**, the firmware version should be returned.

27.1. Start Up

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.

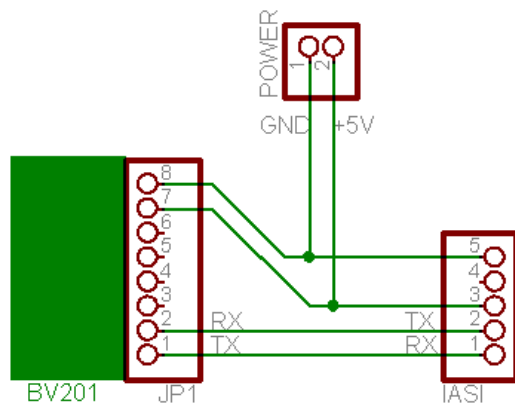


Figure 10 Using non-inverted

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

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

IASI-2 ALWAYS starts with the output (Tx pin 2) set to inverted mode. If the above connections are used then the device or devices need to be changed so that the output is non-inverted. This is easily achieved by issuing command 4. Once this command is issued all output is then non-inverted.

Note that the device will revert back to non-inverted if reset or powered off and on again. This provides a consistent and easy to use interface without the additional complication of configuring the device.

28. Microcontroller Use

The output from a microcontroller UART is non-inverted, the Tx pin of the microcontroller will go to the Rx pin 1 of the IASI-2 device.

The start up code could consist of the following:

- 1) Set the Baud rate of the UART to match one of the rated for IASI-2.
- 2) Send CR (#13) 3 times: this will establish the Baud rate for any connected device.
- 3) Issue command 4: this will make all the devices use non-inverted output from now on.
- 4) Issue command 1: the devices on the bus will respond and reply in non-inverted mode through pin 2.

All of the devices are now ready to be used in the non-inverted mode. Each time the device is reset, either command N or 4 should be used to set the output to non-inverted.

28.1. Multiple Devices

In both modes, inverted and non-inverted many devices can be connected together and all will receive the correct input.

The output however on pin 2 is connected to an open collector and this must have ONE resistor to complete the circuit. This resistor is on each device by default and is permanently connected via a PCB shorting track.

To connect more than one device ideally only one resistor (one track shorted) should be used, the other tracks cut to accommodate this. In practice however several devices can be connected without any ill effects.

IASI - 1			
Jumper x			
3	5	1	2

IASI -2			
Jumper			
3	5	1	2

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

DC / Stepper Motor Controller

BV4113

The above illustrates this principle where only one jumper is connected.

A side effect of this is that signalling can only be obtained by pulling the output low and so feedback can only be obtained on **multiple devices using the non-inverted mode.**

29. Restoring Factory Defaults

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

Address = 'a' (#97)

CR value = #13

29.1. Software

Issue the command **aZYeS**.

29.2. Hardware

This is likely to be needed if you have accidentally changed the contents of the first 16 bytes of EEPROM.

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.
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.

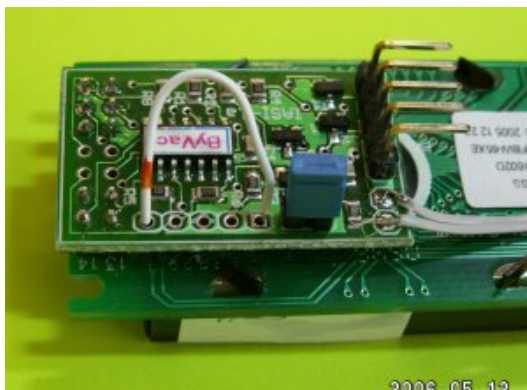


Figure 11 Example Shorting link

DC / Stepper Motor Controller

BV4113
