Serial, I2C LED Display BV4513

BV4513

Serial LED Display

Product specification October 2009 V0.a

Contents

Serial, I2C LED Display BV4513

Rev	Change
October 2009	Preliminary
Feb 2009	Firmware v0.c, added hex to number command and corrected segment display on I2C
Oct 2010	Changed picture of rear following PCB update
Jan 2011	Limited brightness to 25
April 2011	Typing errors in I2C section
March 2012	Brightness limit increased to 250

1. Introduction

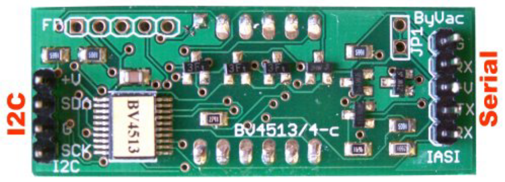
This is a 7 segment 4 digit display that had a serial AND an I2C interface for maximum flexibility.

The product is very simple to use and reduces the number of ports required form the host, the multiplexing and timing are all carried out on board so the user just has to supply the data.

2. Features

- Serial Interface
- I2C interface
- Multiple devices can share the same data lines on the serial and the I2C interface
- Adjustable brightness in software
- Clear easy to see display
- Stackable
- 51 x 21 x 20mm including pins
- 2.5mA no displays on
- 10mA average all on

3. Electrical Specification



The back of the display has two connectors, and ASI (Serial) connector and an I2C connector. By default the display is set to serial, when an I2C bus is applied to the I2C connector then the display will become an I2C display. See the I2C section.

The pin designations are marked on the PCB.

4. Factory Reset

The factory reset will restore the EEPROM to its original values and thus the default device addresses. The two holes mentioned in section 24 for this device are holes 1 and 2 of the pads marked FR above the I2C connector. Hole 1 is the square pad and hole 2 is the next one to it.

5. Serial (ASI)

The serial interface is set by leaving the I2C connector, unconnected at start up. The first byte to be sent through the RX line should be a CR (value 13). This will establish a Baud rate, when this happens the display will light up with the first three LED digits displaying 'ASI'

The Baud rate is selected from:

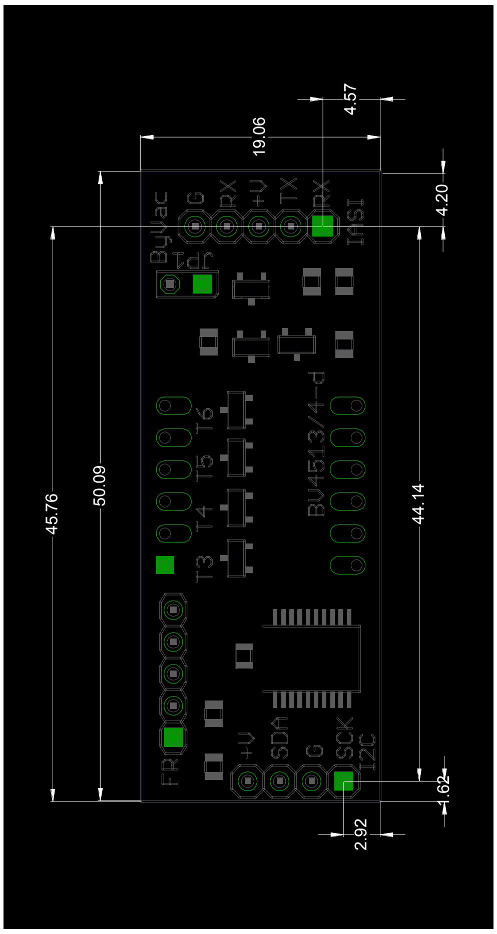
9600,14400,19200,38400

Only those four Baud rates are acceptable and so the host must be set to one of those.

The display is accessed by commands, preceded by an address. **The default address for this device is b** To illuminate digit 0 with 3 for example the command 'bn0 3' is sent

For detailed information about the ASI command set and interface see section 7. The command set that applies to this device follows.

6. Physical layout



See the web site www.doc.byvac.com for a larger drawing

7. IASI2 Command set

*NOTE there are two distinct command sets. The system command set, commands normally using upper case and the device command set. The communication command set is described in the introduction to IASI-2 (Intelligent Asynchronous Serial Interface) section **10**.*

*** Commands are case sensitive ***

Device default address is 'b'

Command	Device Command Set
b	Brightness
c	Clear (blank) display
s	Send byte to digit
n	send number to digit
p	Decimal point

Table 1 Device Command Set

Table 1 is a command summary of all of the device specific commands. Note that these are all in lower case as the IASI2 Protocol is case sensitive.

There are also another set of system commands that are in upper case, see section 19 for details of these.

Note that values given are in ASCII and so the command 'bb10' will in fact be:

98,98,49,48

7.1. b

Name: **Brightness**

Command Parameters: **<n>(0-25)**

Typical Use **bb10**

Sets the brightness of the display, 0 is dim and 250 is bright. Any value greater than 250 will default to 250.

NOTE: Depending on the display a high brightness will reduce the life of the display.

7.2. c

Name: **Clear display**

Command Parameters: **none**

Typical Use **bc**

Clears (blanks) display.

7.3. s

Name: **Send byte to digit**

Command Parameters: **<digit><byte>**

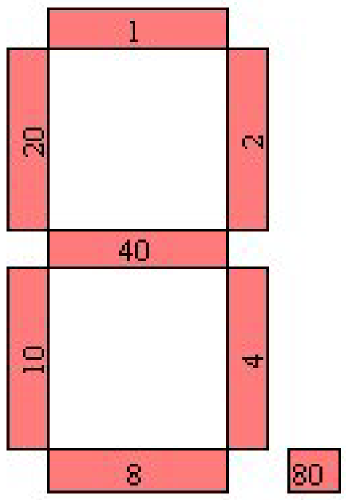
Typical Use **bs0 6**

This will place the byte at the given digit. Note there must be a space between the digit number and the byte value.

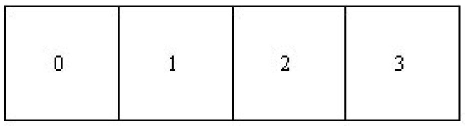
Digit is in the range 0-3

Byte is in the range 0-0xff

The values are in HEX



The byte vales are shown above and can be added together, so for example to display the number 3: 1+2+40+4+8 is added, these are HEX values and so the result is 0x4f. To display '3' onto the first digit the command 'bs0 4f' is used.



The digits are numbered from left to right as shown.

7.4. n

Name: **Send number to digit**

Command Parameters: **<digit><number>**

Typical Use **bn0 6**

There is a built in look up table and this command uses that in order to illuminate the correct segment to represent a number.

0-15 will display the corresponding number on the requested digit as follows:

0,1,2,3,4,5,6,7,8,9,A,b,C,d,E,F

16 will blank the digit.

7.5. p

Name: **Decimal point**

Command Parameters: **<digit><0 or 1>**

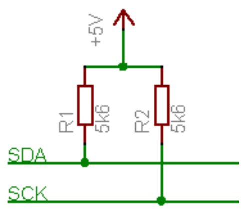
Typical Use **bp1 1**

The decimal points are treated separately from the other segments this is to enable a digit to be displayed with or without the decimal point. To illuminate the decimal point on the first digit 'bp1 1' is used to extinguish it 'bp1 0' is used.

8. I2C

This display has two interface options. To use the I2C option the display musts be connected to an I2C bus before power up. The pull up resistors on the I2C bus are detected and this will put the display in I2C mode. This is verified by the display showing '12C'.

The display shows '12C' when in this mode.



It is essential that the i2c interface is properly terminated otherwise the display will revert to the serial mode. This is only checked at start up.

When in I2C mode the **default 8 bit address is 0x62 the 7 bit address is 0x31**. For an explanation of 7 and 8 bit addresses see http://doc.byvac.com/index.php5?title=I2C_Troubleshooting under the Addressing section.

A full explanation of the system commands can be found in section 25.

9. I2C Command set

The format used by this device consists of a command, this is a number, followed by other bytes depending on that command.

There are two type of command, those referring to the device and those referring to the system. The system commands enable changing of the device address etc.

Device default address is 0x62

Command	I2C Command Set
1	Brightness
2	Clear (blank) display
3	Send byte to digit
4	send number to digit
5	Decimal point

The method of writing to the device using the I2C protocol follows a consistent format, typically:

<S-Addr><Command><data..><Stop>

Where S-Addr is the start condition followed by the device address (0x62). Command is one of the commands given in the table. Data is one or more bytes and Stop is the stop condition.

Reading data requires a restart and this will be in the format:

<S-Addr><Command><R-Addr><data..><Stop>

The restart address will be one greater then the start address, thus if the start address is 0x62, the restart address will be 0x63. Again the data can be one or more bytes read from the device.

9.1. Command 1

Name: **Brightness**

Format:<**S-addr**><**1**><**0 to 25**><**Stop>**

Sets the brightness of the display, 0 is dim and 25 is bright. Any value greater than 25 will default to 25.

9.2. Command 2

Name: **Clear display**

Format:<**S-addr**><**2**><**Stop>**

Clears or blanks the display

9.3. Command 3

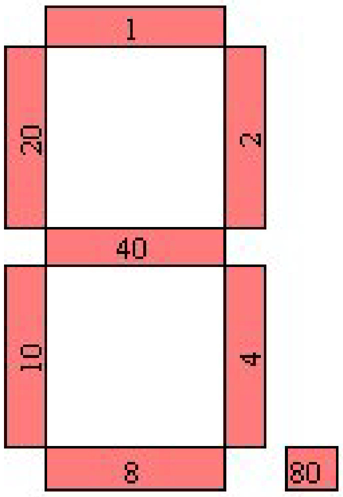
Name: **Send Byte to digit**

Format:<**S-addr**><**3**><**digit**><**byte**><**Stop>**

This will place the byte at the given digit.

Digit is in the range 0-3

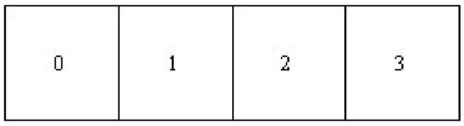
Byte is in the range 0-0xff



The byte vales are shown above and can be added together, so for example to display the number 3: 1+2+40+4+8 is added, these are HEX values and so the result is 0x4f. To display '3' onto the first digit the command

s 3 0 4f p

is used, where s is the start condition followed by the address and p is the stop condition



The digits are numbered from left to right as shown.

9.4. Command 4

Name: **Send number to digit**

Format:<**S-addr**><**4**><**digit**><**byte**><**Stop>**

There is a built in look up table and this command uses that in order to illuminate the correct segment to represent a number.

0-15 will display the corresponding number on the requested digit as follows:

0,1,2,3,4,5,6,7,8,9,A,b,C,d,E,F

16 will blank the digit.

9.5. Command 5

Name: **Decimal point off / off**

Format:<**S-addr**><**5**><**digit**><**0or1**><**Stop>**

The decimal points are treated separately from the other segments this is to enable a digit to be displayed with or without the decimal point. 1 will illuminate the DP 0 will turn it off.

10. Revisions to the IASI-2 Section

Rev	Change
Oct 2008	Preliminary
July	Added new command H that returns the device number

11. 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.

12. 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 microcontroller pin or UART and pin 4 will accept positive an 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.

13. 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.

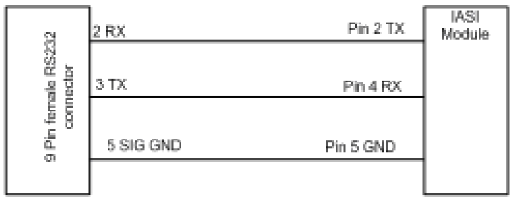


Figure 1 Connection to a PC

Figure shows the connections to a 9 pin D type

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 23.1)
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 E2 Serial Connection Details

connector found on most PC's.

14. 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 then one CR just in case. Once the Baud rate has been established the interface is ready to receive commands.

15. 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.

16. 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 commend is required as in the above example, it would probably be sent as text:

Send("aj4");

17. 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.

18. 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

19. 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.

19.1. 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>

19.2. 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.

19.3. 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:

```
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.

20. 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.

20.1. 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
H	Returns device number
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

20.2. 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 EERPOM – 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.

20.3. 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.

20.4. 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.

20.5. 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';

20.6. 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.

20.7. 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'.

20.8. 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>

20.9. H

Name: **Returns device number**

Command Parameters: **none**

Typical Use **aH**

Returns a number representing the device product number.

20.10. 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 form accidentally issuing a command, change of address for example.

20.11. 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

20.12. 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.

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.

20.13. 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**.

20.14. 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.

20.15. V

Name: **Version information**

Command Parameters: **none**

Typical Use **aV**

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

20.16. 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.

20.17. 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 ';' .

21. 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.

22. Connecting and Configuration

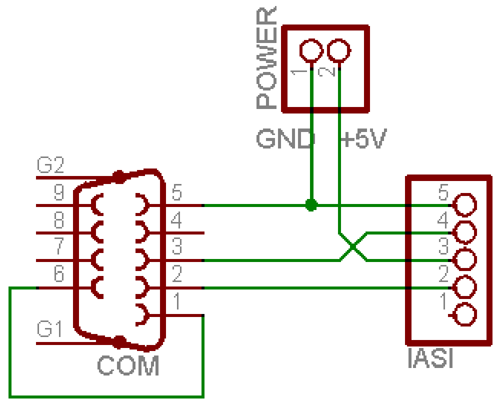


Figure 2 Connection wiring

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.

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.

22.1. 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.

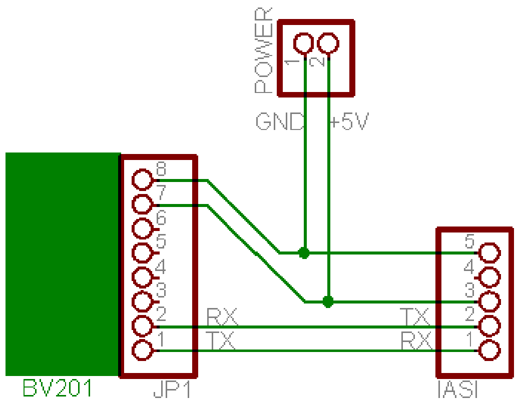


Figure 3 Using non-inverted

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 out put is then non-inverted.

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

23. 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.

23.1. 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

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**.

24. Restoring Factory Defaults

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

Address = 'a' (#97)

CR value = #13

24.1. Software

Issue the command **aZYeS**.

24.2. 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.

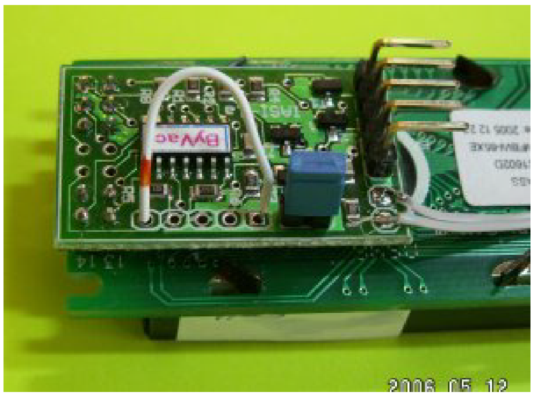


Figure 4 Example Shorting link

Rev	System Section Changes
Oct 2008	Preliminary
Dec 2008	Removed command 96
Aug 2008	Added command 0xa1 (not applicable to all devices)

25. I2C System Commands

The following section deals with system commands that are common to all I2C devices. Note that not all of theses commands are available for all devices.

Command	System Command Set
0x55	Test
0x90	Read EEPROM
0x91	Write EEPROM
0x93	End of EEPROM
0x94	Sleep
0x95	Reset
0x98	Change Device Address Temporary
0x99	Change Device Address Permanent
0xA0	Firmware Version
0xA1	Returns device ID

Most but not all devices contain an EEPROM that can store data when the power is off. The first 16 bytes of the memory is reserved for system use and should not be changed by using these commands.

If the contents of the first 16 bytes are changed then, depending on the device unpredictable results may occur. A factory reset will put the contents back to normal. In some devices not all 16 bytes are used.

The rest of the EEPROM can be used by the user for any purpose.

25.1. 0x55

Name: **Test**

Format: < **S-addr**><**0x55**><**start**><**R-Addr**><**Value..**><**NACK**>**<Stop>**

BV4221 Example

```
0x42>s 55 r g-3 p
```

The above command will return 1,2,3 if the device is connected and working correctly.

This command simply returns an incrementing value until NACK is sent by the master prior to stop. This can be useful for testing the interface.

It can be used for testing the presence or other wise of a device at a particular address. It is also useful during the development stage to ensure that the I2C is working for that device.

25.2. Command 0x90

Name: **Read EEPROM**

Format: <**S-addr**><**0x90**><**EE-Address**><**R-Addr**><**data…**>**<Stop>**

BV4221 Example

```
0x42>s 90 0 r g-3 p
```

The above will fetch 3 bytes from the EERPOM addresses 0, 1 and 2

This command will allow a single or several bytes to be read from a specified EEPROM address.

25.3. Command 0x91

Name: **Write EEPROM**

Format: <**S-addr**><**0x91**><**EE-Address**><**data…**>**<Stop>**

BV4221 Example

```
0x42>s 91 10 1 2 3 p
```

The above write 1,2 and 3 to EEPROM addresses 0x10, 0x11 & 0x12

This command will write one or more, up to a maximum of 30 bytes at any one time, to be written to the EEPROM. Address 0 of the EEPROM is the device address and this cannot be written to by this command. A special command 0x99 is used for this purpose.

The first 16 bytes 0 to 15 are reserved for system use.

25.4. Command 0x93

Name: **End of EEPROM**

Format: <**S-addr**><**0x93**><**R-Addr**><**data>**<**Stop>**

BV4221 Example

```
0x42>s 93 r g-1 p
```

Returns the address of the end of the EEPROM, normally 0xff

The system only uses a small portion of the first part of the EEPROM, the rest of the EEPROM can be used for user data or other purposes depending on the device. This command returns a single byte that will determine the last writeable address of EEPROM, normally 0xFF.

25.5. Command 0x94

Name: **Sleep**

Format: <**S-addr**><**0x94**>**<Stop>**

BV4221 Example

0x42>s 94 p

This will put the IC into sleep mode. Any other command will wake the IC. Depending on the device this can be a considerable power saving.

25.6. Command 0x95

Name: **Reset**

Format: <**S-addr**><**0x95**>**<Stop>**

BV4221 Example

0x42>s 95 p

Resets the device, this is equivalent to disconnecting and then connecting the power again.

25.7. Command 0x98

Name: **Change Device Address Temporary**

Format: <**S-addr**><**0x98**><**New-Addr**>**<Stop>**

BV4221 Example

0x42>s 98 62 p

Changes the device address to 0x62, the device will revert back to 0x42 at reset.

This will change the device address with immediate effect and so the next command must use the new address. The address must be a write address (even number) Odd numbers will simply be ignored. The effect will last as long as the device is switched on. Resetting the device will restore the address to its original value. The address is stored in EEPROM location 0.

25.8. Command 0x99

Name: **Change Device Address Permanent**

Format: <**S-addr**><**0x99**><**New-Addr**>**<0x55><0xaa><Current-Addr><Stop>**

BV4221 Example

0x42>s 99 62 55 aa 42 p

Permanently changes the device address to 0x62.

This command changes the address immediately (the next command will need to use the new address) and permanently (see hardware reset). The address must be a write address (even number) and follow the sequence exactly.

Permanent in this case means that the device will retain this address after power down, i.e. it is stored in EEPROM. Should anything go wrong the default address can be restored by using a hardware reset.

25.9. Command 0xA0

Name: **Firmware version**

Format: <**S-addr**><**a0**><**R-Addr**><**byte**><**byte**>**<Stop>**

BV4221 Example

0x42>s a0 r g-2 p

This will return the two firmware bytes.

This simply returns two bytes that represents the firmware version.

25.10. Command 0xA1

Name: **Device ID**

Format: <**S-addr**><**a0**><**R-Addr**><**byte**><**byte**>**<Stop>**

BV4221 Example

0x42>s a1 r g-2 p

This returns two bytes that represent the device ID. This is a later addition to the command sent and so may not be available on all devices.

26. Hardware Reset

A hardware reset has been provided should the device address be changed to some unknown value.

The method of restoring the factory defaults and thus the default device address is as follows:

1) Remove power
2) Hold the designated pin low or high or connect two pins together. The actual pins are device dependant and will be referenced in the sections above this text.
3) Apply power
4) Remove power
5) Remove shorting link

When power is now restored the device will have the default I2C address, normally 0x42.

27. Command Diagrams

To further explain the format of the commands this section has been provided. The design of the interface has been purposely kept simple and so there are only a few standard sequences required.

Key

🟩 Master

🟥 Slave

🟩 S Start condition

P Stop Condition

A Acknowledge = 1

N Not acknowledge = 0

27.1. Sending a single command

This is designated in the list as:

<**S-addr**><**cmd**>**<Stop>**

This sequence is used for simple functions where no data is involved. The I2C sequence, using the default address is:

S	0	1	0	0	0	0	1	0	A	Command	A	P

27.2. Sending a command with a parameter byte/s

This is designated in the list as:

<**S-addr**><**cmd**><**data…**>**<Stop>**

Some commands expect a parameter after the command. In this case the bytes are sent one after the other up to the maximum of 31 bytes. The stop command tells the slave that there is a command ready to be executed.

S	0	1	0	0	0	0	1	0	A	Command	A	→

Byte	A	Byte…	A	P

27.3. Receiving bytes from the salve

<**S-addr**><**cmd-Addr**><**byte**>**<Stop>**

When receiving one or a number of bytes from the device a restart is required.

The command is sent with an even address (the R/W bit 0). When the slave acknowledges the command another start condition is sent with the R/W bit set to 1. This is called a restart. After this restart the slave will continue to send bytes until the master sends a not acknowledge (N or NACK).

If the master does not send a NACK at the last byte the slave will be expecting another byte to be requested and this may cause either unpredictable results or the I2C bus to lock.

28. Trouble Shooting

This section has been added to answer frequently asked questions. The problems are usually caused because the master device has not had the I2C specification fully implemented.

28.1. Pulse Stretching

This is a method of I2C handshaking which is used in BV slave devices but it is not always supported by the master system. The symptoms are erratic behaviour, some commands will be accepted an others will not.

To explain: when the slave device is busy it holds the clock line low (normally only the master controls the clock line), the master should check that the clock line is high before sending the start condition. If it is low the master should wait until it is free.

Quite a few slave devices do not use pulse stretching and so this not being implemented in the master does not show up. However when dealing with relatively slow hardware, an LCD display for example (i.e. BV4219), this will become a problem. The work round is to make sure that the master recognises pulse stretching properly or introduce delays after each command.

28.2. Last Read NACK

When optionally multiple reads of a slave is required (the 0x55 command is a good example) the last read should send a NACK rather then a ACK. This informs the slave that no more reads from that command are required.

It has been found that some master implementations do not send a NACK on the last read. This causes the BV slave to remain in the (multi read) command effectively blocking any other commands.

The typical symptoms are that when the 0x55 command is implemented no other commands will work after that.

28.3. Pull Up's

The most common problem when trying to get a new device going is to forget to put the pull up resistors somewhere on the bus. BV Slave devices do not have pull up resistor on board so they must be provided by the master (the BV4221 has pull up's) or provided externally. A value of around 5k is okay but this is not usually critical.

S	0	1	0	0	0	0	1	0	A	Command	A	→
S	0	1	0	0	0	0	1	1	A	Byte	A	→

Byte	A	Byte	N	P