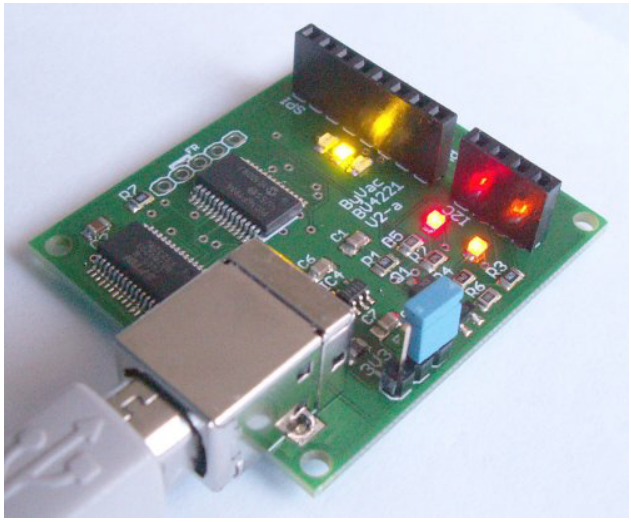
USB to I2C/SPI Terminal BV4221-V2



BV4221-V2

USB to I2C/SPI Terminal

Product specification August 2010 V0.a

©ByVac

Contents

Rev	Change
Aug 2010	Preliminary
Feb 2011	Upgrade information + command 'f' changed to command 'x', new firmware 2.2
Jan 2013	Update: Fixed SPI bug that only retuned a byte, Added commands N and O
Feb 2013	Altered the 'r' command – see the command reference

1. Introduction

The BV4221-V2 is a device for using I2C and SPI compatible equipment via USB. It provides a terminal like interface and is primarily intended for debugging and experimenting with I2C devices although could be used to drive I2C devices from a PC, see the Foundation Guide

In addition to the above there is an 'inspector' mode that allows 'viewing' of the I2C bus which is useful for debugging existing applications.

2. Features

- I2C up to 1MHz [1]
- I2C clock stretch detect
- SPI max speed 100k
- Command Driven
- Automatic Baud rate select
- USB driver easily obtainable
- On board 3v3 and 5V regulators
- Switchable 3v3 or 5V
- 5V or 3v3 out to drive external I2C circuit – short circuit protected (100mA)
- I2C pull up resistors incorporated
- Inspect I2C up to 100kHz bus speed
- LED indication of SDA & SCL lines
- LED indication of CS lines
- Size 45x40x15 high (mm)
- Power (idle) 15mA (powered from USB)

[1] Minimum 32kHz. The maximum can exceed 1MHz if the slave is up to it.

3. Enhancements from V1

The differences between version 1 and 2 are listed here.

- SPI interface added
- I2C address finder added
- Master clock rate selectable
- Inspector mode operates at 100k
- 5V or 3V3 logic switchable
- Two on board voltage regulators

4. Physical Specification

The device consists of a USB connector a 4 pin I2C connector and an 8 pin SPI connector. The USB supplies power to the devices via a 5V or 3V3 regulator. This gives short circuit protection for the USB bus.

4.1. I2C Interface

The I2C output pins have the following designated pins.

Pin	Description
1	SCK – I2C clock
2	GND
3	SDA – I2C data line
4	V+ output to drive external equipment

Table 1 IC Pin Description

One or more external devices can be connected to this bus, the BV4221-V2 acts as a master device that can write to and read from the external devices.

Two LED indicators are provided that give an indication of the SDA & SCL lines. When they are on the lines are high and when off the lines are low. This is useful as it indicates if the bus is free or has become locked up. There are two 5k6 resistors, one connected to the SCL line and the other connected to the SDA line. These are the required pull up resistors for the I2C bus so there is no need to provide them elsewhere.

V+ This is the power source for external equipment (absolute maximum power 200mA) and is selectable between 5V and 3V3 by the on board jumper.

4.2. SPI Interface

Pin	Description
1	MOSI (Master Out Slave In) (OUT)
2	MISO (Master In Slave Out) (IN)
3	SCK (Clock) (OUT)
4	CS1 (OUT)
5	CS2 (OUT)
6	CS3 (OUT)
7	V+
8	GND

The SPI interface is a 4 wire bus (in, out, clock and chip select) and this connector can handle three separate SPI devices because it has three chip select lines.

MOSI Data output from the master and input to the slave, this naming convention has been used to avoid any ambiguity.

MISO Data input to the master which is output from the slave.

SCL This is the clock output from the master that controls the rate of data transfer.

CS1,2,3 Output pins that can be selected either high or low by terminal commands. One of these will go to each SPI device connected to the bus (in,out,clock). When the line is set low then the device to which it is connected will be selected.

There are also three LED indicators that reflect the state of the lines. When CSn is low the LED will be illuminated.

V+ This is the power source for external equipment (absolute maximum power 200mA) and is selectable between 5V and 3V3 by the on board jumper.

5. Terminal Interface

The device should be plugged into a spare USB socket. It will then ask for a USB driver. The chip used for the interface is a FTDI chip (FT232R) and a driver for this can be found here:

http://www.ftdichip.com/

The driver is the **VCP** driver and should be downloaded for your operating system. The Unbutu Linux system already has the driver installed.

When running the USB presents itself as a Com port. HyperTerminal or BV-Terminal (better) can be used to communicate. BV-Terminal is free and can be downloaded from:

www.byvac.com (see the page fro BV4221-V2)

There is also a Foundation Guide that is a comprehensive tutorial/projects for getting to know the I2C and SPI busses and this device better. Included in the zip file is BV-COM.

Alternatively a programming language such as VB or Just Basic can be used for automated applications. The requirement is that it can talk to the COM device.

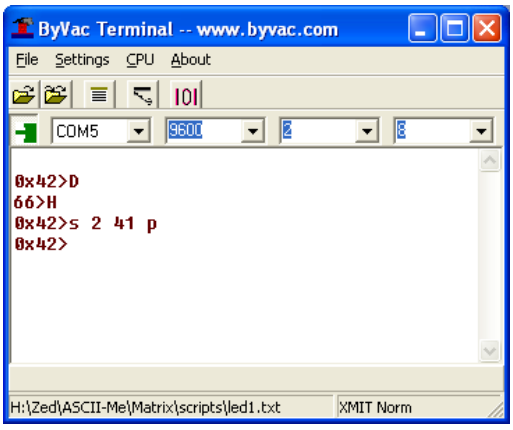


Figure 1 Using BV Terminal

The Baud rate is selected by interrogating the first CR (Carriage Return) character from the following five Baud rates:

- 2,400
- 4,800
- 9,600
- 14,400
- 19,200
- 38,400
- 56,700
- 11,5200

It will select the closest to the above and so the terminal should be set to one of those. There is no handshake or parity and the number of stop bits can be one or two.

When the device is first switched on a CR character (#13) should be sent first to establish the Baud rate, sending any other character will give unpredictable results.

There are two modes of operation, I2C mode (default) and SPI mode. Each are dealt with in the next sections.

6. Commands & Entry

In general upper case commands refer to both modes and lower case commands are specific to each mode.

A command consists of a single upper or lower case letter. It will not be sent to the I2C or SPI device until and EOL (End of Line) byte is received. The default EOL byte is 13 which corresponds to the ASCII Carriage Return (CR). As an example using the I2C mode:

s 27 15 p

The above will stay in the input buffer until, say on a terminal (BV-COM) CR is pressed. This gives the opportunity when interacting directly with the device to correct mistakes before sending to the I2C device. It is also perfectly legitimate to send EOL at any time.

The buffer size is **69 bytes** and so this value must not be exceeded before sending CR

Text: All data sent to the device is in TEXT which includes numbers. So the above example 's 27 15 p' would actually be the following values (in decimal):

115 50 55 49 53 112 13

It is important to understand this when sending bytes from a program. Sending the value 27:

putc(27) // WRONG

puts("27") // correct

Hex entry MUST be in lower case and is NOT preceded by 0x so for example to enter 0xAF the following is entered:

0x42>af

Where 0x42> is the prompt .

7. I2C Mode

This is an enhanced implementation of version 1 and has some subtle differences and so this section should be read carefully if migrating from version 1 to version 2.

By default the device starts up in I2C mode, to explicitly enter this mode use the 'I' command.

It is easy to tell that the device is in I2C mode as once the Baud rate has been established there is a prompt:

nn>

The nn is the current I2C address. This can be changed with the A or s commands. There are two further modes of operation, Decimal and Hex. In Hex mode the 'nn' is shown with a leading 0x. In hex mode all values are assumed to be hex so entering 37 for example would give a decimal value of 55.

At the prompt the BV4221-V2 is ready to send commands to the connected I2C device/s.

Sending I2C information to an I2C device usually consists of a start condition, the start address followed by the information and then a stop condition. Each device is different and so the commands that are sent will depend on the device.

The terminal will send whatever is requested using the command format, this can be done individually (line by line) or it can be done as a long line up to the capacity of the input buffer'

7.1. Address Notation

This text uses 8 bit address notation which means that even numbered addresses are write commands and odd numbered addresses are read command. For further information about this see this link:

http://www.i2c.byvac.com/ar_trouble.php

at the section on addressing.

8. SPI Mode

Entry to the SPI mode is acquired by issuing the 'S' command. The decimal and hex modes still apply and are reflected in the prompt:

Hn> or Dn>

Where H is hex mode and numbers entered will be hex value, for example 10 will be decimal 16 if entered in hex mode. The 'n' following the hex or decimal indicator is the SPI mode which can have a value from 0 to 3 following the SPI convention:

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

The manufacturers data sheet for the SPI part will normally specify which mode to use. If in doubt use mode 0.

The SPI interface specification requires the device to be selected by hardware. This is done using one of the chip select lines and its associated commands. When the chip select line is low the LED associated with that line illuminates as most SPI devices will be selected when the line is low.

Most SPI devices use an 8 bit data length however quite a few don't and this is catered for by being able to set the data length. This in turn has an effect on the values that can be sent and returned. If the data length is less than 9 bits a byte will be returned, the user should then extract the required bits if less then 8.

If the data length is greater than 8 bits a word (16 bits) is returned, again it is down to the user to extract the relevant bits. Although a 16 bits are returned only the exact specified number of bits are retrieved from the SPI bus the rest are padded with zeroes. This ensures that the slave devices specifications are being met.

The interface has two main command for input and output. To output to the SPI interface the 's' command can be used thus:

s 3 6 9

Will send the values 3,6 and 9 to the SPI interface. The values are not sent until CR is pressed or in an automated system byte 13 is sent. In C this would be something like:

puts("s 3 6 9\r");

There must also be exactly ONE space between each number.

Retrieving information from the SPI interface is carried out using the 'g-n' command (get). The 'n' is the number of values to retrieve so

g-7

will return 7 values, the values are separated by commas.

Because the SPI standard is full duplex it is possible for the master to send out a value and receive a value at the SAME time, in which case the above commands would not work. This full duplex is achieved by simply entering a value or values. If a value is entered without a leading command then it will be sent to the interface and the value got back form the slave will be displayed.

9. Restore Defaults

It is possible to set some configuration values that render the device inoperable through the USB port. Setting the Baud rate to one for example.

To restore the factory information, i.e. reset the EEPROM contents to their default values carry out the following procedure:

1) Turn the device off.
2) There is a row of 5 holes with two of the pads marked FR, short out those tow pads.
3) Turn the device back on.
4) Turn the device off
5) Remove the short.

The defaults will be restored

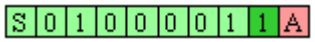
10. Firmware Upgrade

The firmware on this device is upgradable. There is a separate set of upgrade instructions in the resources zip file.

11. I2C Commands

There is a simple command set that controls, reads and writes to the device and the external I2C devices.

A command is a single letter and is **case sensitive**.

Command	Name	Description
s	Start	This will send the start condition AND the default address which is the number shown on the default prompt.
s[-nn]	Start	This will send the start condition AND the address specified after the minus sign, these should be no space between the 's' '-' and address. Example to send a start condition and address 0xa4 (assuming in hex mode) s-a4
r	Restart	Send restart command with default address+1, if the default address (at the prompt) is say 0x42 then this command will send:  The darker '1' is the read bit and the 'A' is the ACK from the slave device. NOTE: As from version 2.5 this will also send the address+1 if it has been specified with the 's' command, for example: 0x42>s-a0 0 r g-1 p This will send o to the I2C address a0 and the r command will use the address a1 to read the device.
g	Get	Gets one byte from slave. A point to not here is that when receiving from the slave the master (this device) will ACK on each byte received. However it will NOT-ACK on the last byte requested to inform the slave that no more bytes will be requested. Some slaves are unaware of this and so this point does not matter, however others rely on it for correct operation.
g[-nn]	Get	Gets multiple bytes from slave, the number of bytes to retrieve up to a maximum of 255 is specified after the '-'. If more than 255 bytes are required then multiple 'g' command can be used. Bytes are separated by a comma thus "g-3" will give an output something like: 01,02,03 Note that this is in text and so the actual bytes received would be (in decimal): 48 49 44 48 50 44 48 51
p	Stop	Send stop condition
nn	Byte	any numbers that do not have a command in them are sent to the salve device, so in the example s 27 p

		the '27' would be sent to the device as a number 27.
x	Find	This will list the address all of the I2C slave devices on the bus separated by a comma. As an example if two devices were attached to the bus address 0xd0 and 0x42 then the output from this command would be: 42,d0 If in hex mode. ** This was the 'f' command in version 2.1
tn	Speed	This is the clock rate that is used for I2C when sending and receiving data. The default value is 100k but can be set anywhere between 32,000 and 2,600,000. Example to set the clock speed to 50k: t50000 Value entered must be in DECIMAL regardless of mode.
i	Inspect I2C data	There is no exit from this mode other than resetting the device. It will passively monitor all traffic on the bus and report the results back. It can operate up to 100k I2C bus speed but is limited by the input buffer size. Data is captured as an I2C slave device would, beginning with a start condition, address, data and then a stop condition. This information is buffered at high speed and is only reported when a stop condition is received. Special characters (all upper case) are used to indicate certain information: S start P stop A address K acknowledge (low from slave) N not acknowledge (high from slave) As an example sending "s 55 r g-2 p" to a ByVac I2C device will result in that device outputting 1 and 2, this is shown in the inspector mode as: SA42K55KSA43K01K02NP Colours have been added to make the explanation more readable, green is the master and pink is the slave. As can be seen from the output start, address and 0x42 have been sent, this was then acknowledges 'K' by the slave. 55 is then sent and this is also acknowledged by the slave. Next is the restart condition which is start followed by the address + 1, i.e. 'SA43', this is also acknowledged by the slave. In response to 'g-2', the slave outputs 01, to which the master acknowledges and then 02. This time the master not-acknowledges to indicate that no more bytes will be requested. The final stop condition is sent by the master which will release the bus.
A	Change Address	Changes the default I2C address, for example if in hex mode: Ad0

		Will change the default I2C address to 0xd0 and this will be reflected at the prompt

Table 2 I2C Command Set

12. SPI Commands

Command	Name	System Command Set
ln	Low	(lower case L) sets one of the three chip select pins to low for example to set chip select 2 pin to low: l2
hn	High	Sets one of the chip select pins to high, for example to set chip select pin 1 to high: h1
tn	Speed	Sets the SPI data speed for transfer. This is a fixed option of speeds as follows: 1 100k 2 75k 3 30k 4 25k The above speeds are approximate.
nn	Data Length	Sets the data length form 1 to 16 (default is 8). This value will determine how many bits are transferred on each request. To set the bit length to 13: n13
p	MISO	Gets the value on the MISO (Master in Slave out) pin. This is sometimes used by slave devices to indicate an acknowledge.
s	Send	Send one or more values to the MOSI pin, values can range from 0 to 0xffff and will be sent according to the length set with the 'n' command. Values must be separated by a SINGLE space. Assuming an 8 bit length, to send 3,0,6 the following would be required: s 3 0 6 The values will be stored in a buffer and sent when the end of line character (default 13 [CR]) is sent. This allows for mistakes to be corrected when used in conjunction with a terminal (BV-COM).
g-n	Get values	Gets a value/s from the slave, the values will be determined by the data length. If in hex mode and the data length is less then 9 the output will be in byte format occupying two digits thus 00. If the data length is greater then 8 then the data will be in word format occupying 4 digits thus 0000. In decimal mode the output is as long as it needs to be. The values are separated by commas. As an example this would be a typical output: 02,21,a4,5b

mn	Mode	SPI can have any one of 4 standard modes, the default is 0. See text for more information.
<>	Value	Any number value that is not a command will be sent to the SPI interface and a number will be presented that has been returned from the slave.

13. System Commands

Command	Name	System Command Set
H	Hex	Set Hex mode
D	Decimal	Set Decimal mode
P1[0]	Prompt on	The prompt can be turned on or off, it is on by default. This may be useful in automatic systems where it may get in the way. To turn the prompt on specify P1 (P one), anything else e.g. P0 (P zero) will turn the prompt off.
Kn	ACK	When Prompt is on this will have no effect, but when prompt is switched off a character is sent by the device to indicate that a command or set of commands have been completed. This can be used by the host to send commands only when the device is ready. The default character sent is '*' (0x2a). To set another character for example CR, send the ASCII code of that character following the 'K': K0d (assuming hex mode) To turn off the ACK mechanism use 0 (zero) e.g. K0
En	End of line	The end of line by default is #13 (CR), this in fact marks the end of the packet. This command will change that to another value. So for example this default sequence: V <enter> would print out the version number, if E1 is entered then V ^A (where ^A is control A) would print out the version number. Note that the new end of line should be used immediately after entering it. This can be fixed using the F command BUT if Auto Baud is used this still requires #13 to establish the Baud rate.
V		Show firmware version
B	Baud rate	Sets baud rate, this must be in decimal regardless of mode. The default is 0 (zero) which is the automatic Baud rate detect. If this is set to any other value that rate will be permanently set. This will now be the Baud rate at start up so be careful when using this command. If a Baud rate has been set that is unreachable this will render the device unusable and a hardware factory reset must be

		performed. See text on how to do this. Example: setting Baud rate to 9600 B9600 Example setting Auto Baud rate B0 NOTE The device may need resetting after this command.
R	Reset	Resets, just as switch on.
I	I2C mode	Sets the I2C mode commands will now respond to this main command table and the I2C command table.
S	SPI mode	Sets the SPI mode, commands will respond to this table and the SPI table
F	Fix	Writes current value semi-permanently to EEPROM
Ln	Delay	Delay operation for n number of milliseconds 1-255
N	Input CS	Inputs a value from the CS lies as a single byte. The CS to byte relationship is the lower 3 bits: Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 CS3 CS2 CS1 As an example if CS2 was high and the other CS lines were low then the output would be 2, if all three CS lines were high then the o/p would be 7
O	Set CS	This will set the CS lines to either input or output. At reset the CS lines are now set to I/P, if SPI mode is selected by using the 'S' command then all three lines will be switched to output. To set the CS lines to either i/p or o/p a byte is sent that has the same relationship as the 'N' command. A 1 will set the CS line to be an i/p and a 0 to be an output. As an example if CS2 is input and the other CS lines are o/p then O5 would do this. To set all lines to be an o/p then O0 and all lines to be an i/p is O7

14. Examples

14.1. Using a 24LC256 EEPROM

Write 1 to 5 to EEPROM address 0. The EEPROM has a device address of 0xA0

H (set hex mode)

Aa0 (set device address)

s 0 0 1 2 3 4 5 p (write 1-5)

Read from those locations

s 0 0 r g-2 p

In the above read operation the EEPROM address counter has been reset back to 0000 and a restart command issued to read the slave. g-2 reads the first 2 bytes which will return 1 and 2.

s-a1 g-2 p

This next operation sets the address to read (odd number) and reads the next two bytes that return 3 & 4.

14.2. Using BV4218 LCD & Keypad Board

Assuming the default address of 0x42 is used:

s-42 1 1 p

This will clear the display

s 2 41 42 43 p (no need to set address again)

This writes ABC to the display

s 90 28 r g-5 p

This gets the first 5 bytes of the sign on message

15. Error Codes

The following error codes may be seen:

1. End of input buffer reached
2. Unknown command (sometimes displays as 3)
3. Number expected
4. I2C bus not free on master start condition, caused by slave not releasing bus.

There is also the LED indicators for the I2C SDA and SCL lines. When the bus is free these should both be high, illuminated. The master will release the bus on issuing a stop command ‘p’. If both lights are not on after this command then one or more slave devices are holding the bus. Sometimes issuing a start ‘s’ and then a stop ‘p’ can clear a slave device.

16. Control by Software

The BV4221 can be controlled by software hosted on a PC, this will enable simple hardware to perform complex applications.

The website www.pin1.org contains some example code written in Just Basic (JB). This language was chosen because it is extremely simple, has a good serial interface and is FREE.

Other languages of course can be used provided they can talk to the COMM port. Here are some hint's and tips that will be needed if a programming interface is used.

16.1. End of Line

The serial communication works by receiving any and all characters up to and including the CR (Carriage Return 13) OR LF (Line Feed 10). When a CR or LF is received the whole line is processed and the instructions within that line are passed to the I2C bus.

It is important that only CR is sent and not a combination of CRLF or LFCR as this will slow down the interface and delays may need to be added.

This is often carried out in languages using an escape character such as:

Print("some commands \r");

The \r or whatever it may be tells the output only to append ASCII 13, i.e. CR

In JB this is handled explicitly by a print routine that suppressed the normal CRLF behaviour and replaces it with a CR.

16.2. Waiting

The prompt **0x42>** will only appear once all of the commands have been processed so it is important to wait for this before sending the next command. The examples given in JB wait the '**>**' character.

17. Inspector Mode

Command	Bus Debug
I	Inspect I2C data interactive
i	Inspect I2C data buffered

The BV4221 is also capable of 'spying' on the I2C bus and displaying what it sees. Because of the limitation of the display, the maximum speed is 50k but this is useful enough for debugging purposes.

There are two options; the first option will display anything on the I2c bus immediately and the second option will buffer any received data until a stop condition is received whereby it will display what has been captured.

This second option allows faster capturing as the device does not have to display each event.

Both modes, monitor the I2C bus and report on the results. Pressing any key on the console will break out of the inspector mode.

When displaying the results the following key is used:

S Start condition

P Stop condition

A ACK

N NACK

To give an example, writing values 1,2 and 3 to the beginning of a serial EEPROM would be this command:

s 0 0 1 2 3 p

(The address is set to 0xA0)

The inspector would display:

SA0A00A00A01A02AP

Note, this command would need to be sent by another I2C master device for the inspector to see it. From the output it can be seen that the sequence begins with the Start condition (S) followed by the address (A0) followed by and ACK (A), etc. up until the stop command (P)

Reading back from the EEPROM:

s 0 0 r g-2 p

Produces this from the inspector:

SA0A00A00A**SA1A**01A02**N**P

There are some things to note about this. The 'r' command produces this SA1A, highlighted in bold above. The restart is S followed by the address plus 1 (A1) and the last A is the acknowledge from the slave. Note also that the last byte received is followed by a NACK prior to the stop command.

17.1. Limitations

I2C Analyser

The inspector mode will not work for a busy I2C, it is not intended as an I2C analyser and so don't expect to be able to connect it to a fully operational system and inspect packet after packet at high speed.

The inspector mode is intended for simple debugging and even learning about how I2C works. It is extremely useful for getting a system to work particularly with unfamiliar devices.

Buffer

The buffer when used in the 'I' mode is limited to about 90 bytes. There is no warning or checking for overflow. If more than 90 bytes are received before a valid stop condition than this is likely to crash the BV4221 so that it will need restarting.