BV4543

I2C or Serial 16x2 with Keypad interface

Date	Firmware	Revision
February 2018		Preliminary
11 Feb. 2018	1.1.1	Updated how serial works
16 Feb. 2018	1.1.3	Sleep updated

Introduction

This is an I2C or Serial user interface for use with microcontrollers, for example the Arduino, Raspberry Pi etc..

The output is in the form of an LCD display and the input is a membrane type keypad or any other switches.

Full I/O control can be realised with only 2 wires. The keypad has an optional 79 byte buffer relieving the host microcontroller of a considerable burden.

Description

The device consists of a PCB with a COG LCD display mounted to the front. There is a 10 way interface to attached a membrane keypad or any other form of cross point switch.

The device is access either by I2C OR Serial depending on the device supplied.

BV4541 Serial

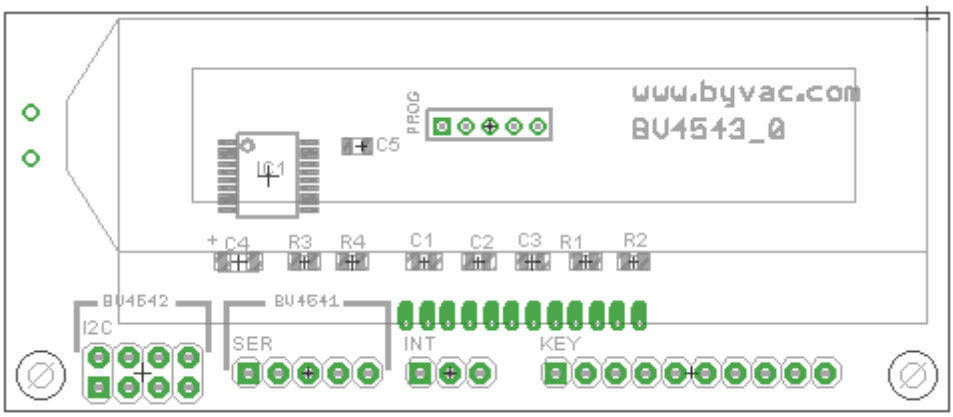
BV4542 I2C

For ease of use the keypad can buffer keys so they can be read at a later time by the host microcontroller. Writing to the device will go directly to the display and reading from the device will read the keypad / buffer.

Features

- I2C display 16x1 AND 16x2, mode selectable
- User selectable I2C address (BV4542)
- Software adjustable contrast
- Software variable back light, variable brightness
- 79 key buffer
  25 way keypad interface
- Interrupt pins
- Wide voltage 3.3V to 5V
- 5V: 10mA BL on, 1.5mA BL off, 280uA sleep
  3V3: 4.71mA BL on, 1.3mA BL off, 170uA sleep
- Only 2 wires for full I/O control
- Wake from sleep by keypad

Physical Description



(PCB Version 0)

The PCB has 4 connection options, all the connector pads are in place but not all of them may be active for the particular device.

I2C Applicable only to the BV4532

Serial Applicable only to the BV4531

Int Output for keypad monitoring

Key Row and column interface for keypad

Pin	**Description**
R1 to R5	Connect to rows on keypad
C1 to C5	Connect to columns on keypad

Keypad Connector

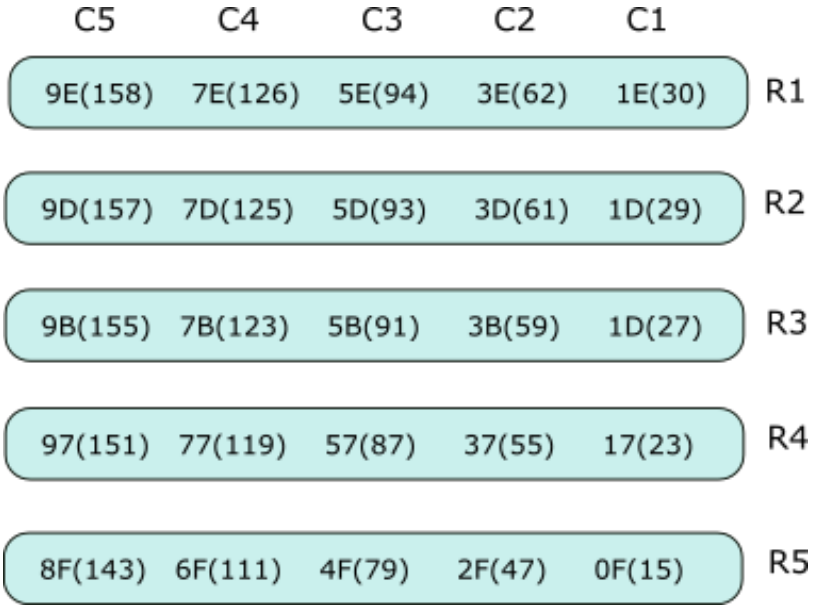
The keypad connector has 5 rows and 5 columns, these can be connected to a membrane keypad or any other type of switch. The interface allows for a maximum of 25 keys but fewer keys (switches) can be used if required.

The image shows the expected scan codes in hex and (decimal). As an example connecting R3 to C4 would produce a key code of (123). By default keeping the pins shorted will produce a key repeat of about 1 second intervals, this can be changed in the EEPROM.

These is also a command that allows direct reading, bypassing the keyboard buffer. This is useful for volume control type commands where the user keeps a key down to increase or decrease a value.

It should follow from this description that any row or column can be connected together to form an input, so if only one switch is available then it can be connected to R1 and C1 that will produce an output of (30) when switched on.

IMPORTANT: The Row/Col matrix will only accept one connection at a time.



Key Buffer

There is a 79 key, key buffer to store pressed keys. It is a circular buffer for maximum flexibility. It is up to the user to ensure that the buffer does not become full as this will overwrite previous keys.

The output from the keys will be a key code (1 byte) that can be decoded by the host. The actual code values will depend on how the keypad is connected up.

Pin	**Description**
Ky (output)	This normally high pin goes low when a key is being pressed and returns high when not being

	pressed. It can be used as an interrupt or connected to a beeper.
In (output)	This pin is normally high but goes low when there is a key in the keypad buffer. It will remain low until all of the keys have been read out.
Rs	No connection (later versions of the PCB have 2 pads)

Interrupts

The connections here are optional as the main interface (I2C or Serial) are capable of determining the keypad status, however they are available for particular applications where polling is undesirable.

LCD

The LCD is a chip on glass (COG) type.

The LCD uses the ST7032i controller. This is almost identical to the very familiar HD44780 controller in that commands and data can be written to it. It also has more advanced features such as double height characters and software controlled contrast. The character set itself is more comprehensive.

Access to the LCD is via I2C commands or serial escape codes, depending on the device. The commands will enable any manipulation of the LCD to be carried out.

Sleep

There are two modes of sleep, deep sleep 0 and light sleep 1. In deep sleep mode the LCD is switched off and this gives the greatest amount of saving. In light sleep mode the LCD remains switched on and so the contents can still be seen. Depending on the supplied voltage the difference is:

Deep sleep about 100uA

Light sleep about 180uA

The above is with a 3.3V supply and the back light off.

Wake up from sleep is achieved by input on the keypad, serial or I2C. The serial interface takes slightly more current in light sleep than does the I2C.

Wake up from deep sleep will cause a reset

Wake up from light sleep will continue from after sleep occurred.

I2C: There will be a delay in responding and so depending on the master device, the first read or write may result in a timeout,

some experimentation will be required.

Serial: Depending on the host some experimentation may be needed but usually serial will work immediately after light sleep.

EEPROM Locations

The values in the EERPOM effect the operation of the device and are read on start up, some values are stored in RAM and so can be temporarily changed and some commands change the EEPROM values.

Adr	**Default**	**Description**
0	0x55	System Use
1	68	I2C address (only relevant for I2C)
2	17	Trigger (only relevant for I2C)
4	38	LCD Contrast
5	0	LCD mode 1 is single line
6	60	Back light value
8	4	Baud rate (applies to serial only)
10	0 Ser 1 I2C	Buffered keypad (0 is not)
11	15	Key repeat
12	4	Beep time – duration of beep when key pressed (assuming a beeper is connected to the key pin)
14	68	I2C address copy (only relevant for I2C)
16		Sign on message starts here, ends with 0xFF
240	68	I2C address copy (only relevant for I2C)

EEPROM Locations and default values

The user is free to use any locations that are not occupied by the system but for future use it is best to avoid locations below 32.

Most EEPROM values are only read on start up so when changing values they may not take effect until the device is reset.

I2C Address

This is only applicable to I2C devices, it is stored in 3 places for security, should one location become corrupt then it will be reset to the other two values. There is a command to change the I2C address but it can also be done by directly writing to the EEPROM, if so than at least two locations need changing.

Trigger

Is used for general call (address 0). This device will respond to a general call but expects the next byte to be this trigger value.

Contrast

This is only read at start up and is the value that the LCD contrast is set to for start up, it can be changed at any time after that using the appropriate command.

Mode

This LCD can display 2 lines x 16 (mode 0) or 1 line by 16 but double height (mode 1)

Back Light

This is the back light brightness at start up and can be set between 0 and 130 with 130 being maximum brightness.

Values greater then 130 will make no difference.

Baud Rate

Applies only to serial devices. This is a code between 1 and 8 as follows:

1 1200 Baud
2 2400 Baud
3 4800 Baud
4 9600 Baud (Default)
5 19200 Baud
6 38400 Baud
7 57500 Baud
8 115200 Baud

Buffered Keys

Serial

If this is set to 0 then any key presses will immediately be shown at the TX output, no internal buffering will take place. If this is set to 1 then the keypad input will go to the internal buffer and to read the keys a command is required. (default 0)

I2C

As this interface needs to initiate communication from the master buffering is normally required, an exception to this is when using the scan command (default 1)

Key Repeat

When a key is pressed a key will go to TX or the buffer depending on the buffer flag. If the key is held down then another key of the same value will be sent to the buffer after a time specified

in this EEPROM location. The time is approximately 16mS x value in this location. The default of 15 therefore gives about 240mS delay.

Sign on message

The sign on message is read beginning at this location by simply transferring the contents of the EEPROM until 0xFF is encountered.

BV4542 (I2C)

This has the I2C interface.

I2C Interface

Pin	**Description**
SCL (*2)	Clock
GND (*2)	Ground
SDA (*2)	Data
V+ (*2)	Supply voltage for device *

I2C Electrical Connection

The I2C connector has duplicate pins, this is to allow for daisy chaining of I2C devices. Please remember that unless buffered the length of the cable should be limited.

*__Voltage__: The device can work with 5V or 3.3V, there are NO pull up resistors on the device as these are normally provided by the master device. The device can be supplied with 5V and still work with 3.3V without and detrimental effect as the pull up resistors take care of the 'high' value of the voltage.

The device has a standard I2C interface and will act as a slave device.

0x44 (0x22 7 bit) Keypad & LCD address

Writing to this address will appear on the LCD screen and reading from this address will retrieve keys from the keyboard buffer.

Commands are realised through a general call address 0. This address cannot be changed but the read write address (0x44 default) can be set to any address via the EEPROM

Address

Device I2C address is stored in EEPROM in three places (see eeprom locations). This address should be set to between 10 and 250 using EVEN numbers ONLY. It is the 8 bit address value so an address of 68 will be on some systems the read and write address 34.

There is a command however that will change all 3 addresses at

once.

Commands

In order to get information such as how many bytes there are in the keypad buffer, or to clear the buffer a general call is used, followed by a trigger byte.

The general call is address 0 and a command sequence is 0 + trigger followed by the command and any other data. For commands that return information a normal read is used after the command sequence.

The default trigger is 17

Command	Range	Notes
<n/a>	N/a	**EEPROM reset** This will restore the default EEPROM settings which has the I2C address. No effect will take place until the device is reset. This can take a few ms and so some master I2C devices will need to take this into account if a timeout is to be avoided. This is a special command in that it can be called regardless of the state of the EEPROM, in other words it does not rely on the I2C address or trigger value, so if the EEPROM values are random then this will restore as factory defaults. Example i2c.write(0,0x55)
1	N/a	**Clear keypad buffer** Example i2c.write(0,trigger,1)
2	0-79	**Gets number of keys in buffer** Example i2c.write(0,trigger,2) i2c.read(1)
5	N/a	**Gets scan code and clears key buffer** This will only return a value when the key is being pressed otherwise it returns zero i2c.write(0,trigger,5) i2c.read(1)
Use read		**Get key value from buffer** Use the read address, default (0x7a or 0x3d) 0 is returned if no keys are in the buffer, returns a scan code i2c.read(n)
30		**Reset LCD** Examplesi2c.write(0,trigger,30)
31		**Send LCD commands** See LCD controller data sheet for the available commands to send. For example sending 1 will clear the screen. Example (clear screen)

		i2c.write(0,trigger,31,1)
35		**Display stored sign on message** For debugging to see if the new message displays okay Example i2c.write(0,trigger,35)
36	0 - 130	**Back light level** Sets back light level Example i2c.writei2c.write(0,trigger,36,50) // set bl to 50
37	0 - 60	**Set contrast** Sets LCD contrast level, the effect of this will depend on the voltage input Example i2c.writei2c(0,trigger,27,20) set to 20
38	0-1	**Set mode** 0 is 16 x 2 1 is 16 x 1 double height Example i2c.write(0,trigger,28,1) set double ht
60	Adr 6-255	**Change I2C address** The address must be an even value, this is the 8 bit or full address where an even address is write and an odd address is read. It will not take effect until reset. The same effect can be achieved by writing to the EEPROM Example i2c.write(0,trigger,60,adr)
61	Adr 0-255	**Read EEPROM** This will read a single value from an address in EEPROM Example i2c.write(0,trigger,61,adr) i2c.read(1) Send the trigger, command and then the address to be read.
62	Adr 0-255 data 0-255	**Write EEPROM** Writes a single byte to the given address Example i2c.write(0,trigger,62,adr,data)
63		**Gets device ID as 2 bytes** The first byte is the high byte of a 16 bit number and the second byte is the low byte Example i2c.write(0,trigger,63) i2c.read(2)
64		**Gets firmware version as 3 bytes** x.x.x Example i2c.write(0,trigger,64) i2c.read(3)
65	0 or 1	**Sleep**

		Places device in sleep mode and turns off the back light. The device can be awakened by any key on the keypad (connecting and row to any column) or input from I2C. See text Example i2c.write(0,trigger,65,0)
66	N/a	**Reset** Resets the device as at first switch on. Depending on the I2C master this will likely cause a time out as there will be no reply from the device and so this may cause an I2C error from the master Example i2c.write(0,trigger,66)

Notes:

The I2C interface does not have any text positioning commands and so these must be implemented on the host by sending the appropriate command to the LCD using 31.

i2c.write(0,trigger,31,0x80) is row 0, column 0

i2c.write(0,trigger,31,0xc0) is row 1, column 0

Add the column value to either 0x80 or 0xc0 to set a column position for that row.

BV4541 (Serial)

The serial version of the device is simpler to use and more versatile but cannot be 'bussed' and so only one device is allowed per serial interface.

Pin	**Description**
TX (output)	Transmit
RX (input)	Receive
V+	Power supply to device either 3.3V or 5V
Rs	No Connection
Gn	Ground

Serial Connector

The serial expects (TTL logic either 3.3V or 5V) NOT + and -12V RS232. If you have the old 9 pin connector then a conversion device will be required to transform the 12V levels to 5V or 3.3V levels.

If a 5V supply is used then 5V logic levels will be in place and if 3.3V supply is used then 3.3V logic levels will be in place.

The serial interface uses 1 start bit, 8 data bits,1 or 2 stop bits, no parity and no handshake, the default Baud rate is 9600 but this can be changed via an EEPROM setting

Writing to the serial interface will result on characters going to the LCD. To send commands to the LCD (and the keypad buffer if enabled) and escape character (value 27) is required forllowed by the command.

The command will not take place until CR (value 13 or \r) is received and when the command is complete CR is returned.

Where commands require a number it follows the command in deciaml, where commands require more than one number it is separated by a comma so for example reading 5 bytes form EEPROM location 0 would be:

escR0,5\r

General Rules

- <esc> is Escape (byte value 27) and must prefix all commands.
- CaSe is important.
- Values returned are in decimal followed by CR (byte value 13)
- Command act as soon as the CR is received.

Command	Range	Notes
Key Pad Note for these commands to be useful the Buffered Keypad setting in the EEPROM needs to be set to 1 (there is a command for this below)		
<esc>j	N/a	**Clear keypad buffer**
<esc>n	0-79	**Gets number of keys in buffer**
<esc>k		**Get key value from buffer** Reads the next key in buffer and decrements the buffer, a value of 0 means there is no values in the buffer. The buffer is arranged as first in first out so if keys 1,2,3 were pressed then reading the buffer four times would reveal 1,2,3,0
<esc>i<n>	0 or 1	**Set Buffered / Unbuffered** Sets the Buffered Keypad value in EEPROM to either 1 or 0. 1 is for buffered
<esc>r<n>	0 to 255	**Set Key repeat time** Key repeat time in multiples of about 16ms
LCD		
<esc>x		**Reset LCD** Does a software reset of the LCD should it ever be needed.
<esc>d<n>	1 to 255	**Send LCD commands** This will send the number <n> as a command to the LCD, see the data sheet for the LCD controller. For example the command 1 will clear the screen so <esc>d1\r will clear the screen.
<esc>s		**Display stored sign on message** For debugging to see if the new message displays okay
<esc>a<msg>	Text	**New sign on message** As an example to create “My Message”

		<esc>aMy Message\r End the message with CR and check if it is okay with <esc>s This message will be retained even after a power cycle.
<esc>v<n>	0 - 130	**Back light level**
<esc>o<n>	0 - 60	**Set contrast**
<esc>c		**Clear Screen** And home cursor
<esc>h		**Home Cursor** Without clearing screen
<esc>m<row>, <col>		**Move position** Moves position of cursor (or text insertion point) to that given by <row> 0 to 1 and <column> 0 to 15
<esc>g<from> ,<to>c		**Clear from to** Clears a section of text from position to position. The position starts at 0 which is row,col (0,0) to position 31 which is row,col(1,15)
<esc>u<n>	0 to 2	**Cursor Type** 0 no cursor (default) 1 Block flashing 2 Underscore
<esc>e<n>	0-1	**Set mode** 0 is 16 x 2 1 is 16 x 1 double height Has immediate effect but is not stored to EEPROM so only temporary
System		
<esc>E	N/a	**EEPROM reset** This will restore the default EEPROM settings No effect will take place until the device is reset.
<esc>B<n>	**1 to 8**	**Set Baud Rate** This is a temporary setting, for a permanent change the EEPROM values must be set. The value is a choice of Baud rates from 1 to 8, see the table in the EEPROM settings section.
<esc>R<a>,<n>	Adr 0-255	**Read EEPROM** Read the eeprom starting at address <a> for the number of bytes <n>
<esc>W<a>,<b>	a 0-255 b 0-255	**Write EEPROM** Writes byte <b> to address <a>
<esc>I		**Gets device ID** As a decimal number
<esc>F		**Gets firmware version** As a string
<esc>S		**Sleep** Places device in sleep mode and turns off the back light. The device can be awakened by any key on the keypad (connecting and row to any column) When the device wakes, it goes into the reset state.
<esc>X		**Reset** Resets the device as at first switch on