BV4212

I2C Front Panel with rotary input

Product specification

Contents

Rev	Change
01/02/17	Preliminary
17/05/17	Error in description of command 38 - corrected

1. Introduction

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

The output is in the form of an LCD display and the input is a rotary encoder with attached switch, there is also an additional button for further input.

Full I/O control can be realised with only 2 wires. The rotary encoder an switches are buffered leaving the host microcontroller free for other interrupts

2. Description

The device consists of two parts, an LCD display and encoder and switches. Both operate from the same I2C address (0x46 (0x23 7 bit))

3. Features

- Display 128x64 Graphic
- 3 Fonts
- I2C 100KHz
- I2C address user changeable
- Software adjustable contrast
- Software variable back light
- Rotary encoder
- Dual Voltage 3.3V & 5V
- 16mA @ 3V3 BL full on, 13mA off
- Sleep Mode 8.4mA
- Only 2 wires for full I/O control

4. Physical Description

The interface has a 2.54mm spaced 2 rows by 4 pins.

Pin	**Description**
3.3V interface	
SCL	I2C clock
GND	Ground
SDA	I2C data
3V3	3.3V
5V interface	
SCL	I2C clock
GND	Ground
SDA	I2C data
3V3	3.3V

Pin connections

The choice of interface depends on the voltage of the system.

4.1. Power Supply

The device works on 3.3V but there is an on board 3.3V regulator and so it can operate from 3.3V or 5V as follows:

3.3V

Use the 3v3 connector and supply with 3.3V

5V

Use the 5V connector and supply with 5V (max 6V)

5. I2C Interface

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

0x46 (0x23) for LCD and input

All commands go through the single I2C address that can be changed if required by the user.

NOTE: The address is stored in EEPROM in three places and a check is made at each reset to verify the value. At leas two address location values have to agree, if this is the case the third is set to that. If no addresses agree then the default address is used.

This is a robust method of storing addresses in a semi-volatile memory and in nearly all cases the address set by the user is maintained for ever. However if it is critical that the address cannot change under any circumstances then the part can be ordered with a fixed address.

6. Input

6.1. Rotary Encoder

The encoder is fully handled by the device, very simply increments when turned right and decrements when turned left. This value can be set and read via the I2C.

No intervention is required by the host processor other than reading or setting the initial value.

6.2. Rotary Switch

The encoder incorporates a push switch that can be read via the I2C. The switch once pressed will remain high until read, when read it is set low.

6.3. Button

The button can be read via the I2C. The button once pressed will remain high until read, when read it is set low.

6.4. Sleep Mode

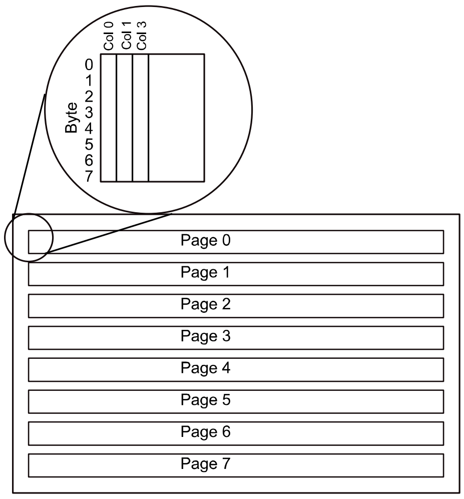
The device can be set to sleep mode via an I2C command. In this mode the input is inactive however the device can be awakened by an I2C read or write.

7. LCD

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

The LCD uses a UC1701 controller via an internal SPI interface. The interface is write only which limits the display to addressing columns and pages only.

Even with this limitation is possible to produce images using a free utility.



The layout of the display is in 8 pages, each page is 128 columns wide and 8 bits high. When writing to the display a byte is written at a time. This produced 8 pixels in a vertical line.

The x co-ordinate can be 0 to 127, however the y co-ordinate must specify a page 0 to 7. It is possible to specify an exact (0-63) y co-ordinate by specifying the correct page and then writing a byte that corresponds to that pixel. This is how images and fonts are produced.

The user need not worry too much about this as the device has a built in font generator.

7.1.1. Fonts

Although this is a graphic display the device is really designed to display text, 8 lines worth which is much more then the normal 16x2 or 20x4.

Fonts can be specified to begin on any x pixel but must start on a page (0-7).

There are three fonts, font 1 is 8 bits high and 6 bits wide and thus will fit into a page line, font 2 is the same as font 1 but 8 bits wide and so looks 'bold'. Font 3 is double height occupying 2 pages.

7.2. Images

Images are sent to the device as binary. These are raster images pre-created by the utility. They must be stored on the host and then sent to the device.

<number of pages> <number of columns> <data>

The data is in a particular format which suits the page layout of the display. There is a utility written in Python that will convert a 32bit colour BMP image into a monochrome data block suitable for incorporating into either a C for ByPic file.

Images must not exceed 128x64 otherwise distortion will occur. They can of course be smaller.

7.3. I2C

Pseudo code for sending an I2C image..

```
i2c_start(106)
i2c_putc(34) // command
pages = img[0]
bpp = img[1]
i2c_putc(pages) // pages
i2c_putc(bpp) // bytes per page
for j = 0 to (pages*bpp)-1
    i2c_putc(img[j+2]) // binary
next
```

8. Device Parameters

The EEPROM contains important values that control the way the device behaves. All of the values can be changed by the user using the i2c interface.

The EEPROM consists of 255 bytes and in general the first 16 bytes are used by the system

Adr	Default Value	Description
0	0	System Use
1	106	Device address 'j'
2	20	Default contrast
3	1	Indicator flag
4	13	EOL
5	8	Default back light
14	106	Device address copy
250	106	Device address copy

Table 1 System EEPROM

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.

8.1. Address

These EEPROM locations contains the device address. By convention the address is set to values between the values 10 to 250, no checking is made by the device so setting values outside this range may or may not work.

The address is the 8 bit address and so MUST bean even number. For example if the address was set to say 0x64 then on the Arduino which uses a 7 bit system, accessing the device would need the Arduino to address 0x32.

For security the address is stored in three places and to change the address of the device at least two of the locations need to be set otherwise the device will detect the anomaly at start up and revert to the majority value.

Normally to change the address of a device locations 1 and 14 are both changed. The device will detect this at start up and change the address in location 250 to match.

8.2. Contrast

This is the default contrast setting for the LCD display and the default value will give good results in normal conditions.

The contrast can be set at any time so this value does not need changing it is simply the value that is used for initialisation.

8.3. Indicator Flag

At start up there is a sign on message similar to:

BV412 I2C adr: 0x46

This can be suppressed by setting the indicator flag to 0

8.4. EOL Character

By default this is 13 which is the standard ASCII CR . This is only used by the write text command to indicate the end of text.

8.5. Back Light

This is the back light intensity can be varied form 0 (off) to 10 full on. This value holds the default value but can be changed at any time.

9. Keypad Commands

9.1. I2C

Key pad commands **I2C address 0x46 (0x23 7 bit address)**

All I2C transactions start with a command for example:

8 bit pseudo code get value from buffer	7 bit pseudo code get value from buffer

i2c_start(0x46) // write
i2c_putc(3)
i2c_stop()
i2c_start(0x47) // read
value = i2c_getc()
i2c_stop()
	i2c_start()
	i2c_write(0x23,3)
	i2c_stop()
	i2c_start()
	value = i2c_read(0x23)
	i2c_stop()

I2C Command	Value	Returns	Description
Rotary and Switch input			
5	N/a	0-255	Get Rotary Count Turning clockwise will increment the count, anticlockwise will decrement the count. After 255 it will roll over to 0
6	0-255	N/a	Set Rotary count Sets the count to a pre-determined value. This is useful for say setting the count to zero before a menu operation.
7	N/a	0-1	Get rotary switch value When the switch is pressed this value is set to 1. Reading will reset the value back to 0
8	N/a	0-1	Get button value When the button is pressed this value is set to 1. Reading will reset the value back to 0
LCD Display			
30	N/a	N/a	Reset LCD Resets the display
31	0-255	N/a	Sends Command Sends a command to the LCD display, see data sheet for the display. This will only be used for specialist applications. The display uses a UC1701 controller and this command will give direct access to that.
32	0-255	N/a	Sends Data Sends data to the display. See data sheet for display. This will only be used for specialist applications. The display uses a UC1701 controller and this command will give direct access to that.
33	See text	N/a	Write Text This is for writing a string of text to the display. The command should be followed by EOL (default 13) to terminate the command.

34	See text	N/a	Write Image See text. The image data is created by a Python utility.
36	0-10	N/a	Back Light Sets the brightness of the back light
37	0-63	N/a	Sets contrast

38	0-7,0-127,n,s..	N/a	Send Bytes to LCD
			Sends a string of bytes s to the display at one particular row as specified in the command. The byte effect on the display is described in section 7.
			This command is useful for drawing special characters or lines.
			Example: send bytes 101,103,107 and 109 to row 1 starting at column 50
			i2c_start(0x23) // start with the default address of device
			i2c_put(1); // row
			i2c_put(50); // col
			i2c_put(4); // number of bytes
			i2c_put(101);i2c_put(103),i2c_put(107);i2c_put(109);
			i2c_stop()

40	1 – 3	N/a	Set Font 1 is 8x6, normal text, 2 is 8x8 bold text and 3 is double height.
41	N/a	N/a	Home and Clear Screen
42	0 – 127	N/a	Set Column
43	0 – 7	N/a	Set row (page)
44	0-63	N/a	Initial Scroll line Experiment with this – see data sheet for the display to see what it does,
45	0 – 255	N/a	Write Character Writes a single character to the screen in the current font at the current location.
System			
144	Byte	Byte	Read EEPROM Reads a **single byte** from the EEPROM, specify command, address.
145	Byte		Write a Byte to EEPROM Specify command, address, value.
149	N/a	N/a	Reset Resets the whole device, similar to a power cycle.
160		2 bytes	Firmware Version After sending command read 2 bytes

161		Word	Device ID After sending command read 2 bytes. The first byte is the high byte of a 16 bit word.
163	N/a	N/a	Sleep Puts device into sleep mode.