BV4611

128x64 Serial + I2C Graphic Controller

Product specification Apr 2013 V0.a

Contents

128x64 Serial + I2C Graphic Controller BV4611

Rev	Change
Apr. 2013	Preliminary
Jan 2014	Version 2.0 * font 3 fixed * added to I2C esc{g * device id reports 4611 *Baud rate initially fixed at 9600

1. Introduction

The BV4611 is a 128x64 serial graphic display controller designed for displays that use two KS0108B controllers.

It provides a universal serial interface that can easily be controlled by escape commands. It has full scrolling capability and 3 fonts.

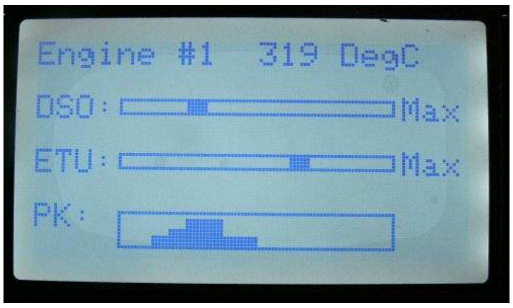
The serial interface can be either a standard asynchronous **OR** I2C, this is fully selectable by the user.

This datasheet describes the controller and thus the display capabilities, the controller is normally supplied attached to a display, however the controller can be supplied separately.

2. Features

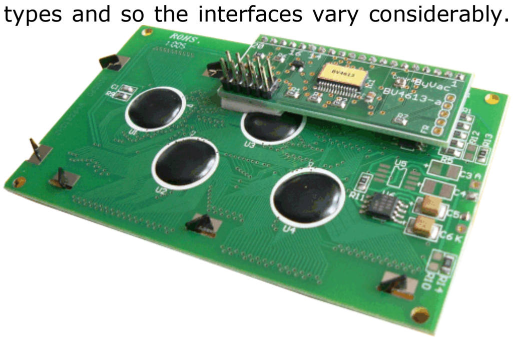
- Designed for displays using the KS0108B LCD controller
- Serial interface
- I2C Interface
- Three fonts
- Baud rate selectable by the user, initial rate 9600
- Up to 21 characters by 8 lines
- Graphic drawing, lines, pixels, circles rectangles filled or not.
- I2C address set by user
- Back light control
- ACK mechanism to eliminate the need for hardware handshaking
- Voltage: 3.3V or 5V (see text)
- Size: 53x22mm

3. Physical Specification



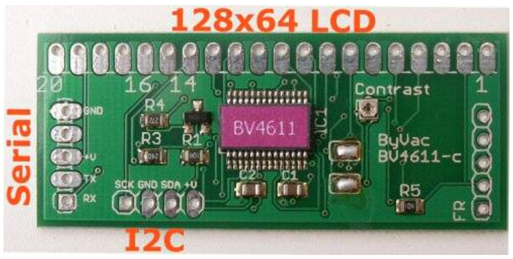
Controller with display

The controller can be supplied with an LCD display attached. Where this is not the case please make sure that the controller is suitable for your display. This kind of display is no where as near standardised as the character types and so the interfaces vary considerably.



Rear view, typical

3.1. Control Interface



The device has a connector that can either be used for serial or I2c. This is user selectable option, see the section on I2C for more details.

3.2. Serial Interface

The serial interface has 5 pins, one of the pins is not used.

Pin	Function
5	Ground
4	not used
3	+V (3.3 to 5V)
2	TX
1	RX

The serial connector will mate directly with the BV101/3 .

The board will operate from 3.3V and work correctly on 3.3V logic. It will also operate at 5V and work with 5V logic. Most LCD displays however require 5V and the display supplied by ByVac will need a minimum of 4.9V to operate correctly.

RX This is the serial receive and expects signals 0 to +V. The idle state is high. The input is a standard byte frame of 1 start bit 8 data bits and 1 or 2 stop bits. In other words a standard asynchronous serial signal.

NOTE the interface will NOT directly accept voltages from a standard COM port that output +ve and negative voltages. There must be some kind of voltage translation, see the BV103 at www.byvac.com which is an ideal convertor but any USB to serial interface will do.

TX This is the output from the device.

+V Is the main power supply for the device which should be capable of providing sufficient current when the display is connected. Most LCD device will not work correctly if the voltage is below 5V. The maximum voltage the device can accept is 5.5V

3.3. Multiple Devices

Only one device at once can be used on a serial bus. For multiple devices use the I2C interface.

4. Handshake and ACK

The ideal interface would implement the hardware handshaking mechanism using RTS/CTS. This is where the device's RTS is connected to the host CTS, when the device is busy, it raises the RTS line and the host temporarily stops transmitting.

Where it is not possible to do this and also where software can be simplified, the ACK mechanism is available. This is switched off by default and so must be enabled using the appropriate command.

Because the device takes a finite time to carry out a command, say clearing the screen, it is useful to have an indication of when that has happened (completed) and more accurately when the device is ready to accept another command. This is what the ACK mechanism is for. When enabled and when the device has completed a command and is ready to accept another command it will send an ACK character. The actual character (value from 1 to 255) can be specified by the user.

Not all commands will send an ACK where this is so it is indicated on the command table. Sending plane text for example does not send an ACK character. To get round this without resorting to a hardware handshake a command is provided that will just return the ACK character.

If this is used for every 60 or so characters sent, if the buffer is filling up then sending this character and waiting for the ACK will effectively empty the buffer so more characters can be sent.

5. Control Commands

There are TWO distinct methods of controlling the device, one is by serial commands and the other is by I2C. Only one can be active at any one time, see the section on I2C for how to activate this method of command.

The device when used serially is controlled by issuing control commands that always begin with and escape byte (0x1b). The escape is followed by various sequences to make up individual commands.

The command is accepted as soon as the last byte of the command is entered, so for example when clearing the screen with esc[2J the screen will clear as soon as 'J' is entered.

6. Serial

The serial interface uses 1 start bit, 8 data bits and 1 or 2 stop bits, no parity. By default the Baud rate **9600**. The user can change this. The rate is selected from the following rates:

2400, 4800, 9600, 14400, 19200, 38400, 56700 and 115200

No other rates are acceptable and so the host must be set to one of these. It is possible to fix one of these rates (see the command table).

There is a sign-on screen that will indicate when the device is in serial or I2C mode

7. I2C

(Default address 0x68, 8 bit) (0x32, 7 bit)

Function	SCK	GND	SDA	+V
Pin	1	2	3	4

The device can operate from serial **OR** I2C. This is determined by the SDA line which, when disconnected, is held low by a high value resistor. The mode is determined at reset or switch on.

With no I2C bus connected the SDA line is low and thus the device detects this and starts up in serial mode. With an I2C bus connected the pull up resistors on the master or bus hold this line high and the device goes into I2C mode. The mode remains until reset.

It should be obvious that this device does not have any pull-up resistors, these are normally provided on the master device or on the bus by two discrete pull up resistors.

The commands entered are exactly those that are used for the serial communication. In order to give some examples the following notation is used.

7.1.1. Write

1. Send start condition
2. Send device address with write
3. Send command

4. Send stop condition

7.1.2.Read

1. Send start with write
2. Send device address with write
3. Send command
4. Send restart condition with read
5. Read bytes
6. Send stop condition

ALL commands on this device begin with a write, even commands that read something a command is sent first. There is a notation that is used in the examples as follows:

'**s**' means send the start condition **and** device address with write. i.e. the least significant bit is set to 0 telling the I2C device that the next byte should be written to the device.

'**number**' A number will simply be sent to the device.

'**r**' This is the restart command; it sends a stop condition followed by a start condition and the device address with the read bit set (LSB set to 1). The device will now send out bytes that are clocked by the master.

'**g-n**' will clock 'n' number of bytes out of the device, 'g-3' will get three bytes.

'**p**' is the stop condition.

This is the notation that is used for the BV4221, USB to I2C convertor, for more information about this visit www.byvac.com or www.i2c.byvac.com

As an example, to send clear screen to the display which is esc[2J the following would do, all values in hex

s 1b 5b 32 4a p

Note that the '32' above is the ASCII code for 2. To illustrate this point further to move the cursor to line 3 column 21 would need the command esc[3;27H which would be:

s 1b 5b 33 3b 32 37 48 p

7.2. Address

The I2C address is set to **0x68** by default. This is the 8 bit address which includes the read/write bit the 7 bit address would be 0x34.

This can be changed to any address in the range 2 to 254 by the command given in the command table.

8. LCD Connector

The controller is designed to operate a specific type of display as supplied by ByVac, however it may well be possible that these can be obtained elsewhere. The information given here should help determine if an alternative display is suitable.

The connector is a row of pads at the top of the PCB, pin 1 is marked.

Pin	Function
1	GND
2	+V
3	Vo
4	RS
5	WR
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	CS1
16	CS2
17	RESET
18	Vee
19	LED-A
20	LED-K

D0-D7: Data lines, output and input for writing to the display and reading from it.

E & RS: Control lines for the display, output.

R/W: Controls the direction of the data lines, output

Vee & Vo: VOUT is an output voltage from the display. This goes to a potentiometer the centre tap of which goes to Vo to control the contrast. The potentiometer is in the form of a trimmer on the board.

+V & GND: These are connected directly to the +V and GND on the input connector.

CS1 & CS2: Control which LCD controller (KS0108B) is selected.

RESET: Is an input that resets the display, this is invoked when the BV4613 is powered up or the reset command is issued.

Pin 19 & 20 (back light): Pin 19 goes to the back light on the display and pin 20 goes to a switching transistor through a very low ohm value resistor. When on pin 20 is more or less connected to grround.

9. Start Up & Macro

By default the start up screen gives information as to what interface is being used, I2C or Serial and in the case of serial it will inform the user when communication has been established, or more accurately when the Baud rate has been determined. By default when this happens the controller will also send '*' to indicate to the host that communication is established.

The user can specify a macro that will replace the start up screens, the macro can consist of commands and text so there the only limit as to what can be shown on the screen is the number of bytes that can be saved, see the command table for this information.

It may not be desirable to have anything on the start up screen in which case it can be switched off.

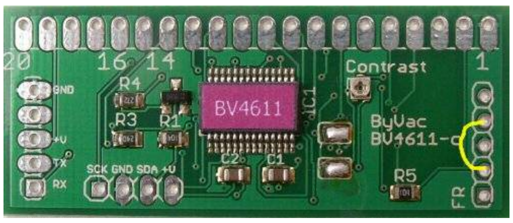
10. Factory Reset

Using the appropriate commands defaults can be set that in certain circumstances make the device difficult to communicate with. For example if an unknown address has been inadvertently set.

In this case there is a hardware option for setting all of the defaults back to a known condition. To do this use the following procedure:

1. Remove the power
2. Short out two pads on the PCB, this is pad 2 and 4 as shown below in yellow. (pad 1 is a square shape)
3. Apply power
4. Remove power and remove the short.

The device will now be restored to its factory defaults.



11. Programming Examples

The following shows some typical examples using a C like language. The routines have not been tested and are here for example only:

com_putc(n); puts a character 'n' to the open com port.

com_puts(s); sends a sting to the open com port.

com_getc(); gets a character from the open com port.

The examples assume that a com port has been opened.

11.1. Serial

Initialisation; when the device is set for autobaud consists of sending byte 13 (CR) and waiting for a response. In this example the ACK mechanism will be activated as there is no HW handshake available

```
int initialise()
{
int j;
char c;
        com_putc(13);
        j=10;
        while(j--) {
      c=com_getc();
      if(c='*') break;
      delay(1); // ms
        }
        if(j) { // initialised ok if J!=0
          // send esc[6E
          com_putc(27);
          com_putc(91); // [
          com_putc(6);
          com_putc(69); // E
        }
        return j;
}
```

The above will return not 0 if initialised properly or 0 otherwise. The options here are to control the reset line with hardware and have the software try again but this should not be necessary.

The device has also been initialised so that commands send back char(6) when finished.

Sending Commands

```
void vt_cmd(char* cmd)
{
int j=10; // set to time out
char c;
    // send command
    com_putc(27);
    com_puts(cmd);
    // wait for ACK
    while(j--) {
        c=com_get();
        if(c==6) break;
        // some delay or bigger timeout
    }
    // if j==0 then handle an error
}
```

When using the ACK system the above will wait for the ACK to be returned before proceeding

Sending Text

```
void vt_text(char* s)
{
int j=0;
    while(*s) {
        com_putc(*s++);
        j++;
        if(j > 60) {
            vt_cmd("[e")
            j=0;
        }
    }
}
```

When sending text, in order to ensure that the device buffer does not become full, text is sent char by char and the special ACK command is used. This will stop filling the buffer until the command is received and acknowledged.

NOTE: The above routine complexities are only necessary when hardware handshaking is not available. If it is then simply send to the com port.

11.2. I2C

The ACK is not necessary for I2C, however clock stretching is essential and the master must provide this. Any errors of the type that work up to a point, i.e. only half a page of text is sent instead of a full one is because the master is not recognising clock stretching properly.

The following example will position the cursor on line 4, column 27. The command for this is esc[4;27H

```
void i2c_pos()
{
    i2c_start(); // send start condition
    i2c_send(0x68); // send i2c address with
write set
    i2c_send(27); // esc
    i2c_send(91); // [
    i2c_send(34); // 4
    i2c_send(59); // ;
    i2c_send(32); // 2
    i2c_send(37); // 7
    i2c_send(72); // H
    i2c_stop(); // stop condition
}
```

Note that the line and column positions are specified as ASCII codes rather than actual binary values. A function could be devised to do this automatically

11.3. Implemented Escape Codes

In the table columns indicate:

S is the VT100 standard, Y is standard, P is partially standard, N is non-standard

Code is the escape sequence to invoke the command

DEF column shows the default values from the factory, and F in this column will indicate that this value will be stored when writing the EEPROM

ACK column indicates that this command is capable of sending an ACK character when it has finished.

Grayed out cells are not implemented on this device. Unless otherwise stated all numbers are in decimal.

S	Code	DEF	ACK	Name	Description
	Cursor Movement and Text Placement				
Y	LF			Line feed (10)	Moves the cursor down one line, if the cursor is on the last line then the command is ignored.
Y	CR			Carriage return (13)	Moves the cursor to the beginning of the current line
Y	BS			Back space (8)	Moves the cursor back one space and deletes the character immediately to the left before moving the cursor. When the cursor reached the left margin it stops.
Y	esc[<row>;<col>H		Y	Move cursor to line and column	Moves cursor to specified line and character position. The row and column values are effected by the current font. For the standard font the row value will be between 0 & 7 and the column between 0 and about 31. Zero is the top left and so esc[0;0H will move the cursor to the home position. Values out of range will be ignored. This command is generally used for text and will place the cursor on a text boundary rather than a pixel boundary. For more precise pixel level positioning see the graphics and pixel drawing commands.
Y	esc[<num>A esc[A		Y	Move cursor up.	Moves the cursor up one or more lines, specified on its own 'esc[A' it will move the cursor up one line, specified with a number it will move the cursor up that many lines. When the cursor reaches the first line subsequent calls are ignored. esc[A can be used instead of esc[1A
Y	esc[<num>B esc[B		Y	Move cursor down.	As esc[<num>A but moves cursor down, this is the equivalent of line feed. esc[B can be used instead of esc[1B
Y	esc[<num>C		Y	Move cursor right.	As esc[<num>A but moves the cursor to the right.

	esc[C				esc[C can be used instead of esc[1C
Y	esc[<num>D esc[D		Y	Move cursor left.	As esc[<num>A but moves cursor to left. esc[D can be used instead of esc[1D
Y	esc[H		Y	Cursor home	Moves cursor to home position does not clear the display. This is the same as esc[0;0H. The home position is top left.
	System Settings				
N	esc[<num>E	Off		ACK	Sets and activates the ACK character as an alternative to hardware handshake. Some commands will send a character to the receiving device, this can be used to determine when the command has fished. The host can wait for this character before sending the next command. To set up the ACK to send ‘X for example the following is used: esc[88E 88 Is the ASCII code for ‘X’. To turn off the ACK system use esc[0E (esc[zeroE).
N	esc[e		Y	ACK	This will return the ACK character regardless of the ACK setting. This command can be useful for commands that don’t normally return ACK and there is no hardware handshake.
N	esc[?10a	On F	Y	Add Line Feed	Adds Line feed when CR is received as some terminals only send CR
N	esc[?10b		Y	Remove Line Feed	Turns off the additional line feed set by esc[?10a
Y	esc[?25I		Y	Hide cursor	turns the cursor off
Y	esc[?25h	On F	Y	Show cursor	turns the cursor on
N	esc[?26n		Y	No Scroll	Stops the display from scrolling, this can be useful for graphic presentations as when the text gets to the end of the line, the cursor will move to the next line. This can have an undesirable effect on the contents of the display. When this is set the cursor will move to the top of the display when it reached the end of the display.
N	esc[?26y	On F	Y	Scroll on	Reverses the effect of esc[?26n, this is the default.
N	esc[?26I		Y	Back light off	Turns off the back light
N	esc[?26h	On F	Y	Back light on	Turns the back light on
N	esc[<n>f			Cursor Flash Rate	Sets cursor flash rate. This is an arbitrary number, the larger the number the slower the flash rate, the default is 30000 esc[20000f

N	esc[<n>b			Sets Baud rate	Seta an alternative Baud rate to the default 9600. Set <n> to be the new Baud rate. The rate is selected from the following:
					
					1 = 2400;
					2 = 4800;
					3 = 9600;
					4 = 14400;
					5 = 19200;
					6 = 38400;
					7 = 56700;
					8 = 115200
					
					For example to set the Baud rate to 38400:
					
					esc[6b
					
					The effect will be immediate but only temporary, until the device is reset or power cycled. To f fix this rate and any other settings follow the command with esc[?27D, at the new Baud rate of course.

N	esc[?27D		Y	Writes defaults to EEPROM	Any variable indicated by an 'F' in the 'DEF' column of this table will be written to the EEPROM. This means that when the device is reset or switched on the new values will take effect (whatever the user has set them to) rather than the factory defaults.
N	esc[?27M		Y	Writes a sign on message to the EEPROM	All characters that follow this command will be displayed on sing on, providing the Com Flag is active (it is by default). The characters can consist of any command so lines and other items on this table can be included. To terminate the command use two escapes. As an example this will display "Hello World" at start uo: esc[?27MHello World<esc><esc> To turn off the macro use: esc[?27M<esc><esc> If the macro is active, i.e. something in it then the default sign on will not be displayed. The maximum number of characters is 150. NOTE: The macro will turn off the default screens.
N	esc[<num>a		Y	Set I2C Address	This will set the I2C address and must be followed by EEPROM write. The new address will not take effect until reset. The default address is 0x68 (104). As an example to set the address to 0x42 (66) : <esc>[66a<esc>[?27D Both of the commands must be specified although the write to EEPROM command does not need to directly follow this command.

					IMPORTANT An even number MUST be specified to give a valid 8 bit address, the device will not work if an odd address is specified, the device does not check the address you enter.
N	esc[?31d		Y	Device ID	Returns device ID as a number **I2C**: This is returned as a 16 bit value, in two bytes, high byte first.
N	esc[?31y	On F		Com Flag (0n)	When communication is established with the device, i.e. at switch on or when the auto Baud mechanism has selected the correct Baud rate, ASCII code 42 '*' is sent to indicate this has happened. This commands switches this on, it is on by default
N	esc[?31n			Com Flag (off)	Switches the flag off see esc[?31y
N	esc[?31f		Y	Firmware version	Returns firmware version, this is a 16 bit number **I2C**: This is returned as a 16 bit value, in two bytes, high byte first.
	esc[?32d			Debug	Shows the input as hex rather than text, this is useful for seeing what is actually being sent to the display when trying to figure out command sequences. The only way out of this mode is to reset the display.
P	escc			Reset device	Carries out a software reset
	Font and display settings				
P	esc(1	On F	Y	Font size 1	This is normal font with a character size of 6x8
P	esc(2		Y	Font size 2	This is a bold font with a character size of 8x8
P	esc(3		Y	Font size 3	This is a double height font with a character size of 8x16
N	esc[I		Y	Invert	Inverts the font colour and background each time it is called
	Screen clearing				
Y	esc[2J		Y	Clear screen	Clears display and homes cursor
N	esc[3J		Y	Clear screen and reset all defaults	Clears display, homes cursor and resets to the defaults values. Note if the values have been set by the user then this will use those values.
Y	esc[K		Y	Clear line from cursor right	All characters are cleared form the cursor to the end of the line, the cursor remains where it is and the character under the curser is also erased. (clear right)
Y	esc[1K		Y	Clear line from cursor left	The characters are cleared form the display starting at the beginning of the line to the current cursor position. The cursor remain where it is. (clear left)

Y	esc[2K		Y	Clear entire line	The whole current line is cleared and the cursor remains where it is. (clear all)
	Graphic Commands and Pixel Drawing				
	NOTE			Curly brackets	When used for drawing and bitmap commands co-ordinates are specified with a non-numeric character between, for example a space, comma or something else. The x co-ordinate is the width or horizontal which can have a value of 0-191 and the y co-ordinate is the height that can have a value 0-64. Example of valid co-ordinate specifications: esc{10 10 30 30L or esc{10;10;30;30L Drawing is carried out using the current foreground and background colours. Some commands have alternative formats where shown. When the co-ordinates are not given the current co-ordinates are used.
N	esc{x0 y0 x1 y1L esc{x1 y1L		Y	Draws a line	Draws a line in any direction from the co-ordinates x0,y0 to x1,y1. There must be a non-numeric character between the co-ordinates, this is acceptable esc{10,10,40,40L or esc{10;10;40;40L
N	esc{ x0 y0 x1 y1R esc{x1 y1R		Y	Draws a rectangle	The co-ordinates are the top left and bottom right of the rectangle
N	esc{ x0 y0 x1 y1F esc{x1 y1F		Y	Draws a filled rectangle	As the R command but fills with foreground colour.
N	esc{x0 y0 radC esc{radC		Y	Draws a circle	A circle is drawn with the given radius (rad) at the specified x and y co-ordinates.
N	esc{x0 y0P esc{P		Y	Draws a pixel	A single pixel is drawn at the given co-ordinates, where no co-ordinates are given a pixel is drawn at the current co-ordinates.
N	esc{x0 y0p		Y	Moves to pixel	This is similar to the 'P' command but no pixel is drawn, this command will move the cursor to an exact pixel and is useful for aligning text for example.
N	esc{g			Returns current co-ordinates	Returns the pixel x (horizontal) and y(vertical) co-ordinate pixel values. The format is: x,y<CR> **I2C:** (from 1.5) Returns two bytes representing x and y

Notes