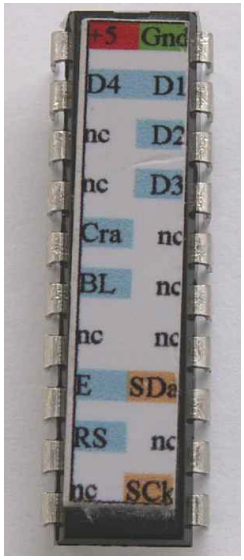


BV4208

I2C-LCD IC

Product specification August 2008 V0.a

©ByVac

Contents

Rev	Change
Sept 2007	Preliminary
Oct 2008	Missing description of system commands

1. Introduction

The BV4208 is an I2C two wire compatible integrated circuit for interfacing to the HD44780 or similar LCD display. This is just about every display module available.

The chip works by converting commands derived from the I2C bus and translating that into signals suitable for the LCD display.

2. Features

- I2C up to 400kHz
- Simple command set for direct interface to LCD module
- Back light output
- Contrast pin allows display blanking
- 1,2 and 4 line displays
- 31 character buffer for I2C
- Operating voltage 2.0 to 5.5V
- Current <1mA @ 5V

3. Electrical Specification

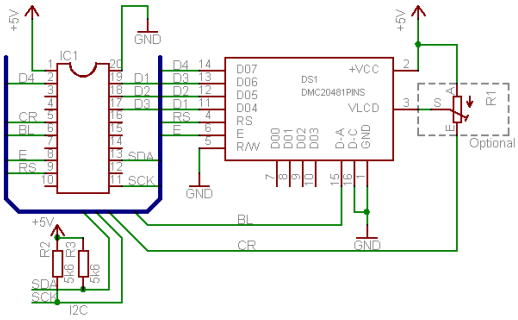


Figure 1 Wiring Diagram

The chip is connected according to the wiring diagram. It should be noted that this is a 4 wire interface where only the top 4 data lines are connected to the display.

The contrast potentiometer is optional, the CR line when active provides a ground connection thus the potentiometer will set the contrast. There is an I2C command that will set this line high which will blank the display. As an option the Cr line can go directly to the VCLD pin of the LCD display.

The BL (Back Light) line is capable of sourcing or sinking up to 20mA so this can be connected to either the anode or the cathode of the LED backlight

The I2C specification calls for two pull up resistors, these must be included somewhere on the bus and so have been shown on this diagram.

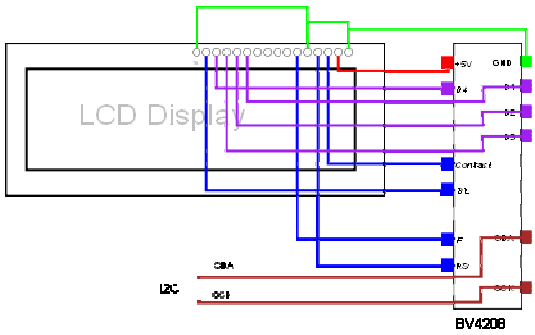


Figure 2 Minimal Wiring

This diagram shows the minimal wiring for making the display work.

The default for the device is set for 2 display lines with a 5 x 7 dot matrix character set. This is determined by the commands issued to the display at start up, this of course can be changed when the I2C communication is established.

Figure 3 Sign on

The picture in Figure 3 shows a 16 x 2 line display using the minimal wiring illustrated in Figure 2. The sign on message is 11 characters so should fit on a small display. This can be changed using a system command.

Pin	Description
1	+2.0V to 5V
2	D4: connect to D7 of the LCD display
3	Hardware reset pin: see section on hardware reset. Leave this pin unconnected
4	n/c
5	Contrast: this pin is a digital o/p that can be either high or low
6	Back Light: this pin is a digital output capable of supplying or sourcing up to 20mA
7	n/c
8	E: connect to the E pin of the LCD display
9	RS: Connect to the RS pin of the LCD display

10	n/c
11	SCK: this is the I2C clock, a pull up resistor is required on this pin 5k6 will do.
12	n/c
13	SDA: this is the I2c data, a pull up resistor is required on this pin 5k6 will do.
14	n/c
15	n/c
16	n/c
17	D3: connect this to the D6 of the LCD display
18	D2: connect this to the D5 of the LCD display
19	D1: connect this to the D4 of the LCD display
20	Ground

Table 1 IC Pin Description

4. 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 LCD and those referring to the system. The system commands enable changing of the device address etc.

Device default address is 0x42

Command	LCD Command Set
1	Send LCD Command
2	Send LCD Data
3	Back Light
4	Print Message
5	Display on
6	Message Read
Command	**System Command Set**
0x90	Read EEPROM
0x91	Write EEPROM
0x92	Confirm Command Complete
0x93	End of EEPROM
0x95	Reset
0x96	Factory Reset
0x98	Change Device Address Temporary
0x99	Change Device Address Permanent

Table 2 LCD & System Command Set

Table 2 is a command summary of all of the LCD and system commands.

5. The LCD Command Set

The LCD command set is a direct control of the standard internal controller chip, typically a HD44780, SPLC780D that allows simple control over the LCD display. For projects requiring simple displays this is ideal.

There are just two basic inputs to the LCD display and that is one for controlling how the display works and the other for writing characters to the display. This is taken care of by commands 1 and 2 as described later.

A useful source of information on how to control the lCD display can be found at http://www.doc.ic.ac.uk/~ih/doc/lcd/ . This is one of many on the net.

The method of writing to the display 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 (0x42). 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 0x42, the restart address will be 0x43. Again the data can be one or more bytes read from the device.

Each command will have it's own format and is described in the following text. A start condition and address is always followed by a command. The device has an internal 32 byte I2C buffer, the effect of this is two fold:

1. Only 32 bytes can be sent to the I2C bus at any one time, this includes the command itself and any restart address that may be required.
2. The device will only respond after the stop command is received.

Using a buffer enables the I2C bus to work at full speed even though the device it may be connected to may be slower.

5.1. Command 1

Name: **Send LCD Command**

Format: <**S-addr**><**1**><**byte**><**Stop**>

This command will send data directly to the LCD as a command, i.e. with the RS line low. Using this enables the display behaviour to be altered to suit the application.

Examples for **<byte>**:

1 = Home and clear display

4 = This will move the cursor from right to left when a character is displayed.

6 = Default move cursor to right

0x0c = Turns cursor off

0x0e = Turns cursor on

0x0F = Cursor blinks whole character

All of the above are determined by the display controller, command 1 simply sends a byte of information to the display with the RS line low.

5.2. Command 2

Name: **Send LCD Data**

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

This will send data to be displayed on the display. The data should be ASCII coded text and <**byte..**> can be up to 30 bytes at any one time.

There is no warring that the buffer is full so it is up to the application to make sure less than 32 bytes are sent.

Examples for <**byte..**>

0x46 0x72 0x65 0x64

Writes **Fred** to the display.

5.3. Command 3

Name: **Back light**

Format: <**S-addr**><**3**><**1 or 0**>**<Stop>**

This command controls the back light pin as a digital output. It is the inverse of the number sent so a 1 would place a logic 0 on the pin and a 0 would place a logic 1 on the pin.

If not used for the back light this can be used as a general purpose output pin.

5.4. Command 4

Name: **Print Message**

Format: <**S-addr**><**4**><**EEPROM Address**>**<Stop>**

This will print a string contained in the EEPROM to the display at the address given at EEPROM Address. The string can consist of commands and characters and follows this format:

0x0 Terminates the string

0x01 Next byte that follows is an LCD command

Anything else will be sent to the display as LCD data. See command 6 for creating a new sign-on message.

Example:

0x01 0x01 0x46 0x72 0x65 0x64 0x00

The first byte 0x01 tells the controller to interpret the next byte as a command, the command in byte 2 0x01 will be sent to the display as a command and thus it will clear the display and home the cursor. The sting "0x46 0x72 0x65 0x64" is ASCII for Fred and this is terminated with 0x00.

5.5. Command 5

Name: **Display on**

Format: <**S-addr**><5><0 or 1>**<Stop>**

This directly controls the contrast output and is the inverse. A one in this command will take the contrast pin low, thus enabling the display. A 0 on this command will take the contrast pin high and thus blank the display.

5.6. Command 6

Name: **Message Area**

Format: <**S-addr**><**6**><**R-Addr**><**data**>**<Stop>**

The value returned from this command is the EEPROM address of the sign on message. A new message can be placed here to customise the start up message.

This address up to the end of EEPROM can be used for any purpose including other custom messages. The end of the EEPROM can be determined by system command 0x93

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

6. 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.

6.1. 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.

6.2. 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.

6.3. 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.

6.4. 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.

6.5. 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.

6.6. 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.

6.7. 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.

6.8. 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.

6.9. 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.

6.10. 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.

7. 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.

8. 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

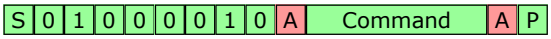
N Not acknowledge = 0

8.1. 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:

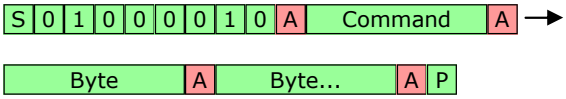


8.2. Sending a command with a parameter byte/s

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.



8.3. 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.

9. 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.

9.1. 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.

9.2. 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.

9.3. 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.