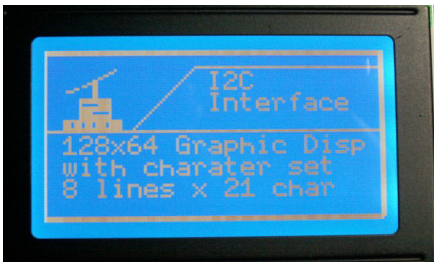
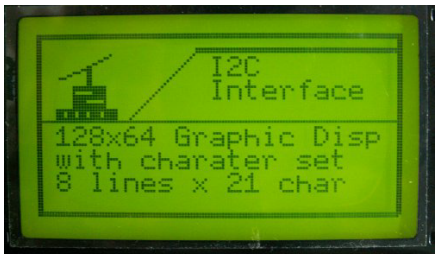
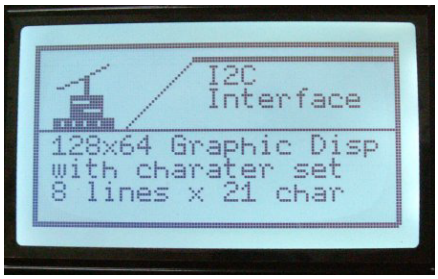
I2C-LCD- 128 x 64 Graphic BV4512







BV4512

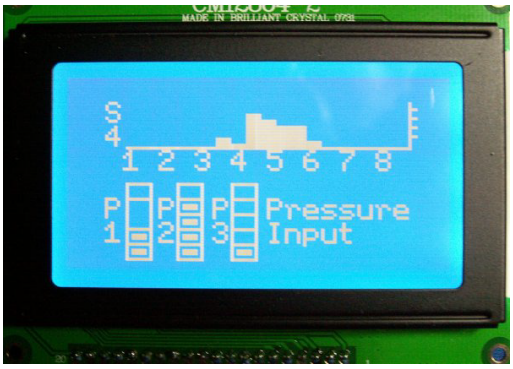
I2C-LCD- 128 x 64 Graphic

Product specification November 2008 V0.a

Contents

Rev	Change
Nov 2007	Preliminary
June 2009	Minor update to command 20
Aug 2009	Name change from BV4219 to BV4511

1. Introduction



The BV4511 (formally BV4219) is a graphical LCD display that has white pixels on a blue background. The 128 x 64 pixels can be individually switched on or off producing monochrome graphics. In addition to this the I2C controller provides a text interface giving a display that is capable of displaying 21 characters over 8 rows. Text and graphics can be mixed to produce a comprehensive display system.

The display is controlled by simple I2C commands over a two wire interface thus considerably reducing component cont and interface complexity.

There is built in line, box, pixel and text drawing commands reducing the complexity of the software interface.

2. Features

- I2C up to 400kHz
- Simple command set for direct interface to LCD module
- Back light
- 128 x 64 pixels
- Built in character set
- 21 characters by 8 lines giving a total of 168 displayable characters
- Line drawing functions
- Box drawing functions, with or without fill.
- Pixel drawing functions
- Size 95mm wide, 70mm tall and 23mm deep including interface board

• Power

The following values are average as the displays vary, all readings are taken at 5V

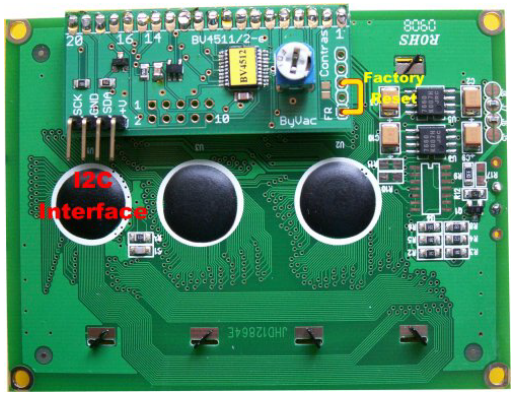
Back Light On

Device	Ma	Mw
Dark Blue	40.5	202.5
Green	42	225
White	42	225

Back Light Off

Device	Ma	Mw
Dark Blue	8.4	42
Green	10.1	50.5
White	8.4	42

3. Connection Specification



I2C Interface

Attached to the back of the display is the I2C controller board. This has four inputs as follows:

Pin	Name
1	Clock (SCK)
2	GND
3	Data (SDA)
4	+5V

The connection is terminated as 4 PCB pins. There is also one other row of holes to the right of the contrast trimmer which is used for factory reset. It is marked FR and is a 'square' hole.

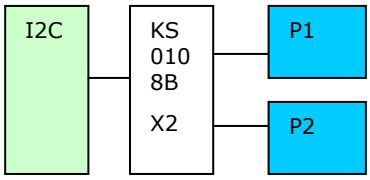
Should the EEPROM contents become overwritten the following actions will restore the factory conditions and the default I2C address of 0x42.

1) Remove power
2) Connect pins 1 and 3 of jumper 1 together
3) Apply power
4) Remove power
5) Remove shorting link

The device will now be restored to the factory default settings.

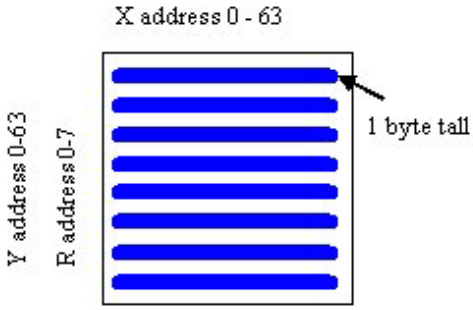
There are also other holes on the board that should be ignored as these are used for the BV4512 which is not an I2C device.

4. Device Description



Block Diagram

As shown in the block diagram the display consists of an I2C interface connected to the KS0108 controller system. This in turn is connected to two 64 x 64 panels that are laid out side by side.



Each panel is arranged as 8 strips of 8 bits tall, as two panels are used this makes the display 64 pixels high by 128 pixels long.

The implication of this layout is that there are two vertical addresses, the Y address that refers to the pixel and the R (row) address that refers to one of the horizontal strips. The row address is used for text and writing to the display directly with a byte (command 3). The Y address is used for drawing and placing pixels.

The work of addressing the panels and drawing functions are taken care of by the I2C interface.

5. White or Dark

The display is capable of offering white pixels on a dark background or dark pixels on a white background. Most of the commands have two alternatives for this reason.

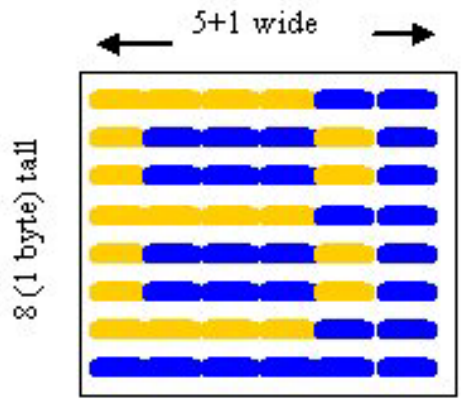
By default, the display will clear to a dark background.

6. Text

Character Set

There is no built in text on the KS0108 controller as there is in the more common HD44780 type controller and so all of the text, fonts, display positioning etc. is taken care of by the I2C controller.

The font is based on a 5 wide x 8 tall matrix with 1 x 8 pixel space between the characters and so 1 character occupies 6 pixels



This is a close up of the letter 'B' The actual font occupies 5 pixels wide but there is always and extra column to separate it from the next character.

This format allows 21 characters to fit into the length of the display with 1 pixel space at x=0 and one at x=127.

The character set consists of the standard ASCII codes from 32 (0x20) to 126 (0x7e). There are also user defined characters. The full set is displayed in the above picture beginning with space (32) and ending with '}', (126).

7. Timing

This device along with other LCD devices, in certain circumstances, requires time to complete a command. Some commands are also internally complex and require more time than other to complete, filling a box for example.

It is important that the host software takes this into account when sending multiple instructions to the display otherwise commands may be missed completely. See also section 10.1 on clock stretching.

Longer commands have the time shown in the command description.

8. Co-ordinates

This is fully described in the commands section but as a general rule co-ordinate 0 is top left. Lines and boxes emanate either to the right or downward.

9. 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 types of command, those referring to the LCD display 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	Turn on/off display
2	Direct command to the KS0108
3	Write byte
4	Read byte
5	Reset and clear display
6	Back light on/off
7	Set X position
8	Set R position
9	Set X,R and send byte
0x0a	Set X,Y and send byte
0x10	Fill display with byte
0x11	X,Y, Pixel
0x12	Draw Horizontal line (White)
0x13	Draw Horizontal line (Dark)
0x14	Draw Vertical line (White)
0x15	Draw Vertical line (Dark)
0x16	Draw Box no-fill (White)
0x17	Draw Box no-fill (Dark)
0x18	Draw Box with-fill (White)
0x19	Draw Box with-fill (Dark)
0x20	Character (White)
0x21	Character (Dark)
0x22	User defined char (White)
0x23	User defined char (Dark)
0x25	Set R,C position
0x27	Get X position
0x28	Get R position
Command	**System Command Set**
0x55	Test
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 1 LCD & System Command Set

Table 1 is a command summary of all of the LCD and system commands.

10. The LCD Command Set

Most of the work for this display is carried out in the I2C controller as the KS0108 does not do that much and so there is only one direct command for writing to this device.

The data sheet for the KS0108 can be found on the web or downloaded form the downloads section of www.byvac.co.uk

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.

10.1. Clock Stretching

The LCD device can be slow particularly when pixel addressing and drawing lines. The I2C method of handshake is to use clock stretching, the slave holds the clock line low to tell the master it is not free yet.

This device makes use of this on the slower commands. Unfortunately not all master devices will detect this and if this is the case, delays must be introduced into the driving software. See the timing table later on.

10.2. Command 1

Name: **Display on/off**

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

Simply turns the display on (1) or off (0). This is the display controllers and not the back light

10.3. Command 2

Name: **Direct command**

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

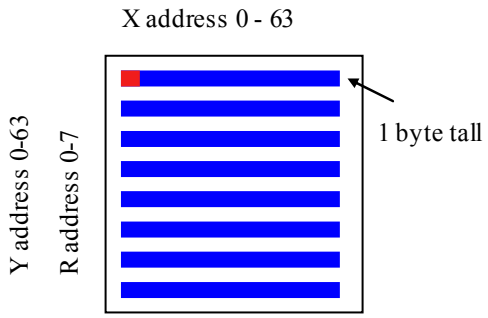
Sends a byte to the KS0108 controller as a command. For example sending 0x3e will turn off the display and sending 0x3f will turn it on. There are a limited set of commands for this controller. Refer to the KS0108 datasheet.

10.4. Command 3

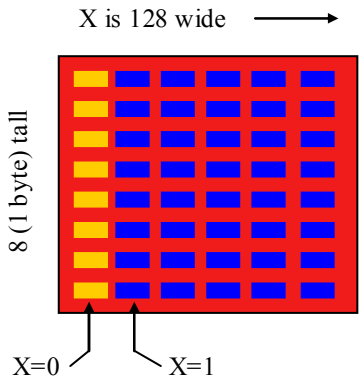
Name: **Write byte**

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

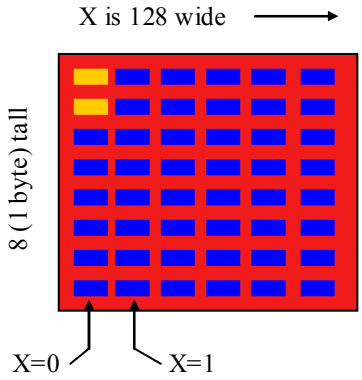
This will write a byte directly to the display at the current position. The current position at reset is row 0, x=0



To illustrate this point the above shows how the bytes on the display are laid out for half of the display. Each one of the blue lines consists of 64 bytes, 8 bits high side by side.



This is a 'zoomed' in view of the left hand side of just the top line. If the current row is 0 and the current value for X is zero, writing 0xff will turn on all of the pixels on that byte.



Writing 0x3 to the same location would produce this result.

This command is most useful for creating bitmaps as it writes directly to the LCD controller it will work as fast as the I2C interface can go.

Once the byte as been written the X value automatically increments.

10.5. Command 4

Name: **Write byte**

Format: <**S-addr**><**cmd**><**r**><**byte...**><**Stop**>

This will read the current byte on the display, the actual byte read can be set using commands 7 & 8.

10.6. Command 5

Name: **Reset**

Format: <**S-addr**><**cmd**><**Stop**>

(Time 45ms)

This command clears the display by writing 0x0 to all of the bytes and places the next read or write position to the beginning, top left.

10.7. Command 6

Name: **Back Light**

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

Turns the back light on or off as determined by the byte after the command set to 1 or 0.

10.8. Command 7

Name: **Set X Position**

Format: <**S-addr**><**cmd**><**0 to 127**><**Stop**>

See command 3 for a detailed explanation of what the X value is. Zero is far left and 127 is far right.

10.9. Command 8

Name: **Set R Position**

Format: <**S-addr**><**cmd**><**0 to 7**><**Stop**>

Sets the current row value. This should not be confused with the Y value. The row value is 8 pixels high. Zero is the top most row and 7 is the bottom row.

10.10. Command 9

Name: **X,R,Byte**

Format: <**S-addr**><**cmd**><**X**><**R**><**byte**><**Stop**>

This command combines commands 7, 8 and 3 into one command.

10.11. Command 0xa

Name: **X,Y,Byte**

Format: <**S-addr**><**cmd**><**X**><**Y**><**byte**><**Stop**>

This command will enable a byte (8 pixels high) to be placed anywhere on the X-Y grid and not be restricted to row positions as command 9. This is convenient for bit mapping but the will be a small time penalty as the I2C processor has to read two bytes before calculating the correct pixel values when crossing row boundaries.

For fast bit mapping it is recommended that command 9 be used.

10.12. Command 0x11

Name: **X,R,Pixel**

Format: <**S-addr**><**cmd**><**X**><**R**><**1 or 0**><**Stop**>

This will turn on (1) or turn off (0) a pixel at a particular X,Y co-ordinate. This is a complex internal command that will require reading two bytes before calculating values and writing up to two bytes, where the bytes cross a row boundary.

The consequences of this is that it will take a finite time to complete and so mapping hundreds of pixels this way will take more time than directly writing to the display using commands 3 or 9.

10.13. Command 0x12

Name: **Draw Horizontal Line (white)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**Length**><**Stop**>

(time – 18ms for 127 long line, shorter lines take proportionately less time)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a horizontal line by turning on the pixels.

The line will emanate to the right of the starting co-ordinates. The maximum length of the line is 127.

10.14. Command 0x13

Name: **Draw Horizontal Line (dark)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**Length**><**Stop**>

(time – 18ms for 127 long line, shorter lines take proportionately less time)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a horizontal line by turning off the pixels.

The line will emanate to the right of the starting co-ordinates. The maximum length of the line is 127.

10.15. Command 0x14

Name: **Draw Vertical Line (white)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**Length**><**Stop**>

(time – 18ms for 127 long line, shorter lines take proportionately less time)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a vertical line by turning on the pixels.

The line will be drawn downwards of the starting co-ordinates. The maximum length of the line is 63.

10.16. Command 0x15

Name: **Draw Vertical Line (dark)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**Length**><**Stop**>

(time – 18ms for 127 long line, shorter lines take proportionately less time)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a vertical line by turning off the pixels.

The line will be drawn downwards of the starting co-ordinates. The maximum length of the line is 63.

10.17. Command 0x16

Name: **Draw Box No Fill (white)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**X-Length**><**Y-Length**><**Stop**>

(time – This uses the line drawing functions, consideration should be given to the length of time the command takes that in turn depends on the size of the box)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a box by turning on the pixels.

The x,y co-ordinates specify the top left corner of the box. The lines used to make the box are 1 pixel wide.

10.18. Command 0x17

Name: **Draw Box No Fill (dark)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**X-Length**><**Y-Length**><**Stop**>

(time – This uses the line drawing functions, consideration should be given to the length of time the command takes that in turn depends on the size of the box)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a box by turning off the pixels.

The x,y co-ordinates specify the top left corner of the box. The lines used to make the box are 1 pixel wide.

10.19. Command 0x18

Name: **Draw Box With Fill (white)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**X-Length**><**Y-Length**><**Stop**>

(time – This uses the line drawing functions, consideration should be given to the length of time the command takes that in turn depends on the size of the box)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a box by turning on the pixels. The pixels within the box are also turned on

The x,y co-ordinates specify the top left corner of the box.

10.20. Command 0x19

Name: **Draw Box With Fill (dark)**

Format: <**S-addr**><**cmd**><**X**><**Y**><**X-Length**><**Y-Length**><**Stop**>

(time – This uses the line drawing functions, consideration should be given to the length of time the command takes that in turn depends on the size of the box)

Given the starting co-ordinates anywhere on the x,y grid, this will draw a box by turning off the pixels. The pixels within the box are also turned off

The x,y co-ordinates specify the top left corner of the box.

10.21. Command 0x20

Name: **Character (white)**

Format: <**S-addr**><**cmd**><**ASCII-byte...**><**Stop**>

Places a character on the X,R position set by command 0x25. At reset the X,R position is 0,0. This command expects a byte that represents the ASCII code of that character thus using 0x41 will be 'A'.

Once a character is placed the X,R value is incremented ready for the placement of the next character. Continually placing characters will overflow on to the next line below, when the end of the last line is reached the next placement will be the top left of the screen.

More than one byte can be sent at any one time before the stop command, thus a sting can be placed using a single command.

10.22. Command 0x21

Name: **Character (dark)**

Format: <**S-addr**><**cmd**><**ASCII-byte...**><**Stop**>

As command 0x20 but the character bitmap is inverted.

10.23. Command 0x22

Name: **Character (white)**

Format: <**S-addr**><**cmd**><**0 – 34**><**Stop**>

User characters are stored in EEPROM from location 0x30 onwards and each character occupies 6 bytes so 0=0x30, 1=0x36 etc.

This command will display the character defined in the EEPROM space, The factory setting defines 6 of these as examples and can be addresses by using 0 to 6 as the second parameter to the command.

To create a user defined character the system command 0x91 is used to place the 6 bytes in EEPROM starting at location 0x30. A character is made up of placing 6 bytes side by side that will be displayed vertically, bit 0 is at the top and bit 7 is at the bottom. To produce a horizontal line in the middle of the character for example, 6 bytes all of value 0x18 could be used. This will produce a line two pixels high.

Command 3 gives more details on how the bytes are arranged within the display.

10.24. Command 0x23

Name: **Character (dark)**

Format: <**S-addr**><**cmd**><**0 – 34**><**Stop**>

This is the inverse of 0x22 where bits having a value of '1' will appear dark instead of light.

10.25. Command 0x25

Name: **R, C Cursor Pos**

Format: <**S-addr**><**cmd**><**R**><**C**><**Stop**>

R (row) = 0 to 7

C (column) = 0 to 20

This positions the cursor (cursor is not visible) anywhere on the display for writing a character to.

For example to write a character somewhere in the middle of the display the following may be used:

BV4221 Example

0x42>s 25 3 10 p

0x42>s 22 41 p

The above moves the 'cursor' to the centre of the display and then outputs the character 'A'

10.26. Command 0x27

Name: **Get X Position**

Format: <**S-addr**><**cmd**><**r**><**byte**><**Stop**>

X = 0 to 127

The device keeps track of the X position as this is normally incremented when writing to the display.

The command returns that value. To calculate the column position when using text -1 and device by 6, thus:

C = (pos -1) / 6

Where pos is the value returned by this command.

10.27. Command 0x28

Name: **Get R Position**

Format: <**S-addr**><**cmd**><**r**><**byte**><**Stop**>

R = 0 to 7

The device also keeps track of the row position, this command returns the current row.

11. Command Timing

The following table shows how long each command will take from the first start condition to the stop condition.

This device will use clock stretching; this means it will hold the clock line low until the command has finished. Providing the master I2C software takes this into account no additional delays will be required in the software.

Command	Description	Time
1	**Turn on/off display**	**290uS**
2	Direct command to the KS0108	300uS
3	Write byte	315uS
4	Read byte	426uS
5	Reset and clear display	44mS
6	Back light on/off	320uS
7	Set X position	350uS
8	Set R position	375uS
9	Set X,R and send byte	575uS
0x0a	Set X,Y and send byte	812uS
0x10	Fill display with byte	44mS
0x11	X,Y, Pixel	660uS
0x12	Draw Horizontal line (White)	[1]
0x13	Draw Horizontal line (Dark)	[1]
0x14	Draw Vertical line (White)	[1]
0x15	Draw Vertical line (Dark)	[1]
0x16	Draw Box no-fill (White)	[2]
0x17	Draw Box no-fill (Dark)	[2]
0x18	Draw Box with-fill (White)	[3]
0x19	Draw Box with-fill (Dark)	[3]
0x20	Character (White)	730uS
0x21	Character (Dark)	730uS
0x22	User defined char (White)	800uS
0x23	User defined char (Dark)	800uS
0x25	Set R,C position	615uS
0x27	Get X position	600uS
0x28	Get R position	600uS
System		
0x55	Test	6.7mS[4]
0x90	Read EEPROM	9.2mS[4]

0x91	Write EEPROM	10mS [5]
0x93	End of EEPROM	350uS
0x94	Enter sleep mode	170uS
0x95	Reset	170uS[6]
0x96	Factory Reset	9mS[6]
0x96	Change Device Address Temporary	[7]
0x99	Change Device Address Permanent	824uS
0xa0	Firmware version	7.2ms

Notes

[1] Setup time 650uS and then 140uS per pixel

[2] Formula is (273*(X+Y))+500 where X and Y are the lengths of the box sides.

[3] Formula is (142*X*Y)+500 where X and Y are the lengths of the box sides

[4] To return one byte

[5] To write one byte

[6] This is the time it takes to release the bus but the device may need longer to perform its internal start up routines.

[7] Device dependant

Rev	System Section Changes
Oct 2008	Preliminary
Dec 2008	Removed command 96

12. 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

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.

12.1. 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.

12.2. 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.

12.3. 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.

12.4. 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.

12.5. 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.

12.6. 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.

12.7. 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.

12.8. 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.

12.9. 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.

13. 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.

14. 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

14.1. 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:

S	0	1	0	0	0	0	1	0	A	Command	A	P

14.2. 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.

14.3. 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.

15. 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.

15.1. 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.

15.2. 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.

15.3. 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.