BV4206

I2C-GP I/O

Product specification February 2008 V0.a

©ByVac

Contents

Rev	Change
Feb 2008	Preliminary
Oct 2008	Command 2 channels missing in format, corrected text in command 99 and missing command descriptions
Dec 2008	Minor corrections

1. Introduction

The BV4206 is an I2C two wire compatible integrated circuit with 15 general purpose, user configurable input or output lines. In addition it has a Pulse Width Modulated output and Timer features.

2. Features

- I2C up to 400kHz
- Simple command set
- 15 inputs or outputs or mixed
- High current source/sink for direct LED drive (25mA)
- PWM running at 19.6 KHz
- Timer with internal or external clock source
- EEPROM for general use
- Sleep mode to save power
- Operating voltage 2.0 to 5.5V
- Current 1.8mA @ 5V
- Sleep current 200uA

3. Physical Specification

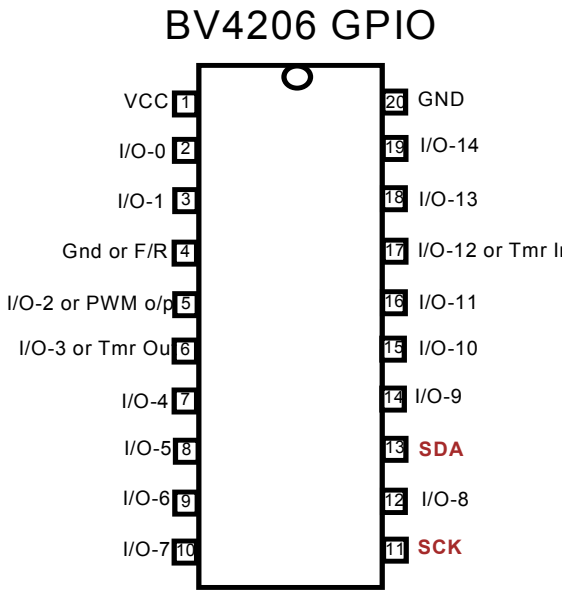


Figure 1 BV4206 Pin-out

Pin	Description
1	+2.0V to 5V
2	I/O 0
3	I/O 1
4	GND or Factory [1]
5	I/O 2 or PWM output
6	I/O 3 or Timer out
7	I/O 4
8	I/O 5
9	I/O 6
10	I/O 7
11	SCK: this is the I2C clock, a pull up resistor is required on this pin 5k6 will do.
12	I/O 8
13	SDA: this is the I2c data, a pull up resistor is required on this pin 5k6 will do.
14	I/O 9
15	I/O 10
16	I/O 11
17	I/O 12 or Timer input
18	I/O 13
19	I/O 14
20	Ground

Table 1 IC Pin Description

[1] This pin must be tied low for normal operation, see text.

The IC is a standard narrow 20 pin DIL with the pin designations as given in Table 1. The device comes with an I2C address of 0x42 but this can be changed at any time to any 8 bit value. The address is stored in an internal EEPROM. If the address was changed to some unknown value this can be reset back to the default by using the factory reset pin (4). See section 8 for further details.

Important for normal use pin 4 should be tied to ground if this is not done unpredictable results will be obtained.

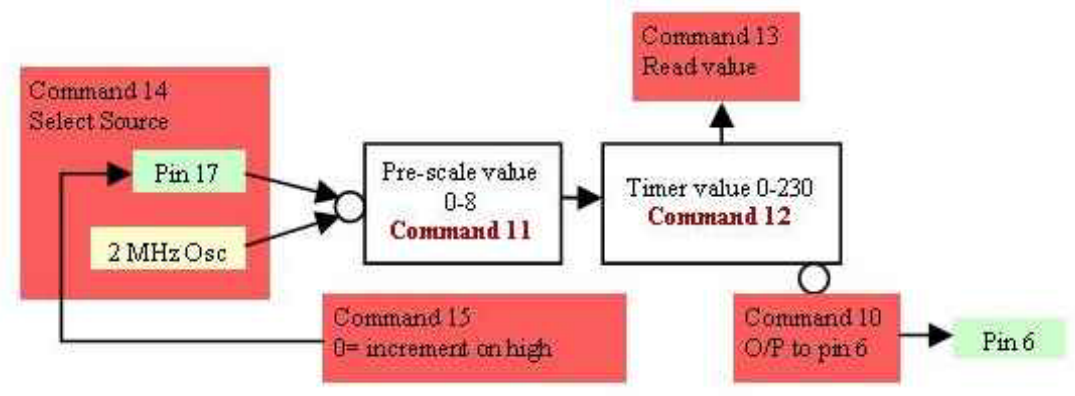
3.1. Digital I/O

Any of the I/O pins can be configured to be either an input or an output. This is done with command 1. In output mode any pin can either source or sink up to 25mA.

When set to input, some of the pins have an internal weak pull up that can be configured with command 4. By default the weak pull up feature is switched off.

3.2. Timer

There is a timer that can be driven from an external source or the internal 2MHz oscillator. The value of the timer register can be read from the I2C bus or the output can be fed to an external pin. The external pin will toggle at each overflow of the register.



Commands Used With Timer

The timer is working all of the time but by default it is set to get it's input from pin 17 so reading the timer value with command 13 will not reveal any changes unless there is some change on pin 17 which is set to input at reset.

The timer consists of an 8 bit register and an 8 bit presaler. The values of the prescaler can be set to the following:

Value	Rate
0	1:2
1	1:4
2	1:8
3	1:16
4	1:32
5	1:64
6	1:128
7	1:256
8	Off (1:1)

Table 2 Prescaler values

Note that value 8 will trun off the prescaler and allow the clock source to go straight to the timer counter.

The timer value can be loaded with any value from 0 to 230. This value is loaded into the counter which then increments until it overflows at value 255. At this point, if set the output at pin 6 will toggle and the value will be loaded into the timer again.

The result of this is that a value of 0 will give the largest delay (lowest frequency) and 230 will give the highest delay.

Prescale (11)	Timer (12)	Period at 6
8	230	50uS
8	0	275uS
7	0	65mS

Table 3 Example timer periods

This table gives some examples of the output at pin 6 when the internal clock is used. The period is one full cycle, i.e. pin 6 goes from low to high and then back to low.

See the scope output: this is for the lowest frequency when the prescaler is set at 7 and the timer is set to 0. The period shown in Figure 2 is 65mS.

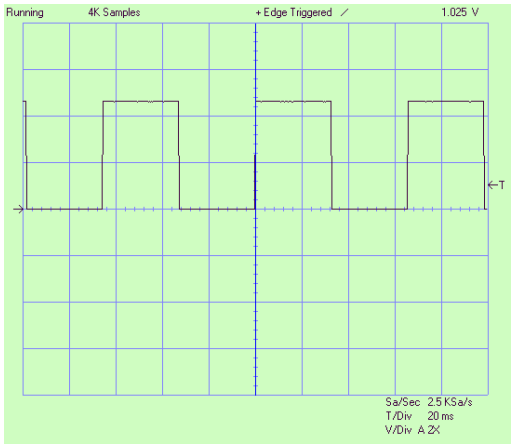


Figure 2 Output at pin 6 for longest period example

Command 13 will automatically set pin 6 (i/o 3) to be an output and using command 2 to set it either high or low will make no difference.

3.3. PWM

The pulse width modulator out put at pin5 (i/o 2) is activated by command 0x20. This sets pin 5 to be an output and fixes the PWM frequency at 19.61 KHz. The duty cycle within that period is set by command 0x21. This is 9 bits with the minimum value of 0 (off) and a maximum value of 408 (full on).

Two bytes are required to set this value, the high byte is sent first.

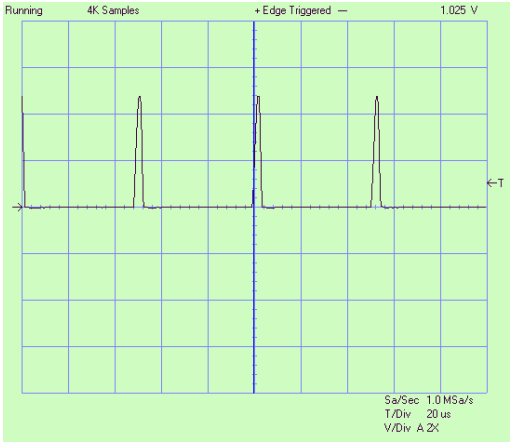


Figure 3 PWM value 10

The example output from pin 5 is with the duty cycle (command 0x21) set at 10. Command 20 has been used with a parameter of 1 so that high is ON.

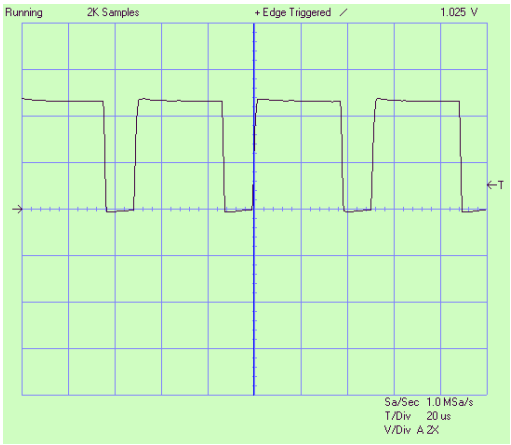


Figure 4 PWM value 300

As the PWM value increases so more of the time is spent in the 'on' state. A value of 408 would be fully high with no pulses.

The 'On' state can be specified with command 0x20 and this can be either high or low.

4. EEPROM

The device contains a 256 byte EEPROM that can be read and written to by the user for any purpose. The first 16 bytes are reserved for system use. The EEPROM has 1,000,000 Write endurances.

5. I2C Interface

The I2C interface is provided on pins 11 and 13, pull up resistors should be used for this to work correctly.

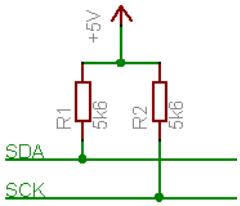


Figure 5 I2C Bus pull up resistors

6. 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 General purpose Input Output (GPIO) and those referring to the system. The system commands enable changing of the device address etc.

Device default address is 0x42

Command	Command Set
1	Set input or output
2	Set pin high or low
3	Read pin
4	Weak pull ups
0x10	Timer o/p control
0x11	Timer prescaler
0x12	Timer value
0x13	Timer read value
0x14	Timer source
0x15	Timer pin 17 control
0x20	PWM activate
0x21	PWM duty cycle

Table 4 ADC Command Set

Table 4 is a command summary of all of the GPIO and system commands.

The method of writing to the GPIO 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 than 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.

For more information about the command formats see section 9.

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.

6.1. Command 1

Name: **Set I/O**

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

The channel is a value from 0 to 14 and refers to the I/O number for a given pin. If this is followed by a 0 then that pin will be an output, if followed by a 1 then it will be an input.

Thus to set pin 16 to an output, **channel** will be 11 and this will be followed by a **0**.

S	0x42	1	11	0	P

At reset all of the pins are configured for input.

6.2. Command 2

Name: **Set Pin (channel)**

Format: <**S-addr**><**2**><**channel**><**1 or 0**><**Stop**>

This will only work when the selected pin is set to an output using command 1. The pin will reflect the second byte thus if the second byte were a 0 the pin would be low.

Assuming that pin 6 (i/o 3) has been set to an output then using channel 3 followed by 1 will set that pin high.

S	0x42	2	3	1	P

6.3. Command 3

Name: **Read Pin**

Format:<**S-addr**><**3**><**channel**><**R-Addr**><**bit**><**Stop**>

This returns the value on the selected pin (channel) which will be either 0 or 1.

To see the value on channel 5, pin 8 the following would be required.

S	0x42	3	5	S	0x43	Byte	P

The Byte would either be a 1 or 0.

6.4. Command 4

Name: **Weak pull ups**

Format: < **S-addr**><**4**><**0 or 1**><**Stop**>

This enables (1) or disables (0) weak pull ups on the following channels.

0,1,7,8,12,13 and 14

At reset the week pull ups are disabled. The effect of this can be seen when reading the pins using command 3.

To turn on weak pull ups:

S	0x42	4	1	P

6.5. Command 0x10

Name: **Timer o/p Control**

Format: <**S-addr**><**0x10**><**0 or 1**><**Stop**>

By default pin 6 is an input, using this command with '1' will set this pin to an output and each time the timer overflows it will toggle this pin.

Once this command is used to set pin 6 to be a timer output it will remain an output until command 1 is used to set it back to an input.

If '0' is used the pin will remain an output but will not be toggled by the timer.

For pin 6 to be the o/p of the timer the following is used:

S	0x42	0x10	1	P

0 sets pin 6 as normal i/o pin (default)

1 sets pin 6 to be o/p from timer

6.6. Command 0x11

Name: **Timer prescaler**

Format: <**S-addr**><**0x11**><**0 to 8**><**Stop**>

This sets the prescaler value, see Table 2 for values.

To turn off the prescaler so that the timer input is fed straight to the timer register use:

S	0x42	0x11	8	P

6.7. Command 0x12

Name: **Timer value**

Format: <**S-addr**><**0x12**><**0 to 230**><**Stop**>

This value along with the prescaler value will set the time frequency or period. Table 3 gives some examples of what can be expected for the various combinations of values.

Note that the maximum value for this is 230, a value larger than this will be ignored.

6.8. Command 0x13

Name: **Read Timer**

Format: <**S-addr**><**0x13**><**R-Addr**><**value**><**Stop**>

This will read the value of the timer register. The timer register is loaded with the value set by command 0x12, it then increments until it reached 255 and then is loaded again with the value set by command 0x12.

The minimum value that can be read is therefore set by command 0x12.

6.9. Command 0x14

Name: **Timer Source**

Format: <**S-addr**><**0x14**><**0 or 1**><**Stop**>

The timer can be fed from the internal oscillator or an external pin (pin 17). This command select the source of the timer.

At reset the source is from pin 17 that corresponds to '0' for the second byte. A 1 will set the source to the internal oscillator.

0 sets input from pin 17 (default)

1 sets input from internal oscillator

6.10. Command 0x15

Name: **Timer External i/p control**

Format: <**S-addr**><**0x15**><**0 or 1**><**Stop**>

When the timer source is provided externally through pin 17, this controls if the timer is incremented from a low to high transition or a high to low transition.

0 = increment on low-high

1 = increment on high-low.

6.11. Command 0x20

Name: **PWM Activate**

Format: <**S-addr**><**0x20**><**0 or 1**><**Stop**>

This command will set pin 5 to be an output and start the PWM working.

Using a 1 as the second byte will make the pin high when the PWM is fully on. Using a 0 as the second byte will make the PWM low when fully on.

The PWM period is set to 19.61kHz.

6.12. Command 0x21

Name: **PWM Pulse Width**

Format: <**S-addr**>
<**0x21**><**high**><**low**><**Stop**>

The width of the pulse within the PWM period is determined by this value made up of a high byte and a low byte. The actual value can be between 0 and 408. 0 is fully off and 408 is fully on, larger numbers than 408 will not make any difference.

The value must be sent as two bytes. As an example a value of 290 is 0x122 so the first byte would be 0x01 and the second 0x22.

e.g.

S	0x42	0x21	1	0x22	P

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

7. 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.

7.1. 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.

7.2. 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.

7.3. 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.

7.4. 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.

7.5. 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.

7.6. 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.

7.7. 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.

7.8. 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.

7.9. 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.

7.10. 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.

8. 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.

9. 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

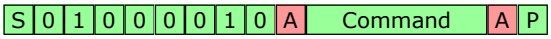
N Not acknowledge = 0

9.1. 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:

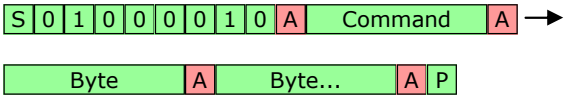


9.2. Sending a command with a parameter byte/s

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.



9.3. 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.

10. 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.

10.1. 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.

1

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.

10.2. 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.

10.3. 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.