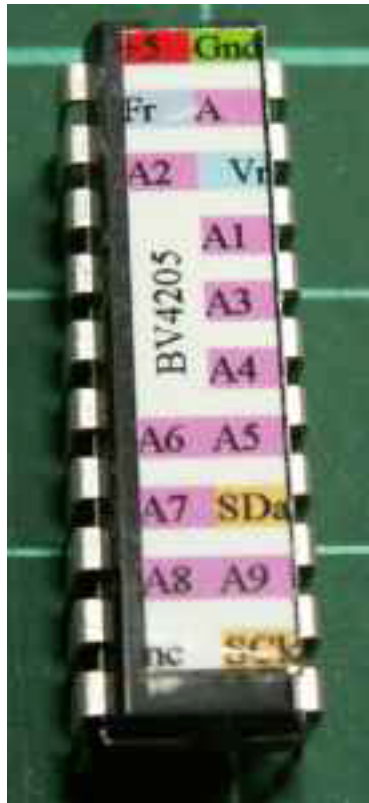

I2C-10 Channel A to D

BV4205



BV4205

I2C-10 Channel A to D

Product specification

January 2008 V0.a

I2C-10 Channel A to D**BV4205****Contents**

1.	Introduction	4
2.	Features	4
3.	Physical Specification	4
3.1.	Factory (hardware) reset	4
3.2.	Analogue Inputs	4
3.3.	Voltage Reference.....	5
3.4.	Analogue Conversion.....	5
4.	EEPROM	5
5.	I2C Interface.....	5
6.	I2C Command set.....	5
6.1.	Command 1	6
6.2.	Command 2	6
6.3.	Command 3	6
6.4.	Command 4	6
6.5.	Command 5	6
6.6.	Command 6	6
6.7.	Command 7	6
6.8.	Command 8	6
6.9.	Command 9	6
6.10.	Command 10	6
6.11.	Command 11	6
6.12.	Command 12	6
7.	Electrical Specifications	6
8.	I2C System Commands	6
8.1.	0x55	6
8.2.	Command 0x90.....	6
8.3.	Command 0x91.....	6
8.4.	Command 0x93.....	6
8.5.	Command 0x94.....	6
8.6.	Command 0x95.....	6
8.7.	Command 0x98.....	6
8.8.	Command 0x99.....	6
8.9.	Command 0xA0.....	6
8.10.	Command 0xA1.....	6
9.	Hardware Reset.....	6
10.	Command Diagrams.....	6
10.1.	Sending a single command	6
10.2.	Sending a command with a parameter byte/s	6
10.3.	Receiving bytes from the salve.....	6
11.	Trouble Shooting	6
11.1.	Pulse Stretching	6

I2C-10 Channel A to D

BV4205

11.2. Last Read NACK6

11.3. Pull Up's6

I2C-10 Channel A to D**BV4205**

Rev	Change
Jan 2008	Preliminary
Oct 2008	Manual errors
Oct 2008	Error with description of command 7, channel to read should be doubled.
Dec 2008	Version 2.a Default device address: 0x62
April 2009	Added Electrical Specification
April 2011	Version 2.b

1. Introduction

The BV4205 is an I2C two wire compatible integrated circuit with a 10 channel 10 bit A to D converter. The IC works independently of the microcontroller and can acquire analogue signals continuously until required.

2. Features

- I2C up to 400kHz
- Simple command set
- 10 channels x 10 bits
- Differential input ± 1023
- 2 μ s Conversion time
- 5 μ s Acquisition time
- Can continuously monitor all channels
- Vref selectable
- 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

Pin	Description
1	+2.0V to 5V
2	n/c
3	AN2
4	Ground or Factory [1]
5	n/c
6	n/c
7	AN6
8	AN7
9	AN6
10	n/c
11	SCK: this is the I2C clock, a pull up resistor is required on this pin 5k6 will do.
12	AN9
13	SDA: this is the I2c data, a pull up resistor is required on this pin 5k6 will do.
14	AN5
15	AN4
16	AN3
17	AN1
18	Vref
19	AN0
20	Ground

NOTES[1] This pin must be connected to ground for proper operation

Table 1 IC Pin Description

The IC is a standard narrow 20 pin with the pin designations as given in Table 1. The device comes with an I2C address of 0x62 but this can be changed at any time to any 8 bit value. The address is stored in an internal EEPROM.

3.1. Factory (hardware) reset

If the address was changed to some unknown value this can be reset back to the default by using pin 4. It is important this pin be connected to ground for normal use. To reset back to the factory address (0x62), connect this pin to +V and follow the instructions in section 9.

3.2. Analogue Inputs

There are 10 analogue inputs AN0 to AN9, internally these are multiplexed onto a single A to D converter. The conversion time by default is set to the minimum 2uS but this can be increased if required. The converter has a 10 bit resolution (0 to 1023) and is read in a field width of 16 bits (2 bytes). The reference for the converter is software selectable, see the next section.

The maximum source impedance for each of the channels is 10k.

I2C-10 Channel A to D

BV4205

3.3. Voltage Reference

By default the voltage reference is set to VDD (pin 1) but this can be selected by an I2C command so that the reference is taken from pin 18.

3.4. Analogue Conversion

There are two methods of acquiring analogue voltages from the available channels. The first method involves selecting the correct channel (command 1), issuing a convert command (command 2) and then reading the result (command 4). The second method is to tell the device to do the above steps continuously (command 6) and store the results internally. The values can then be read at any time with command 7 or 0xA.

The first method is useful low power applications where a sleep command can be issued between conversions. The second method will have minimum impact on the host, the result will be delivered as fast as the I2C bus with a single command. This method also allows the use of command 0x0A that will obtain the difference between two adjacent channels.

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.

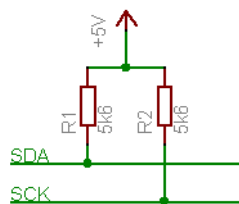


Figure 1 I2C Bus pull up resistors

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

Device default address is 0x62

Command	AtOD Command Set
---------	------------------

1	Select channel
2	Do conversion
3	Conversion status
4	Fetch result of conversion
5	ADC enable / disable
6	ADC autoscan
7	Read autoscan
8	Acquisition delay
9	Read autoscan differential
10	Result justification
11	Voltage ref. source
12	Conversion clock
0x55	Test
Command	System Command Set
0x90	Read EEPROM
0x91	Write EEPROM
0x92	Confirm Command Complete
0x93	End of EEPROM
0x94	Sleep mode
0x95	Reset
0x98	Temporarily change address
0x99	Change Device Address Permanent
0xA0	Firmware Version

Table 2 ADC & System Command Set

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

The method of writing to the ADC 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 (0x62). 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 0x62, 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 10.

I2C-10 Channel A to D

BV4205

Each command will have its 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 condition is received.
3. Receiving the stop condition will clear the buffer.

6.1. Command 1

Name: **Select Channel**

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

The channel is a value from 0 to 9. This command must be used prior to a convert command. It sets the required channel, at reset this is set to 0. Any conversion will refer to the channel number set by this command.

6.2. Command 2

Name: **Do Conversion**

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

The voltage on the channel selected by command 1 will be read and converted into a digital value with reference to the Vref voltage. Unless using the autoscan feature this must be done prior to reading the analogue value with command 4.

6.3. Command 3

Name: **Conversion Status**

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

This returns a value of 0 if the conversion has completed and is ready to be read using command 4 or 1 if the converter is still working. In practice it is unlikely that this will be needed as the conversion times are quicker than the I2C can obtain the data at 400kHz.

6.4. Command 4

Name: **Result of conversion**

Format: **< S-addr><4><R-Addr><high><low><Stop>**

This is the result of a conversion, the value is in a 16 bit field (see command 10) and so two bytes are fetched, high byte first.

To have meaningful results the correct channel number must be selected and a conversion (command 2) carried out first.

6.5. Command 5

Name: **ADC Enable**

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

By default the ADC is enabled but it is possible to disable it with this command. Sending 0 will disable the ADC and 1 will enable the ADC. When the ADC is disabled it uses slightly less power.

6.6. Command 6

Name: **ADC Autoscan**

Format: **<S-addr><6><0 or 1><Stop>**

This is disabled (set to 0) at reset. When enabled the device will scan all the 10 channels and store the results internally to RAM in a round robin fashion. With this working there is no need to select (command 1), convert (command 2) and then read (command 4) a channel as they are being monitored on a continuous basis.

Each channel takes approximately 25uS with an acquisition delay of 2 so all of the channels will be scanned in approximately 450uS.

Reading any channel will therefore be a maximum of 450uS old.

6.7. Command 7

Name: **Read Autoscan**

Format: **: < S-addr><7><start><R-Addr><data16...><NACK><Stop>**

Command 6 will carry out a conversion and store the results to 20 bytes of RAM contained within the chip. This command will read one or more of those ram locations.

Each read is 2 bytes followed by a ACK or NACK. If ACK is used it informs the IC that another two bytes will be read, if NACK is used it informs the IC that this will be the last byte pair.

Using this method one or more locations can be read. Start is the first value (byte pair) to read.

Example

To read channels 4 to 6 use a start value of 8 and read 3 byte pairs (6 bytes). To read just channel 8, use a start value of 16 and read just one byte pair.

6.8. Command 8

Name: **Acquisition delay**

Format: **<S-addr><8><1 to 255><Stop>**

The acquisition delay is a small delay that occurs before a channel is changed. This prevents one channel value from affecting the next channel value and mainly applies when using the Autoscan feature. The delay is in uS so setting this to 1 will give 1uS delay, the default is 10.

I2C-10 Channel A to D

BV4205

6.9. Command 9

Name: **Read Autoscan - Differential**

Format: : < **S-addr**><**0x09**><**Dchan**><**R-Addr**><**Byte-H**><**byte-L**><**Stop**>

This command returns the difference of two analogue channels represented as a signed 16bit integer giving a range of -1022 to +1023. Dchan is a value from 0 to 5 that represents channel pairs as follows:

DChan	Result
0	AN0 – AN1
1	AN2 – AN3
2	AN4 – AN5
3	AN6 – AN7
4	AN8 – AN9

The result is the effect of subtracting one channel from the other. For example of channel 2 has a value of 120 and channel 3 had a value of 500 the result of Dchan 1 would be -380 (minus). This would be 0xFE84 as a signed 16 bit number.

6.10. Command 10

Name: **Result justification**

Format: : < **S-addr**><**0x0a**><**1** or **0**><**Stop**>

By default the result is given in the lower part of the 16 bit value bits 9:0, this gives the most sensible result giving a value from 0 to 1023. However by setting this to 0 it is possible to have the results given in the upper half of the 16 bit value bits 15:6. This will give the values 65472 to 65535. When using this mode it is up to the user to mask off the lower 6 bits as they are undefined.

1 = Normal justification (default)

0 = Left (upper) justification

6.11. Command 11

Name: **Voltage Reference Source**

Format: : < **S-addr**><**0x0b**><**1** or **0**><**Stop**>

The voltage reference can come from two sources, the power supply +V or pin 18 (vRef). This command specifies that source.

0 = Voltage reference from +V (default)

1 = Voltage reference from vRef pin

6.12. Command 12

Name: **Conversion Clock**

Format: : < **S-addr**><**0x0c**><**0** to **2**><**Stop**>

Sets the conversion clock time as follows:

0 = 2uS (Default)

1 = 4uS

2 = 8uS

7. Electrical Specifications

	Min	Max
Supply Voltage V	3.0	5.5
Supply Current mA		2.4
Resolution		10bits
Integral error	±1 bit	
Full scale range V	2.2	5.5
Vref V	2.0	V+
Vref uA	0	±150
Input Voltage	0	V+
Recommended source impedance		10k

I2C-10 Channel A to D**BV4205**

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

8. I2C System Commands

The following section deals with system commands that are common to all I2C devices. Note that not all of these 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.

8.1. 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.

8.2. 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 EEPROM addresses 0, 1 and 2

This command will allow a single or several bytes to be read from a specified EEPROM address.

8.3. 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.

8.4. 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.

8.5. 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.

8.6. 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.

8.7. 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.

8.8. 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.

8.9. 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.

8.10. 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.

9. 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.

10. 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

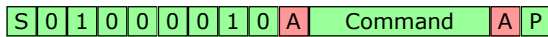
N Not acknowledge = 0

10.1. 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:

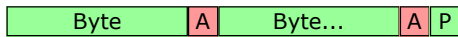
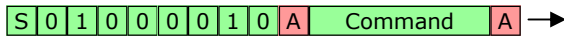


10.2. Sending a command with a parameter byte/s

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.



10.3. Receiving bytes from the slave

<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.

11. 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.

11.1. 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 and 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.

11.2. 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 than 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.

11.3. 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.

