QBridge-I2C-USB
Manual

Revision: 1.4
Date: 04-20-2007

1. Introduction

QBridge-I2C-USB is a fast and reliable I2C bus host adaptor via USB interface. It can simply turn any Windows based PC into an I2C master. No power is needed since the PC USB port can provide standard +5 V and current of as much as 500 mA.

2. Disclaimer

Q-Proto System will not be liable for direct, indirect, incidental, general or consequential damage from the use of the products, including software and hardware, from Q-Proto System. If you do not agree with these terms, do not buy the products.

While every effort has been made to ensure that the information contained in this document is accurate and complete, no liability can be accepted for any error or omission. Q-Proto System reserves the rights to change the specification of the hardware and software described herein at any times prior notice.

3. Quick Start

- Install **QBridge Control Center** application from the installation CD to your local drive. The latest version of the application can be downloaded from Q-Proto System Web site http://www.qprotos.com/downloads.htm.
- Connect the QBridge-I2C-USB device to your PC USB port.
- Install the Windows device driver for the QBridge-I2C-USB.
- Launch the QBridge control center application from Windows <Start> → <QBridge Control Center>
- Ready for use.

4. System requirements

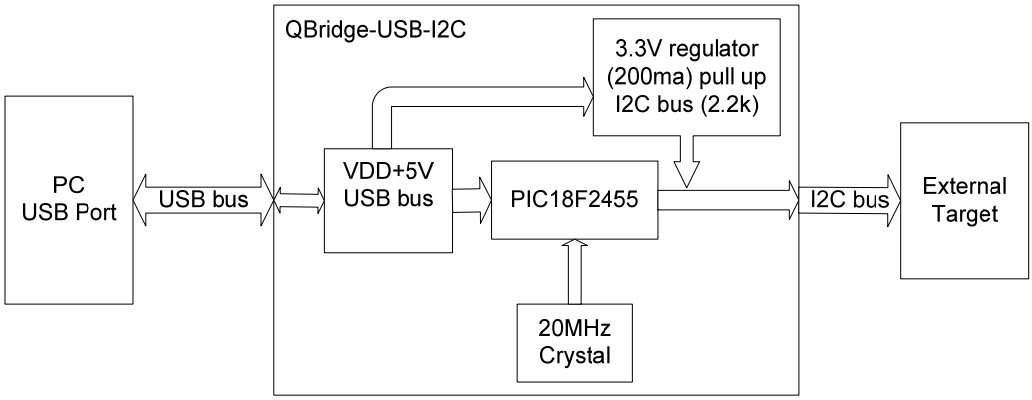
- Windows 2000, XP operating system.
- Minimum 256M RAM.
- One available USB port.

5. Features

- Microchip PIC18F2455 microprocessor with built in USB functionality.
- Self power by USB port for up to 500 mA.
- Internal 2.2K pull up for I2C bus with +5 V.
- Male DB 9 for easy external target connection.
- Windows version of QBridge I2C control panel included.
- Windows XP and 2000 device driver included.
- Documented USB to I2C API extends flexibility for user development.
- Configurable I2C speed between 100 K and 400 K.
- Field upgradeable for the application software via USB port.
- Example program written in Visual C++.
- LED for device status indication.

- Faster BULK USB transfer.

6. Diagram

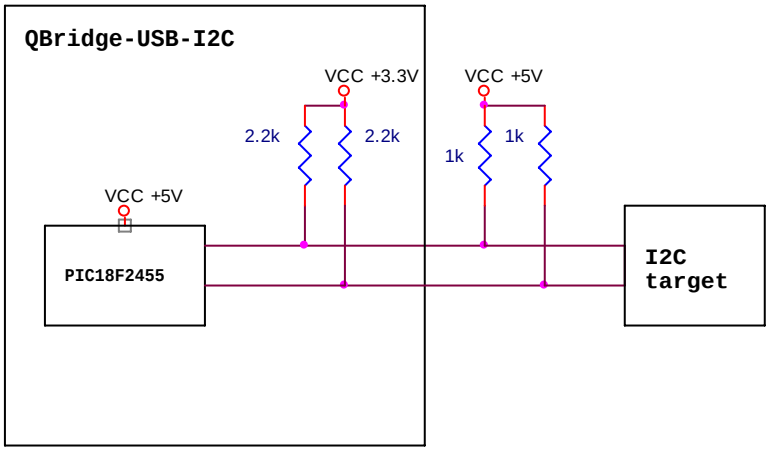


7. Hardware specification

7.1. I2C bus voltage level

The microcontroller PIC18F2455 is running off the USB port power at +5V VDD. There is an internal 3.3V regulator in the QBridge-USB to pull up the I2C bus at the 3.3V level. The maximum current draw for the 3.3 v regulator is up to 200 mA.

Because the PIC18F2455's VDD is +5 Volts, both SDA and SCK pins can still tolerate +5 Volts signal even they are internally pulled up to +3.3 Volts. The actual voltage level depends on the external pull up resistors. An example diagram is illustrated below:

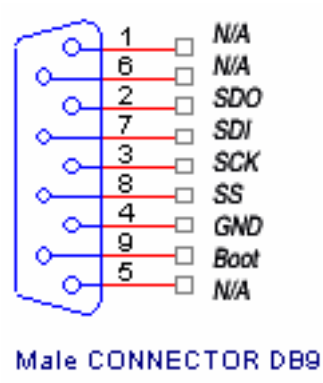


The actual voltage on the I2C bus is calculated as follow:

I2C Bus Voltage level = 3.3v+ (5v-3.3v)*(2.2k/(2.2k+1k) = 4.47 V

7.2. External target connector layout.

The QBridge-I2C-USB has two external ports. One is the USB port that connects to the PC, the other is the male DB 9 connector that connects to the external I2C target (may be enhanced to SPI target in the future). The male DB 9 pins layout is shown below.



1: N/A
2: SDO (used for SPI)
3: SCK
4: GND
5: N/A
6: N/A
7: SDA/SDI
8: SS (used for SPI)
9: Boot

If you use a female DB 9 connector or other break out boxes to connect to your target, the pin numbers could be reversed or changed. Incorrect connection may permanently damage the device.

7.3. LED indication.

The LED will perform power up sequence with on -> off 1 second -> on once the QBridge is connected to the USB port.

This LED can also be controlled through application software.

The LED will be OFF if the USB bus is in suspended state.

7.4. Power limitation.

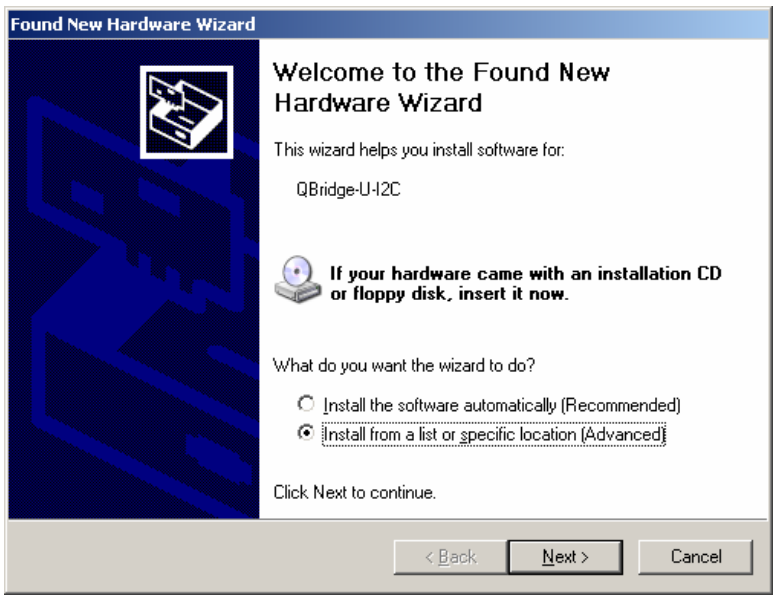
The QBridge device is powered by the PC USB. Therefore, no external power is required for the device.

The maximum current limit for each USB port is 500 mA according to the USB specification. This provides approximately 400 mA for the QBridge device and I2C bus to use. Therefore, this device is not desirable for the I2C bus requiring a large amount current.

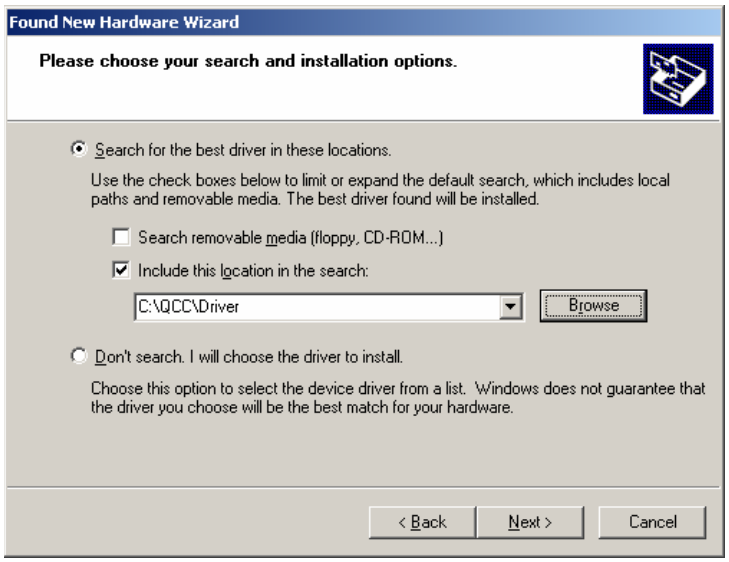
The internal pull up resistors inside the QBridge device for SCK and SDA are 2.2K.

8. Device driver installation

8.1. Plug in the QBridge device to your USB port through the USB A-B cable. The setup wizard should run automatically and a Welcome screen should appear as follows. Select option <Install from a list or specified location (Advanced)>. Then click the <Next> button.



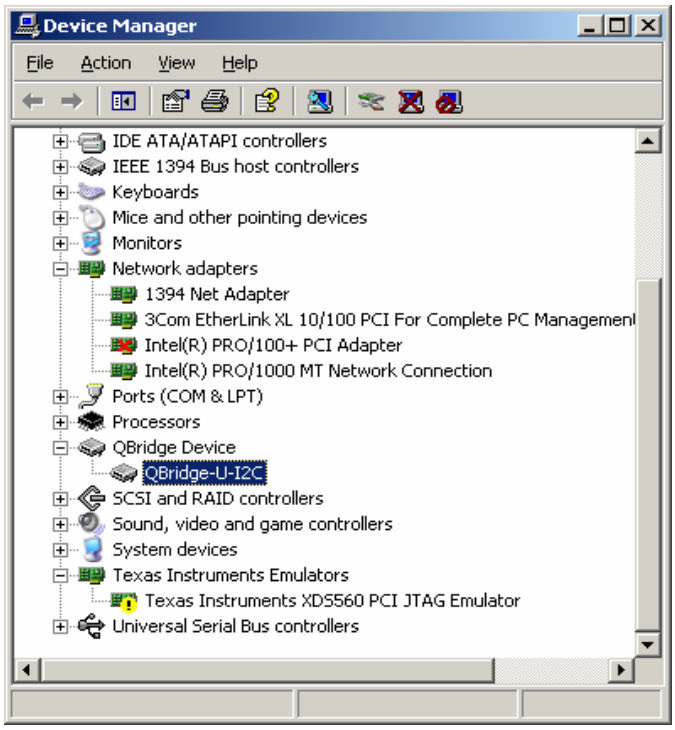
8.2. Select option <Search for the best driver in these locations>. Set the search location to the directory where the QBridge driver is located, i.e. c:\QCC\driver if you have installed to the c:\QCC (default). Then click the <Next> button.



8.3. Occasionally, on some operating systems, you may get a warning of “device driver is not certified”. This is because QBridge USB driver has not been certified from Microsoft at this time. However, this warning should not affect the operation of your system. Click the <Continues Anyway> button to continue installation.

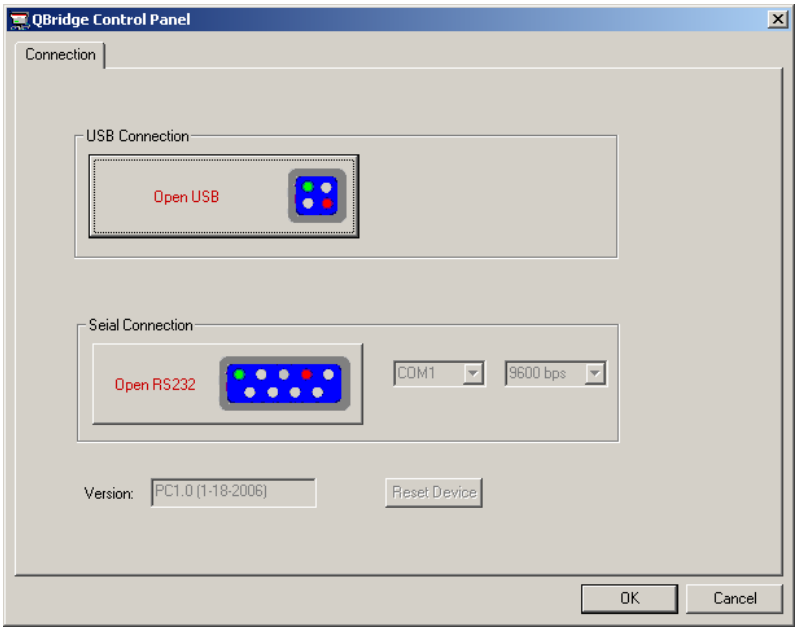
8.4. Click the <Finish> button to complete the installation after the hardware wizard finishes installing QuickCAN device driver.

8.5. To verify the successful installation of the driver, open Windows <control panel> → <System> → <Hardware> → <Device Manager>, the device **QBridge-U-I2C** shall be shown under “QBridge device” tree.



9. QBridge control panel

9.1. Connection panel

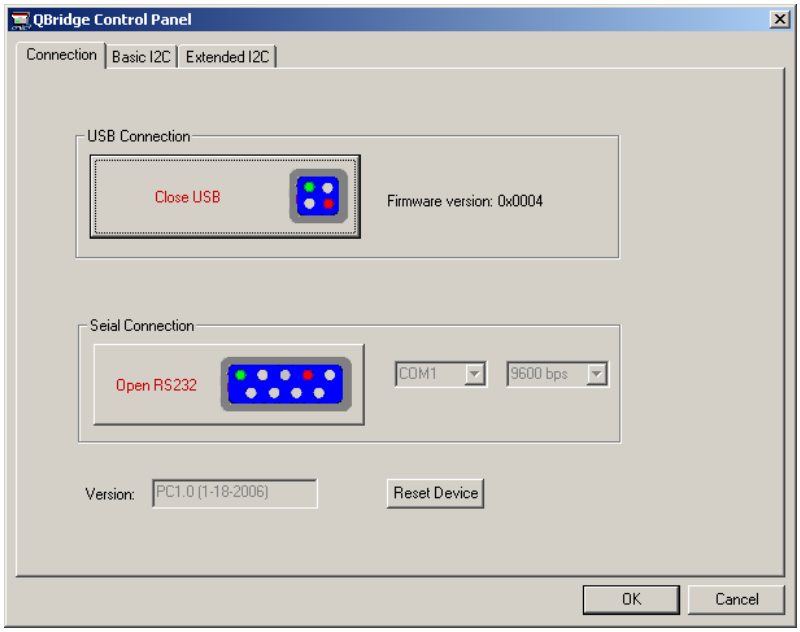


The connection panel is the first Window when the QBridge control application is launched. This window allows users to connect the QBridge Control Center to the actual hardware.

- **USB Connection**

  Click the <Open USB> button to establish connection to the QBridge-I2C-USB device. The QBridge device should have been plugged to the PC USB port.

  Once the connection is successfully established (see picture below), the <Open USB> button becomes the <Close USB> button, the firmware version from the QBridge device is shown in the panel and two tabs, <Basic I2C> and <Extended I2C>, are displayed.



- **Serial Connection**

  Not supported now.

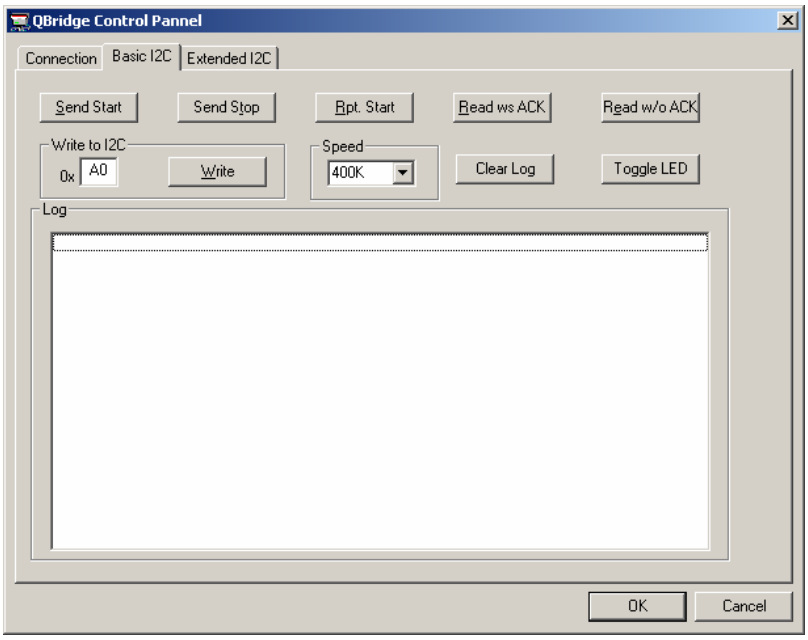
- **Version**

  Release version of the QBridge control panel.

- **Reset Device**

  Reset the device. The QBridge device will take 3-4 seconds to come out of the reset state. The USB connection will be closed automatically when the button is pressed.

9.2. Basic I2C function



The Basic I2C panel allows users to perform all the basic I2C operations such as send SATRT, send STOP, write one bye, read one byte and etc.

The log records the status of the command and response.

- **Send START**

  Send the START bit.

- **Send STOP**

  Send the STOP bit.

- **Rpt START**

  Send the repeated SATRT bit.

- **Read w/ ACK**

  Read one byte from the target with acknowledgement.

- **Read w/o ACK**

  Read one byte from the target without acknowledgement.

- **Write to I2C**

  Write data to I2C

- **Toggle LED**

  Toggle the LED on/off.

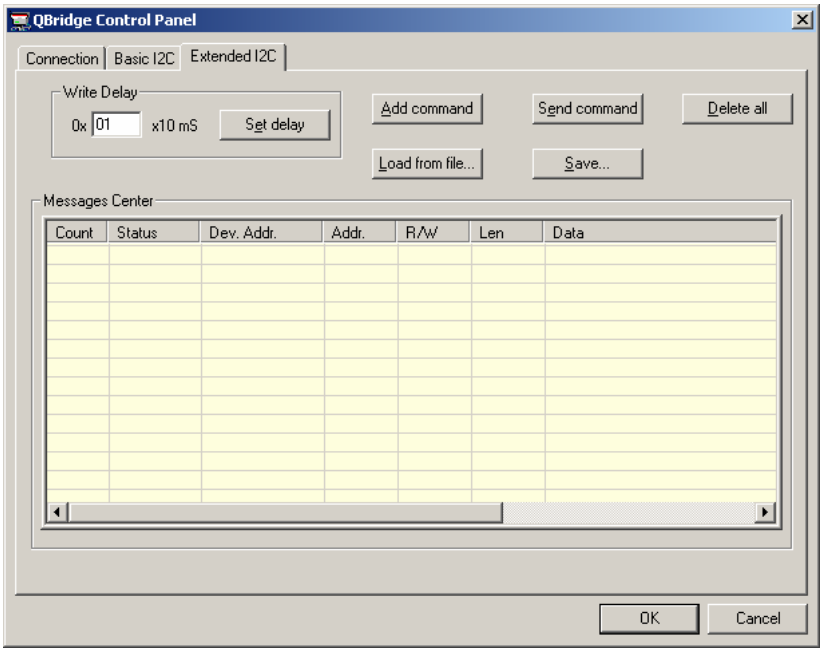
- **Clear Log**

  Clear the log content.

- **Speed**

  Change the I2C bus speed.

9.3. Extended I2C function



Extended I2C panel allows users to perform more sophisticated functions like get a string from a specified address or write a string to a specified location of the I2C target.

Users can also load the initial data to the <Messages Center> from an ASCII file. File format will be described below.

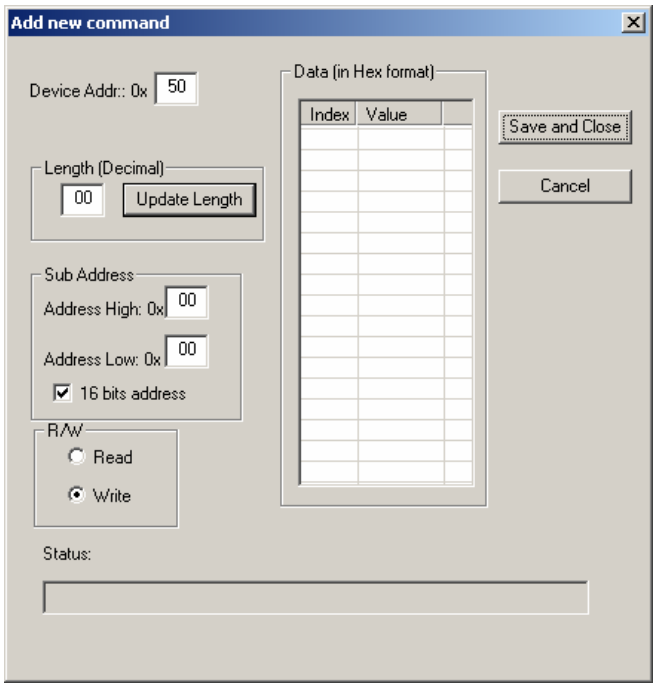
- **Write delay**

  Set the interval between the writes when performing the Write operation.
  The default delay is 0mS. The delay can be incremented in the basis of 10mS.

- **Add command**

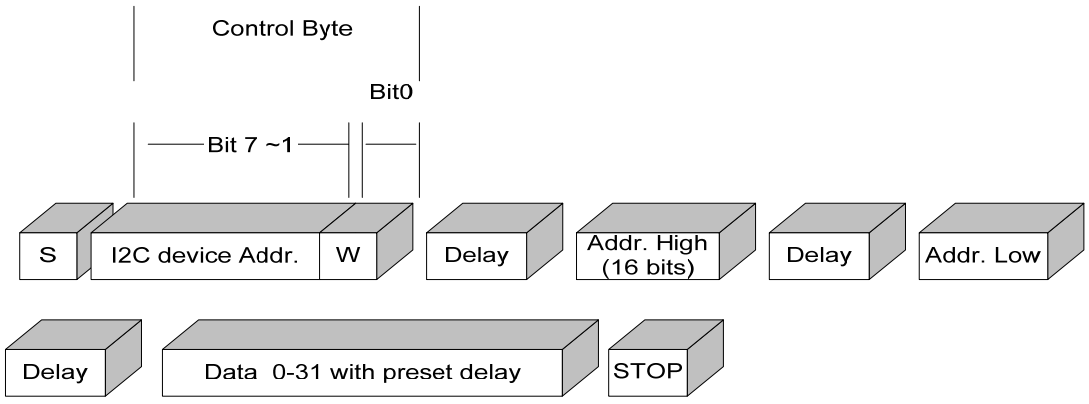
  Compose the READ/WRITE command and save the command to <Messages Center>. The <Command Dialog> window will be shown when this button is pressed. Users can specify the I2C device address, sub-address, read or write operation and data. The START and STOP bit are set by the application in the command.

This command is useful when performing the I2C based EEPROM write or read operation.



Note that the **Device address** is entered instead of **Control Byte** in QCC version 1.2 or later. The QCC will form a control byte based on the device address and read/write operation.

1. Write operation:

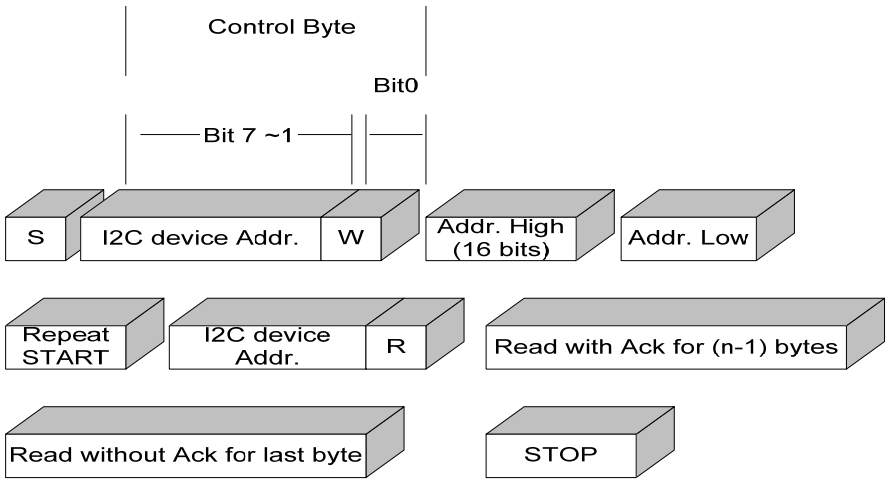


S: Start Bit
W: 0 for write operation
Delay: Preset delay
STOP: STOP bit

According to the I2C specification, the WRITE control byte is formed by the I2C slave address on the upper 7 bits and zero at the LSB.

A preset interval (write delay) will be inserted between data bytes.

2. Read operation



S: Start Bit
W: 0 for write operation
R: 1 for READ operation
Delay: Preset delay
Repeat START: Repeat START bit
STOP: STOP bit

Note: There is no delay for the read operation.

Based on the I2C specification, The READ control byte is formed by I2C salve address on the upper 7 bits and one at the LSB.

- **Send command**

  Send the selected command to the I2C target. The **Status** column will be updated to reflect the status of the execution.

- **Delete all**

  Delete all entries from the <Messages Center>.

- **Load from file**

  Load predefined commands from a file.

  The format of the file is defined as following:

1. Any line proceeding with '#' is comment.
2. Data must be defined in HEX format.
3. Each byte of data is separated by ','.
4. The order of the data is <control addr> <addr high> <addr low> <16 bits addr mode> <read/write> <length> <data0>...<data31>.

   <16 bits addr mode> field

   1: 16 bit address mode 0: 8 bit address mode

   <read/write> field

   1: read operation 0: write operation

Example:

#comment <control addr> <addr high> <addr low> <16 bits addr mode> <read/write> <length> <data0>...<data31>
0xa1,0x00,0x10,1,0,10,1,2,3,4,5,6,7,8,9,10 <← 16 bits mode, address 0x0010, write operation, length is 10, data is 1,2,3,4,5,6,7,8,9,10

0xa1,0x13,0x54,0,1,10,11,2f,3,4e,5,6,7,8,9,10 <← 8 bits address mode, address 0x54, read operation, length is 10 bytes, data is ignored.

- **Save…**

  Save all data in the <Messages Center> to an ASCII file. This file can be reloaded later by clicking **< Load from file>**.

10. USB to I2C API

The QBridge-I2C-USB device provides a set of communication protocol for users to develop their own applications. This allows users to easily create customized applications to control the I2C bus.

The communication protocol is asynchronous, which means that there is one command and one response. No command shall be issued before the previous response is received.

- All commands start with command identifier '@' followed by QBridge command byte and ended with checksum.
- All responses start with '@' followed by QBridge command byte plus returned data, if any, and the checksum.
- If any error occurs during the operation, the first two bytes of the response are '@e' followed by the error code and checksum.
- The user application shall check for error status from the response before proceeding to the next operation.

The XOR checksum method is applied to all the commands and responses. For example, if a request 0x40 0x41 is sent to the device, a checksum byte 0x01 will be appended to the original data. Therefore the real time outgoing data is 0x40 0x41 0x01; the same XOR method applies to responses from the device.

Error code from the QBridge firmware

Error code	Description
0	Success
1	No data from target
2	Collision when writing to target
3	I2C Bus collision
4	No ACK from the I2C target
5	Timeout when ACK to the target
6	Time out when writing to the target
7	Error when sending START bit
8	Error when sending STOP bit
9	Error when sending repeated START
10	Unrecognized command
11	Incorrect checksum
12	Missing command identifier '@'
13	Time out when waiting for the TX completion
14	Unexpected error

10.1. Command 0x00

Description: read the firmware version.

Format:

Byte 0	Byte 1	Byte 2

@	0x00	Checksum

Normal response:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
@	0x00	Major version	Minor version	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.2. Command 0x01

Description: read the boot firmware version.

Format:

Byte 0	Byte 1	Byte 2
@	0x01	Checksum

Normal response:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
@	0x01	Major version	Minor version	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.3. Command ‘O’

Description: toggle the LED.

Format:

Byte 0	Byte 1	Byte 2
@	‘O’	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	‘O’	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.4. Command 'N'

Description: open the I2C port. This function must be called before other I2C operations can be performed.

Format:

Byte 0	Byte 1	Byte 2
@	'N'	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	'N'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.5. Command 'C',

Description: Close the I2C port.

Format:

Byte 0	Byte 1	Byte 2
@	'C'	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	'C'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.6. Command 'S'

Description: send the I2C start bit.

Format:

Byte 0	Byte 1	Byte 2
@	'S'	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	'S'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.7. Command 'T'

Description: send the I2C STOP bit.

Format:

Byte 0	Byte 1	Byte 2
@	'T'	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	'T'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.8. Command 'W'

Description: Write one byte to I2C target.

Format:

Byte 0	Byte 1	Byte 2	Byte 3
@	'W'	data	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	'W'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.9. Command 'P'

Description: send the repeated SATRT bit.

Format:

Byte 0	Byte 1	Byte 2

@	‘P’	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	‘P’	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.10. Command 'R'

Description: read one byte from I2C target with the Acknowledgement.

Format:

Byte 0	Byte 1	Byte 2
@	‘R’	Checksum

Normal response:

Byte 0	Byte 1	Byte 2	Byte 2
@	‘R’	Data	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.11. Command ‘D’

Description: read one byte from I2C target without Acknowledgement.

Format:

Byte 0	Byte 1	Byte 2
@	‘D’	Checksum

Response:

Byte 0	Byte 1	Byte 2
@	‘D’	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.12. Command 'A'

Description: get the I2C delay. This delay is the interval between the writes for the command 'Q'. The delay is 10 mS basis. For example, if the value of the delay is 0x2, then the actual delay is 20 mS.

.

Format:

Byte 0	Byte 1	Byte 2
@	'A'	Checksum

Normal response:

Byte 0	Byte 1	Byte 2	Byte 3
@	'L'	delay	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.13. Command 'L'

Description: set the I2C delay. This delay is the interval between the writes for the command 'Q'.

The delay is 10 mS basis. For example, if a delay of 0x2 is set, then 20 mS interval is inserted between the I2C writes. This delay does not apply to command 'W'.

Format:

Byte 0	Byte 1	Byte 2	Byte 3
@	'L'	Delay	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	'L'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.14. Command 'G'

Description: get multiple bytes (up to 32 bytes) from the I2C target. This command will not specify the read address. User should use write command to specify the read address first.

Format:

Byte 0	Byte 1	Byte 2	Byte 3
@	'G'	Length n	Checksum

Where n is the length of the data to read.

Normal response:

Byte 0	Byte 1	Byte 2 ~ Byte (2+n)	Byte (3+n)
@	'G'	Returned data	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.15. Command 'F'

Description: This command is similar to the 'G' except that read address is specified.
Both SATRT and STOP bits are sent in this command.
This command is only in firmware version 0003 or later.

Format:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
@	'F'	Address mode	Device address	Address high	Address low	Length N	Checksum

Where:

- Address mode:

Value	Description
0	8 bits address mode. Only address low byte is used to specify the read address.
1	16 bits address. Both address high and low are used to specify the read address
other	Invalid value

- Device address: I2C salve address. The firmware will form a control byte with this address and read or write operation.
- Address high: only used if the device requires 16 bits sub address. For example, device like EEPROM 24LC512. This value is ignored by the firmware if address mode is 0
- Address low: specifies the lower 8 bits of the address
- N is the length of the data to read.

Normal response:

Byte 0	Byte 1	Byte 2 ~ Byte (2+n)	Byte (3+n)
@	'F'	Returned data	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.16. Command 'Q'

Description: write multiple bytes (up to 32 bytes) to the I2C target. An interval set by command 'L' is inserted between writes.

Both START and STOP bits are sent in this command. Users do not need to send a separate SATRT or STOP bit.

Format:

Byte 0	Byte 1	Byte 2	Byte 3 ~ Byte (3+n)	Byte (4+n)
@	'Q'	Length n	Data	Checksum

Where n is the length of the data to write.

Normal response:

Byte 0	Byte 1	Byte 2
@	'Q'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.17. Command 'B'

Description: set the I2C speed. This command sets the SSPADD register value in the PIC18F2455.

Because the PIC18F2455 runs at 48 MHz peripheral clock, the I2C clock speed can be calculated with the following formula.

$$\text{I2C Speed} = \frac{48\ \text{MHz}}{(\text{SSPADD}+1)*4}$$

For I2C speed at 400 K, SSPADD shall be set to decimal 29.
For I2C speed at 100 K, SSPADD shall be set to decimal 119.

Other speeds can be calculated in a similar way.

Format:

Byte 0	Byte 1	Byte 2	Byte 3
@	'B'	SSPADD value	Checksum

Normal response:

Byte 0	Byte 1	Byte 2
@	'B'	Checksum

Error response:

Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

10.18. Command 'V'

Description: return the I2C speed register SSPADD value from PIC18F2455.

Format:

Byte 0	Byte 1	Byte 2
@	'V'	Checksum

Response:

Byte 0	Byte 1	Byte 2	Byte 3
@	'V'	SSPADD value	Checksum

Error response:

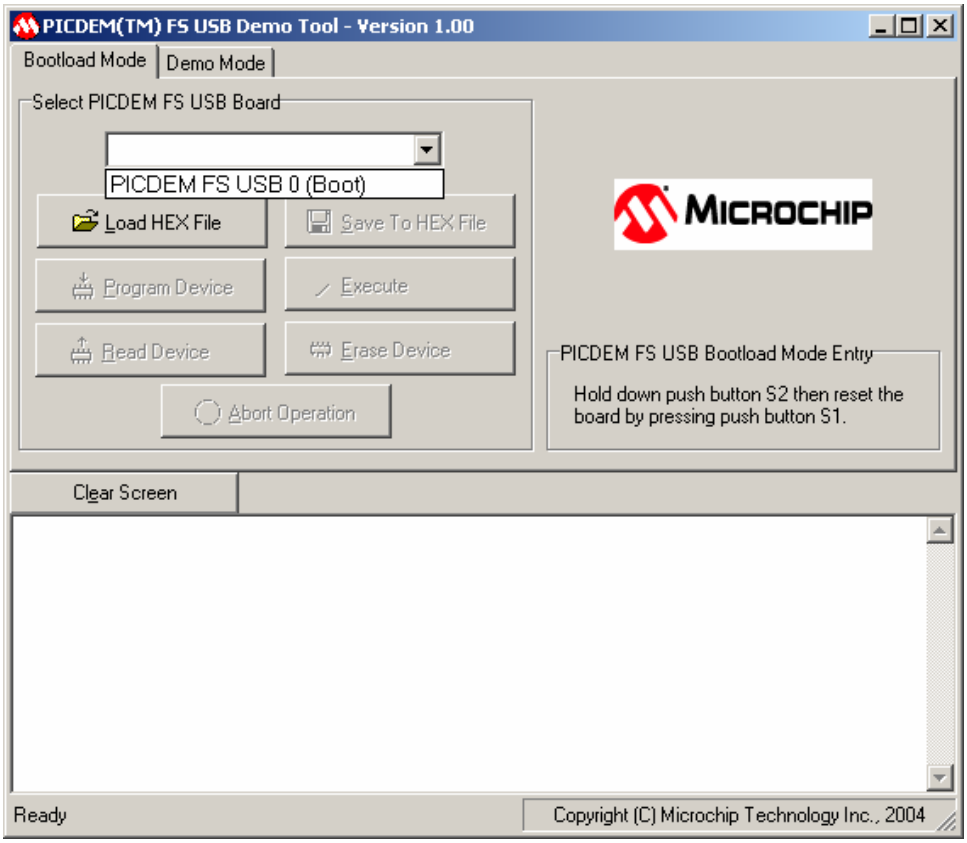
Byte 0	Byte 1	Byte 2	Byte 3
@	e	Error code	Checksum

11. Boot loader mode

Bootloader mode allows users to upgrade the application software in the field if needed. The QBridge-I2C-USB will enter into boot loader mode if the <Boot> pin (pin 9) and <Gnd> (pin 4) are shorted (connected). These pins are located in the male DB 9 connector of the device.

<Boot> pin should not be connected to Ground for normal operation.

The Microchip USB demo tool is used to download the new application.



Procedures to download the new application:

- Unplug the QBridge-I2C-USB device from your PC.
- Use a wire or jumper to short (connect) the <Boot> pin and <GND> pin.
- Attach the QBridge to the PC USB port. The QBridge will enter into the Boot mode. The LED will continuously flash when in the Boot mode.
- Start the Bootloader from Windows toolbar <Start> -> <Programs> -> <QBridge Control Center> -> <Bootloader> and then select the <Bootload Mode>
- Select the "**PICDEM FS USB (boot)**" from the dropdown list.
- Click the <Load Hex file> to load the new application.
- Click the <Program Device> to program the new application to the QBridge. This will take approximate 5~15 seconds.

- Close the Bootloader program. Do not use the <Execute> button to run the downloaded application.
- Disconnect the <Boot> and <GND>.
- Unplug and re-plug the QBridge device.
- The LED will perform On-Off-On to indicate that the device is in the normal operation mode.

12. Revision history

Document Revision	PC Software Revision	Date	Comment
1.0	1.0	Feb-02-2006	Initial release
1.1	1.1	June-30-2006	Fixed the document format error. Fixed the PC software 8 bits address error.
1.2	1.2	Aug-26-2006	1) Added 'F' command. Firmware version 0003 or later is needed to support 'F' command. This command is used in the Extended I2C pane to perform the read operation. 2) Changed the <Control Byte> field to <Device Address> in the Command Dialog. In the Extended I2C pane, The PC application will form a control byte based on the device address and R/W operation. 4) Added a new I2C error 13 to indicate that time out when waiting for the I2C TX completion.
1.3	1.2	Oct-6-2006	Update the document to reflect the change of 3.3V pull up on I2C. Current drain limit is changed from 500ma to200ma for I2C bus
1.4	1.3	Apr-20-2007	In the QBridge firmware, change the USB transfer type from INT to BULK to obtain faster throughput. In the QBridge application, fix the bug for incorrect data parsing when loading user ini data.