I2C232/I2CUSB

I2C to RS232 Adapter
&
I2C to USB Adapter
Application Documentation

Version 3.1
4/27/08

1. Introduction

This document covers both the I2C232 and the I2CUSB. This devices used identical microcontrollers and embedded software and only differ in the interface to the PC.

- The I2C232 adapter connects a PC to the I2C bus using the RS232 port.
- The I2CUSB adapter connects a PC to the I2C bus using the USB port.

The command set is in ASCII allowing a simple terminal emulation program such as Terminal.exe or Hyperterminal.exe to be used to talk to the I2C buss.

The I2CUSB uses a driver to make it look like a typical RS232 device.

1.1. Typical Applications

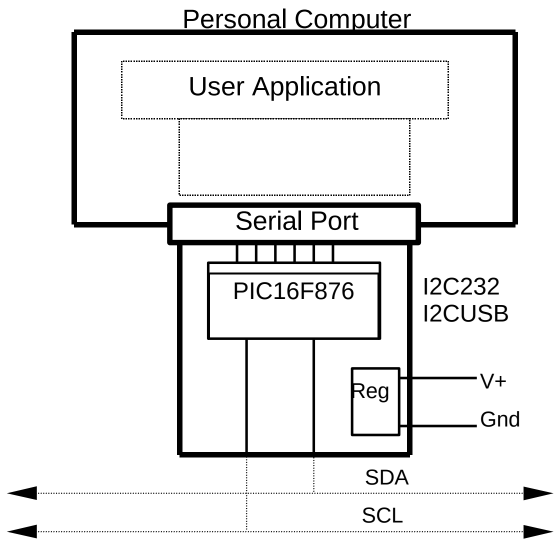
- Send and receive I2C messages
- Develop powerful I2C applications using PC software
- Compact design is ideal for portable applications
- Simulate bus traffic to test module applications

1.2. Design Intentions

The I2C232/I2CUSB is intended to be a low-cost alternative to the PCMCIA, PC Bus Cards, and serial interface cards used to connect a PC to the I2C bus. In order to keep the cost low while maintaining a high degree of functionality, the I2C232 relies on a MicroChip PIC16F876 to provide the interface between the serial channel (RS232) and the I2C bus. PC software enhances the package.

1.3. I2C232/I2CUSB Overview

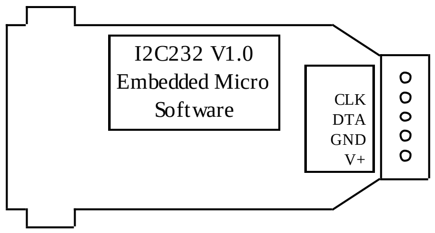
The following block diagram shows the hardware and software model.



1.4. I2C232 Installation

The I2C232 interfaces to the serial port of the PC. The following external connections are required;

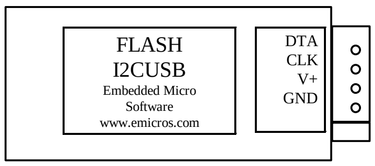
1. Connect the I2C bus to clock and data terminals on the green terminal block.
2. Connect a DC voltage (6 to 24 volts) to V+ and GND on the green terminal block.
3. Verify polarity is correct (I2C232 is protected for reverse polarity and overvoltage).
4. Connect the I2C232 to the serial port on the PC.
5. Turn on power.
6. Test the installation by running any terminal program on the PC. Set communications parameters to 9600 baud, no parity, 8 bits, and 1 stop bit. Enter a V to display the version.



1.5. I2CUSB Installation

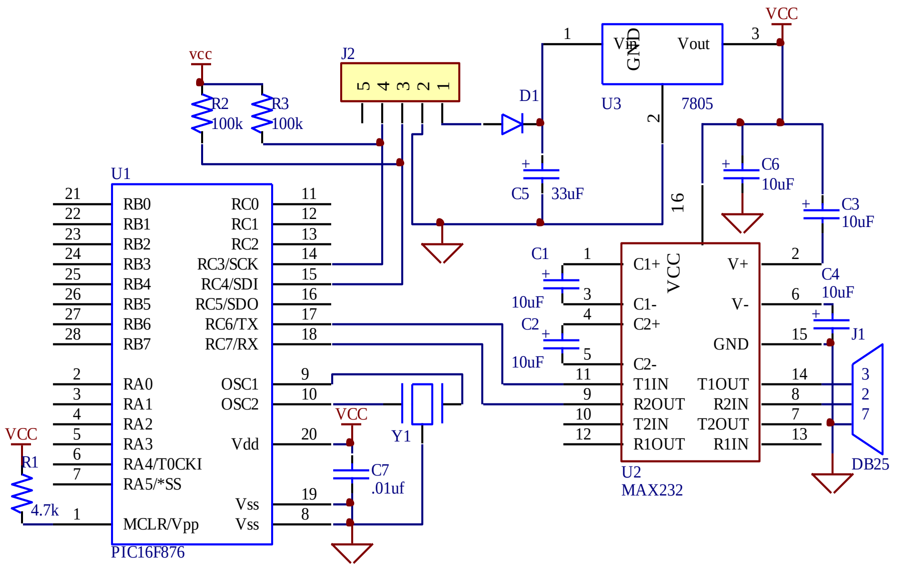
The I2CUSB interfaces to the USB port of the PC and requires a driver to be installed. Refer to www.emicros.com/i2cusb.htm for complete installation.

1. Connect the I2C bus to clock and data terminals on the green terminal block.
2. Connect the I2CUSB to an available USB port on the PC.
3. Test the installation by running any terminal program on the PC. Set communications parameters to 9600 baud, no parity, 8 bits, and 1 stop bit. Enter a V to display the version.

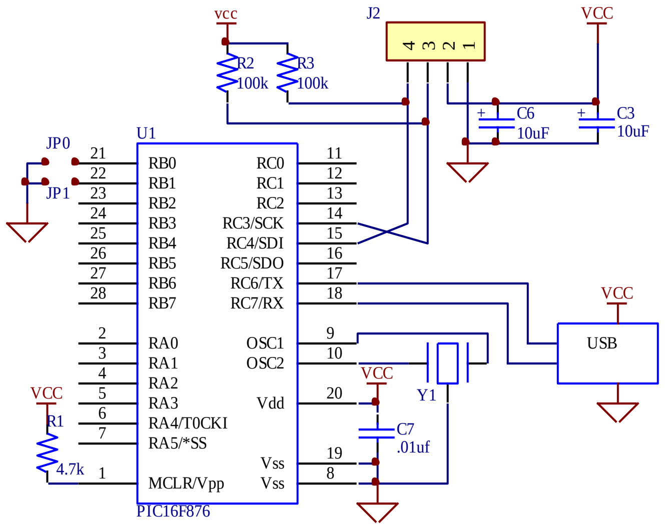


1.6. Hardware Overview

The following schematic shows the I2C232.

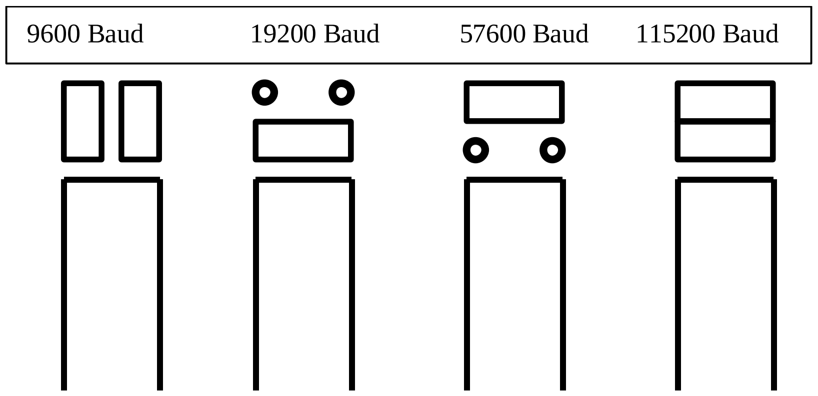

VCC
vcc
J2
5 4 3 2 1
R2 100k
R3 100k
D1
1
Vin
GND
Vout
3
U3
7805
2
C5 33uF
C6 10uF
C3 10uF
16
U1
RB0 RB1 RB2 RB3 RB4 RB5 RB6 RB7
RC0 RC1 RC2 RC3/SCK RC4/SDI RC5/SDO RC6/TX RC7/RX
RA0 RA1 RA2 RA3 RA4/T0CKI RA5/*SS
OSC1 OSC2 Vdd Vss Vss
MCLR/Vpp
R1 4.7k
Y1
C7 .01uf
PIC16F876
C1 10uF
C2 10uF
C1+ C1- C2+ C2- T1IN R2OUT T2IN R1OUT
VCC
V+ V- GND T1OUT R2IN T2OUT R1IN
C4 10uF
J1
DB25
U2
MAX232

The following schematic shows the I2CUSB.


vcc
J2
4 3 2 1
VCC
R2 100k
R3 100k
C6 10uF
C3 10uF
JP0
JP1
U1
RB0 RB1 RB2 RB3 RB4 RB5 RB6 RB7
RC0 RC1 RC2 RC3/SCK RC4/SDI RC5/SDO RC6/TX RC7/RX
RA0 RA1 RA2 RA3 RA4/T0CKI RA5/*SS
OSC1 OSC2 Vdd Vss Vss
MCLR/Vpp
R1 4.7k
Y1
C7 .01uf
USB
PIC16F876

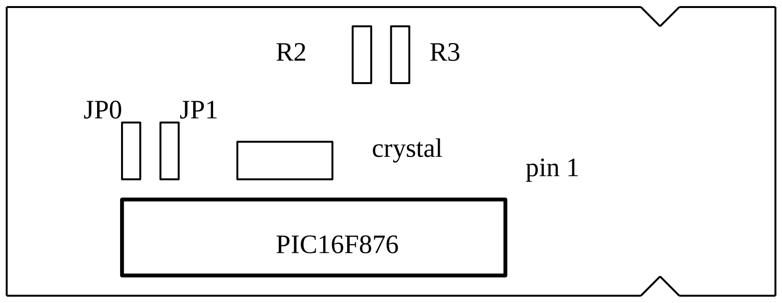
1.7. I2C232 Baud Rate Setting

The I2C232 can be configured to operate at 4 different baud rates. The 2 jumpers are located next to pin 1 on the PIC microcontroller. The default setting is 9600 baud, 8 data bit, no parity, and 1 stop bit.



1.8. I2CUSB Baud Rate Setting

The I2CUSB can be configured to operate at 4 different baud rates. The 2 jumpers are located next to pins 13 and 14 on the PIC microcontroller. The default setting is 9600 baud, 8 data bit, no parity, and 1 stop bit.



JP0	JP1	Baud Rate
out	out	9600
in	out	19.2k
out	in	57.6k
in	in	115.2k

R2 and R3 can be replaced with user's specific pullup resistors if needed.

2. Commands

The commands are shown in this section. Note that these ASCII commands are the same for both the I2C232 and the I2CUSB adapters.

Note: When entering hexidecimal value, use the standard C notation 0x00 to 0xFF. Values from 0x00 to 0x0F must have 2 places, i.e. 0x0 to 0xF will result in unknown values.

Values shown as {,xx} are optional and can be repeated.

2.1. S - Start Command

The Start Command, S, causes a start condition to occur on the I2C bus.

```
Command:    S       ' no carriage return required
Response:   !       ' acknowledge character is explanation point, no carriage return
```

2.2. T - Stop Command

The Stop Command, T, causes a stop condition to occur on the I2C bus.

```
Command:    T       ' no carriage return required
Response:   !       ' acknowledge character is explanation point, no carriage return
```

2.3. V - Display Version

The Display Version Command, V, displays the software version (x.y is replaced by current version).

```
Command:    V       ' no carriage return required
Response:   I2C232 Version x.y: EMICROS[cr]
```

2.4. P - Display Parameters

The Display Parameter Command, P, displays the software version, display mode, and slave address.

```
Command:    P       ' no carriage return required
Response:   I2C232 Version 2.0: EMICROS[cr]
            Display Mode = {Hex | Decimal}[cr]
            Slave Address = 0 (00 hex)[cr]
```

2.5. M - Display Mode Command

The Display Mode Command, M, selects the display mode of displayed values.

```
Command:    M0[cr]  ' display in hex notation (0x00 through 0xff)
```

```
Response:   Hex[cr]

Command:    M1[cr]  ' display in decimal notation (0 through 255)
Response:   Dec[cr]
```

2.6. D - Set Destination Address

The Set Destination Address command sets the slave address sent at the start of the W (write) and R (read) commands. The slave address can be displayed by entering this command without any parameters. The slave address value will be displayed in the currently selected mode, hex or decimal.

```
Command:    Dxx[cr]         ' where xx is decimal or hex representation (i.e. 0x20)
Response:   Dxx[cr]

Command:    D[cr]
Response:   Dxx[cr]
```

Note: The destination address is the 1st 7 bits following the Start sequence and the 8th bit is the R/W bit. Enter the address as the lower 7 bits, 0xxxxxxx. When the destination address is used in the Write or Read command, the address is shifted left 1 bit and the R/W bit is set according to the command entered (R or W).

2.7. A - Write I2C Bytes

The Write I2C Bytes command writes from 1 to 64 bytes to the I2C bus. The Start and Stop conditions are not executed in this command.

```
Command:    Axx{,xx}[cr]    ' where xx is decimal or hex representation (i.e. 0x20)
Response:   xx{,xx}[cr]
```

Example: This example shows the sequence to program 3 bytes of a 24C65 Serial EEPROM.

```
S               ' execute Start command
A0xa0,0,0[cr]   ' Write address 0x50 (shift left by 1) and 2 data bytes of 0
A1,2,3[cr]      ' Write the sequence to the I2C bus, program 3 bytes
T               ' execute the Stop command
```

2.8. B - Read 1 Byte with ACKNOWLEDGE

The Read 1 Byte command reads 1 byte from the I2C bus and displays the results. The Start and Stop conditions are not used and the ACK pulse is generated at the end of the byte read.

```
Command:    B               ' single byte command, no [cr] needed
Response:   xx[cr]
```

Example: This example shows the sequence to read 3 bytes from a 24C65 Serial EEPROM.

```
S               ' execute Start command
A0xa0,0,0[cr]   ' Write address 0x50 (shifted by 1 and R/*W cleared), set 24C65 address to 0
S               ' execute another Start command
A0xa1[cr]       ' Write address 0x50 (shifted by 1 and R/*W set), read 24C65
```

```
B               ' Read 1 byte, address 0 from 24C65 with ACK
B               ' Read 1 byte, address 1 from 24C65 with ACK
B               ' Read 1 byte, address 2 from 24C65 with ACK
C               ' Read 1 byte, address 2 from 24C65 with NO ACK (see section 2.9)
T               ' execute Stop Command
```

2.9. C - Read 1 Byte with NO ACKNOWLEDGE

The Read 1 Byte command with NO ACK reads 1 byte from the I2C bus and displays the results. The Start and Stop conditions are not used and the ACK pulse is NOT generated at the end of the byte read.

```
Command:    C              ' single byte command, no [cr] needed
Response:   xx[cr]
```

See example in section 2.8.

2.10. W - Write Message

The Write Message command, M, writes a series of bytes to the I2C bus. The Start and Stop conditions are used. The Write command uses the destination address entered using the D command as the 1st byte to address the slave.

```
Command:    Wxx{,xx}[cr]   ' where xx is decimal or hex representation (i.e. 0x20)
Response:   xx{,xx}[cr]    ' I2C232 returns bytes written
```

Example: This example writes the values 1,2,3,4 to the current slave destination.

```
W0,0,1,2,3,4,5,6,7,8[cr]                                  ' command to I2C232
0x00,0x00,0x01,0x02,0x03,0x04,0x05,0x05,0x06,0x07,0x08[cr] ' response from I2C232
```

2.11. R - Read Message

The Read Message command, R, reads the specified number of bytes from the I2C bus. The Start and Stop conditions are used. The maximum number is clamped at 64 bytes, i.e. any number entered greater than 64 is set to 64.

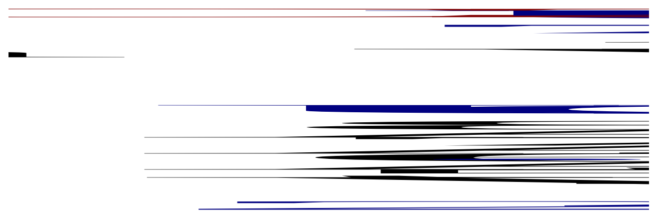
```
Command:    Rxx{,xx}[cr]   ' where xx is decimal or hex representation (i.e. 0x20)
Response:   xx{,xx}[cr]
```

Example: This example reads 2 values from the current slave destination.

```
R2[cr]              ' command to I2C232
0xFF,0xFF[cr]       ' response from I2C232, 0xFF if no device is responding
```

3. Example

The following schematic shows the I2C232 connected to a 24C65 Smart Serial EEPROM from Microchip. The 24C65 is an I2C device and contains 8k x 8 bytes of Electrically Erasable PROM (EEPROM). Note that external pullup resistors may be required.



The slave address for the 24C65 is set according to A0, A1, and A2. The example has the 3 address lines grounded resulting in an address of 0x50. To write the 1st 8 bytes of EEPROM, the following commands is issued;

```
D0x50                   ' set destination address of device
W0,0,1,2,3,4,5,6,7,8[cr] ' first 2 bytes following the W sets the address to 0 (0,0), then
                        '    program the 8 bytes starting at address 0 to 1,2,3,4,5,6,7,8
```

The 24C65 will program the bytes when the stop condition is executed, done automatically by the Write command.

To read back the bytes, the following sequence is executed;

```
S            ' execute start
A0xA0,0,0[cr] ' Sets the starting address to 0, slave address 0x50 shifted by 1
S            ' execute start again
A0xA1[cr]    ' Read bit set
B            ' Read 1 byte, I2C232 returns value
B            ' 2nd data byte
B            ' 3rd data byte
B            ' 4th data byte
B            ' 5th data byte
B            ' 6th data byte
B            ' 7th data byte
C            ' 8th data byte, NOACK
T            ' Stop condition
```

4. Revision History

2/5/00 Added baud rate selection capability. Compiled to new version of compiler.
4/27/08 Added I2CUSB