Code (ASM):
```
'Author: languer (©2012)
'Pin Allocation:
'PIN# Main_Fn                 Secondary_Fn
'RA0 -> not used
'RA1 -> not used
'RA2 -> not used
'RA3 -> not used
'RA4 -> not used
'RA5 -> MCLR
'RA6 -> OSC
'RA7 -> OSC
'RB0 -> I2C START Interrupt
'RB1 -> TX_RS232 (PC_RX)
'RB2 -> I2C STOP Interrupt
'RB3 -> I2C SCL Interrupt (using CCP)
'RB4 -> RX_RS232 (PC_TX)
'RB5 -> not used
'RB6 -> I2C SDA               PGC (Programming clock)
'RB7 -> CTS#                  PGD (Programming data)
'Usage Information:
'RS232 Baud Rate: 115.2kbps
'Version Info:
'rs232 comms on array at main loop
'max speed is 100kHz
'no indication if buffer overflows

'General Configuration
'for external 10MHz w PLL (40MHz)
Define CONFIG1L = 0x00
Define CONFIG1H = 0x06
Define CONFIG2L = 0x0a
Define CONFIG2H = 0x00
Define CONFIG3L = 0x00
Define CONFIG3H = 0x80
Define CONFIG4L = 0x80
Define CONFIG4H = 0x00
Define CONFIG5L = 0x03
Define CONFIG5H = 0xc0
Define CONFIG6L = 0x03
Define CONFIG6H = 0xe0
Define CONFIG7L = 0x03
Define CONFIG7H = 0x40

'Oscillator/Clock Configuration
Define CLOCK_FREQUENCY = 40

'HW UART Setup
Hseropen 115200

'RS232 Definitions
Symbol io_rs232tx = RB1  'mcu rs-232 output
Symbol io_rs232rx = RB4  'mcu rs-232 input
Symbol io_rs232ctsn = RB7  'mcu rs232 cts# handshake signal

'I2C Definitions
Symbol io_i2cstart = RB0
Symbol io_i2cstop = RB2
Symbol io_i2cscl = RB3
Symbol io_i2csda = RB6

'Constants
Const trisa1 = %11111111
Const trisb1 = %01111101
Dim _true As Bit
Dim _false As Bit
_true = True
_false = False

'Variables
Dim flag_i2cstart As Bit
Dim flag_i2cstop As Bit
Dim i2cack As Byte
Dim i2cdata As Byte
Const i2cbuffersize = 200
Dim i2carray(200) As Byte
Dim i2cnextin As Byte
Dim i2cnextout As Byte
Dim bitcount As Byte

'Main Program
main:
WaitMs 2500
Call init()

'enable interrupt routines
INTCON.INT0IE = True  'enable RB0/I2CSTART interrupt
CCP1CON = %00000101  'enable RB3/CCP1/I2CSCL
Enable High  'enable general interrupt

Dim data As Byte
Dim cnt As Byte
While _true
If i2cnextout = i2cnextin Then
'do nothing
Else
'loop-around buffer
If i2cnextout = i2cbuffersize Then
i2cnextout = 0
Endif
'hserout when TXREG is empty
If PIR1.TXIF = True Then
data = i2carray(i2cnextout)
TXREG = data
i2cnextout = i2cnextout + 1
Endif
Endif
Wend
End

Proc init()
AllDigital
TRISA = trisa1
TRISB = trisb1

flag_i2cstart = False
flag_i2cstop = False
i2cack = 0
i2cdata = 255
i2cnextin = 0
i2cnextout = 0
bitcount = 0

'Hserout "Start...", CrLf
End Proc

On High Interrupt
Dim hex1 As Byte
Dim hex2 As Byte

'START/RESTART Condition (3.6us / 29Tcy)
If INTCON.INT0IF = True Then  'RB0/I2CSTART flag
INTCON.INT0IF = False  'RB0/I2CSTART flag
INTCON3.INT2IF = False  'RB2/I2CSTOP flag
INTCON3.INT2IE = True  'enable RB2/I2CSTOP interrupt
```

```
flag_i2cstop = False
bitcount = 0
PIR1.CCP1IF = False  'RB3/CCP/I2CSCL  flag
PIE1.CCP1IE = True  'enable RB3/CCP/I2CSCL interrupt
INTCON.PEIE = True  'enable peripheral interrupt (required for RB3/CCP/I2CSCL)
'loop-around buffer
If i2cnextin = i2cbuffersize Then
i2cnextin = 0
Endif
If flag_i2cstart = True Then
i2carray(i2cnextin) = "R"  'restart condition"
Else
flag_i2cstart = True
i2carray(i2cnextin) = "S"  'start condition
Endif
i2cnextin = i2cnextin + 1
Endif

'STOP Condition (3.0us / 24Tcy)
If INTCON3.INT2IF = True Then  'RB2/I2CSTOP flag
INTCON3.INT2IE = False  'disable RB2/I2CSTOP interrupt
INTCON3.INT2IF = False  'RB2/I2CSTOP flag
INTCON.PEIE = False  'disable peripheral interrupt (required for RB3/CCP/I2CSCL)
PIE1.CCP1IE = False  'disable RB3/CCP/I2CSCL interrupt
PIR1.CCP1IF = False  'RB3/CCP/I2CSCL flag
flag_i2cstart = False
'loop-around buffer
If i2cnextin = i2cbuffersize Then
i2cnextin = 0
Endif
i2carray(i2cnextin) = "P"  'stop condition
i2cnextin = i2cnextin + 1
Endif

'DATA Condition
'(2.4us / 18Tcy for Bits 0-6)
'(7.2us / 55Tcy for Bit 7)
'(7.4us / 58Tcy for ACK/NACK)
If PIR1.CCP1IF = True Then  'RB3/CCP/I2CSCL flag
PIR1.CCP1IF = False  'RB3/CCP/I2CSCL flag
Select Case bitcount
Case < 7
'shift data left and add new i2cdata bit
'i2cdata = ShiftLeft(i2cdata, 1)
ASM:        RLCF i2cdata,1
If io_i2csda = True Then  'add
i2cdata = i2cdata + 1
Endif
bitcount = bitcount + 1
Case 7
'shift data left and add last i2cdata bit
'i2cdata = ShiftLeft(i2cdata, 1)
ASM:        RLCF i2cdata,1
If io_i2csda = True Then  'add
i2cdata = i2cdata + 1
Endif
bitcount = bitcount + 1
'store I2CDATA
'loop-around buffer
'hex1 = ShiftRight(i2cdata, 4)
ASM:        movff i2cdata,hex1
ASM:        movf hex1,w
ASM:        andlw 0xf0
ASM:        movwf hex1
ASM:        RRNCF hex1,1
ASM:        RRNCF hex1,1
ASM:        RRNCF hex1,1
ASM:        RRNCF hex1,1
hex1 = LookUp("0123456789ABDCEF"), hex1
'hex2 = i2cdata And 0x0f
ASM:        movff i2cdata,hex2
ASM:        movlw 0x0f
ASM:        andwf hex2,1
hex2 = LookUp("0123456789ABDCEF"), hex2
Case Else
'ACK/NACK bit
If i2cnextin = i2cbuffersize Then
i2cnextin = 0
Endif
i2carray(i2cnextin) = hex1
i2cnextin = i2cnextin + 1
If i2cnextin = i2cbuffersize Then
i2cnextin = 0
Endif
i2carray(i2cnextin) = hex2
i2cnextin = i2cnextin + 1

bitcount = 0

'store ack/nack STATUS
If io_i2csda = True Then  'test for ACK/NACK
i2cack = "A"
Else
i2cack = "N"
Endif
'loop - around buffer
If i2cnextin = i2cbuffersize Then
i2cnextin = 0
Endif
i2carray(i2cnextin) = i2cack
i2cnextin = i2cnextin + 1

EndSelect
Endif

Resume
```