**Pin Allocation:**

<table>
<thead>
<tr>
<th>PIN#</th>
<th>Main Fn</th>
<th>Secondary_fn</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>-</td>
<td>not used</td>
</tr>
<tr>
<td>RA1</td>
<td>-</td>
<td>not used</td>
</tr>
<tr>
<td>RA2</td>
<td>-</td>
<td>not used</td>
</tr>
<tr>
<td>RA3</td>
<td>-</td>
<td>not used</td>
</tr>
<tr>
<td>RA4</td>
<td>-</td>
<td>not used</td>
</tr>
<tr>
<td>RA5</td>
<td>MCLR</td>
<td>-</td>
</tr>
<tr>
<td>RA6</td>
<td>OSC</td>
<td>-</td>
</tr>
<tr>
<td>RA7</td>
<td>OSC</td>
<td>-</td>
</tr>
<tr>
<td>RB0</td>
<td>I2C</td>
<td>START</td>
</tr>
<tr>
<td>RB1</td>
<td>T</td>
<td>RX_RS232 (PC_RX)</td>
</tr>
<tr>
<td>RB2</td>
<td>I2C</td>
<td>STOP Interrupt</td>
</tr>
<tr>
<td>RB3</td>
<td>I2C</td>
<td>SCL Interrupt (using CCP)</td>
</tr>
<tr>
<td>RB4</td>
<td>RX_RS232</td>
<td>PC_TX</td>
</tr>
<tr>
<td>RB5</td>
<td>-</td>
<td>not used</td>
</tr>
<tr>
<td>RB6</td>
<td>I2C</td>
<td>SDA</td>
</tr>
<tr>
<td>RB7</td>
<td>CTS#</td>
<td>PGD (Programming data)</td>
</tr>
</tbody>
</table>

**Usage Information:**

- RS232 Baud Rate: 115.2kbps
- Max speed is 100kHz
- No indication if buffer overflows

**General Configuration:**
- Define CONFIG1L = 0x00
- Define CONFIG1H = 0x06
- Define CONFIG2L = 0x0a
- Define CONFIG2H = 0x00
- Define CONFIG3L = 0x80
- 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

**RS232 Definitions**
- Symbol io_rs232tx = RB1
- Symbol io_rs232rx = RB4
- Symbol io_rs232ctsn = RB7

**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

**Variables**
- Dim _true As Bit
- Dim _false As Bit
- Dim flag_i2cstart As Bit
- Dim flag_i2cstop As Bit
- Dim i2cack As Byte
- Dim i2cdata As Byte
- Dim 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 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
    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
i2czack = 0
i2cdata = 0x5
i2cnextin = 0
i2cnextout = 0
bitcount = 0

'When TXREG is empty
If PIR1.TXD = True Then
  data = i2carray(i2cnextout)
  TXREG = data
  i2cnextout = i2cnextout + 1
Endif
End
```

**On High Interrupt**
- Dim hex1 As Byte
- Dim hex2 As Byte

```
Hserout "Start...", CrLf
```

**START/RESTART Condition (3.6us / 29Tcy)**

```
If INTCON.INT0IF = True Then
  'Enable RB0/I2CSTART flag
  INTCON.INT0IE = True
Endif
```

End Proc
flag_i2cstop = False
bitcount = 0
PIR1.CCP1IF = False

' RB3/CCP/I2CSCL flag
PIN1.CCP1IF = True 'enable RB3/CCP/I2CSCL interrupt
INTCON.PIE1 = True 'enable peripheral interrupt (required for RB3/CCP/I2CSCL)

'I2C STOP

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

'I2C STOP Condition (3.0us / 24Tcy)
If INTCON.INT2IF = True Then
RB3/CCP/I2CSTOP flag
INTCON.INT2IE = False 'disable RB2/I2CSTOP interrupt
INTCON.INT2IF = False 'RB2/I2CSTOP flag
INTCON.PEIE = False 'disable peripheral interrupt (required for RB3/CCP/I2CSCL)
PIE1.CCP1IE = False 'RB3/CCP/I2CSCL flag
Flag_i2cstart = False
Endif

DATA Condition
'I2C STOP

If i2cnextin = i2cbuffersize Then
i2cnextin = 0
Endif
i2carray(i2cnextin) = "P" 'stop condition
i2cnextin = i2cnextin + 1
Endif

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
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
i2cdata = i2cdata + 1
Endif
bitcount = bitcount + 1
'store I2CDATA
hex1 = ShiftRight(i2cdata,4)
ASM: movff i2cdata,hex1
ASM: andlw 0x0F
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) = hexw
i2cnextin = i2cnextin + 1
EndSelect
Resume

Case Else
Table

'ACK/NACK bit
If i2cnextin = i2cbuffersize Then
i2cnextin = 0
Endif
i2carray(i2cnextin) = hexw
i2cnextin = i2cnextin + 1
Endif
i2cnextin = i2cnextin + 1
Endif

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
Resume