Software Implementation of Asynchronous Serial I/O

INTRODUCTION

The PIC16CXX microcontrollers from Microchip Technology, Inc., are mid-range, high performance EPROM based 8-bit microcontrollers. Some of the members of this series (like PIC16C71 and PIC16C84) do not have an on-chip hardware asynchronous serial port. This application note describes the Interrupt driven Software implementation of Asynchronous Serial I/O (Half Duplex RS-232 Communications) using PIC16CXX microcontrollers. These microcontrollers can operate at very high speeds with a minimum of 250 ns cycle time (with input clock frequency of 16 MHz). To test the RS-232 routines, a simple Digital Volt Meter (DVM)/Analog Data Acquisition Systems has been implemented using PIC16C71 in which upon reception of a command from host (IBM® PC), an 8-bit value of the selected A/D channel is transmitted back to host.

IMPLEMENTATION

A half duplex Interrupt driven software implementation of RS-232 communications using PIC16C71 is described in detail below. The transmit pin used in the example code is RB7 and receive pin is connected to RTCC/RA4 pin (see Figure 2). Of course these pins are connected with appropriate voltage translation to/from RS-232/ CMOS levels. The voltage translation is given described with schematics in the hardware section of this application note.

Transmit Mode

The transmit mode in software is quite straight forward to implement using interrupts. Once the input clock frequency and baud rate is known, the number of clock cycles per bit can be computed. The on-chip Real Time Clock Counter (RTCC) along with the prescaler can be used to generate interrupt on RTCC overflow. This RTCC overflow interrupt can be used as timing to send each bit. The Input clock frequency ("_ClkIn") and the Baud Rate ("_BaudRate") are programmable by the user and the RTCC time-out value (the period for each bit) is computed at assembly time. Whether the prescaler must be assigned to RTCC or not is also determined at assembly time. This computation is done in the header file "rs232.h". Note that very high speed transmissions can be obtained if transmission is done with software delays instead of every interrupt driven, however, the processor will be totally dedicated to this job.

Transmission of a byte is performed by calling "PutChar" function and the data byte in the "TxReg" is transmitted out. Before calling this function ("PutChar"), the data must be loaded into TxReg and also made sure that serial port is free. The serial port is free when both _txmtProgress and _rcvOver bits are cleared (see description of these bits in the Serial Status/Control Reg table given later).

Summary of "PutChar" function :

1) Make sure _txmtProgress & _rcvOver bits are cleared
2) Load TxReg with data to be transmitted
3) CALL PutChar function

Receive Mode

The reception mode implementation is slightly different from the transmit mode. Unlike the transmit Pin (TX pin in the example code is RB7, but could be any I/O pin), the receive pin (RX Pin) must be connected to RTCC/RA4 Pin. This is because in reception, the Start Bit which is asynchronous in nature, must be detected. To detect the start bit, when put in Reception mode, the RTCC module is configured to counter mode . The OPTION register is configured so that RTCC module is put in counter mode (increment on external clock on RTCC/RA4 Pin) and set to increment on falling edge on RTCC/RA4 pin with no prescaler assigned. After this configuration setup, RTCC (File Reg 1) is loaded with 0xFF. A falling edge on RTCC Pin will make RTCC roll over from 0xFF to 0x00, thus generating an interrupt indicating a Start Bit. The RTCC/RA4 pin is sampled again to make sure the transition on RTCC is not a glitch. Once the start bit has been detected, the RTCC module is reconfigured to increment on internal clock and the prescaler is assigned to it depending on input master clock frequency and the baud rate (configured same way as the transmission mode).

The software serial port is put in reception mode when a call is made to function "GetChar". Before calling this function make sure serial port is free (i.e. _txmtProgress and _rcvOver status bits must be 0). On completion of reception of a byte, the data is stored in RxReg and _rcvOver bit is set to 0.

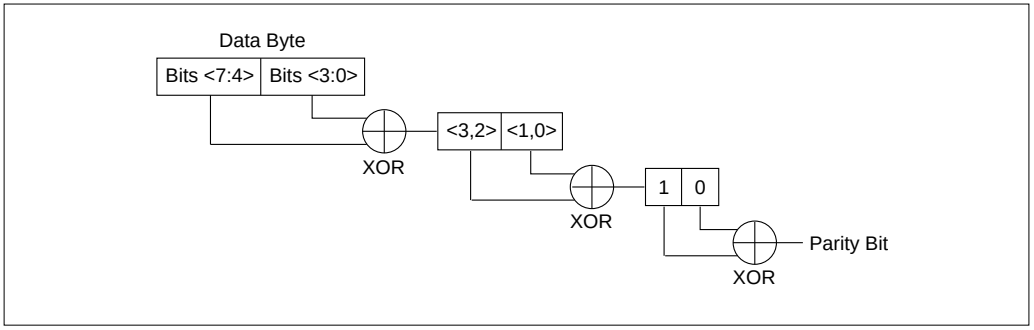
Summary of "GetChar" function:

1) Make sure _txmtProgress & _rcvOver bits are cleared
2) CALL GetChar function
3) The received Byte is in TxReg after _rcvOver bit is cleared

Parity Generation

Parity can be enabled at assembly time by setting "_PARITY_ENABLE" flag to TRUE. If enabled, the parity can be set to either EVEN or ODD parity. In transmission mode, if parity is enabled, the parity bit is computed and transmitted as the ninth bit. On reception, the parity is computed on the received byte and compared to the ninth bit received. If a match does not occur the parity error bit is set in the RS-232 Status/Control Register (_ParityErr bit of SerialStatus reg). The parity bit is computed using the algorithm shown in Figure 1. This algorithm is highly efficient using PIC16CXX's SWAPF and XORWF instructions (with ability to have the destination as either file register itself or W register) and the sub-routine (called "GenParity") is in file "txmtr.asm".

FIGURE 1 - AN EFFICIENT PARITY GENERATION SCHEME IN SOFTWARE



Assembly Time Options

The firmware is written as a general purpose routines and the user must specify the following parameters before assembling the program. The Status/Control register is also described below:

TABLE 1 - LIST OF ASSEMBLY TIME OPTIONS

_ClkIn	Input clock frequency of the processor.
_BaudRate	Desired Baud Rate. Any valid value can be used. The highest Baud Rate achievable depends on Input Clock Freq. 600 to 4800 Baud was tested using 4 MHz Input Clock. 600 to 19200 Baud was tested using 10 MHz Input Clock. Higher rates can be obtained using higher Input Clock Frequencies. Once the _BaudRate & _ClkIn are specified, the program automatically selects all the appropriate timings.
_DataBits	Can specify 1 to 8 data bits.
_StopBits	Limited to 1 Stop Bit. Must be set to 1.
_PARITY_ENABLE	Parity Enable Flag. Set it to TRUE or FALSE. If PARITY is used, then set it to TRUE, else FALSE. See "_ODD_PARITY" flag description below.
_ODD_PARITY	Set it to TRUE or FALSE. If TRUE, then ODD PARITY is used, else mEVEN Parity Scheme is used. This Flag is ignored if _PARITY_ENABLE is set to FALSE.
_USE_RTSCTS	RTS & CTS Hardware handshaking signals. If set to FALSE, no hardware handshaking is used. If set to TRUE, RTS & CTS use up 2 I/O Pins of PortB.

TABLE 2 - BIT ASSIGNMENTS OF SERIAL STATUS/CONTROL REGISTER ("SERIALSTATUS" REG)

Bit #	Name	Description
0	_txmtProgress	1 = Transmission in progress. 0 = Transmission line free.
1	_txmtEnable	Set this bit to 1 on initialization to enable transmission. This bit may be used to abort a transmission. The transmission is aborted if in the middle of a transmission (i.e. when _txmtProgress bit is 1) _txmtEnable bit is set to 0. This bit gets automatically set when PutChar function is called.
2	_rcvProgress	1 = Middle of a byte reception. 0 = Reception of a byte (in RxReg) is complete and is set to 1 when a valid start bit is detected in reception mode.
3	_rcvOver	0 = Completion of reception of a byte. The user's code can poll this bit after calling "GetChar" function and check to see if it is set. When set, the received byte is in RxReg. Other status bits should also be checked for any reception errors.
4	_ParityErr	1 = Parity error on reception (irrespective of Even Or Odd parity chosen). Not applicable if No Parity is used.
5	_FrameErr	1 = Framing error on reception.
6		Unused
7	_parityBit	The 9th bit of transmission or reception. In transmission mode, the parity bit of the byte to be transmitted is set in this bit. In receive mode, the 9th bit (or parity bit) received is stored in this bit. Not Applicable if no parity is used.

Hardware

The hardware is primarily concerned with voltage translation from RS-232 to CMOS levels and vice versa. Three circuits are given below and the user may choose which ever best suits his application. The primary difference between each solution is cost versus number of components. Circuits in Figure 3 and 4 are very low cost but have more components than the circuit in Figure 2. The circuit in Figure 2 interfaces to RS-232 line using a single chip (MAX-232) and single +5V supply. The circuit in Figure 3 is a low cost RS-232 Interface but requires two chips and a single +5V supply source.

Figure 4 shows a very low cost RS-232 Interface to an IBM PC® with no external power requirements. The circuit draws power from RS-232 line (DTR) and meets the spec of drawing power less than 5mA. This requires that the host to communicate must assert DTR high and RTS low. The power is drawn from DTR line and this requires that DTR to be asserted high and must be at least 7V. The negative -5 to -10 V required by LM339 is drawn from RTS line and thus the host must assert RTS low. This circuit is possible because of the low current consumption of PIC16C71 (typical 2 mA).

FIGURE 2 - SINGLE CHIP FOR RS-232 INTERFACE (SINGLE +5V SUPPLY)

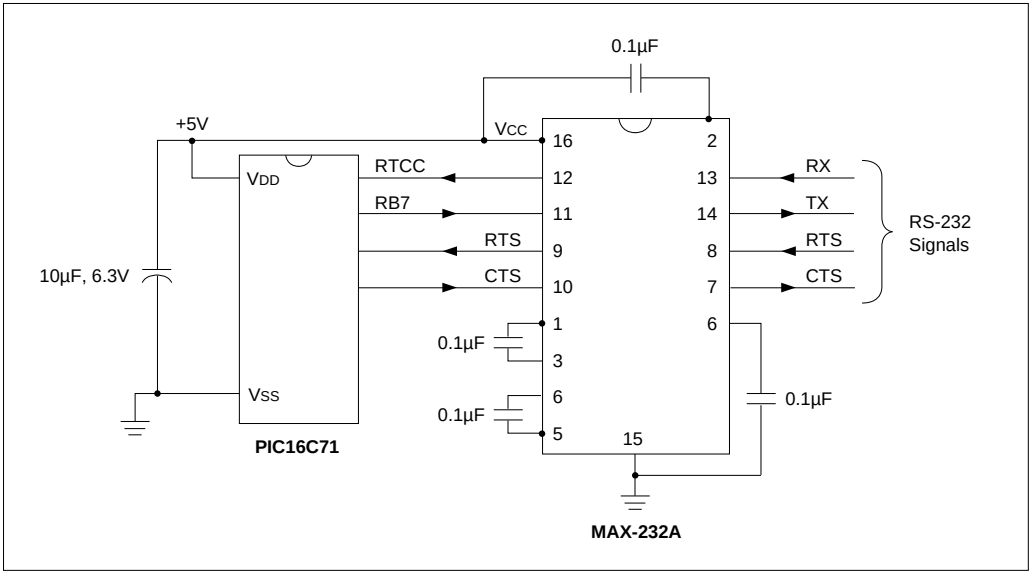


FIGURE 3 - LOW COST RS-232 INTERFACE (TWO CHIPS, SINGLE +5V SUPPLY)

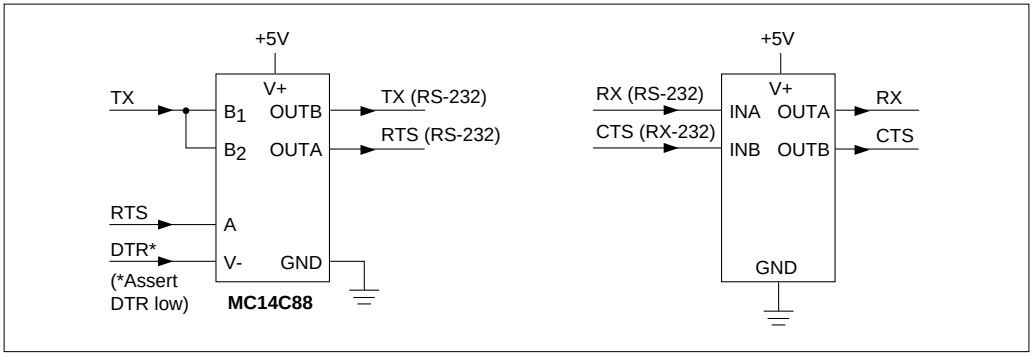
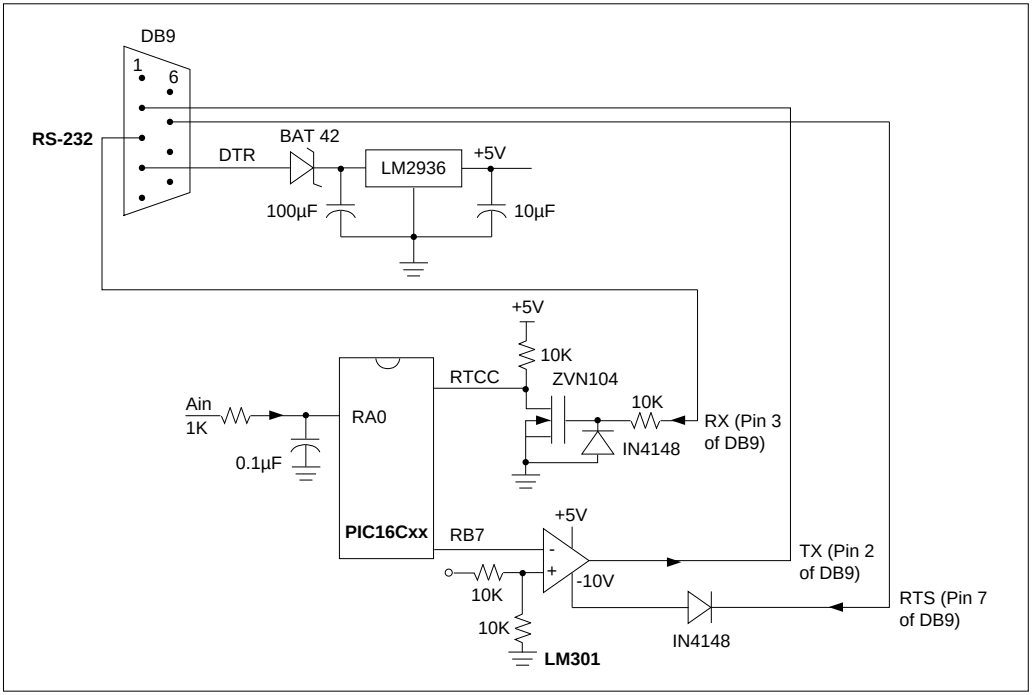


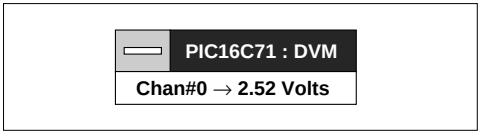
FIGURE 4 - LOW COST, LOW POWER RS-232 INTERFACE (POWER SUPPLIED BY RS-232 LINES)



Test Program

To test the transmission and reception modules, a main program is written in which the PIC16C71 waits to receive a command from a host through the RS-232. On reception of a byte (valid commands are 0x00, 0x01, 0x02 & 0x03), the received byte is treated as the PIC16C71's A/D channel number and the requested channel is selected, an A/D conversion is started and when the conversion is complete (in about 20 uS) the digital data (8 bits) are transmitted back to the host. A Microsoft® Windows® program running on an IBM PC/AT® was written to act as a host and collect the A/D data from PIC16C71 via an RS-232 port. The Windows program (DVM.EXE) runs as a background job and displays the A/D data in a small window (similar to the CLOCK program that comes with MS Windows). The windows program and the PIC16C71 together act like a data acquisition system or a digital volt meter (DVM). The block diagram of the system is shown in Figure 2. The input clock frequency is fixed at 4 MHz and RS-232 parameters are set to 1200 Baud, 8-bits, 1 Stop Bit and No Parity. The program during development stage was also tested at 1200, 2400, 4800 Baud Rates @ 4 MHz Input Clock and up to 19200 Baud @ 10 MHz input clock frequency (all tests were performed with No Parity, Even Parity and Odd Parity at 8 and 7 Data Bits).

FIGURE 5 - MS WINDOWS PROGRAM FETCHING A/D DATA FROM PIC16C71 VIA RS-232



Source Code

The PIC16CXX source code along with the Microsoft® Windows™ DVM Program (executable running on an IBM PC/AT under MS Windows 3.1 or higher) is available on Microchip's BBS. The assembly code for PIC16CXX must be assembled using Microchip's Universal Assembler, MPASM. The code cannot be assembled using the older assemblers without significant modifications. It is suggested that user's who do not have the new assembler MPASM, must change to the new version.

The MS Windows Program (DVM.EXE) runs under MS Windows 3.1 or higher. The program does not have any menus and shows up as a small window displaying A/D Data and runs as a background job. There are a few command line options and are described below :

-Px : x is the comm port number (e.g. - P2 selects COM2). Default is COM1

-Cy : y is the number of A/D channels to display. Default is one channel (channel #1)

-Sz : z is a floating point number that represents the scaling factor (For example - S5.5 would display the data as 5.5*<8bit A/D>/256). The default value is 5.0 volts. -S0 will display the data in raw format without any scaling.

Author: *Amar Palacherla*
Logic Products Division
Code Update: *Scott Fink - Sr. Applications Engineer*
Logic Products Division

Appendix A - RS232.H

```
       NOLIST

;*************************************************************************************
;                       RS-232 Header File
;   PIC16C6X/7X/8X
;*************************************************************************************

_ClkOut        equ     (_ClkIn >> 2)         ; Instruction Cycle Freq = CLKIN/4
;

_CyclesPerBit  set     (_ClkOut/_BaudRate)
_tempCompute   set     (_CyclesPerBit >> 8)
;
;*************************************************************************************
;       Auto Generation Of Prescaler & Rtcc Values
;   Computed during Assembly Time
;*************************************************************************************

;  At first set Default values for RtccPrescale & RtccPreLoad
;
RtccPrescale   set     0
RtccPreLoad    set     _CyclesPerBit
UsePrescale    set     FALSE

 if (_tempCompute >= 1)
  RtccPrescale set     0
  RtccPreLoad  set     (_CyclesPerBit >> 1)

        UsePrescale    set     TRUE
 endif

 if (_tempCompute >= 2)
  RtccPrescale set     1
  RtccPreLoad  set     (_CyclesPerBit >> 2)
 endif

 if (_tempCompute >= 4)
  RtccPrescale set     2
  RtccPreLoad  set     (_CyclesPerBit >> 3)
 endif

 if (_tempCompute >= 8)
  RtccPrescale set     3
  RtccPreLoad  set     (_CyclesPerBit >> 4)
 endif
```

```
    if (_tempCompute >= 16)
     RtccPrescale set    4
     RtccPreLoad  set    (_CyclesPerBit >> 5)
    endif

    if (_tempCompute >= 32)
     RtccPrescale set    5
     RtccPreLoad  set    (_CyclesPerBit >> 6)
    endif

    if (_tempCompute >= 64)
     RtccPrescale set    6
     RtccPreLoad  set    (_CyclesPerBit >> 7)
    endif

    if   (_tempCompute >= 128)
     RtccPrescale set    7
     RtccPreLoad  set    (_CyclesPerBit >> 8)
    endif

;
        if( (RtccPrescale == 0) && (RtccPreLoad < 60))
          messg  "Warning : Baud Rate May Be Too High For This Input Clock"
        endif
;
; Compute RTCC & Presclaer Values For 1.5 Times the Baud Rate for Start Bit Detection
;

    _SBitCycles   set    (_ClkOut/_BaudRate) + ((_ClkOut/4)/_BaudRate)
    _tempCompute  set    (_SBitCycles >> 8)

    _BIT1_INIT    set    08
    SBitPrescale  set    0
    SBitRtccLoad  set    _SBitCycles


    if (_tempCompute >= 1)
     SBitPrescale set    0
     SBitRtccLoad set    (_SBitCycles >> 1)
     _BIT1_INIT   set    0
    endif

    if (_tempCompute >= 2)
```

```
SBitPrescale set   1
SBitRtccLoad set   (_SBitCycles >> 2)
endif

if (_tempCompute >= 4)
 SBitPrescale set   2
 SBitRtccLoad set   (_SBitCycles >> 3)
endif

if (_tempCompute >= 8)
 SBitPrescale set   3
 SBitRtccLoad set   (_SBitCycles >> 4)
endif

if (_tempCompute >= 16)
 SBitPrescale set   4
 SBitRtccLoad set   (_SBitCycles >> 5)
endif

if (_tempCompute >= 32)
 SBitPrescale set   5
 SBitRtccLoad set   (_SBitCycles >> 6)
endif

if (_tempCompute >= 64)
 SBitPrescale set   6
 SBitRtccLoad set   (_SBitCycles >> 7)
endif

if   (_tempCompute >= 128)
 SBitPrescale set   7
 SBitRtccLoad set   (_SBitCycles >> 8)
endif


;
;*****************************************************************************
;
;
#define _Cycle_Offset1 24                ;account for interrupt latency, call time

LOAD_RTCC     MACRO  Mode, K, Prescale

    if(UsePrescale == 0 && Mode == 0)
```

```
  movlw -K + _Cycle_Offset1
    else
  movlw -K + (_Cycle_Offset1 >> (Prescale+1))  ; Re Load RTCC init value + INT Latency Offset
    endif
  movwf _rtcc          ; Note that Prescaler is cleared when RTCC is written

       ENDM
;*****************************************************************************************

LOAD_BITCOUNT  MACRO

  movlw _DataBits+1
  movwf BitCount
  movlw 1
  movwf ExtraBitCount

    if  _PARITY_ENABLE
  movlw 2
  movwf ExtraBitCount
    endif

       ENDM
;
;*************************************************************************************************
;             Pin Assignements
;*************************************************************************************************
#define RX_MASK 0x10          ; RX pin is connected to RA4, ie. bit 4
#define RX_Pin  _porta,4      ; RX Pin : RA4
#define RX      RxTemp,4

#define TX      _portb,7      ; TX Pin , RB7

#define _RTS    _portb,5      ; RTS Pin, RB5, Output signal
#define _CTS    _portb,6      ; CTS Pin, RB6, Input signal


#define _txmtProgress  SerialStatus,0
#define _txmtEnable    SerialStatus,1

#define _rcvProgress   SerialStatus,2
#define _rcvOver       SerialStatus,3
#define _ParityErr     SerialStatus,4
#define _FrameErr      SerialStatus,5

#define _parityBit     SerialStatus,7

;*************************************************************************************************
```

```
_OPTION_SBIT   set    0x38          ; Increment on Ext Clock (falling edge), for START Bit Detect

  if UsePrescale
_OPTION_INIT   set    0x00          ; Prescaler is used depending on Input Clock & Baud Rate
  else
_OPTION_INIT   set    0x0F
  endif

        CBLOCK  0x0C
                TxReg               ; Transmit Data Holding/Shift Reg
                RxReg               ; Rcv Data Holding Reg
                        RxTemp
                SerialStatus        ; Txmt & Rev Status/Control Reg
                BitCount
                ExtraBitCount       ; Parity & Stop Bit Count
                SaveWReg            ; temp hold reg of WREG on INT
                        SaveStatus      ; temp hold reg of STATUS Reg on INT
        temp1, temp2
        ENDC

;*****************************************************************************************

        LIST
```

Appendix B - RS232.ASM

```
  TITLE           “RS232 Communications : Half Duplex : PIC16C6x/7x/8x”
  SUBTITLE        “Software Implementation : Interrupt Driven”

;*****************************************************************************************************
;                       Software Implementation Of RS232 Communications Using PIC16CXX
;                                           Half-Duplex
;
;  These routines are intended to be used with PIC16C6X/7X family. These routines can be
;  used with processors in the 16C6X/7X family which do not have on board Hardware Async
;  Serial Port.
;  MX..
;
;  Description :
;               Half Duplex RS-232 Mode Is implemented in Software.
;               Both Reception & Transmission are Interrupt driven
;               Only 1 peripheral (RTCC) used for both transmission & reception
;               RTCC is used for both timing generation (for bit transmission & bit polling)
;               and Start Bit Detection in reception mode.
;               This is explained in more detail in the Interrupt Subroutine.
;               Programmable Baud Rate (speed depnding on Input Clock Freq.), programmable
;               #of bits, Parity enable/disable, odd/even parity is implemented.
;                       Parity & Framing errors are detected on Reception
;
;                               RS-232 Parameters
;
;The RS-232 Parameters are defined as shown below:
;
;               _ClkIn          :       Input Clock Frequency of the processor
;                                       (NOTE : RC Clock Mode Is Not Suggested due to wide variations)
;               _BaudRate       :       Desired Baud Rate. Any valid value can be used.
;                                       The highest Baud Rate achievable depends on Input Clock Freq.
;                                       300 to 4800 Baud was tested using 4 Mhz Input Clock
;                                       300 to 19200 Baud was tested using 10 Mhz Input Clock
;                                       Higher rates can be obtained using higher Input Clock Frequencies.
;                                       Once the _BaudRate & _ClkIn are specified the program
;                                       automatically selectes all the appropiate timings
;               _DataBits       :       Can specify 1 to 8 Bits.
;               _StopBits       :       Limited to 1 Stop Bit. Must set it to 1.
;               _PARITY_ENABLE  :       Parity Enable Flag. Set it to TRUE or FALSE. If PARITY
;                                       is used, then set it to TRUE, else FALSE. See “_ODD_PARITY” flag
;                                       description below
;               _ODD_PARITY     :       Set it to TRUE or FALSE. If TRUE, then ODD PARITY is used, else
;                                       EVEN Parity Scheme is used.
;                                       This Flag is ignored if _PARITY_ENABLE is set to FALSE.
;
;
```

```
;  Usage :
;               An example is given in the main program on how to Receive & Transmit Data
;               In the example, the processor waits until a command is received. The command is interpreted
;               as the A/D Channel Number of PIC16C71. Upon reception of a command, the desired A/D channel
;               is selected and after A/D conversion, the 8 Bit A/D data is transmitted back to the Host.
;
;                       The RS-232 Control/Status Reg's bits are explained below :
;
;       "SerialStatus"          : RS-232 Status/Control Register
;
;       Bit 0   :       _txmtProgress   (1 if transmission in progress, 0 if transmission is complete)
;                                       After a byte is transmitted by calling "PutChar" function, the
;                                       user's code can poll this bit to check if transmission is complete.
;                                       This bit is reset after the STOP bit has been transmitted
;       Bit 1   :       _txmtEnable     Set this bit to 1 on initialization to enable transmission.
;                                       This bit can be used to Abort a transmission while the transmitter
;                                       is in progress (i.e when _txmtProgress = 1)
;       Bit 2   :       _rcvProgress    Indicates that the receiver is in middle of reception.It is reset when
;                                       a byte is received.
;       Bit 3   :       _rcvOver        This bit indicates the completion of Reception of a Byte. The user's
;                                       code can poll this bit after calling "GetChar" function. Once "GetChar"
;                                       function is called, this bit is 1 and is set to 0 after reception of
;                                       a complete byte (parity bit if enabled & stop bit)
;       Bit 4   :       _ParityErr      A 1 indicates Parity Error on Reception (for both even & odd parity)
;       Bit 5   :       _FrameErr       A 1 indicates Framing Error On Reception
;
;       Bit 6   :       _unused_        Unimplemented Bit
;
;       Bit 7   :       _parityBit      The 9 th bit of transmission or reception (status of PARITY bit
;                                       if parity is enabled)
;
;       To Transmit A Byte Of Data :
;                       1) Make sure _txmtProgress & _rcvOver bits are cleared
;                       2) Load TxReg with data to be transmitted
;                       3) CALL  PutChar function
;
;       To Receive A Byte Of Data :
;                       1) Make sure _txmtProgress & _rcvOver bits are cleared
;                       2) CALL GetChar function
;                       3) The received Byte is in TxReg after _rcvOver bit is cleared
;
;
; Rev 2, May 17,1994 Scott Fink
;       Corrected 7 bit and parity operation, corrected stop bit generation, corrected
;       receive prescaler settings.  Protected against inadvertant WDT reset.
;*************************************************************************************************

        Processor       16C71
```

```
        Radix   DEC
        EXPAND

        include         "16Cxx.h"

;*****************************************************************************************************
;                               Setup RS-232 Parameters
;*****************************************************************************************************

_ClkIn          equ     4000000         ; Input Clock Frequency is 4 Mhz
_BaudRate       set     1200            ; Baud Rate (bits per second) is 1200
_DataBits       set     8               ; 8 bit data, can be 1 to 8
_StopBits       set     1               ; 1 Stop Bit, 2 Stop Bits is not implemented

#define _PARITY_ENABLE  FALSE           ; NO Parity
#define _ODD_PARITY     FALSE           ; EVEN Parity, if Parity enabled
#define _USE_RTSCTS     FALSE           ; NO Hardware Handshaking is Used

        include         "rs232.h"

;*****************************************************************************************************
;
  ORG     _ResetVector
  goto    Start
;
  ORG     _IntVector
  goto    Interrupt
;
;*****************************************************************************************************
;                               Table Of ADCON0 Reg
; Inputs : WREG (valid values are 0 thru 3)
; Returns In WREG, ADCON0 Value, selecting the desired Channel
;
;  Program Memory :     6 locations
;  Cycles         :     5
;
;*****************************************************************************************************

GetADCon0:
  andlw   0x03                  ; mask off all bits except 2 LSBs (for Channel # 0, 1, 2, 3)
  addwf   _pcl
  retlw   (0xC1 | (0 << 3))     ; channel 0
  retlw   (0xC1 | (1 << 3))     ; channel 1
  retlw   (0xC1 | (2 << 3))     ; channel 2
GetADCon0_End:
  retlw   (0xC1 | (3 << 3))     ; channel 3
```

```
   if( (GetADCon0 & 0xff) >= (GetADCon0_End & 0xff))
  MESSG   "Warning : Crossing Page Boundary in Computed Jump, Make Sure PCLATH is Loaded Correctly"
   endif
;
;*****************************************************************************************************
;                               Initialize A/D Converter
;  <RA0:RA3>    Configure as Analog Inputs, VDD as Vref
;   A/D Clock Is Internal RC Clock
;   Select Channel 0
;
;  Program Memory :     6 locations
;  Cycles         :     7
;
;*****************************************************************************************************
InitAtoD:
  bsf     _rp0
  clrf    _adcon1
  bcf     _rp0
  movlw   0xC1
  movwf   _adcon0
  return
;
;*****************************************************************************************************
;                                     Main Program Loop
;
;  After appropriate initilization, The main program wait for a command from RS-232
;  The command is 0, 1, 2 or 3. This command/data represents the A/D Channel Number.
;  After a command is received, the appropriate A/D Channel is seleted and when conversion is
;  completed the A/D Data is transmitted back to the Host. The controller now waits for a new
;  command.
;*****************************************************************************************************

Start:
  call    InitSerialPort
;
WaitForNextSel:
   if _USE_RTSCTS
  bcf     _rp0
  bcf     _RTS                    ; ready to accept data from host
   endif
  call    GetChar                 ; wait for a byte reception
  btfsc   _rcvOver                ; _rcvOver Gets Cleared when a Byte Is Received (in RxReg)
  goto    $-1                     ; USER can perform other jobs here, can poll _rcvOver bit
;
; A Byte is received, Select The Desired Channel & TMXT the desired A/D Channel Data
;
  bcf     _rp0                    ; make sure to select Page 0
```

```
  movf    RxReg,w                 ; WREG = Commanded Channel # (0 thru 3)
  call    GetADCon0               ; Get ADCON0 Reg Constant from Table Lookup
  movwf   _adcon0                 ; Load ADCON0 reg, selecting the desired channel
  nop
;
  bsf     _go                     ; start conversion
  btfsc   _done
  goto    $-1                     ; Loop Until A/D Conversion Done

  movf    _adres,w
  movwf   TxReg
   if _USE_RTSCTS
  bsf     _RTS                    ; Half duplex mode, transmission mode, ask host not to send data
  btfsc   _CTS                    ; Check CTS signal if host ready to accept data
  goto    $-1
   endif
  call    PutChar
  btfsc   _txmtProgress
  goto    $-1                     ; Loop Until Transmission Over, User Can Perform Other Jobs


;
  goto    WaitForNextSel          ; wait for next selection (command from Serial Port)
;
;*****************************************************************************************************
;                               RS-232 Routines
;;****************************************************************************************************
;                               Interrupt Service Routine
;
; Only RTCC Inturrupt Is used. RTCC Inturrupt is used as timing for Serial Port Receive & Transmit
; Since RS-232 is implemented only as a Half Duplex System, The RTCC is shared by both Receive &
; Transmit Modules.
;       Transmission :
;                       RTCC is setup for Internal Clock increments and interrupt is generated when
;                       RTCC overflows. Prescaler is assigned, depending on The INPUT CLOCK & the
;                       desired BAUD RATE.
;       Reception :
;                       When put in receive mode, RTCC is setup for external clock mode (FALLING EDGE)
;                       and preloaded with 0xFF. When a Falling Edge is detected on RTCC Pin, RTCC
;                       rolls over and an Interrupt is generated (thus Start Bit Detect). Once the start
;                       bit is detected, RTCC is changed to INTERNAL CLOCK mode and RTCC is preloaded
;                       with a certain value for regular timing interrupts to Poll RTCC Pin (i.e RX pin).
;
;*****************************************************************************************************

Interrupt:
  btfss   _rtif
  retfie                          ; other interrupt, simply return & enable GIE
```

```
;
; Save Status On INT : WREG & STATUS Regs
;
  movwf   SaveWReg
  swapf   _status,w               ; affects no STATUS bits : Only way OUT to save STATUS Reg ?????
  movwf   SaveStatus
;
  btfsc   _txmtProgress
  goto    _TxmtNextBit            ; Txmt Next Bit
  btfsc   _rcvProgress
  goto    _RcvNextBit             ; Receive Next Bit
  goto    _SBitDetected           ; Must be start Bit
;
RestoreIntStatus:
  swapf   SaveStatus,w
  movwf   _status                 ; restore STATUS Reg
  swapf   SaveWReg                ; save WREG
  swapf   SaveWReg,w              ; restore WREG
  bcf     _rtif
  retfie
;
;*****************************************************************************************************
;
;
;
; Configure TX Pin as output, make sure TX Pin Comes up in high state on Reset
; Configure, RX_Pin (RTCC pin) as Input, which is used to poll data on reception
;
;  Program Memory :     9 locations
;  Cycles         :     10
;*****************************************************************************************************

InitSerialPort:
  clrf    SerialStatus
;
  bcf     _rp0                          ; select Page 0 for Port Access
  bsf     TX                            ; make sure TX Pin is high on powerup, use RB Port Pullup
  bsf     _rp0                          ; Select Page 1 for TrisB access
  bcf     TX                            ; set TX Pin As Output Pin, by modifying TRIS
  if _USE_RTSCTS
  bcf     _RTS                          ; RTS is output signal, controlled by PIC16Cxx
  bsf     _CTS                          ; CTS is Input signal, controlled by the host
  endif
  bsf     RX_Pin                        ; set RX Pin As Input for reception
  return
;
;*****************************************************************************************************
```

```
        include "txmtr.asm"  ; The Transmit routines are in file "txmtr.asm"
        include "rcvr.asm"   ; The Receiver Routines are in File "rcvr.asm"

;*********************************************************************************************************

        END
```

Appendix C - RCVR.ASM

```
;*******************************************************************************************
;             GetChar Function
;  Receives a Byte Of Data
;      When reception is complete, _rcvOver Bit is cleared
;      The received data is in RxReg
;
;  Program Memory :    15 locations (17 locations if PARITY is used)
;  Cycles         :    16 (18 if PARITY is USED)
;
;*******************************************************************************************
GetChar:
  bcf   _rp0
  bsf   _rcvOver                    ; Enable Reception, this bit gets reset on Byte Rcv Complete
  LOAD_BITCOUNT
        clrf   RxReg
  bcf   _FrameErr
        bcf    _ParityErr           ; Init Parity & Framing Errors
  clrf  _rtcc
  clrwdt
  bsf   _rp0
  movlw 07h
  movwf _option
  bcf   _rp0
  clrf  _rtcc
  bsf   _rp0
  movlw 0Fh
  movwf _option
  clrwdt
        movlw  _OPTION_SBIT         ; Inc On Ext Clk Falling Edge
  movwf _option                     ; Set Option Reg Located In Page 1
  bcf   _rp0                        ; make sure to select Page 0
  movlw 0xFF
  movwf _rtcc                       ; A Start Bit will roll over RTCC & Gen INT
  bcf   _rtif
  bsf   _rtie                       ; Enable RTCC Interrupt
  retfie                            ; Enable Global Interrupt
;
;*******************************************************************************************
;             Internal Subroutine
; entered from Interrupt Service Routine when Start Bit Is detected.
;
;  Program Memory :    14 locations
;  Cycles         :    12 (worst case)
;
;*******************************************************************************************
_SBitDetected:
```

```
    bcf   _rp0
    btfsc RX_Pin              ; Make sure Start Bit Interrupt is not a Glitch
    goto  _FalseStartBit      ; False Start Bit
    bsf   _rcvProgress
    clrf  _rtcc
    clrwdt
    bsf   _rp0
    movlw 07h
    movwf _option
    bcf   _rp0
    clrf  _rtcc
    bsf   _rp0
    movlw 0Fh
    movwf _option
    clrwdt
    movlw (_BIT1_INIT | SBitPrescale)   ; Switch Back to INT Clock
    movwf _option                       ; Set Option Reg Located In Page 1
    bcf   _rp0                          ; make sure to select Page 0
    LOAD_RTCC  1,(SBitRtccLoad), SBitPrescale
    goto  RestoreIntStatus
  ;
  _FalseStartBit:
    movlw 0xFF
    movwf _rtcc               ; reload RTCC with 0xFF for start bit detection
    goto  RestoreIntStatus
  ;*****************************************************************************
  ;
  ;       Internal Subroutine
  ; entered from Interrupt Service Routine when Start Bit Is detected.
  ;
  ;  Program Memory :   28 locations ( 43 locations with PARITY enabled)
  ;  Cycles         :   24 Worst Case
  ;
  ;*****************************************************************************
  _RcvNextBit:
    clrwdt
    bsf   _rp0
    movlw 07h
    movwf _option
    bcf   _rp0
    clrf  _rtcc
    clrwdt
    bsf   _rp0
    movlw 07h
    movwf _option
    bcf   _rp0
    clrf  _rtcc
    bsf   _rp0
```

```
    movlw 0Fh
    movwf _option
    clrwdt
    movlw (_OPTION_INIT | RtccPrescale)  ; Switch Back to INT Clock
    movwf _option                        ; Set Option Reg Located In Page 1
  ;
    bcf   _rp0
    movf  _porta,w                       ; read RX pin immediately into WREG
    movwf RxTemp
    LOAD_RTCC  0,RtccPreLoad, RtccPrescale  ; Macro to reload RTCC
    movf  _porta,w
    xorwf RxTemp,w
    andlw RX_MASK                        ; mask for only RX PIN (RA4)
    btfsc _z
    goto  _PinSampled                    ; both samples are same state
  _SampleAgain:
    movf  _porta,w
    movwf RxTemp                         ; 2 out of 3 majority sampling done
  _PinSampled:
    movf  BitCount,1
    btfsc _z
    goto  _RcvP_Or_S
  ;
    decfsz      BitCount
    goto  _NextRcvBit
  ;
  _RcvP_Or_S:
      if _PARITY_ENABLE
    decfsz      ExtraBitCount
        goto   _RcvParity
      endif
  ;
  _RcvStopBit:
    btfss RX                             ; may be framing Error or Glitch
    bsf   _FrameErr                      ; disable further interrupts
    bcf   _rtie
    bcf   _rcvProgress                   ; Byte Received, Can RCV/TXMT an other Byte
    bcf   _rcvOver
    if _PARITY_ENABLE
    movf  RxReg,w                        ; Generate Parity, for Parity check
    call  GenParity
    movlw 0
    btfsc _parityBit                     ; to mask off Received Parity Bit in _ParityErr
    movlw 0x10                           ; _ParityErr bit is set accordingly
    xorwf SerialStatus
    endif
    if _DataBits == 7
    rrf   RxReg,1
```

```
    bcf   RxReg,7
    endif
    goto  RestoreIntStatus
    ;
_NextRcvBit:
    bcf   _carry              ; prepare bit for shift
    btfsc RX
    bsf   _carry              ; shift in received data
    rrf   RxReg
    goto  RestoreIntStatus
    ;
    if _PARITY_ENABLE
_RcvParity:
    bcf   _ParityErr          ; Temporarily store PARITY Bit in _ParityErr
    btfsc RX                  ; Sample again to avoid any glitches
    bsf   _ParityErr
    goto  RestoreIntStatus
    endif
    ;
    ;*****************************************************************************
    ;
```

Appendix D - TXMTR.ASM

```
;*************************************************************************************************
;                    PutChar Function
;
; Function to transmit A Byte Of Data
; Before calling this routine, load the Byte to be transmitted into TxReg
; Make sure _txmtProgress & _rcvOver bits (in Status Reg) are cleared before
; calling this routine
;
;  Program Memory :    6 locations (10 locations if PARITY is Used)
;  Cycles         :    8 (13 if PARITY is Used)
;
;*************************************************************************************************
PutChar:
  bsf   _txmtEnable                     ; enable transmission
  bsf   _txmtProgress
  LOAD_BITCOUNT                         ; Macro to load bit count
  decf  BitCount,1
  if _DataBits == 7
  bsf   TxReg,7
  endif
;
   if _PARITY_ENABLE
  movf  TxReg,W
  call  GenParity                       ; If Parity is used, then Generate Parity Bit
   endif
;
  call  _TxmtStartBit
  bsf   _rtie                           ; Enable RTCC Overflow INT
  retfie                                ; return with _GIE Bit Set
;
;*************************************************************************************************
;             Internal Subroutine
; entered from Interrupt Service Routine when Start Bit Is detected.
;
;  Program Memory :   30 locations  (38 locations if PARITY is used)
;  Cycles         :   15 Worst Case
;
;*************************************************************************************************

_TxmtNextBit:
  bcf   _rp0
  LOAD_RTCC  0,RtccPreLoad, RtccPrescale     ; Macro to reload RTCC
;
  movf  BitCount                        ;done with data xmission?
  btfsc _z
  goto  _ParityOrStop                   ;yes, do parity or stop bit
```

```
;
    decf  BitCount                  ;no, send another
    goto  _NextTxmtBit
    ;
_ParityOrStop:
    if _PARITY_ENABLE
btfsc ExtraBitCount,1               ;ready for parity bit?
      goto   _SendParity
    endif
    movf  ExtraBitCount,1           ;check if sending stop bit
    btfsc _Z
    goto  DoneTxmt
    decf  ExtraBitCount,1
    ;
_StopBit:
    bsf   TX                        ; STOP Bit is High
    goto  RestoreIntStatus
    goto  DoneTxmt
    ;
_NextTxmtBit:
    bsf   _carry
    rrf   TxReg
    btfss _carry
          bcf   TX
    btfsc _carry
    bsf   TX
    ;
    btfss _txmtEnable               ; disable further interrupts, Transmission Aborted
    bsf   _rtie
    ;
    goto  RestoreIntStatus
    ;
    if _PARITY_ENABLE
_SendParity:
    decf  ExtraBitCount,1           ;subtract parity from count
    btfss _parityBit
    bcf   TX
    btfsc _parityBit
          bsf   TX
    goto  RestoreIntStatus
    endif

DoneTxmt
    bsf   TX                        ;STOP Bit is High
    bcf   _rtie                     ;disable further interrupts
    bcf   _txmtProgress             ;indicates end of xmission
    goto  RestoreIntStatus
    ;
```

```
;*****************************************************************************************************************
;       Internal Subroutine
; entered from Interrupt Service Routine when Start Bit Is detected.
;
; Program Memory :   9 locations
; Cycles         :   10
;
;*****************************************************************************************************************
_TxmtStartBit:
  bcf   _rp0
  clrf  _rtcc
  clrwdt
  bsf   _rp0
  movlw 07h
  movwf _option
  bcf   _rp0
  clrf  _rtcc
  bsf   _rp0
  movlw 0Fh
  movwf _option
  clrwdt
  movlw (_OPTION_INIT | RtccPrescale)
  movwf _option                 ; Set Option Reg Located In Page 1
  bcf   _rp0                    ; make sure to select Page 0
  bcf   TX                      ; Send Start Bit
  movlw -RtccPreLoad            ; Prepare for Timing Interrupt
  movwf _rtcc
  bcf   _rtif
  return
;*****************************************************************************************************************
;               Generate Parity for the Value in WREG
;
;  The parity bit is set in _parityBit (SerialStatus,7)
;  Common Routine For Both Transmission & Reception
;
; Program Memory :   16 locations
; Cycles         :   72
;
;*****************************************************************************************************************
  if _PARITY_ENABLE

GenParity:
  movwf temp2                   ;save data
  movf  BitCount,w              ;save bitcount
  movwf temp1
Parityloop
        rrf     temp2
```

```
    btfss _carry                ;put data in carry bit
    goto  NotOne
    xorlw 00h                   ;parity calculated by XORing all data bits
    goto  OneDone
NotOne
    xorlw 01h
OneDone
    decfsz      temp1           ;decrement count
    goto  Parityloop
    movwf temp1
; Parity bit is in Bit 0 of temp1
;
    if _ODD_PARITY
    bsf   _parityBit
    btfsc temp1,0
        bcf    _parityBit
    else
    bcf   _parityBit
    btfsc temp1,0
        bsf    _parityBit
    endif

    return
    endif
;*****************************************************************************************************************
```

Appendix E - RS232.LST

```
MPASM 01.00.02 Alpha  \PICMASTR\CU   5-20-1994  9:13:56                PAGE  1
RS232 Communications : Half Duplex : PIC16C6x/7x/8x
Software Implementation : Interrupt Driven
LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001         TITLE           “RS232 Communications : Half Duplex : PIC16C6x/7x/8x”
                     0002         SUBTITLE        “Software Implementation : Interrupt Driven”
                     0003
                     0004 ;*****************************************************************************************************
                     0005 ;                       Software Implementation Of RS232 Communications Using PIC16CXX
                     0006 ;                                              Half-Duplex
                     0007 ;
                     0008 ;  These routines are intended to be used with PIC16C6X/7X family. These routines can be
                     0009 ;  used with processors in the 16C6X/7X family which do not have on board Hardware Async
                     0010 ;  Serial Port.
                     0011 ;  MX..
                     0012 ;
                     0013 ;  Description :
                     0014 ;               Half Duplex RS-232 Mode Is implemented in Software.
                     0015 ;               Both Reception & Transmission are Interrupt driven
                     0016 ;               Only 1 peripheral (RTCC) used for both transmission & reception
                     0017 ;               RTCC is used for both timing generation (for bit transmission & bit polling)
                     0018 ;               and Start Bit Detection in reception mode.
                     0019 ;               This is explained in more detail in the Interrupt Subroutine.
                     0020 ;               Programmable Baud Rate (speed depnding on Input Clock Freq.), programmable
                     0021 ;               #of bits, Parity enable/disable, odd/even parity is implemented.
                     0022 ;                       Parity & Framing errors are detected on Reception
                     0023 ;
                     0024 ;                                RS-232 Parameters
                     0025 ;
                     0026 ;The RS-232 Parameters are defined as shown below:
                     0027 ;
                     0028 ;               _ClkIn          :       Input Clock Frequency of the processor
                     0029 ;                                       (NOTE : RC Clock Mode Is Not Suggested due to wide variations)
                     0030 ;               _BaudRate       :       Desired Baud Rate. Any valid value can be used.
                     0031 ;                                       The highest Baud Rate achievable depends on Input Clock Freq.
                     0032 ;                                       300 to 4800 Baud was tested using 4 Mhz Input Clock
                     0033 ;                                       300 to 19200 Baud was tested using 10 Mhz Input Clock
                     0034 ;                                       Higher rates can be obtained using higher Input Clock Frequencies.
                     0035 ;                                       Once the _BaudRate & _ClkIn are specified the program
                     0036 ;                                       automatically selectes all the appropiate timings
                     0037 ;               _DataBits       :       Can specify 1 to 8 Bits.
                     0038 ;               _StopBits       :       Limited to 1 Stop Bit. Must set it to 1.
                     0039 ;               _PARITY_ENABLE  :       Parity Enable Flag. Set it to TRUE or FALSE. If PARITY
```

```
;                       is used, then set it to TRUE, else FALSE. See "_ODD_PARITY" flag
;                       description below
;       _ODD_PARITY     :       Set it to TRUE or FALSE. If TRUE, then ODD PARITY is used, else
;                               EVEN Parity Scheme is used.
;                               This Flag is ignored if _PARITY_ENABLE is set to FALSE.
;
;
;       Usage :
;
;               An example is given in the main program on how to Receive & Transmit Data
;               In the example, the processor waits until a command is received. The command is interpreted
;               as the A/D Channel Number of PIC16C71. Upon reception of a command, the desired A/D channel
;               is selected and after A/D conversion, the 8 Bit A/D data is transmitted back to the Host.
;
;                       The RS-232 Control/Status Reg's bits are explained below :
;
;               "SerialStatus"          : RS-232 Status/Control Register
;
;               Bit 0   :       _txmtProgress   (1 if transmission in progress, 0 if transmission is complete)
;                                               After a byte is transmitted by calling "PutChar" function, the
;                                               user's code can poll this bit to check if transmission is complete.
;                                               This bit is reset after the STOP bit has been transmitted
;               Bit 1   :       _txmtEnable     Set this bit to 1 on initialization to enable transmission.
;                                               This bit can be used to Abort a transmission while the transmitter
;                                               is in progress (i.e when _txmtProgress = 1)
;               Bit 2   :       _rcvProgress    Indicates that the receiver is in middle of reception.It is reset when
;                                               a byte is received.
;               Bit 3   :       _rcvOver        This bit indicates the completion of Reception of a Byte. The user's
;                                               code can poll this bit after calling "GetChar" function. Once "GetChar"
;                                               function is called, this bit is 1 and is set to 0 after reception of
;                                               a complete byte (parity bit if enabled & stop bit)
;               Bit 4   :       _ParityErr      A 1 indicates Parity Error on Reception (for both even & odd parity)
;               Bit 5   :       _FrameErr       A 1 indicates Framing Error On Reception
;
;               Bit 6   :       _unused_        Unimplemented Bit
;
;               Bit 7   :       _parityBit      The 9 th bit of transmission or reception (status of PARITY bit
;                                               if parity is enabled)
;
;               To Transmit A Byte Of Data :
;                                       1) Make sure _txmtProgress & _rcvOver bits are cleared
;                                       2) Load TxReg with data to be transmitted
;                                       3) CALL  PutChar function
;
;               To Receive A Byte Of Data :
;                                       1) Make sure _txmtProgress & _rcvOver bits are cleared
;                                       2) CALL GetChar function
;                                       3) The received Byte is in TxReg after _rcvOver bit is cleared
```

```
                    0088 ;
                    0089 ;   Rev 2, May 17,1994 Scott Fink
                    0090 ;           Corrected 7 bit and parity operation, corrected stop bit generation, corrected
                    0091 ;           receive prescaler settings.  Protected against inadvertant WDT reset.
                    0092 ;*****************************************************************************************************
                    0093
                    0094                 Processor       16C71
                    0095                 Radix   DEC
                    0096                 EXPAND
                    0097
                    0098             include         “16Cxx.h”
                    0179
                    0180
                    0181
                    0098
                    0099
                    0100 ;*****************************************************************************************************
                    0101 ;                                 Setup RS-232 Parameters
                    0102 ;*****************************************************************************************************
                    0103
003D 0900           0104 _ClkIn          equ     4000000         ; Input Clock Frequency is 4 Mhz
04B0                0105 _BaudRate       set     1200            ; Baud Rate (bits per second) is 1200
0008                0106 _DataBits       set     8               ; 8 bit data, can be 1 to 8
0001                0107 _StopBits       set     1               ; 1 Stop Bit, 2 Stop Bits is not implemented
                    0108
0046                0109  #define _PARITY_ENABLE  FALSE           ; NO Parity
0047                0110  #define _ODD_PARITY     FALSE           ; EVEN Parity, if Parity enabled
0048                0111  #define _USE_RTSCTS     FALSE           ; NO Hardware Handshaking is Used
                    0112
                    0113             include         “rs232.h”
                    0001
                    0113
                    0114
                    0115 ;*****************************************************************************************************
                    0116 ;
                    0117
                    0118         ORG     _ResetVector
0000 2811           0119         goto    Start
                    0120 ;
                    0121
                    0122         ORG     _IntVector
0004 2823           0123         goto    Interrupt
                    0124 ;
                    0125 ;*****************************************************************************************************
                    0126 ;                                 Table Of ADCON0 Reg
                    0127 ; Inputs : WREG (valid values are 0 thru 3)
                    0128 ; Returns In WREG, ADCON0 Value, selecting the desired Channel
                    0129 ;
```

```
0130 ;  Program Memory :    6 locations
0131 ;  Cycles         :    5
0132 ;
0133 ;*****************************************************************************************************************
0134
0135 GetADCon0:
0136         andlw   0x03                    ; mask off all bits except 2 LSBs (for Channel # 0, 1, 2, 3)
0137         addwf   _pcl
0138         retlw   (0xC1 | (0 << 3))       ; channel 0
0139         retlw   (0xC1 | (1 << 3))       ; channel 1
0140         retlw   (0xC1 | (2 << 3))       ; channel 2
0141 GetADCon0_End:
000A 34D9    retlw   (0xC1 | (3 << 3))       ; channel 3
0142
0143     if( (GetADCon0 & 0xff) >= (GetADCon0_End & 0xff))
0144       MESSG   "Warning : Crossing Page Boundary in Computed Jump, Make Sure PCLATH is Loaded Correctly"
0145     endif
0146 ;
0147 ;*****************************************************************************************************************
0148 ;                                   Initialize A/D Converter
0149 ;
0150 ;  <RA0:RA3>    Configure as Analog Inputs, VDD as Vref
0151 ;  A/D Clock Is Internal RC Clock
0152 ;  Select Channel 0
0153 ;
0154 ;  Program Memory :    6 locations
0155 ;  Cycles         :    7
0156 ;
0157 ;*****************************************************************************************************************
0158 InitAtoD:
0005 3903    bsf     _rp0
0006 0782    clrf    _adcon1
0007 34C1    bcf     _rp0
0008 34C9    movlw   0xC1
0009 34D1    movwf   _adcon0
000B 1683    return
000C 0188
000D 1283
000E 30C1
000F 0088
0010 0008
0165 ;
0166 ;*****************************************************************************************************************
0167 ;                                   Main Program Loop
0168 ;
0169 ;  After appropriate initilization, The main program wait for a command from RS-232
0170 ;  The command is 0, 1, 2 or 3. This command/data represents the A/D Channel Number.
0171 ;  After a command is received, the appropriate A/D Channel is seleted and when conversion is
0172 ;  completed the A/D Data is transmitted back to the Host. The controller now waits for a new
0173 ;  command.
0174 ;*****************************************************************************************************************
0175
0176 Start:
0011 2033    call    InitSerialPort
0177
```

```
                    0178 ;
                    0179 WaitForNextSel:
                    0180    if _USE_RTSCTS
                    0181         bcf     _rp0
                    0182         bcf     _RTS                    ; ready to accept data from host
                    0183    endif
0012 2074           0184         call    GetChar                 ; wait for a byte reception
0013 198F           0185         btfsc   _rcvOver                ; _rcvOver Gets Cleared when a Byte Is Received (in RxReg)
0014 2813           0186         goto    $-1                     ; USER can perform other jobs here, can poll _rcvOver bit
                    0187 ;
                    0188 ; A Byte is received, Select The Desired Channel & TMXT the desired A/D Channel Data
                    0189 ;
0015 1283           0190         bcf     _rp0                    ; make sure to select Page 0
0016 080D           0191         movf    RxReg,w                 ; WREG = Commanded Channel # (0 thru 3)
0017 2005           0192         call    GetADCon0               ; Get ADCON0 Reg Constant from Table Lookup
0018 0088           0193         movwf   _adcon0                 ; Load ADCON0 reg, selecting the desired channel
0019 0000           0194         nop
                    0195 ;
001A 1508           0196         bsf     _go                     ; start conversion
001B 1908           0197         btfsc   _done
001C 281B           0198         goto    $-1                     ; Loop Until A/D Conversion Done
                    0199
001D 0809           0200         movf    _adres,w
001E 008C           0201         movwf   TxReg
                    0202    if _USE_RTSCTS
                    0203         bsf     _RTS                    ; Half duplex mode, transmission mode, ask host not to send data
                    0204         btfsc   _CTS                    ; Check CTS signal if host ready to accept data
                    0205         goto    $-1
                    0206    endif
ww001F 203A         0207         call    PutChar
0020 180F           0208         btfsc   _txmtProgress
0021 2820           0209         goto    $-1                     ; Loop Until Transmission Over, User Can Perform Other Jobs
                    0210
                    0211
                    0212 ;
0022 2812           0213         goto    WaitForNextSel          ; wait for next selection (command from Serial Port)
                    0214 ;
                    0215 ;*************************************************************************************************
                    0216 ;                               RS-232 Routines
                    0217 ;;************************************************************************************************
                    0218 ;                               Interrupt Service Routine
                    0219 ;
                    0220 ; Only RTCC Inturrupt Is used. RTCC Inturrupt is used as timing for Serial Port Receive & Transmit
                    0221 ; Since RS-232 is implemented only as a Half Duplex System, The RTCC is shared by both Receive &
                    0222 ; Transmit Modules.
                    0223 ;       Transmission :
                    0224 ;                       RTCC is setup for Internal Clock increments and interrupt is generated when
```

```
                        0225 ;                       RTCC overflows. Prescaler is assigned, depending on The INPUT CLOCK & the
                        0226 ;                       desired BAUD RATE.
                        0227 ;       Reception :
                        0228 ;                       When put in receive mode, RTCC is setup for external clock mode (FALLING EDGE)
                        0229 ;                       and preloaded with 0xFF. When a Falling Edge is detected on RTCC Pin, RTCC
                        0230 ;                       rolls over and an Interrupt is generated (thus Start Bit Detect). Once the start
                        0231 ;                       bit is detected, RTCC is changed to INTERNAL CLOCK mode and RTCC is preloaded
                        0232 ;                       with a certain value for regular timing interrupts to Poll RTCC Pin (i.e RX pin).
                        0233 ;
                        0234 ;*****************************************************************************************************
                        0235
                        0236 Interrupt:
0023 1D0B               0237         btfss   _rtif
0024 0009               0238         retfie                          ; other interrupt, simply return & enable GIE
                        0239 ;
                        0240 ; Save Status On INT : WREG & STATUS Regs
                        0241 ;
0025 0092               0242         movwf   SaveWReg
0026 0E03               0243         swapf   _status,w               ; affects no STATUS bits : Only way OUT to save STATUS Reg ?????
0027 0093               0244         movwf   SaveStatus
                        0245 ;
0028 180F               0246         btfsc   _txmtProgress
0029 2844               0247         goto    _TxmtNextBit            ; Txmt Next Bit
002A 190F               0248         btfsc   _rcvProgress
002B 28A8               0249         goto    _RcvNextBit             ; Receive Next Bit
002C 2890               0250         goto    _SBitDetected           ; Must be start Bit
                        0251 ;
                        0252 RestoreIntStatus:
002D 0E13               0253         swapf   SaveStatus,w
002E 0083               0254         movwf   _status                 ; restore STATUS Reg
002F 0E92               0255         swapf   SaveWReg                ; save WREG
0030 0E12               0256         swapf   SaveWReg,w              ; restore WREG
0031 110B               0257         bcf     _rtif
0032 0009               0258         retfie
                        0259 ;
                        0260 ;*****************************************************************************************************
                        0261 ;
                        0262 ;
                        0263 ;
                        0264 ; Configure TX Pin as output, make sure TX Pin Comes up in high state on Reset
                        0265 ; Configure, RX_Pin (RTCC pin) as Input, which is used to poll data on reception
                        0266 ;
                        0267 ;  Program Memory :     9 locations
                        0268 ;  Cycles         :     10
                        0269 ;*****************************************************************************************************
                        0270
                        0271 InitSerialPort:
```

```
0033 018F            0272         clrf    SerialStatus
                     0273 ;
0034 1283            0274         bcf     _rp0                          ; select Page 0 for Port Access
0035 1786            0275         bsf     TX                            ; make sure TX Pin is high on powerup, use RB Port Pullup
0036 1683            0276         bsf     _rp0                          ; Select Page 1 for TrisB access
0037 1386            0277         bcf     TX                            ; set TX Pin As Output Pin, by modifying TRIS
                     0278   if _USE_RTSCTS
                     0279         bcf     _RTS                          ; RTS is output signal, controlled by PIC16Cxx
                     0280         bsf     _CTS                          ; CTS is Input signal, controlled by the host
                     0281   endif
0038 1605            0282         bsf     RX_Pin                        ; set RX Pin As Input for reception
0039 0008            0283         return
                     0284 ;
                     0285 ;*****************************************************************************************************
                     0286
                     0287     include “txmtr.asm”    ; The Transmit routines are in file “txmtr.asm”
                     0001 ;*****************************************************************************************************
                     0002 ;                          PutChar Function
                     0003 ;
                     0004 ;   Function to transmit A Byte Of Data
                     0005 ;   Before calling this routine, load the Byte to be transmitted into TxReg
                     0006 ;   Make sure _txmtProgress & _rcvOver bits (in Status Reg) are cleared before
                     0007 ;   calling this routine
                     0008 ;
                     0009 ;  Program Memory :       6 locations (10 locations if PARITY is Used)
                     0010 ;  Cycles    :    8 (13 if PARITY is Used)
                     0011 ;
                     0012 ;*****************************************************************************************************
                     0013 PutChar:
003A 148F            0014         bsf     _txmtEnable                    ; enable transmission
003B 140F            0015         bsf     _txmtProgress
                     0016     LOAD_BITCOUNT                              ; Macro to load bit count
003C 3009               M         movlw   _DataBits+1
003D 0090               M         movwf   BitCount
003E 3001               M         movlw   1
003F 0091               M         movwf   ExtraBitCount
                        M     if  _PARITY_ENABLE
                        M         movlw   2
                        M         movwf   ExtraBitCount
                        M     endif
0040 0390            0017         decf    BitCount,1
                     0018   if _DataBits == 7
                     0019         bsf     TxReg,7
                     0020   endif
                     0021 ;
                     0022    if _PARITY_ENABLE
                     0023         movf    TxReg,W
                     0024         call    GenParity                     ; If Parity is used, then Generate Parity Bit
```

```
                      0025    endif
                      0026 ;
0041 2060             0027         call    _TxmtStartBit
0042 168B             0028         bsf     _rtie                          ; Enable RTCC Overflow INT
0043 0009             0029         retfie                                 ; return with _GIE Bit Set
                      0030 ;
                      0031 ;*****************************************************************************************************
                      0032 ;                  Internal Subroutine
                      0033 ; entered from Interrupt Service Routine when Start Bit Is detected.
                      0034 ;
                      0035 ;  Program Memory :   30 locations  (38 locations if PARITY is used)
                      0036 ;  Cycles    :   15 Worst Case
                      0037 ;
                      0038 ;*****************************************************************************************************
                      0039
                      0040 _TxmtNextBit:
0044 1283             0041         bcf     _rp0
                      0042     LOAD_RTCC  0,RtccPreLoad, RtccPrescale       ; Macro to reload RTCC
                         M     if(UsePrescale == 0 && 0 == 0)
                         M         movlw   -RtccPreLoad + _Cycle_Offset1
                         M     else
0045 3036                M         movlw   -RtccPreLoad + (_Cycle_Offset1 >> (RtccPrescale+1))  ; Re Load RTCC init value + INT La
                         M     endif
0046 0081                M         movwf   _rtcc                          ; Note that Prescaler is cleared when RTCC is written
                      0043 ;
0047 0890             0044         movf    BitCount                       ;done with data xmission?
0048 1903             0045         btfsc   _z
0049 284C             0046         goto    _ParityOrStop                  ;yes, do parity or stop bit
                      0047 ;
004A 0390             0048         decf    BitCount
004B 2853             0049         goto    _NextTxmtBit                   ;no, send another
                      0050 ;
                      0051 _ParityOrStop:
                      0052     if _PARITY_ENABLE
                      0053         btfsc   ExtraBitCount,1                ;ready for parity bit?
                      0054         goto    _SendParity
                      0055     endif
004C 0891             0056         movf    ExtraBitCount,1                ;check if sending stop bit
004D 1903             0057         btfsc   _z
004E 285C             0058         goto    DoneTxmt
004F 0391             0059         decf    ExtraBitCount,1
                      0060 ;
                      0061 _StopBit:
0050 1786             0062         bsf     TX                             ; STOP Bit is High
0051 282D             0063         goto    RestoreIntStatus
0052 285C             0064         goto    DoneTxmt
                      0065 ;
```

```
0066 _NextTxmtBit:
0053 1403 0067          bsf     _carry
0054 0C8C 0068          rrf     TxReg
0055 1C03 0069          btfss   _carry
0056 1386 0070          bcf     TX
0057 1803 0071          btfsc   _carry
0058 1786 0072          bsf     TX
          0073 ;
0059 1C8F 0074          btfss   _txmtEnable             ; disable further interrupts, Transmission Aborted
005A 168B 0075          bsf     _rtie
          0076 ;
005B 282D 0077          goto    RestoreIntStatus
          0078 ;
          0079   if _PARITY_ENABLE
          0080 _SendParity:
          0081          decf    ExtraBitCount,1         ;subtract parity from count
          0082          btfss   _parityBit
          0083          bcf     TX
          0084          btfsc   _parityBit
          0085          bsf     TX
          0086          goto    RestoreIntStatus
          0087   endif
          0088
          0089 DoneTxmt
005C 1786 0090          bsf     TX                      ;STOP Bit is High
005D 128B 0091          bcf     _rtie                   ;disable further interrupts
005E 100F 0092          bcf     _txmtProgress           ;indicates end of xmission
005F 282D 0093          goto    RestoreIntStatus
          0094 ;
          0095 ;*****************************************************************************
          0096 ;
          0097 ; entered from Interrupt Service Routine when Start Bit Is detected.
          0098 ;
          0099 ;  Program Memory :      9 locations
          0100 ;  Cycles         :   10
          0101 ;
          0102 ;*****************************************************************************
          0103 _TxmtStartBit:
0060 1283 0104          bcf     _rp0
0061 0181 0105          clrf    _rtcc
0062 0064 0106          clrwdt
0063 1683 0107          bsf     _rp0
0064 3007 0108          movlw   07h
0065 0081 0109          movwf   _option
0066 1283 0110          bcf     _rp0
0067 0181 0111          clrf    _rtcc
0068 1683 0112          bsf     _rp0
0069 300F 0113          movlw   0Fh
```

```
006A 0081  0114          movwf   _option
006B 0064  0115          clrwdt
006C 3001  0116          movlw   (_OPTION_INIT | RtccPrescale)
006D 0081  0117          movwf   _option                 ; Set Option Reg Located In Page 1
006E 1283  0118          bcf     _rp0                    ; make sure to select Page 0
006F 1386  0119          bcf     TX                      ; Send Start Bit
0070 3030  0120          movlw   -RtccPreLoad            ; Prepare for Timing Interrupt
0071 0081  0121          movwf   _rtcc
0072 110B  0122          bcf     _rtif
0073 0008  0123          return
           0124
           0125 ;*****************************************************************************************************************
           0126 ;
           0127 ;                   Generate Parity for the Value in WREG
           0128 ;
           0129 ; The parity bit is set in _parityBit (SerialStatus,7)
           0130 ; Common Routine For Both Transmission & Reception
           0131 ;
           0132 ; Program Memory :      16 locations
           0133 ; Cycles       :    72
           0134 ;
           0135 ;*****************************************************************************************************************
           0136   if _PARITY_ENABLE
           0137 GenParity:
           0138          movwf   temp2                   ;save data
           0139          movf    BitCount,w              ;save bitcount
           0140          movwf   temp1
           0141 Parityloop
           0142          rrf     temp2
           0143          btfss   _carry                  ;put data in carry bit
           0144          goto    NotOne
           0145          xorlw   00h                     ;parity calculated by XORing all data bits
           0146          goto    OneDone
           0147 NotOne   xorlw   01h
           0148 OneDone
           0149          decfsz  temp1                   ;decrement count
           0150          goto    Parityloop
           0151          movwf   temp1
           0152 ; Parity bit is in Bit 0 of temp1
           0153 ;
           0154   if _ODD_PARITY
           0155          bsf     _parityBit
           0156          btfsc   temp1,0
           0157          bcf     _parityBit
           0158   else
           0159          bcf     _parityBit
           0160          btfsc   temp1,0
           0161
```

```
                     0162         bsf     _parityBit
                     0163     endif
                     0164
                     0165         return
                     0166   endif
                     0167 ;*****************************************************************************************************
                     0287
                     0288     include “rcvr.asm”    ; The Receiver Routines are in File “rcvr.asm”
                     0001 ;*****************************************************************************
                     0002 ;                 GetChar Function
                     0003 ;  Receives a Byte Of Data
                     0004 ;          When reception is complete, _rcvOver Bit is cleared
                     0005 ;          The received data is in RxReg
                     0006 ;
                     0007 ;  Program Memory :       15 locations (17 locations if PARITY is used)
                     0008 ;  Cycles     :    16 (18 if PARITY is USED)
                     0009 ;
                     0010 ;*****************************************************************************
                     0011 GetChar:
0074 1283            0012         bcf     _rp0
0075 158F            0013         bsf     _rcvOver                ; Enable Reception, this bit gets reset on Byte Rcv Complete
                     0014     LOAD_BITCOUNT
0076 3009               M         movlw   _DataBits+1
0077 0090               M         movwf   BitCount
0078 3001               M         movlw   1
0079 0091               M         movwf   ExtraBitCount
                        M     if  _PARITY_ENABLE
                        M         movlw   2
                        M         movwf   ExtraBitCount
                        M     endif
007A 018D            0015         clrf    RxReg
007B 128F            0016         bcf     _FrameErr
007C 120F            0017         bcf     _ParityErr              ; Init Parity & Framing Errors
007D 0181            0018         clrf    _rtcc
007E 0064            0019         clrwdt
007F 1683            0020         bsf     _rp0
0080 3007            0021         movlw   07h
0081 0081            0022         movwf   _option
0082 1283            0023         bcf     _rp0
0083 0181            0024         clrf    _rtcc
0084 1683            0025         bsf     _rp0
0085 300F            0026         movlw   0Fh
0086 0081            0027         movwf   _option
0087 0064            0028         clrwdt
0088 3038            0029         movlw   _OPTION_SBIT         ; Inc On Ext Clk Falling Edge
0089 0081            0030         movwf   _option              ; Set Option Reg Located In Page 1
008A 1283            0031         bcf     _rp0                ; make sure to select Page 0
008B 30FF            0032         movlw   0xFF
```

```
008C 0081           0033         movwf   _rtcc               ; A Start Bit will roll over RTCC & Gen INT
008D 110B           0034         bcf     _rtif
008E 168B           0035         bsf     _rtie               ; Enable RTCC Interrupt
008F 0009           0036         retfie                      ; Enable Global Interrupt
                    0037 ;
                    0038 ;*************************************************************************************
                    0039 ;                 Internal Subroutine
                    0040 ; entered from Interrupt Service Routine when Start Bit Is detected.
                    0041 ;
                    0042 ;  Program Memory :       14 locations
                    0043 ;  Cycles    :    12 (worst case)
                    0044 ;
                    0045 ;*************************************************************************************
                    0046 _SBitDetected:
0090 1283           0047         bcf     _rp0
0091 1A05           0048         btfsc   RX_Pin                      ; Make sure Start Bit Interrupt is not a Glitch
0092 28A5           0049         goto    _FalseStartBit              ; False Start Bit
0093 150F           0050         bsf     _rcvProgress
0094 0181           0051         clrf    _rtcc
0095 0064           0052         clrwdt
0096 1683           0053         bsf     _rp0
0097 3007           0054         movlw   07h
0098 0081           0055         movwf   _option
0099 1283           0056         bcf     _rp0
009A 0181           0057         clrf    _rtcc
009B 1683           0058         bsf     _rp0
009C 300F           0059         movlw   0Fh
009D 0081           0060         movwf   _option
009E 0064           0061         clrwdt
009F 3002           0062         movlw   (_BIT1_INIT | SBitPrescale)     ; Switch Back to INT Clock
00A0 0081           0063         movwf   _option                         ; Set Option Reg Located In Page 1
00A1 1283           0064         bcf     _rp0                            ; make sure to select Page 0
                    0065     LOAD_RTCC  1,(SBitRtccLoad), SBitPrescale
                       M     if(UsePrescale == 0 && 1 == 0)
                       M         movlw   -(SBitRtccLoad) + _Cycle_Offset1
                       M     else
00A2 3081              M         movlw   -(SBitRtccLoad) + (_Cycle_Offset1 >> (SBitPrescale+1))  ; Re Load RTCC init value + INT La
                       M     endif
00A3 0081              M         movwf   _rtcc           ; Note that Prescaler is cleared when RTCC is written
00A4 282D           0066         goto    RestoreIntStatus
                    0067 ;
                    0068 _FalseStartBit:
00A5 30FF           0069         movlw   0xFF
00A6 0081           0070         movwf   _rtcc                           ; reload RTCC with 0xFF for start bit detection
00A7 282D           0071         goto    RestoreIntStatus
                    0072 ;
                    0073 ;*************************************************************************************
                    0074 ;                 Internal Subroutine
```

```
                     0075 ; entered from Interrupt Service Routine when Start Bit Is detected.
                     0076 ;
                     0077 ;  Program Memory :       28 locations ( 43 locations with PARITY enabled)
                     0078 ;  Cycles    :     24 Worst Case
                     0079 ;
                     0080 ;*****************************************************************************************
                     0081 _RcvNextBit:
00A8 0064            0082         clrwdt
00A9 1683            0083         bsf     _rp0
00AA 3007            0084         movlw   07h
00AB 0081            0085         movwf   _option
00AC 1283            0086         bcf     _rp0
00AD 0181            0087         clrf    _rtcc
00AE 0064            0088         clrwdt
00AF 1683            0089         bsf     _rp0
00B0 3007            0090         movlw   07h
00B1 0081            0091         movwf   _option
00B2 1283            0092         bcf     _rp0
00B3 0181            0093         clrf    _rtcc
00B4 1683            0094         bsf     _rp0
00B5 300F            0095         movlw   0Fh
00B6 0081            0096         movwf   _option
00B7 0064            0097         clrwdt
00B8 3001            0098         movlw   (_OPTION_INIT | RtccPrescale)    ; Switch Back to INT Clock
00B9 0081            0099         movwf   _option                          ; Set Option Reg Located In Page 1
                     0100 ;
00BA 1283            0101         bcf     _rp0
00BB 0805            0102         movf    _porta,w                        ; read RX pin immediately into WREG
00BC 008E            0103         movwf   RxTemp
                     0104     LOAD_RTCC  0,RtccPreLoad, RtccPrescale      ; Macro to reload RTCC
                        M     if(UsePrescale == 0 && 0 == 0)
                        M         movlw   -RtccPreLoad + _Cycle_Offset1
                        M     else
00BD 3036               M         movlw   -RtccPreLoad + (_Cycle_Offset1 >> (RtccPrescale+1))  ; Re Load RTCC init value + INT La
                        M     endif
00BE 0081               M         movwf   _rtcc           ; Note that Prescaler is cleared when RTCC is written
00BF 0805            0105         movf    _porta,w
00C0 060E            0106         xorwf   RxTemp,w
00C1 3910            0107         andlw   RX_MASK         ; mask for only RX PIN (RA4)
00C2 1903            0108         btfsc   _z
00C3 28C6            0109         goto    _PinSampled     ; both samples are same state
                     0110 _SampleAgain:
00C4 0805            0111         movf    _porta,w
00C5 008E            0112         movwf   RxTemp          ; 2 out of 3 majority sampling done
                     0113 _PinSampled:
00C6 0890            0114         movf    BitCount,1
00C7 1903            0115         btfsc   _z
```

```
                        0116          goto    _RcvP_Or_S
                        0117 ;
00C8 28CB               0118          decfsz  BitCount
00C9 0B90               0119          goto    _NextRcvBit
00CA 28D1               0120 ;
                        0121 _RcvP_Or_S:
                        0122   if _PARITY_ENABLE
                        0123          decfsz  ExtraBitCount
                        0124          goto    _RcvParity
                        0125   endif
                        0126 ;
                        0127 _RcvStopBit:
00CB 1E0E               0128          btfss   RX
00CC 168F               0129          bsf     _FrameErr               ; may be framing Error or Glitch
00CD 128B               0130          bcf     _rtie                   ; disable further interrupts
00CE 110F               0131          bcf     _rcvProgress
00CF 118F               0132          bcf     _rcvOver                ; Byte Received, Can RCV/TXMT an other Byte
                        0133   if _PARITY_ENABLE
                        0134          movf    RxReg,w
                        0135          call    GenParity               ; Generate Parity, for Parity check
                        0136          movlw   0
                        0137          btfsc   _parityBit
                        0138          movlw   0x10                    ; to mask off Received Parity Bit in _ParityErr
                        0139          xorwf   SerialStatus            ; _ParityErr bit is set accordingly
                        0140   endif
                        0141   if _DataBits == 7
                        0142          rrf     RxReg,1
                        0143          bcf     RxReg,7
                        0144   endif
00D0 282D               0145          goto    RestoreIntStatus
                        0146 ;
                        0147 _NextRcvBit:
00D1 1003               0148          bcf     _carry
00D2 1A0E               0149          btfsc   RX                      ; prepare bit for shift
00D3 1403               0150          bsf     _carry
00D4 0C8D               0151          rrf     RxReg                   ; shift in received data
00D5 282D               0152          goto    RestoreIntStatus
                        0153 ;
                        0154   if _PARITY_ENABLE
                        0155 _RcvParity:
                        0156          bcf     _ParityErr              ; Temporarily store PARITY Bit in _ParityErr
                        0157          btfsc   RX                      ; Sample again to avoid any glitches
                        0158          bsf     _ParityErr
                        0159          goto    RestoreIntStatus
                        0160   endif
                        0161 ;
                        0162 ;*****************************************************************************
                        0288
```

```
0289
0290 ;*****************************************************************************************************************
0291
0292            END
0293
0294
0295
0296

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : X-XXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX

0080 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXX XXXXXX---------- ---------- ----------

All other memory blocks unused.

Errors   :   0
Warnings :   0
```

NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770 640-0034 Fax: 770 640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408 436-7950 Fax: 408 436-7955

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

Korea
Microchip Technology
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku,
Seoul, Korea
Tel: 82 2 554 7200 Fax: 82 2 558 5934

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 Muenchen, Germany
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95