AN712

RS-232 Autobaud for the PIC16C5X Devices

Author: Thomas Schmidt
Microchip Technology

INTRODUCTION

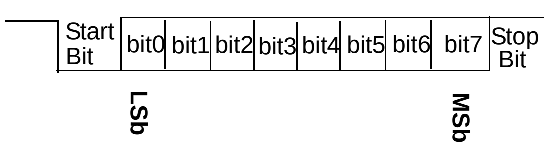
This application note describes an implementation of a RS-232 Autobaud routine on a PIC16C54B microcontroller.

Many microcontroller applications require chip-to-chip serial communication. Since the PIC16C54B has no USART, serial communication must be performed in software. Some applications use multiple transmission rates. Multiple transmission rates require software which detects the transmission rate and adjusts the receive and transmit routines according to the transmission rate.

ASYNCHRONOUS SERIAL I/O COMMUNICATION

Figure 1 shows the format of a data byte transferred via a serial communication line. Before the actual data byte is going to be transmitted, the data line is set to a high level. The first bit transmitted is called the start-bit and is always low, followed by the actual data. The data is transmitted with the LSb (last-significant-bit) first and the MSb (most significant bit) last. A high level represents a one bit and a low level a zero bit. The final bit transmitted is the stop-bit. The stop-bit is always a logic high.

FIGURE 1: DATA BYTE



The number of bits transmitted per second is equal to the baud rate. The inverse of the baud rate equals to the transmission time for one bit.

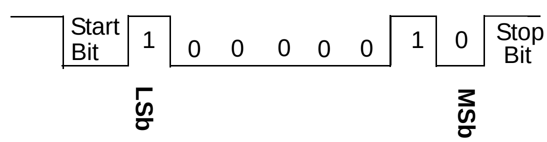
EXAMPLE 1: BAUDRATE CALCULATION

$$t_{one-bit} = \frac{1}{9600 Baud} = 104 \mu s$$

In asynchronous communication, the receiver must know the baud rate of the transmitter, because only the data shown in Figure 1 is transmitted. No clock is provided by the transmitter.

Example 2 depicts the asynchronous transmission of the character 'A'. The character 'A' has the value 41_h (ASCII).

EXAMPLE 2: ASYNCHRONOUS TRANSMISSION OF CHARACTER 'A'



Note: Character 'A' is equivalent to 41_H . 41_H=01000001_b

Autobaud and Asynchronous Serial Communication

In some systems, the transmission is not fixed to a baud rate. In this case, the received has to adjust the baud rate to that of the transmitter. Autobaud means that the receiver measures the transmission time of a calibration character and adjusts the delay routines for the baud rate generation accordingly .

THE SYSTEM

This chapter gives an overview on the setup of the hardware and software.

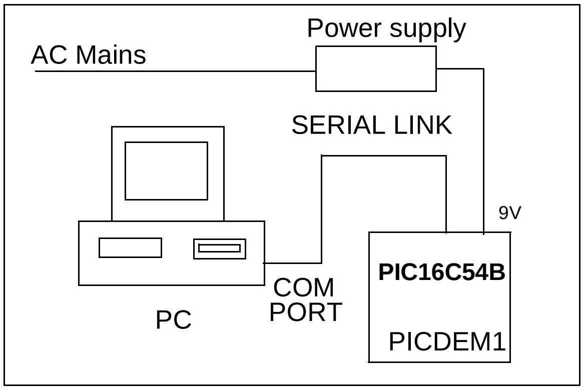
The Hardware

In this application, a PIC16C54B is connected to a PC. The PIC16C54B is placed on a PICDEM1 board. The PICDEM1 board provides a DSUB9 connector to a PC and a MAX232 interface circuit.

The PICDEM1 board is connected via the DSUB9 connector and a serial cable to the serial port of the PC. In this application, the PC sends a calibration character to the PIC16C54B. The PIC16C54B detects the transmission rate by measuring the bit length of transmitted zeros in a calibration character. The transmission time is measure by a software counter. The value of the software counter represents the value of the transmission rate for one bit. This value is used to generate a delay for bit sampling.

The hardware setup for this application note is shown in Figure 2.

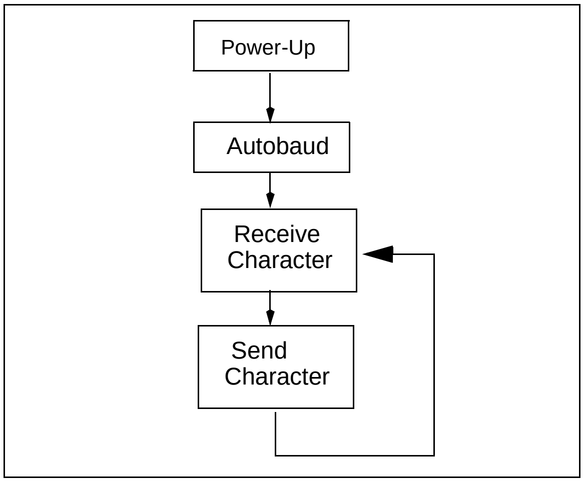
FIGURE 2: HARDWARE SETUP



The Program Flow

The program flow is shown in Figure 3.

FIGURE 3: PROGRAM FLOW OF THE MAIN ROUTINE



After power-up, the PIC16C54B initializes the I/O ports and waits for a calibration character from the PC. When the PC sends the calibration character, the PIC16C54B measures the transmission rate. This is done within the autobaud routine. Once the transmission rate has been detected, the PC has to send a second character. This character is received and echoed to the PC by the PIC16C54B. This process, receiving and transmitting characters, runs in an infinite loop.

The software is divided into three modular routines:

- Autobaud routine
- Receive routine
- Transmit routine

Each routine is a separate software module and can easily be integrated in custom code.

The communication between the PC and the PIC16C54B is half-duplex. In order to implement a full-duplex communication, please refer to *AN510 Implementation Of An Asynchronous Serial I/O*.

THE AUTOBAUD ROUTINE

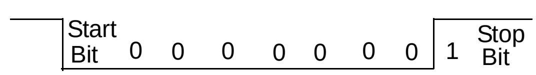
This chapter describes the theory of operation and the implementation of the autobaud routine.

Note: The software is designed for a 8-N-1 communication. Where 8 equals the number of data bits (start and stop bit not included), N is equal to the no parity bit and 1 is equal to the one stop bit.

In order to adjust to the transmission rate on the receiver side, the transmitter has to send a known character to the receiver. This character is called the calibration character. The receiver must know the pattern of the character, so it can measure the time to receive one or more bits. From the measured time, the receiver calculates the transmission time for one bit. This time is used in a receive or transmit routine to generate the baud rate.

The calibration value used for the autobaud routine in this application note is shown in Figure 4.

FIGURE 4: CALIBRATION CHARACTER FOR AUTOBAUD ROUTINE



In the first step, the autobaud routine looks for the start-bit. After the start-bit has been detected, a 16-bit software counter will increment until the next low to high transition is detected (see Figure 4). This means the autobaud routine measures the transmission time of eight zeros (including the start-bit).

The value in the counter represents the value for the transmission rate for 8 zeros. In order to calculate the transmission time for one bit, the value of the 16-bit counter is divided by 8. The result is the transmission time for one bit.

While measuring the transmission time and calculating the transmission time for one bit, the autobaud routine has to check if the 16-bit counter overflows or the result of the division could be zero. A counter overflow means that the transmitted signal is to slow. If the division by 8 equals zero, that means that the incoming signal is too fast.

The Implementation

The implementation of the autobaud routine can be broken up into 6 sections.

1. Check for start-bit
2. Measure time (increment counter)
3. Divide measured time by eight
4. Calculate time for half the transmission time for one bit (divide previous result by two). Half the baudrate is used in the receive routine to place the sampling of the bits in the middle.
5. Adjust result for receive and transmit routines
6. Check if both calculated results are greater than zero. If one of the results is zero, the baudrate cannot be generated because the received signal was to fast.

Each of this sections will be explained separately in the following text. The entire source code for the autobaud, as well as the receive and transmit routines, are given in the Appendix.

Check for Start-Bit

In the first step, the autobaud routine is called and the registers are initialized (see Figure 5). The low and the high byte of the autobaud counter are set to zero. The autobaud status register is also cleared. The autobaud status register contains two error flags, which indicate if the incoming signal was too fast or too slow. After the initialization, the receive pin RX is checked for a high to low transition. When this is detected, the autobaud routine starts measuring.

FIGURE 5: CHECK FOR START BIT

```
Autobaud        clrf    AUTOBAUD_LOW        ; reset register
                clrf    AUTOBAUD_HIGH       ; reset register
                clrf    AUTOHALF_LOW        ; reset register
                clrf    AUTOHALF_HIGH       ; reset register
                clrf    AUTOB_STATUS        ; reset autobaud
                                            ; status register
TestStartBit
                btfsc   PORTA, RX           ; check for start-bit
                goto    TestStartBit        ; Start-bit not found
```

Measure Time To Receive Calibration Word

After the start-bit is detected, the autobaud routine measures the time to receive the calibration character. The source code of this section is shown in Figure 6. The calibration character has the pattern 10000000b. The autobaud routine increments a 16-bit counter until a low to high transition is found. The registers for the 16-bit counter are called AUTOBAUD_HIGH (high byte) and AUTOBAUD_LOW (low byte). If the high byte overflows the error flag SIGNAL_SLOW in the register, AUTOBAUD_STATUS will be set. An overflow means that the incoming signal is too slow, because it takes more cycles to increment the counter than to transmit the full calibration character. See Figure 6.

Calculate Transmission Time For One Bit

After all bits are received the measured time has to be divided by eight, because the time to receive eight zeros was measured. The division is simply done by shifting the 16-bit counter three times to the right. Zeros are shifted into the counter from the left side. The transmission time for one bit is stored in the registers AUTOBAUD_LOW and AUTOBAUD_HIGH.

FIGURE 6: MEASURE TIME TO RECEIVE CALIBRATION WORD

```
Autobaud       clrf   AUTOBAUD_LOW      ; reset register
TestBitHigh    btfsc  PORTA, RX         ; Test for end of bit stream
               goto   Calculate         ; End of bit stream, now calculate
                                        ; bit time for one bit
               incfsz AUTOBAUD_LOW, f   ; increment Autobaud low register
               goto   TestBitHigh       ; test for high bit
               incfsz AUTOBAUD_HIGH, f  ; increment high byte of autobaud register
               goto   TestBitHigh       ; test for end of bit stream
               goto   Signal2Slow       ; High byte got an overflow. Transmitted
                                        ; signal is to slow for clock speed of the uc
```

FIGURE 7: CALCULATION OF TRANSMISSION TIME FOR ONE BIT

```
Autobaud       clrf   AUTOBAUD_LOW      ; reset register
                                        ; divide by measure time by 8 (8 zero where transmitted
                                          including
                                        ; start-bit)
Calculate      movlw  0x03              ; Initialize count register
               movwf  COUNTER           ; Counter for number for rotates = 3
Divide         bcf    STATUS, C         ; clear carry bit
               rrf    AUTOBAUD_HIGH,f   ; rotate autobaud high register
               rrf    AUTOBAUD_LOW, f   ; rotate autobaud low register
               decfsz COUNTER, f        ; decrement counter
               goto   Divide            ; divide
```

Calculate Half The Bit Time

After the transmission time for one bit is calculated, the transmission time for half the bit time has to be computed. This value is needed in the received routine to place sampling in the middle of each bit. After the start bit has been detected in the receive routine, the routine waits 1.5 bit times before the first data bit is sampled. This ensures that the sampling always happens in the middle of the bit.

The calculation of half the bit time is done by simply shifting the 16-bit counter to the right once. The result of the division is stored in the registers AUTOHALF_HIGH and AUTOHALF_LOW. The source code for this section of the autobaud routine is shown in Figure 8.

Adjust Transmission Times For Receive and Transmit Routine

The value of the 16-bit counter for the full bit time and the value for half the bit time have to be adjusted for the receive and transmit routine. Each count in the register AUTOBAUD_LOW and AUTOHALF_LOW stands for 5 instruction cycles, because it took five instruction cycles to get one count. Since the receive and transmit routines have a software overhead for storing or restoring data, this overhead has to be subtracted from the counter values.

After each adjustment, the result is checked to see if it is negative. If this is the case, error flag SIGNAL2FAST will be set. See Figure 9.

FIGURE 8: CALCULATION OF HALF THE BIT TIME

```
Autobaud       clrf    AUTOBAUD_LOW       ; reset register
                                          ; Calculate half the bit time
CalcHalfBit    bcf     STATUS, C          ; clear carry bit
               rrf     AUTOBAUD_HIGH,w    ; rotate autobaud high register
               movwf   AUTOHALF_HIGH      ; copy result into AUTOHALF_HIGH register
               rrf     AUTOBAUD_LOW, w    ; rotate autobaud high register
               movwf   AUTOHALF_LOW       ; copy result into AUTOHALF_LOW register
```

FIGURE 9: COUNTER ADJUSTMENT AND CHECK IF COUNTERS ARE NEGATIVE

```
Autobaud       clrf    AUTOBAUD_LOW       ; reset register
AdjustLowByte  movlw   0x3                ; 18-19 instruction cycles overhead from
                                          ; transmit and receive routine. This overhead
                                          ; must be subtracted from iterations
               subwf   AUTOBAUD_LOW, f    ; Adjust low byte from Autobaud counter
               btfss   STATUS, C          ; Is result negative? (equal=0 will be checked
                                          ; at ErrorCheck). C=0 result is negative
               goto    Signal2Fast        ; Signal is to fast for receive and transmit routine
               movlw   0x02               ; subtract 2 from low byte of half the bit time
               subwf   AUTOHALF_LOW, f    ; subtract from low byte of half the bit time
               btfss   STATUS, C          ; Is result negative? (equal=0 will be checked
                                          ; at ErrorCheck). C=0 result is negative
               goto    Signal2Fast        ; Signal is to fast for receive and transmit routine
```

Check If Both Counter Values Are Zero

After the adjustment, both counter values for the full and half bit time are checked for zeros. If this is the case, the error flag SIGNAL2FAST is set. If both counters are greater than or equal to one, the autobaud routine returns to the main routine. The source code for this section of the autobaud routine is shown in Figure 10.

FIGURE 10: CHECK OF COUNTER VALUES

```
Autobaud       clrf   AUTOBAUD_LOW              ; reset register
                                                ; check if AUTOBAUD_HIGH and AUTOBAUD_LOW are
                                                  zero.
                                                ; This means the transmission time for one byte
                                                  is too high
ErrorCheck     movf   AUTOBAUD_HIGH,w           ; copy high byte of autobaud counter register into
                                                ; w-register
               xorwf  AUTOBAUD_LOW, w           ; AUTOBAUD_HIGH = AUTOBAUD_LOW?
               btfss  STATUS, Z                 ; is result zero?
               goto   ErrorCheckHalf            ; Result is not zero, therefore finish autobaud
                                                ; routine
               goto   Signal2Fast               ; Signal is to fast for routine
ErrorCheckHalf movf   AUTOHALF_HIGH,w           ; copy high byte of autobaud counter register into
                                                ; w-register
               xorwf  AUTOHALF_LOW, w           ; AUTOBAUD_HIGH = AUTOBAUD_LOW?
               btfss  STATUS, Z                 ; is result zero?
               goto   EndAutoBaud               ; Result is not zero, therefore finish autobaud
                                                ; routine
                                                ; Error: delay for half the bit time is zero,
                                                  therefore a
                                                ; delay cannot be generated with the delay
                                                  routines. Incoming signal
                                                ; is to fast for clock speed.
Signal2Fast    bsf    AUTOB_STATUS, SIGNAL_FAST ; set error flag
               retlw  0x00                      ; return to operating system
Signal2Slow    bsf    AUTOB_STATUS, SIGNAL_SLOW ; set error flag
EndAutoBaud    retlw  0x00                      ; Return to operating system
```

THE TRANSMIT ROUTINE

The source code for the transmit routine is shown in Figure 11.

FIGURE 11: SOURCE CODE OF THE TRANSMIT ROUTINE

```
                                              ; Transmit routine
                                              ; Transmits LSB first
                                              ; Software overhead = 10 instruction cycles
                                                (including call
                                              ; to DelayFullBit routine, return from
                                              ; delay routine not included)
Transmit       movlw      BITS                ; Number of bit’s to transmit
               movwf      COUNTER             ; Initialize count register
               bcf        PORTA, TX           ; Generate start-bit
               call       DelayFullBit        ; Generate Delay for one bit-time
TransmitNext   rrf        RXTX_REG, f         ; Rotate receive register
               btfsc      STATUS, C           ; Test bit to be transmitted
               bsf        PORTA, TX           ; Transmit one
               btfss      STATUS, C           ; Check carry bit if set
               bcf        PORTA, TX           ; Transmit a zero
               call       DelayFullBit        ; call Delay routine
               decfsz     COUNTER, f          ; Decrement count register
               goto       TransmitNext        ; Transmit next bit
               bsf        PORTA, TX           ; Generate Stop bit
               call       DelayFullBit        ; Delay for Stop bit
               retlw      0x00                ; Return to operating system
                                              :
```

In the first step, the transmit routine initializes the register Count to 8. After the initialization, the RXTX_REG register is rotated by one position to the right. The bit-0 of the RXTX_Reg is now stored in the carry flag. The carry bit is checked whether it is a ‘1’ or a ‘0’. If the carry bit is set, the TX-pin is also set, otherwise the TX-pin is cleared. After all bits are transmitted, the stop-bit is send.

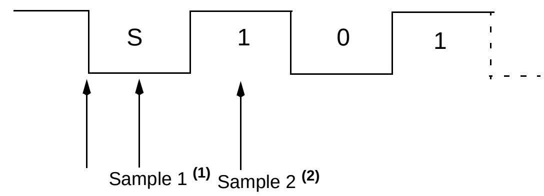
The delay for the transmission is generated by the DELAYFULLBit routine.

THE RECEIVE ROUTINE

The source code for the receive routine is shown in Figure 13.

The receive routine first resets the receive register to ‘0’ and initializes the Count register with 8. After the initialization, the routine checks for the start-bit. When the start bit is detected ,the receive routine waits 1.5 times the transmission time of one bit before sampling the next bit. This ensures that the bits are sampled in the middle and not at the beginning or end of the bit (see Figure 12). The delay for half the bit time is generated by the routine DelayHalfBit. After the delay, the bit is sample and stored in the register RXTX_REG.

FIGURE 12: RECEIVE ROUTINE SAMPLING



Note 1: Delay is generated by using delay value from register AutoBaud2

Note 2: Delay is generated by using delay value from register AutoBaud

FIGURE 13: SOURCE CODE OF THE RECEIVE ROUTINE

```
                                                       ; Receive Routine
                                                       ; receive routine = 11 instruction cycles per
                                                         iteration
                                                       ; including call to DelayFullBit routine
Receive            clrf    RXTX_REG                    ; Clear receive register
                   movlw   BITS                        ; Number of bits to receive
                   movwf   COUNTER                     ; Load number of bits into counter register
ReceiveStartBit    btfsc   PORTA, RX                   ; Test for start bit
                   goto    ReceiveStartBit             ; Startbit not found
                   call    DelayHalfBit                ; Wait until middle of start bit
                   call    DelayFullBit                ; Ignore start-bit and sample first
                                                       ; data bit in the middle of the bit
ReceiveNext        btfsc   PORTA, RX                   ; Is bit a zero or a one
                   bsf     STATUS,C                    ; bit is a one => set carry bit
                   btfss   PORTA, RX                   ; Is bit a one or a zero
                   bcf     STATUS,C                    ; bit is a zero => clear carry bit
                   rrf     RXTX_REG, f                 ; Rotate receive register
                   call    DelayFullBit                ; Call Delay routine
                   decfsz  COUNTER, f                  ; decrement receive count register by one
                   goto    ReceiveNext                 ; Receive next bit
                   retlw   0x00                        ; back to operation system
                                                       ;
```


The time is measured by using a software timer. The software timer is started when the start-bit is detected. The start-bit is detected when a transition from high to low occurs. Once the start-bit is detected, the software timer counts until a low to high transition is detected.

THE DELAY ROUTINES

The delay routine for half the bit time and the full bit time are identical in program flow. If the high byte is zero, only the low byte will be decremented. For decrementing, the low byte is stored in a temporary register. When the low byte is zero, the delay routine returns to either the receive or transmit routine.

If the high byte is not zero, the low byte will be decremented n-times, where n is the value stored in the high byte.

OTHER POSSIBLE AUTOBAUD IMPLEMENTATIONS

There a several other methods to implement an autobaud routine. These methods are briefly described below. The implementations are not given within this application note.

Measuring The Bit Length Using A Timer

Instead of using a software counter, a timer can be used. This would require modifications in the autobaud and the receive and transmit routines. The disadvantage of this method is that one timer has to be dedicated to the autobaud routine.

Measuring The Bit Length Of The First Bit For Each Character Transmitted

This method measures the transmission time of the first bit from a transmitted character. The measured value is used to adjust the delay counter for receiving the following bits. The measurement is done for each character received. Variations in the oscillator frequency are compensated for using this method. The disadvantage of this method is that the transmitted characters need a zero to one transition in the first bit. This limits the number of characters which can be transmitted.

SOFTWARE PERFORMANCE

The performance of the autobaud routine is shown.

TABLE 1: SOFTWARE PERFORMANCE

Oscillator Frequency	Min. Baudrate	Max. Baudrate
4 MHz	110 Baud	19200 Baud
10 MHz	110 Baud	38400 Baud
20 MHz	110 Baud	57600 Baud

APPENDIX

```
MPASM 02.20.04 Intermediate  AUTO16B3.ASM   3-17-1999  11:28:13

                    00001 ; ***********************************************************************************************
                    00002 ; * Title           : RS-232 Autobaud routine                                                   *
                    00003 ; * Author          : Thomas Schmidt                                                            *
                    00004 ; *                   Application Engineer for Standard Microcontroller and ASSP Products       *
                    00005 ; * Date            : 04.01.1999                                                                *
                    00006 ; * Revision        : 1.0                                                                       *
                    00007 ; * Last Modified   : 04.01.1999                                                                *
                    00008 ; * Description     : The purpose of this program to detect automatically the Baudrate of a RS-232*
                    00009 ; * transmitter. The detected baudrate is used to adjust a delay routine for a transmit and     *
                    00010 ; * receive routine.                                                                            *
                    00011 ; * This program measures the transmission time of an incoming calibration character. Based on  *
                    00012 ; * the measured time the transmission time for one bit is calculated. This value is used in    *
                    00013 ; * a software delay routine to generate a delay for on bit. The delay routine is called from   *
                    00014 ; * a transmit and receive routine. The user is free to modify the main routine. If the user    *
                    00015 ; * chooses to modify the receive and transmit routine he has to modify as well the software    *
                    00016 ; * adjustment in the autobaud routine.                                                         *
                    00017 ; ***********************************************************************************************
                    00018
                    00019
                    00020               LIST P=16C54B, r=hex
                    00021
                    00022               ; **********************************************************************
                    00023               ; * Include files                                                      *
                    00024               ; **********************************************************************
                    00025               #include "P16C5X.INC"
                    00001         LIST
                    00002 ; P16C5X.INC  Standard Header File, Version 4.00    Microchip Technology, Inc.
                    00313         LIST
                    00026
                    00027               ; **********************************************************************
                    00028               ; * Pin definitions                                                    *
                    00029               ; **********************************************************************
                    00030               #define RX             2              ; receive pin, connected to RA2
                    00031               #define TX             3              ; transmit pin, connected to RA3
                    00032
                    00033
                    00034               ; **********************************************************************

                    00035               ; * Register definitions                                               *
                    00036               ; **********************************************************************
                    00037               cblock  0x08
  00000008          00038                       AUTOBAUD_LOW    ; low byte of bit-time counter
  00000009          00039                       AUTOBAUD_HIGH   ; high byte of bit-time counter
```

```
0000000A            AUTOHALF_LOW    ; low byte of half the bit time
0000000B            AUTOHALF_HIGH   ; high byte for half the bit time
0000000C            AUTOB_STATUS    ; status byte for Autobaud routine
0000000D            TEMP1, TEMP2    ; temporary registers
0000000F            RXTX_REG        ; receive register
00000010            COUNTER         ; receive & Transmit counter register
            endc

            ; ************************************************************************
            ; * Bit definitions in register AUTOB_STATUS                             *
            ; ************************************************************************
            #define SIGNAL_FAST     0               ; signal-to-fast flag in AUTOB_STATUS
                                                    ; byte. This bit indicates that the
                                                    ; incoming signal was too fast
                                                    ; AUTOB_STATUS.SIGNAL_FAST=0  Signal was OK
                                                    ; AUTOB_STATUS.SIGNAL_FAST=1  Signal was too fast
            #define SIGNAL_SLOW     1               ; signal-to-slow flag in AUTOB_STATUS
                                                    ; byte. This bit indicates that the
                                                    ; incoming signal was too slow
                                                    ; AUTOB_STATUS.SIGNAL_SLOW=0  Signal was OK
                                                    ; AUTOB_STATUS.SIGNAL_SLOW=1  Signal was to slow



            ; ************************************************************************
            ; * Other definitions                                                    *
            ; ************************************************************************
            #define BITS            8               ; number of bits to receive


            ; ************************************************************************
            ; * Fuse configuration                                                   *
            ; ************************************************************************
0FFF 0FF9   __CONFIG _CP_OFF&_WDT_OFF&_XT_OSC

            ; ************************************************************************
            ; * Reset vector                                                         *
            ; ************************************************************************
01FF        ORG 0x1FF
01FF 0A00   goto    Begin

            ; ************************************************************************
            ; * Program Start                                                        *
            ; ************************************************************************
0000        ORG 0x00

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

```
                      ; * Initialization                                                    *
                      ; *********************************************************************
0000 0066   Begin     clrf    PORTB            ; set all latches of PORTB to '0'
0001 0040             clrw                     ; reset W-Register
0002 0006             tris    PORTB            ; initialize TRIS register
0003 0065             clrf    PORTA            ; reset latches of PortA
0004 0CF7             movlw   b'11110111'      ; R2=RX, RA3=TX
0005 0005             tris    PORTA            ; initialize TRIS register for PORTA

                      ; *********************************************************************
                      ; * Main routine. The main routine detects first the transmission     *
                      ; * time of the incoming calibration character. After that the        *
                      ; * routine receives and transmits incoming characters.               *
                      ; *********************************************************************
0006 090F             call    Autobaud         ; call Autobaud routine
0007 020C             movf    AUTOB_STATUS, w ; check if an error occurred
0008 0643             btfsc   STATUS, Z        ; is AUTOB_STATUS=0 (means no error occurred)
0009 0A0C             goto    Main             ; goto Main

                      ; An error occurred. The incoming signal was either too fast or too slow.
                      ; The autobaud status register AUTOB_STATUS is displayed on PORTB in
                      ; order to indicated that an error occurred. The receive and transmit
                      ; routine will not be called.
000A 0026             movwf   PORTB            ; display AUTOB_STATUS on PORTB
000B 0A0B   DoForever goto    DoForever        ; an error occurred. This error is displayed on PORTB.
                                               ; Because of this error, the receive and transmit
                                               ; routine will not be called.

                      ; No error occurred. There receive and transmit characters.
000C 0952   Main      call    Transmit         ; transmit received character back to transmitter
000D 0942             call    Receive          ; receive next character
000E 0A0C             goto    Main             ; do forever

                      ; *********************************************************************
                      ; * Autobaud routine                                                  *
                      ; *********************************************************************
000F 0068   Autobaud  clrf    AUTOBAUD_LOW     ; reset register
0010 0069             clrf    AUTOBAUD_HIGH    ; reset register
0011 006A             clrf    AUTOHALF_LOW     ; reset register
0012 006B             clrf    AUTOHALF_HIGH    ; reset register
0013 006C             clrf    AUTOB_STATUS     ; reset autobaud status register
0014 0645   TestStartBit btfsc PORTA, RX       ; check for start-bit
0015 0A14             goto    TestStartBit     ; start-bit not found

0016 0645   TestBitHigh btfsc PORTA, RX        ; test for end of bit stream
0017 0A1D             goto    Calculate        ; end of bit stream, now calculate
                                               ; bit time for one bit
```

```
0018 03E8                   incfsz  AUTOBAUD_LOW, f ; increment Autobaud low register
0019 0A16                   goto    TestBitHigh     ; test for high bit
001A 03E9                   incfsz  AUTOBAUD_HIGH,f ; increment high byte of autobaud register
001B 0A16                   goto    TestBitHigh     ; test for end of bit stream
001C 0A40                   goto    Signal2Slow     ; high byte got an overflow. Transmitted
                                                    ; signal is too slow for clock speed

                            ; Calculation of transmission time for one bit
001D 0C03      Calculate    movlw   0x03            ; initialize count register
001E 0030                   movwf   COUNTER         ; counter for number for rotates = 3
001F 0403      Divide       bcf     STATUS, C       ; clear carry bit
0020 0329                   rrf     AUTOBAUD_HIGH,f ; rotate autobaud high register
0021 0328                   rrf     AUTOBAUD_LOW, f ; rotate autobaud low register
0022 02F0                   decfsz  COUNTER, f      ; decrement counter
0023 0A1F                   goto    Divide          ; divide


                            ; Calculate the transmission time for half the bit time (means
                            ; divide transmission time of one bit by two).
0024 0403      CalcHalfBit  bcf     STATUS, C       ; clear carry bit
0025 0309                   rrf     AUTOBAUD_HIGH,w ; rotate autobaud high register
0026 002B                   movwf   AUTOHALF_HIGH   ; copy result into AUTOHALF_HIGH register
0027 0308                   rrf     AUTOBAUD_LOW, w ; rotate autobaud high register
0028 002A                   movwf   AUTOHALF_LOW    ; copy result into AUTOHALF_LOW register

                            ; Adjust 16-bit counter for receive and transmit routine. This means
                            ; that the overhead of instruction cycles in of the receive/transmit
                            ; routine has to be subtracted from the transmission time of one bit
                            ; and half a bit.
0029 0C03    AdjustLowByte  movlw   0x3             ; 18-19 instruction cycles overhead from
                                                    ; transmit/receive routine. This overhead
                                                    ; must be subtracted from iterations
002A 00A8                   subwf   AUTOBAUD_LOW, f ; adjust low byte from Autobaud counter
002B 0703                   btfss   STATUS, C       ; is result negative? (equal=0 will be checked
                                                    ; at ErrorCheck). C=0 result is negative
002C 0A3E                   goto    Signal2Fast     ; signal is too fast for receive and transmit routine
002D 0C02                   movlw   0x02            ; subtract 2 from low byte of half the bit time
002E 00AA                   subwf   AUTOHALF_LOW, f ; subtract from low byte of half the bit time
002F 0703                   btfss   STATUS, C       ; is result negative? (equal=0 will be checked
                                                    ; at ErrorCheck). C=0 result is negative
0030 0A3E                   goto    Signal2Fast     ; signal is too fast

                            ; check if AUTOBAUD_HIGH and AUTOBAUD_LOW are zero. This
                            ; means the transmission time for one byte is too high
0031 0229      ErrorCheck   movf    AUTOBAUD_HIGH,f ; copy high byte of autobaud counter register onto itself
0032 0743                   btfss   STATUS, Z       ; is zero-flag set?
0033 0A38                   goto    ErrorCheckHalf  ; no, therefore check next byte
```

```
0034 0228                        movf    AUTOBAUD_LOW, f ; copy low byte of autobaud register onto itself
0035 0743                        btfss   STATUS, Z       ; is zero-flag set?
0036 0A38                        goto    ErrorCheckHalf  ; no, low byte is not zero therefore check next byte
0037 0A3E                        goto    Signal2Fast     ; yes, signal is too fast. Therefore set flag
0038 022B        ErrorCheckHalf  movf    AUTOHALF_HIGH,f ; copy high byte of autobaud counter onto itself
0039 0743                        btfss   STATUS, Z       ; is zero-flag set?
003A 0A41                        goto    EndAutoBaud     ; finish autobaud routine
003B 022A                        movf    AUTOHALF_LOW, f ; check low byte
003C 0743                        btfss   STATUS, Z       ; is zero-flag set?
003D 0A41                        goto    EndAutoBaud     ; no, therefore finish autobaud routine
                                                         ; yes, High and low byte of AUTOHALF register are zero
                                                         ; there the incoming signal was too fast to generate a delay
                                                         ; Therefore set SIGNAL_FAST flag

                                 ; Error: delay for half the bit time is zero, therefore a
                                 ; delay cannot be generated with the delay routines. The incoming signal
                                 ; was too fast for clock speed.
003E 050C        Signal2Fast     bsf     AUTOB_STATUS, SIGNAL_FAST       ; set error flag
003F 0800                        retlw   0x00                            ; return to main routine

0040 052C        Signal2Slow     bsf     AUTOB_STATUS, SIGNAL_SLOW       ; set error flag


0041 0800        EndAutoBaud     retlw   0x00                            ; return to main routine



                                 ; ***********************************************************************
                                 ; * Receive Routine                                                     *
                                 ; ***********************************************************************
0042 006F        Receive         clrf    RXTX_REG        ; clear receive register
0043 0C08                        movlw   BITS            ; number of bits to receive
0044 0030                        movwf   COUNTER         ; load number of bits into counter register
0045 0645        ReceiveStartBit btfsc   PORTA, RX       ; test for start bit
0046 0A45                        goto    ReceiveStartBit ; start-bit not found
0047 0972                        call    DelayHalfBit    ; wait until middle of start-bit
0048 0961                        call    DelayFullBit    ; ignore start-bit and sample first
                                                         ; data bit in the middle of the bit
0049 0645        ReceiveNext     btfsc   PORTA, RX       ; is RX zero or a one?
004A 0503                        bsf     STATUS,C        ; bit is a one => set carry bit
004B 0745                        btfss   PORTA, RX       ; is RX one or a zero?
004C 0403                        bcf     STATUS,C        ; RX is zero => clear carry bit
004D 032F                        rrf     RXTX_REG, f     ; rotate value into receive register
004E 0961                        call    DelayFullBit    ; call Delay routine
004F 02F0                        decfsz  COUNTER, f      ; decrement receive count register by one
0050 0A49                        goto    ReceiveNext     ; receive next bit
0051 0800                        retlw   0x00            ; return to main routine
```

```
                            ; *********************************************************************
                            ; * Transmit routine                                                  *
                            ; *********************************************************************
0052 0C08 Transmit          movlw   BITS             ; number of bit's to transmit
0053 0030                   movwf   COUNTER          ; initialize count register
0054 0465                   bcf     PORTA, TX        ; generate start-bit
0055 0961                   call    DelayFullBit     ; generate Delay for one bit-time
0056 032F TransmitNext      rrf     RXTX_REG, f      ; rotate receive register
0057 0603                   btfsc   STATUS, C        ; test bit to be transmitted
0058 0565                   bsf     PORTA, TX        ; transmit a one
0059 0703                   btfss   STATUS, C        ; check carry bit if set
005A 0465                   bcf     PORTA, TX        ; transmit a zero
005B 0961                   call    DelayFullBit     ; call Delay routine
005C 02F0                   decfsz  COUNTER, f       ; decrement counter register
005D 0A56                   goto    TransmitNext     ; transmit next bit
005E 0565                   bsf     PORTA, TX        ; generate Stop bit
005F 0961                   call    DelayFullBit     ; delay for Stop bit
0060 0800                   retlw   0x00             ; return to main routine

                            ; *********************************************************************
                            ; * Delay routine 16-bit counter (delay for full bit time)            *
                            ; *********************************************************************
0061 0209 DelayFullBit      movf    AUTOBAUD_HIGH,w ; copy content of Autobaud high register into
0062 0743                   btfss   STATUS, Z        ; is high byte = 0?
0063 0A65                   goto    LoadHighByte     ; no, high byte is not zero
0064 0A6B                   goto    DecLowByteOnly   ; decrement only low byte

0065 002E LoadHighByte      movwf   TEMP2            ; load TEMP2 with content of AUTOBAUD_HIGH
0066 006D                   clrf    TEMP1            ; reset TEMP1 register
0067 02ED DecLowByte1       decfsz  TEMP1, f         ; decrement low byte
0068 0A70                   goto    DecLowByte11     ; do until result is zero
0069 02EE                   decfsz  TEMP2, f         ; decrement low byte
006A 0A67                   goto    DecLowByte1      ; decrement low byte again

006B 0208 DecLowByteOnly    movf    AUTOBAUD_LOW, w ; copy low byte from autobaud register
006C 002D                   movwf   TEMP1            ; into TEMP1
006D 02ED DecLowByte2       decfsz  TEMP1, f         ; decrement low byte until zero
006E 0A71                   goto    DecLowByte22     ; extra two cycle delay
006F 0800                   retlw   0x00             ; return from subroutine
0070 0A67 DecLowByte11      goto    DecLowByte1      ; additional two cycle delay
0071 0A6D DecLowByte22      goto    DecLowByte2      ; additional two cycle delay


                            ; *********************************************************************
                            ; * Delay routine 16-bit counter (delay for half bit time)            *
                            ; *********************************************************************
```

```
0072 020B           00274 DelayHalfBit    movf    AUTOHALF_HIGH,w ; copy content of Autobaud high register into
0073 0743           00275                 btfss   STATUS, Z       ; is high byte = 0?
0074 0A76           00276                 goto    LoadHighByteH   ; no, high byte is not zero
0075 0A7C           00277                 goto    DecLowByteOnlyH ; decrement only low byte
                    00278
0076 002E           00279 LoadHighByteH   movwf   TEMP2           ; load TEMP2 with content of AUTOHALF_HIGH
0077 006D           00280                 clrf    TEMP1           ; reset TEMP1 register
0078 02ED           00281 DecLowByteH1    decfsz  TEMP1, f        ; decrement low byte
0079 0A81           00282                 goto    DecLowByteH11   ; do until result is zero
007A 02EE           00283                 decfsz  TEMP2, f        ; decrement low byte
007B 0A78           00284                 goto    DecLowByteH1    ; decrement low byte again
                    00285
007C 020A           00286 DecLowByteOnlyH movf    AUTOHALF_LOW, w ; copy low byte from autobaud register
007D 002D           00287                 movwf   TEMP1           ; into TEMP1
007E 02ED           00288 DecLowByteH2    decfsz  TEMP1, f        ; decrement low byte until zero
007F 0A82           00289                 goto    DecLowByteH22   ; extra two cycle delay
0080 0800           00290                 retlw   0x00            ; return from subroutine
0081 0A78           00291 DecLowByteH11   goto    DecLowByteH1    ; additional two cycle delay
0082 0A7E           00292 DecLowByteH22   goto    DecLowByteH2    ; additional two cycle delay
                    00293
                    00294
                    00295                 END

Program Memory Words Used:   132
Program Memory Words Free:   380
```

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

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

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

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

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

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

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02