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.

ASYNCHEONOUS 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.

EXAMPLE 1: BAUDRATE CALCULATION

\[ t_{\text{one-bit}} = \frac{1}{9600\text{Baud}} = 104\mu s \]

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 41H (ASCII).

EXAMPLE 2: ASYNCHRONOUS TRANSMISSION OF CHARACTER ‘A’

Note: Character ‘A’ is equivalent to 41H.

41H=01000001b

Autobaud and Asynchronous Serial Communication

In some systems, the transmission is not fixed to a baud rate. In this case, the receiver 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 too 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 too 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

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.
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**

<table>
<thead>
<tr>
<th>Autobaud</th>
<th>clrf AUTOBAUD_LOW ; reset register</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestBitHigh</td>
<td>btfsc PORTA, RX; Test for end of bit stream</td>
</tr>
<tr>
<td></td>
<td>goto Calculate; End of bit stream, now calculate</td>
</tr>
<tr>
<td></td>
<td>; bit time for one bit</td>
</tr>
<tr>
<td></td>
<td>incfsz AUTOBAUD_LOW, f; increment Autobaud low register</td>
</tr>
<tr>
<td></td>
<td>goto TestBitHigh; test for high bit</td>
</tr>
<tr>
<td></td>
<td>incfsz AUTOBAUD_HIGH, f; increment high byte of autobaud register</td>
</tr>
<tr>
<td></td>
<td>goto TestBitHigh; test for end of bit stream</td>
</tr>
<tr>
<td></td>
<td>goto Signal2Slow; High byte got an overflow. Transmitted</td>
</tr>
<tr>
<td></td>
<td>; signal is too slow for clock speed of the uc</td>
</tr>
</tbody>
</table>

**FIGURE 7: CALCULATION OF TRANSMISSION TIME FOR ONE BIT**

<table>
<thead>
<tr>
<th>Autobaud</th>
<th>clrf AUTOBAUD_LOW ; reset register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate</td>
<td>movlw 0x03; Initialize count register</td>
</tr>
<tr>
<td></td>
<td>movwf COUNTER; Counter for number for rotates = 3</td>
</tr>
<tr>
<td>Divide</td>
<td>bcf STATUS, C; clear carry bit</td>
</tr>
<tr>
<td></td>
<td>rrf AUTOBAUD_HIGH, f; rotate autobaud high register</td>
</tr>
<tr>
<td></td>
<td>rrf AUTOBAUD_LOW, f; rotate autobaud low register</td>
</tr>
<tr>
<td></td>
<td>decfsz COUNTER, f; decrement counter</td>
</tr>
<tr>
<td></td>
<td>goto Divide; divide</td>
</tr>
</tbody>
</table>
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
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 ; reset register
    clrf AUTOBAUD_LOW ; check if AUTOBAUD_HIGH and AUTOBAUD_LOW are zero.
    ; This means the transmission time for one byte is too high

ErrorCheck ; copy high byte of autobaud counter register into w-register
    movf AUTOBAUD_HIGH, w
    xorwf AUTOBAUD_LOW, w
    btfss STATUS, Z
    goto ErrorCheckHalf ; Signal is too fast for routine
    ; Signal to fast for routine
    goto Signal2Fast

ErrorCheckHalf ; copy high byte of autobaud counter register into w-register
    movf AUTOHALF_HIGH, w
    xorwf AUTOHALF_LOW, w
    btfss STATUS, Z
    ; 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.
    goto EndAutoBaud

Signal2Fast ; set error flag
    bsf AUTOB_STATUS, SIGNAL_FAST
    retlw 0x00 ; return to operating system

Signal2Slow ; set error flag
    bsf AUTOB_STATUS, SIGNAL_SLOW
    retlw 0x00 ; Return to operating system

EndAutoBaud

THE TRANSMIT ROUTINE

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

FIGURE 11: SOURCE CODE OF THE TRANSMIT ROUTINE

Transmit ; Transmit routine
    movlw BITS ; Number of bit’s to transmit
    movwf COUNTER ; Initialize count register
    bcf PORTA, TX ; Generate start-bit
    ; Software overhead = 10 instruction cycles (including call
to DelayFullBit routine, return from delay routine not included)

TransmitNext ; Generate Delay for one bit-time
    rrf RXTX_REG, f
    btfsc STATUS, C
    bsf PORTA, TX ; Transmit one
    btfss STATUS, C
    ; Check carry bit if set
    bcf PORTA, TX ; Transmit a zero
    call DelayFullBit ; call Delay routine
    ; Delay for Stop bit
    decfsz COUNTER, f
    ; Decrement count register
    goto TransmitNext ; Transmit next bit
    bcf PORTA, TX ; Generate Stop bit
    call DelayFullBit ; Return to operating system
    ; Transmit routine
    retlw 0x00 ; Transmits LSB first

    ; Software overhead = 10 instruction cycles (including call
to DelayFullBit routine, return from delay routine not included)
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

```assembly
Receive clrf RXTX_REG ; Clear receive register
movlw BITS ; Number of bits to receive
movwf COUNTER ; Load number of bits into counter register
ReceiveStartBit btfs 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 btfs PORTA, RX ; Is bit a zero or a one
bsf STATUS,C ; bit is a one => set carry bit
btfs 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 are 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

<table>
<thead>
<tr>
<th>Oscillator Frequency</th>
<th>Min. Baudrate</th>
<th>Max. Baudrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 MHz</td>
<td>110 Baud</td>
<td>19200 Baud</td>
</tr>
<tr>
<td>10 MHz</td>
<td>110 Baud</td>
<td>38400 Baud</td>
</tr>
<tr>
<td>20 MHz</td>
<td>110 Baud</td>
<td>57600 Baud</td>
</tr>
</tbody>
</table>
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 LIST P=16C54B, r=hex
00019
00020 ; ***********************************************************************
00021 ; * Include files                                                       *
00022 #include "P16C5X.INC"
00023 ; ***********************************************************************
00024 ; * Pin definitions                                                     *
00025 #define RX              2               ; receive pin, connected to RA2
00026 #define TX              3               ; transmit pin, connected to RA3
00027 ; ***********************************************************************
00028 ; * Register definitions                                                *
00029 cblock 0x08
00030 AUTOBAUD_LOW    ; low byte of bit-time counter
00031 AUTOBAUD_HIGH   ; high byte of bit-time counter
00032
00033
00034
00035
; ***********************************************************************
; * 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
#define SIGNAL_SLOW 1 ; signal-to-slow flag in AUTOB_STATUS
; byte. This bit indicates that the
; incoming signal was too slow

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

; ***********************************************************************
; * Fuse configuration
; ***********************************************************************
__CONFIG_CP_OFF&_WDT_OFF&_XT_OSC

; ***********************************************************************
; * Reset vector
; ***********************************************************************
ORG 0x1FF
goto Begin

; ***********************************************************************
; * Program Start
; ***********************************************************************
ORG 0x00
0000 0066 00087 ; * Initialization
0000 0066 00088 ; *******************************************************
0001 0040 0000 Begin
0002 0006 00090 clrf PORTB ; set all latches of PORTB to '0'
0003 0065 00091 clrw ; reset W-Register
0004 00F7 00092 tris PORTB ; initialize TRIS register
0005 0093 00094 clrf PORTA ; reset latches of PortA
0005 009D 00095 movlw b'11110111' ; R2=RX, RA3=TX
0006 009E 00096 tris PORTA ; initialize TRIS register for PORTA
0007 009F 00097 ; *******************************************************
0008 0050 00098 ; * Main routine. The main routine detects first the transmission *
0009 0050 00099 ; * time of the incoming calibration character. After that the *
000A 0050 00100 ; * routine receives and transmits incoming characters. *
000B 0050 00101 ; *******************************************************
000C 0F90 00102 call Autobaud ; call Autobaud routine
000D 020C 00103 movf AUTOB_STATUS, w ; check if an error occurred
000E 0643 00104 btfsc STATUS, Z ; is AUTOB_STATUS=0 (means no error occurred)
000F 0A0C 00105 goto Main ; goto Main
000F 0068 00106 ; An error occurred. The incoming signal was either too fast or too slow.
0010 0069 00107 ; The autobaud status register AUTOB_STATUS is displayed on PORTB in
0011 006A 00108 ; order to indicate that an error occurred. The receive and transmit
0012 006B 00109 ; routine will not be called.
0013 006C 00110 movwf PORTB ; display AUTOB_STATUS on PORTB
0014 006D 00111 DoForever ; because of this error, the receive and transmit
0015 006E 00112 goto DoForever ; routine will not be called.
0016 006F 00113 ; No error occurred. There receive and transmit characters.
0017 0952 00114 Main call Transmit ; transmit received character back to transmitter
0018 0942 00115 call Receive ; receive next character
0019 0A0C 00116 goto Main ; do forever
001A 0068 00117 ; *******************************************************
001B 0069 00118 ; * Autobaud routine
001C 006A 00119 ; *******************************************************
001D 006B 00120 Autobaud clrf AUTOBAUD_LOW ; reset register
001E 006C 00121 clrf AUTOBAUD_HIGH ; reset register
001F 006D 00122 clrf AUTOHALF_LOW ; reset register
0020 006E 00123 clrf AUTOHALF_HIGH ; reset register
0021 006F 00124 clrf AUTOB_STATUS ; reset autobaud status register
0022 0070 00125 TestStartBit btfscc PORTA, RX ; check for start-bit
0023 0071 00126 goto TestStartBit ; start-bit not found
0024 0072 00127 TestBitHigh btfscc PORTA, RX ; test for end of bit stream
0025 0073 00128 goto Calculate ; end of bit stream, now calculate
0026 0074 00129 ; bit time for one bit
0018 03E8  00134  incfsz AUTOBAUD_LOW, f ; increment Autobaud low register
0019 0A16  00135  goto TestBitHigh ; test for high bit
001A 03E9  00136  incfsz AUTOBAUD_HIGH,f ; increment high byte of autobaud register
001B 0A16  00137  goto TestBitHigh ; test for end of bit stream
001C 0A40  00138  goto Signal2Slow ; high byte got an overflow. Transmitted
0019 03E8  00139  ; signal is too slow for clock speed
001A 0A16  00140
001A 0A40  00141 ; Calculation of transmission time for one bit
001C 0A40  00142 Calculate movlw 0x03            ; initialize count register
001D 0030  00143 movwf COUNTER         ; counter for number for rotates = 3
001E 0403  00144 Divide bcf STATUS, C       ; clear carry bit
001F 0329  00145 rrf AUTOBAUD_HIGH,f ; rotate autobaud high register
0020 0328  00146 rrf AUTOBAUD_LOW, f ; rotate autobaud low register
0021 02F0  00147 decfsz COUNTER, f      ; decrement counter
0022 0A1F  00148 goto Divide          ; divide
0023 00A8  00149
0023 0A1F  00150 ; Calculate the transmission time for half the bit time (means
0024 0403  00151 ; divide transmission time of one bit by two).
0025 0030  00152
0025 0403  00153 CalcHalfBit bcf STATUS, C       ; clear carry bit
0026 0329  00154 rrf AUTOBAUD_HIGH,w ; rotate autobaud high register
0027 0328  00155 movwf AUTOHALF_HIGH ; copy result into AUTOHALF_HIGH register
0028 0308  00156 rrf AUTOBAUD_LOW, w ; rotate autobaud high register
0029 002A  00157 movwf AUTOHALF_LOW ; copy result into AUTOHALF_LOW register
002A 03E9  00158
002A 0A40  00159 ; Adjust 16-bit counter for receive and transmit routine. This means
002B 0A40  00160 ; that the overhead of instruction cycles in of the receive/transmit
002C 0A40  00161 ; routine has to be subtracted from the transmission time of one bit
002D 0A40  00162 ; and half a bit.
002E 0A40  00163 AdjustLowByte movlw 0x03            ; 18-19 instruction cycles overhead from
002F 0030  00164 ; transmit/receive. This overhead
0030 00165  ; must be subtracted from iterations
0031 00A8  00166 subwf AUTOBAUD_LOW, f ; adjust low byte from Autobaud counter
0032 0703  00167 btfs STATUS, C       ; is result negative? (equal=0 will be checked
0033 00168  ; at ErrorCheck). C=0 result is negative
0034 0A19  00169 goto Signal2Fast ; signal is too fast for receive and transmit routine
0035 0170  00170 movlw 0x02        ; subtract 2 from low byte of half the bit time
0036 00AA  00171 subwf AUTOHALF_LOW, f ; subtract from low byte of half the bit time
0037 0703  00172 btfs STATUS, C       ; is result negative? (equal=0 will be checked
0038 00173  ; at ErrorCheck). C=0 result is negative
0039 0A3E  00174 goto Signal2Fast ; signal is too fast
003A 0002  00175
003A 0040  00176 ; check if AUTOBAUD_HIGH and AUTOBAUD_LOW are zero. This
003B 0038  00177 ; means the transmission time for one byte is too high
003C 002A  00178 ErrorCheck movf AUTOBAUD_HIGH,f ; copy high byte of autobaud counter register onto itself
003D 0743  00179 btfs STATUS, Z ; is zero-flag set?
003E 0A3E  00180 goto ErrorCheckHalf ; no, therefore check next byte
0034 0228 00181  movf AUTOBAUD_LOW, f ; copy low byte of autobaud register onto itself
0035 0743 00182  btfs S STATUS, Z ; is zero-flag set?
0036 0A38 00183  goto ErrorCheckHalf ; no, low byte is not zero therefore check next byte
0037 0A3E 00184  goto Signal2Fast ; yes, signal is too fast. Therefore set flag
0038 022B 00185 ErrorCheckHalf  movf AUTOHALF_HIGH, f ; copy high byte of autobaud counter onto itself
0039 0743 00186  btfs S STATUS, Z ; is zero-flag set? 
003A 0A41 00187  goto EndAutoBaud ; finish autobaud routine 
003B 022A 00188  movf AUTOHALF_LOW, f ; check low byte
003C 0743 00189  btfs S STATUS, Z ; is zero-flag set? 
003D 0A41 00190  goto EndAutoBaud ; no, therefore finish autobaud routine
003E 022B 00191  movf AUTOHALF_LOW, f ; copy high byte of autobaud counter onto itself
003F 0743 00192  btfs S STATUS, Z ; is zero-flag set? 
0040 0A41 00193  goto EndAutoBaud ; yes, High and low byte of AUTOHALF register are zero 
0041 022A 00194  movf AUTOHALF_HIGH, f ; copy high byte of autobaud counter onto itself
0042 0743 00195  btfss STATUS, Z ; is zero-flag set? 
0043 0A41 00196  goto ErrorCheckHalf ; no, low byte is not zero therefore check next byte
0044 006F 00197  ; Error: delay for half the bit time is zero, therefore a 
0045 0C08 00198  Signal2Fast bsf AUTOB_STATUS, SIGNAL_FAST ; set error flag 
0046 0030 00199  movwf COUNTER ; load number of bits into counter register
0047 0645 00200  ReceiveStartBit btfsc PORTA, RX ; test for start bit
0048 0972 00201  call DelayHalfBit ; wait until middle of the bit
0049 0961 00202  call DelayFullBit ; ignore start-bit and sample first data bit in the middle of the bit
004A 0645 00203  ReceiveNext  btfsc PORTA, RX ; is RX zero or a one?
004B 006F 00204  Receive  clrf RXTX_REG ; clear receive register
004C 0C08 00205  movlw BITS ; number of bits to receive
004D 0030 00206  movwf COUNTER ; load number of bits into counter register
004E 0645 00211  ReceiveStartBit btfsc PORTA, RX ; test for start bit
004F 0030 00212  movwf COUNTER ; load number of bits into counter register
0050 0645 00213  ReceiveStartBit btfsc PORTA, RX ; test for start bit
0051 0645 00214  goto ReceiveStartBit ; start-bit not found
0052 0972 00215  call DelayHalfBit ; wait until middle of start-bit
0053 0961 00216  call DelayFullBit ; ignore start-bit and sample first
0054 0961 00217  ; data bit in the middle of the bit
0055 0645 00218  ReceiveNext  btfsc PORTA, RX ; is RX zero or a one?
0056 006F 00219  Receive  clrf RXTX_REG ; clear receive register
0057 0743 00220  btfs S STATUS, C ; bit is a one => set carry bit
0058 0743 00221  bcf STATUS, C ; RX is zero => clear carry bit
0059 0743 00222  rrf RXTX_REG, f ; rotate value into receive register
005A 0743 00223  btfss STATUS, C ; RX is zero => clear carry bit
005B 0743 00224  rrf RXTX_REG, f ; rotate value into receive register
005C 0743 00225  bsf AUTOB_STATUS, SIGNAL_FAST ; set error flag
005D 0800 00226  retlw 0x00 ; return to main routine

;***********************************************************************
;* Receive Routine                                                     *
;***********************************************************************
0042 006F 00200  Receive  clrf RXTX_REG ; clear receive register
0043 0C08 00201  movlw BITS ; number of bits to receive
0044 0030 00202  movwf COUNTER ; load number of bits into counter register
0045 0645 00207  ReceiveStartBit btfsc PORTA, RX ; test for start bit
0046 0A45 00208  goto ReceiveStartBit ; start-bit not found
0047 0972 00209  call DelayHalfBit ; wait until middle of start-bit
0048 0961 00210  call DelayFullBit ; ignore start-bit and sample first
0049 0961 00211  ; data bit in the middle of the bit
004A 0645 00212  ReceiveNext  btfsc PORTA, RX ; is RX zero or a one?
004B 006F 00213  Receive  clrf RXTX_REG ; clear receive register
; * Transmit routine
; ***********************************************************************
00227
00228
00229
00230
00231 Transmit
00232 movlw     BITS
00233                 ; number of bit’s to transmit
00234                 ; ***********************************************************************
00235                 ; * Delay routine 16-bit counter (delay for full bit time)              *
00236                 ; ***********************************************************************
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250 DelayFullBit
00251 movf     AUTOBAUD_HIGH, w
00252                 ; copy content of Autobaud high register into
00253                 ; ***********************************************************************
00254
00255
00256
00257
00258
00259
00260
00261
00262 DecLowByteOnly
00263 movf     AUTOBAUD_LOW, w
00264                 ; copy low byte from autobaud register
00265                 ; ***********************************************************************
00266
00267
00268
00269
00270
00271
00272
00273
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 DecLowByteH2 decfsz TEMP1, f ; decrement low byte
007B 0A78 00284                goto DecLowByteH22 ; 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

Program Memory Words Used:  132
Program Memory Words Free:  380
Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable".
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KeelLoq® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.
<table>
<thead>
<tr>
<th>Region</th>
<th>Country</th>
<th>Address</th>
<th>Phone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMERICAS</td>
<td>Corporate Office</td>
<td>2335 West Chandler Blvd. Chandler, AZ 85224-6199</td>
<td>480-792-7200</td>
<td>480-792-7277</td>
</tr>
<tr>
<td></td>
<td>Technical Support</td>
<td>480-792-7267</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web Address</td>
<td><a href="http://www.microchip.com">http://www.microchip.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rock Mountain</td>
<td>2335 West Chandler Blvd.</td>
<td>Chandler, AZ 85224-6199</td>
<td>480-792-7200</td>
<td>480-792-7456</td>
</tr>
<tr>
<td></td>
<td>Atlanta</td>
<td>500 Sugar Mill Road, Suite 200 A</td>
<td>770-640-0034</td>
<td>770-640-0307</td>
</tr>
<tr>
<td></td>
<td>West Hartford</td>
<td>13004 Route 91, Route 91, Route 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Boston</td>
<td>2 Lan Drive, Suite 120</td>
<td>781-767-7000</td>
<td>781-767-7000</td>
</tr>
<tr>
<td></td>
<td>Chicago</td>
<td>333 Pierce Road, Suite 180</td>
<td>312-999-8999</td>
<td>312-999-8999</td>
</tr>
<tr>
<td></td>
<td>Dallas</td>
<td>4570 Westgrove Drive, Suite 160</td>
<td>972-818-7423</td>
<td>972-818-2924</td>
</tr>
<tr>
<td></td>
<td>Detroit</td>
<td>2767 S. Albright Road</td>
<td>404-848-7969</td>
<td>404-848-7969</td>
</tr>
<tr>
<td></td>
<td>Kokomo</td>
<td>2767 S. Albright Road</td>
<td>765-864-8360</td>
<td>765-864-8387</td>
</tr>
<tr>
<td></td>
<td>Los Angeles</td>
<td>18201 Von Karman, Suite 109</td>
<td>949-263-1888</td>
<td>949-263-1338</td>
</tr>
<tr>
<td></td>
<td>San Jose</td>
<td>2107 North First Street, Suite 59</td>
<td>408-436-7955</td>
<td>408-436-7955</td>
</tr>
<tr>
<td></td>
<td>Toronto</td>
<td>6285 Northland Drive, Suite 108</td>
<td>905-673-0699</td>
<td>905-673-6509</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASIA/PACIFIC</td>
<td>Australia</td>
<td>Microchip Technology Australia Pty Ltd</td>
<td>61-2-9868-6733</td>
<td>61-2-9868-6755</td>
</tr>
<tr>
<td></td>
<td>China - Beijing</td>
<td>Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office</td>
<td>Unit 915, Beijing, 100027, No. China</td>
<td>86-10-85282100 Fax: 86-10-85282104</td>
</tr>
<tr>
<td></td>
<td>China - Chengdu</td>
<td>Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office</td>
<td>Rm. 2401, 24th Floor, Ming Xing Financial Tower No. 88 TIDU Street Chengdu 610016, China</td>
<td>86-28-6766200 Fax: 86-28-6766599</td>
</tr>
<tr>
<td></td>
<td>China - Fuzhou</td>
<td>Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office</td>
<td>No. 71 Wus Road Fuzhou 350001, China</td>
<td>86-591-7530506 Fax: 86-591-7530521</td>
</tr>
<tr>
<td></td>
<td>China - Shanghai</td>
<td>Microchip Technology Consulting (Shanghai) Co., Ltd., Shanghai Liaison Office</td>
<td>Room 701, Bldg. B Far East International Plaza No. 317 Xian Xia Road Shanghai, 200005</td>
<td>86-21-6275-7500 Fax: 86-21-6275-5060</td>
</tr>
<tr>
<td></td>
<td>China - Shenzhen</td>
<td>Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office</td>
<td>Rm. 1315, 13/F, Shenzhen Kerry Centre, Remennuin Lu Shenzhen 518001, China</td>
<td>86-755-2350361 Fax: 86-755-2366086</td>
</tr>
<tr>
<td></td>
<td>Hong Kong</td>
<td>Microchip Technology Hong Kong 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</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>India</td>
<td>Microchip Technology Inc.</td>
<td>91-80-2290061</td>
<td>91-80-2290062</td>
</tr>
<tr>
<td></td>
<td>Japan</td>
<td>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</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Korea</td>
<td>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</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Singapore</td>
<td>Microchip Technology Singapore Pte Ltd. 200 Middle Road #07-02 Prime Centre Singapore, 188980 Tel: 65-334-8870 Fax: 65-334-8850</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Taiwan</td>
<td>Microchip Technology Taiwan 11F-3, No. 207 Tung Hua North Road Taipei, 105, Taiwan Tel: 886-2-2717-7175 Fax: 886-2-2545-0139</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EUROPE</td>
<td>Denmark</td>
<td>Microchip Technology Nordic ApS Regus Business Centre Lautrup høj 1-3 Ballerup DK-2750 Denmark Tel: 45 4420 9895 Fax: 45 4420 9910</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>France</td>
<td>Microchip Technology SARL Parc d’Activite du Moulin de Massy 43 Rue du Saule Trapu Batiment A - 1er Etage 91300 Massy, France Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Germany</td>
<td>Microchip Technology GmbH Gustav-Heinemann Ring 125 D-81739 Munich, Germany Tel: 49-89-627-144 0 Fax: 49-89-627-144-44</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Italy</td>
<td>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</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>United Kingdom</td>
<td>Arizona Microchip Technology Ltd. 505 Eskdale RoadWinners Triangle Wokingham Berkshire, England RG41 5TU Tel: 44 118 921 5869 Fax: 44-118 921-5820</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>