INTRODUCTION

The MCP2122 has up to four signals that can be interfaced to the embedded systems Host Controller. These signals are the transmitted and received data (TX and RX), the 16XCLK signal and the RESET signal.

The MCP2122 is pinout compatible with the Agilent® HSDL-7000 Encoder/Decoder.

The 16x clock is used for the baud clock timing (transmit and receive). There are 16 16x clocks (16XCLK) for each bit time. In systems that have already been designed with the HSDL-7000, the 16XCLK signal is already present. For systems that are to be designed, the generation of a 16x clock signal will need to be done. For embedded systems using a PICmicro® microcontroller unit (MCU) as the Host Controller, the Capture/Compare/PWM (CCP) and Timer2 modules can be used to generate the 16XCLK signal. This is accomplished using the CCP module in PWM mode.

This application note will discuss methods with which to interface the MCP2122 to a Host Controller and how to use the PICmicro MCU's CCP and Timer2 modules to generate the 16XCLK signal.
HOST UART INTERFACE

The Host UART interface has two signals, the Transmit signal (TX) and the Receive signal (RX). These signals are connected to the Host Controller’s UART, which can either be a hardware module or implemented via firmware.

IR INTERFACE

The IR interface has two signals, the Transmit signal (TXIR) and the Receive signal (RXIR). These signals are connected to the optical transceiver circuitry, which can either be an integrated device or implemented with discrete components.

16XCLK

For the MCP2122 state machine to operate, the device must be clocked. The frequency of the clock on the 16XCLK pin must be 16 times the desired baud rate (see Table 1). Figure 2 shows the relationship of the 16XCLK signal to either the UART signals (TX and RX signals).

The source of this clock could be from any number of sources, including:

• Board System Clock
• Host controller-generated clock
  - Firmware-generated clock on I/O pin
  - Hardware-generated clock using time-based module (such as the PICmicro MCU CCP module)

The clock source is one of the differences between the MCP2122 and the MCP2120. The MCP2120 requires a crystal to create the device clock. The MCP2120’s internal 16XCLK signal (device baud rate) is then specified by the state on the BAUD2:BAUD0 pins. The MCP2120’s BAUD2:BAU0 pins can be configured so that the baud rate is determined by the state of these pins (Hardware mode) or controlled by the Host Controller (Software mode).

Board System Clock

The Board System Clock is the easiest since it is already available. All that needs to be done is to connect the clock to the 16XCLK pin. This means that the baud rate will be 1/16th of the frequency of this clock.

<table>
<thead>
<tr>
<th>UART/IR Baud Rate</th>
<th>16XCLK</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>9,600</td>
<td>153,600</td>
<td></td>
</tr>
<tr>
<td>19,200</td>
<td>307,200</td>
<td></td>
</tr>
<tr>
<td>38,400</td>
<td>614,400</td>
<td></td>
</tr>
<tr>
<td>57,600</td>
<td>921,600</td>
<td></td>
</tr>
<tr>
<td>115,200</td>
<td>1,843,200</td>
<td>Maximum device baud rate</td>
</tr>
</tbody>
</table>

FIGURE 2: 16XCLK TO TX OR RX SIGNAL
Host Controller Firmware-Generated Clock

The Host Controller’s firmware can implement a 16XCLK signal. However, this may add complexity of timing-related issues for other parts of the application firmware. As the desired baud rate increases (relative to the system clock), the number of “free” instruction cycles performing other firmware operations decreases.

Every 16XCLK clock (pulse) has a rising edge and a falling edge that must be generated. With a PICmicro MCU, the rising edge can be generated with the BSF instruction and the falling edge with the BCF instruction. So two instructions are required in each 16XCLK clock (pulse). Table 3 shows the available PICmicro MCU instruction cycles (at a given device frequency) for the desired Host UART/IR interface baud rate. This probably means that any application that was planning to generate the 16XCLK in firmware would not want to exceed an IR baud rate of 19200. In most cases, 9600 would probably be desired to ensure sufficient CPU bandwidth to accomplish other tasks.

Host Controller Hardware-Generated Clock

Using hardware features of the Host Controller can eliminate the processing overhead of a firmware implementation, as well as allow much faster baud rates to be used.

A typical PICmicro device system implementation would consist of using the CCP module to generate the clock, with the CCP module being configured in PWM mode. The PWM frequency would be the required 16XCLK frequency (for the desired baud rate). The PWM duty cycle should be configured for about 50%. After initial configuration of the module (after a reset of the PICmicro device), no additional software overhead is required. This clock will be synchronized to the operation of the PICmicro device UART.

Note: There are other techniques that could be used with PICmicro devices to generate the 16XCLK signal. These techniques include:

1. Use the hardware USART to generate the 16XCLK clock output and implement the TX and RX functions in firmware. This is done by using the USART in Synchronous Master mode with the Continuous Receive enabled (CREN bit is set). The selected clock of the Baud rate will be output on the TX/CLK pin of the PICmicro MCU.
2. With devices that have an accurate 8 MHz internal RC clock, select the option to output that clock and use an external circuit to create the desired 16XCLK frequency.

RESET

The RESET pin is used to put the MCP2122 into a known state. Forcing the RESET pin low will immediately force the output pins to their default output state, as shown in Table 2.

<table>
<thead>
<tr>
<th>Input Pin</th>
<th>Output Pin State</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>State RX TXIR</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>L H L</td>
<td>Device in reset mode</td>
</tr>
</tbody>
</table>

In the section entitled “Using the PICmicro® MCU’s CCP Module to Generate the 16XCLK” discusses the details of using the PICmicro CCP module.

### Table 3: PICmicro® MCU Frequency/Instruction Cycles vs. Host UART/IR Baud Rate

<table>
<thead>
<tr>
<th>UART/IR Baud Rate</th>
<th>16XCLK</th>
<th>Instruction Frequency (4 X)</th>
<th>Instruction Cycles @ Device Frequency (MHz)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>9,600</td>
<td>153,600</td>
<td>614,400</td>
<td>14.7456 18.432 19.6608</td>
<td></td>
</tr>
<tr>
<td>19,200</td>
<td>307,200</td>
<td>1,228,800</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>38,400</td>
<td>614,400</td>
<td>2,457,600</td>
<td>6</td>
<td>7.5</td>
</tr>
<tr>
<td>57,600</td>
<td>921,600</td>
<td>3,686,400</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>115,200</td>
<td>1,843,200</td>
<td>7,372,800</td>
<td>2</td>
<td>2.5</td>
</tr>
</tbody>
</table>
USING THE PICmicro® MCU'S CCP MODULE TO GENERATE THE 16XCLK

The CCP module is a hardware module that can be configured for Capture, Compare or PWM operation. This module works in conjunction with either Timer1 or Timer2, depending on the CCP mode selected.

PWM mode is required for the generation of a 16XCLK. For PWM generation, the Timer2 module is used to create the time-base. A simplified block diagram of the PIC16F877A's CCP module in PWM mode is shown in Figure 3. The PIC16F877A has two CCP modules; this figure is generic for either.

A PWM output waveform is shown in Figure 4. The period of the PWM determines how often the waveform repeats, while the duty cycle determines the relationship between the time the signal is high to the time the signal is low. For the 16XCLK signal, a 50% time is desired.

The PWM module takes advantage of the PICmicro MCU's internal Q clocks (4Q clocks for an instruction cycle) for the generation and duty cycle of the PWM. This allows slower device frequencies (lower power) to be used for a given baud rate. The minimum PWM period can be an instruction cycle (TCY).

Note: There are some small, low-cost PICmicro devices that have the CCP module. These include the 8-pin PIC12F683 and 14-pin PIC16F684.

FIGURE 3: CCP MODULE - PWM MODE BLOCK DIAGRAM

FIGURE 4: PWM OUTPUT WAVEFORM

Duty Cycle = DCxB9:DCxB0 (approximately 50%)
Period = PR2 + 1

Before PWM Initialized 1st PWM Cycle Subsequent PWM Cycles

1. Timer2 is cleared and new duty cycle value is loaded from the duty cycle latch into the duty cycle slave register
2. Timer2 value equals to value in duty cycle latch register, with CCP pin driven low
3. Timer2 matches PR2 register, value from duty cycle latch is loaded into slave register, CCP pin driven high, with Timer2 being forced clear
PWM period
The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using Equation 1.

**EQUATION 1: CALCULATION FOR PWM PERIOD**

\[
T_{PWM \text{ period}} = [(\text{PR2}) + 1] \times 4 \times \text{Tosc} \\
\times (\text{TMR2 prescale value})
\]

Where:
- PR2 = Value in PR2 Register
- Tosc = Oscillator Clock

PWM frequency is defined as 1/ [PWM period].
When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

1. TMR2 is cleared
2. The CCPx pin is set (exception: if PWM duty cycle = 0%, the CCPx pin will not be set)
3. The PWM duty cycle is latched from CCPRxL into CCPRxH

The period of the PWM determines the frequency of the 16XCLK signal. This corresponds to the UART/IR baud rate. Table 4 shows, at the common UART/IR baud rates and PICmicro MCU frequencies, what the PR2 (PWM period) and CCPRxL (PWM duty cycle) values are.

PWM Duty Cycle
The PWM duty cycle is specified by writing to the DCxB9:DCxB0 bits. DCxB9:DCxB2 are contained in the CCPRxL register, while DCxB1:DCxB0 are located at CCPxCON<5:4>. The CCPRxL contains the eight MSbs, while CCPxCON<5:4> contains the two LSbs. This 10-bit value is represented by DCxB9:DCxB0. **Equation 2** is used to calculate the PWM duty cycle.

**EQUATION 2: CALCULATING THE PWM DUTY CYCLE**

\[
PWM \text{ Duty Cycle} = (\text{DCxB}<9:0> \text{ bits value}) \times \text{Tosc} \\
\times (\text{TMR2 prescale value})
\]

Where:
- PWM Duty Cycle = PWM Duty Cycle Time
- Tosc = Oscillator Clock

Though the DCxB<9:0> bits can be written to at any time, the duty cycle value is not latched into CCPRxH until after a match between PR2 and TMR2 occurs (which marks the end of the current period). In PWM mode, CCPRxH is a read-only register.

The CCPRxH register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

When CCPRxH and a 2-bit latch match the value of TMR2 concatenated with the internal 2-bit Q clock (or two bits of the TMR2 prescaler), the CCPx pin is cleared. This marks the end of the duty cycle.

**Table 4:** PICmicro® MCU FREQUENCY VS. HOST UART/IR BAUD RATE

<table>
<thead>
<tr>
<th>UART/IR Baud Rate</th>
<th>16XCLK Frequency</th>
<th>Instruction Frequency (4 X)</th>
<th>TMR2 Prescaler</th>
<th>Device Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14.7456</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PR2</td>
</tr>
<tr>
<td>9,600</td>
<td>153,600</td>
<td>614,400</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>19,200</td>
<td>307,200</td>
<td>1,228,800</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>38,400</td>
<td>614,400</td>
<td>2,457,600</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>57,600</td>
<td>921,600</td>
<td>3,686,400</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>115,200</td>
<td>1,843,200</td>
<td>7,372,800</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note:**
- The CCPxX:CCPxY bits are always set to ‘11'b
- CCPxX:CCPxY = '00'b will supply approximately a 50% duty cycle when PR2 = 1 and CCPRxL = 1.
- CCPxX:CCPxY = '10'b will supply a 50% duty cycle when PR2 = 0 and CCPRxL = 0.
- A PR2 value cannot be selected for this baud rate (too much clock error).
- PWM can not be configured for this baud rate at this device frequency.
CCP Pin

The CCPx pin is multiplexed with the PORT data latch and the corresponding TRIS bit must be cleared to make the CCPx pin an output.

Set-up for PWM Operation

The following steps configure the CCP module for PWM operation:
1. Establish the PWM period by writing to the PR2 register.
2. Establish the PWM duty cycle by writing to the DCxB9:DCxB0 bits (CCPR1L register and CCP1CON<5:4> bits).
3. Make the CCPx pin an output by clearing the appropriate TRISC bit.
4. Establish the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP module for PWM operation.

Sleep Operation

When the PIC16F877A is placed in sleep, Timer2 will not increment and the state of the module will not change. If the CCP pin is driving a value, it will continue to drive that value. When the device wakes up, it will continue from this state.

Effects of a Reset

When the PIC16F877A is reset, the Timer2 and CCP modules are forced off. The Timer2 and CCP module will need to be reconfigured for PWM operation.

SUMMARY

This application note has shown how a Host Controller can be interfaced to the MCP2122. Several methods have been discussed, with a detailed discussion on how to use the PICmicro MCU's CCP module. The source code for the example PIC16F877A code is provided in Appendix A. Appendix B shows screen captures of the PWM (16XCLK) and data bit timings from the PIC16F877A code. This should help bootstrap the development of any new applications that plan on using the MCP2122.

Note:  If it is desired to place the PICmicro device in Sleep mode, it is good practice to force the PWM pin to a known state before going to “sleep”. This can be done by disabling the PWM (CCP and Timer2 modules) and forcing the I/O pin to the desired state.
LIST  C=132
include P16F877A.inc
ERRORLEVEL -302, -306

FUNCTION MAIN

MCP2122 Transmit Data Code
This code will generate the 16x clock and continuously transmit a
known byte value (bit stream).
The variables "BaudRate" and "TXValue" must first be selected and the program then
assembled and downloaded for these configurations to become effective.
After Reset of the PICmicro(r) MCU, the device baud rate will determine the
values to load into the UART and CCP1 (PWM mode)/Timer2
registers. The TXValue will be the value that is continuously transmitted.

Device: PIC16F877A
Platform: PICDEM™ 2 Plus
Device Frequency: Fosc = 14.7456MHz

PORTC is used to interface to the MCP2122 device. The signals on this
port are: TX, RX, 16XCLK, and RESET

Revision History
1.0 06/10/04 Initial Release

SOFTWARE LICENSE AGREEMENT

The software supplied herewith by Microchip Technology Incorporated
(the "Company") is intended and supplied to you, the Company's
customer, for use solely and exclusively with products manufactured
by the Company.

The software is owned by the Company and/or its supplier, and is
protected under applicable copyright laws. All rights are reserved.
Any use in violation of the foregoing restrictions may subject the
user to criminal sanctions under applicable laws, as well as to
civil liability for the breach of the terms and conditions of this
license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
**FIGURE A-2: MCP2122TX.ASM - PAGE 2**

```assembly
; PICDEM 2 Plus Requirements
; Clock Frequency: 14.7456 MHz
; UART Baud: 9600, 19200, 38400, 57600, 115200
; -->
; 16XCLK: 153600, 307200, 614400, 921600, 1843200
; Instructions/bit (@ 115200 Baud) = 14.7456/1.8432 = 8 (80 instructions/byte)
; PIC16F877 PORT Functions
; PORTA
; Function         ---   ---   NA    NA    NA    NA    NA    NA
; TRIS Direction   ---   ---    O     O     O     O     O     O
; Initial value    ---   ---    H     H     H     H     H     H
; PORTB
; Function         NA    NA    NA    NA    LCD3  LCD2  LCD1  LCD0
; TRIS Direction   O     O     O     O     O     O     O     O
; Initial value    H     H     H     H     H     H     H     H
; PORTC
; Function         RX    TX   RST213X NA    NA   CCP1   NA   NA
; TRIS Direction   I     I     O     O     O     O     O     O
; Initial value    ---   ---    H     H     H     H     H     H
; PORTD
; Function         NA    NA    NA    NA    NA    NA    NA    NA
; TRIS Direction   O     O     O     O     O     O     O     O
; Initial value    H     H     H     H     H     H     H     H
; PORTE
; Function         ---   ---   ---   ---   ---   NA    NA    NA
; TRIS Direction   ---   ---   ---   ---   ---    O     O     O
; Initial value    ---   ---   ---   ---   ---    H     H     H

#define reset   H'00' ;Reset vector
```

**Configuration Bits**

```assembly
__CONFIG _CP_OFF & _PWRTE_ON & _HS_OSC & _WDT_OFF & _BODEN_OFF & _LVP_OFF & _WRT_HALF & _CPD_OFF & _DEBUG_ON
__IDLOCS H'0010'
```

For PIC16F877

```assembly
__CONFIG _CP_OFF & _PWRTE_ON & _HS_OSC & _WDT_OFF & _BODEN_OFF & _LVP_OFF & _WRT_ENABLE_ON & _CPD_OFF & _DEBUG_ON
```
; PortA Bits
#define rxd PORTC, 7 ; input, serial data from MCP2122
#define txd PORTC, 6 ; output, serial data to MCP2122
#define RESET213x PORTC, 5 ; output, used to reset the MCP2122
; high for normal operation,
; low to RESET device
;
; Program Definitions
;
; Data Direction for PORTA (output port)
ddra EQU B'00000000'
; Data Direction for PORTB (output port)
ddrb EQU B'00000000'
; Data Direction for PORTC (input/output port)
ddrc EQU B'11000000'
; Data Direction for PORTD (output port)
ddrd EQU B'00000000'
; Data Direction for PORTE (output port)
ddre EQU B'00000000'

cfgopt equ B'11001000' ; option reg setup

; Constants
;
; Baud Rate
; Baud Rate EQU D'9600' ;
; Baud Rate EQU D'19200' ;
; Baud Rate EQU D'38400' ;
; Baud Rate EQU D'57600' ;
; Baud Rate EQU D'115200' ;

; Value to transmit
; TXValue EQU 0xFF ;
; TXValue EQU 0x00 ;
; TXValue EQU 0x7E ;
; TXValue EQU 0x81 ;
; TXValue EQU 0xFE ;
; TXValue EQU 0x01 ;
; TXValue EQU 0xAA ;
; TXValue EQU 0x55 ;
; TXValue EQU 0xC3 ;
; TXValue EQU 0x3C ;

; Host UART Data Rate/BRG Value
; SPBRG Value
; B9600at14xMHz EQU D'95'
; B19200at14xMHz EQU D'47'
; B38400at14xMHz EQU D'23'
; B57600at14xMHz EQU D'15'
; B115200at14xMHz EQU D'07'
FIGURE A-4: MCP2122TX.ASM - PAGE 4

; PWM Values to generate 16XCLK
; (For the duty cycle register, the CCP1X and CCP1Y bits are set to "1")
;
; PR2 Value      CCPR1L Value
PR2B9600at14xMHz EQU D'23' ; D'11'
PR2B19200at14xMHz EQU D'11' ; D'05'
PR2B38400at14xMHz EQU D'05' ; D'02'
PR2B57600at14xMHz EQU D'03' ; D'01'
PR2B115200at14xMHz EQU D'01' ; D'00'
;
if (BaudRate == D'9600')
  UARTBaud   EQU   B9600at14xMHz ;
  PWMPR2Value EQU   PR2B9600at14xMHz ;
endif
if (BaudRate == D'19200')
  UARTBaud   EQU   B19200at14xMHz ;
  PWMPR2Value EQU   PR2B19200at14xMHz ;
endif
if (BaudRate == D'38400')
  UARTBaud   EQU   B38400at14xMHz ;
  PWMPR2Value EQU   PR2B38400at14xMHz ;
endif
if (BaudRate == D'57600')
  UARTBaud   EQU   B57600at14xMHz ;
  PWMPR2Value EQU   PR2B57600at14xMHz ;
endif
if (BaudRate == D'115200')
  UARTBaud   EQU   B115200at14xMHz ;
  PWMPR2Value EQU   PR2B115200at14xMHz ;
endif
;

;*************************************************************************************************
; Registers
;
cblock H'20'
temp
endc

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

org H'00'     ; use 00h as reset vector
goto START

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

START    clrf    STATUS                    ; Bank 0
movlw   0xFF                      ; Force PORTB to display High when configured as Output
movwf    PORTB
bsf     STATUS, RP0               ; Bank 1
movlw   ddra
movwf    TRISA                     ; Configure PORTA
movlw   ddbb
movwf    TRISB                     ; Configure PORTB
movlw   ddcrc
movwf    TRISC                     ; Configure PORTC
movlw   dddrd
movwf    TRISD                     ; Configure PORTD
movlw   ddeere
movwf    TRISE                     ; Configure PORTE
movlw    cfgopt                    ; setup option reg
movwf    OPTION_REG

;
INITBAUD

CLRF STATUS ; Bank 0
CLRF CCP1CON ; CCP1 Module Off
CLRF T2CON ; Timer 2 Off, Prescaler = 1, Postscaler = 1

BSF STATUS, RP0 ; Bank 1
MOVLW UARTBaud ; Initialize UART
MOVF SPBRG ;
MOVLW 0x24 ; BRGH = 1, 8-bit, TX Enabled, Async.
MOVF TXSTA ;

CLRF STATUS ; Bank 0
MOVLW 0x90 ; Enable serial port, continuous receive
MOVF RCSTA ;

CLRF TMR2 ; TMR2 = 00h
MOVLW 0xFF
MOVF CCP1CON ; PWM mode, CCP1X:CCP1Y = 11b
MOVLW (PWMPR2Value-1)/2 ;
MOVF CCP1L ; Load the PWM Duty Cycle value
BSF STATUS, RP0 ; Bank 1
MOVLW PWMPR2Value ;
MOVF PR2 ; Load the PWM Period value
BSF STATUS ; Bank 0
BSF T2CON, TMR2ON ; Turn On Timer 2, so 16XCLK is generated
CLRF PORTB ; clear outputs (Display on LEDs)

TXData

CLRF STATUS ; NO, Bank 0
BCF RESET213x ; Force MCP2122 RESET pin Low (MCP2122 in RESET)
NOP ; Hold MCP2122 in reset for some time
NOP
NOP
NOP
BSF RESET213x ; Force MCP2122 RESET pin High (Normal Operation)

TXValueLP1

bsf STATUS, RP0 ; Bank 1

UARTWaitLP

btfss TXSTA, TRMT ; check if UART ready to transmit
goto UARTWaitLP ; not ready, wait
bcf STATUS, RP0 ; Bank 0
MOVLW TXValue ; This is the value of the byte to send
MOVF TXREG ; send the byte
GOTO TXValueLP1 ;

LAST NOP

end
APPENDIX B: SCREEN CAPTURE BIT TIMING

The following screen captures (from Figure B-1 to Figure B-5) show the bit time for a single data bit at the indicated baud rate. The circled $1/\Delta t$ time shows the measured baud rate.

FIGURE B-1: BIT TIMING CAPTURE AT 9600 BAUD
FIGURE B-2: BIT TIMING CAPTURE AT 19200 BAUD

15XCLK

TX

File Edit Vertical Horiz/Acq Trig Display Cursors Measure Masks Math MyScope Utilities Help

Tek Stopped Single Seq 1 ACQs 10 Jun 04 15:34:57

Curs1 Pos

$1.894us

Curs2 Pos

-105.8ns

At: -52.0μs

1/At: 19.23kHz

Area(G1): 239.2μV s

μ: 239.22914μ

m: 239.2μ

σ: 0.0

n: 1.0

Ch1 5.0V Ch2 5.0V M 10.0ps 2.5μs/div IT 200ps/div
FIGURE B-3: BIT TIMING CAPTURE AT 38400 BAUD
FIGURE B-4: BIT TIMING CAPTURE AT 57600 BAUD
FIGURE B-5: BIT TIMING CAPTURE AT 115200 BAUD
Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families 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 Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is 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 products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

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 ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company’s quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001:2000 certified.