

Serial Port Utilities

INTRODUCTION

PIC17C42 has an on chip high speed Universal Synchronous Asynchronous Receiver Transmitter (USART). The serial port can be configured to operate either in full-duplex asynchronous mode or half duplex synchronous mode. The serial port has a dedicated 8-bit baud rate generator. Either 8- or 9-bits can be transmitted/received.

This application note provides information on using the serial port, parity generation, serial port expansion, RS-232 interface, I/O port expansion using synchronous mode of serial port.

SERIAL PORT USAGE

Brief code to setup serial port, receive and transmit data is given below. Small sections of code for both asynchronous and synchronous mode are given.

EXAMPLE 1

```

SPBRG : 25 : 9600 baud @ 16 MHz input clock
TXSTA : 00100000 (20h) : 8-bit transmission, async mode
RCSTA : 10010000 (90h) : 8-bit reception, enable serial port, enable reception

;*****
; Sample Code For Asynchronous Mode Serial Port Setup
;*****

#define ClkFreq 16000000 ; input clock frequency = 16 MHz
#define baud(X) ((10*ClkFreq/(64*X))+5)/10 - 1
#define TXSTA_INIT 0xB0
#define RCTSA_INIT 0x90

#include "17C42.h" ; file containing the Register Definitions

Setup_Async_Mode
    movlb 0 ; SPBRG, TXSTA & RCSTA are in bank 0
    movlw baud(9600) ; equals 25 for 9600 baud
    movwf SPBRG ; baud rate generator is reset & initialized
    movlw TXSTA_INIT
    movwf TXSTA ; 8-bit transmission, async mode
    movlw RCSTA_INIT
    movwf RCSTA ; 8-bit reception, enable serial port,
                ; enable reception
    return
;*****

```

Asynchronous Mode Setup

Asynchronous mode set-up requires selection of 8/9-bits of data transfer, baud rate, setting the baud rate generator and configuring the TXSTA & RCSTA control registers. The baud rate generator is set by writing the appropriate value to SPBRG register (bank0, file 17h). The value to be written to SPBRG is given by:

$$SPBRG = \frac{Input_Clk_Freq}{64 * Baud_Rate} - 1$$

For example, to select a baud rate of 9600 bits/sec with input clock frequency of 16 MHz, SPBRG is computed from the above equation to be 25. Once the Baud Rate Generator is setup, it is necessary to configure the TXSTA & RCSTA control registers as follows (please refer to the data sheet) :

Serial Port Utilities

Synchronous Mode Setup

Synchronous mode setup requires selection of 8/9-bits of data transfer, bit rate, setting the baud rate generator and configuring the TXSTA & RCSTA control registers. The baud rate generator is set by writing the appropriate value to SPBRG register (bank0, file 17h). The value to be written to SPBRG is given by:

$$\text{SPBRG} = \frac{\text{Input_Clk_Freq}}{4 * \text{Baud_Rate}} - 1$$

For example, to select a bit rate of 1 Mhz with input clock frequency of 16 Mhz, SPBRG is computed from the above equation to be 3. Once the Baud Rate Generator is setup, it is necessary to configure the TXSTA & RCSTA control registers as follows (please refer to the data sheet) :

EXAMPLE 2

```
SPBRG :      3           : 1 Mbits/sec @ 16 Mhz input clock
TXSTA  :    10110000 (B0h) : 8-bit transmission, Sync mode (MASTER)
RCSTA  :    10010000 (90h) : 8-bit reception, enable serial port, continuous
                               reception

;*****
;          Sample Code For Synchronous Mode (MASTER) Serial Port Setup
;*****

#define ClkFreq      16000000      ; input clock frequency = 16 Mhz
#define baud(X)      ((10*ClkFreq/(4*X))+5)/10 - 1
#define TXSTA_INIT   0xB0
#define RCTSA_INIT   0x90

#include "17C42.h" ; file containing the Register Definitions

Setup_Sync_Master_Mode
    movlb 0 ; SPBRG, TXSTA & RCSTA are in bank 0
    movlw baud(1000000) ; equals 3 for 1 Mbits/sec
    movwf SPBRG ; baud rate generator is reset & initialized
    movlw TXSTA_INIT
    movwf TXSTA ; 8-bit transmission, async mode
    movlw RCSTA_INIT
    movwf RCSTA ; 8-bit reception, enable serial port,
                ; enable reception

    return
;*****
```

Receiving Data (Software Polling)

The sample code provides a way to read the received serial data by software polling (with no serial port interrupts). This applies to both asynchronous and synchronous mode. Software polling is done by checking the RBFL bit (PIR<0>). If this bit is set it means that a word has been received (8 bits are in RCREG and the 9th bit in RCSTA<0>).

EXAMPLE 3

```

;*****
;           Return The 8-bit received Data By Software Polling
;           The received data is returned in location SerInData
;*****
Get_Serial_Data_Poll
    movlb 1           ; PIR is in bank 1
PollRcv    btfss PIR,0      ; chech the RBFL bit
           goto PollRcv    ; loop until char received, assume WDT is off
           movlb 0         ; RCREG is in bank 0
           movpf RCREG,SerInData
           return         ; Received 8-bits are in SerInData
;*****

```

Transmitting Data (Software Polling)

The sample code provides a way to transmit serial data by software polling (no serial port interrupts). Software polling is done by checking the TBMT bit (PIR<0> in bank 1) to be one, indicating the transfer of TXREG to the serial shift register.

EXAMPLE 4

```

;*****
;           Transmit 8-bit Data By Software Polling
;           The data to be transmitted is in location SerOutData
;*****
Send_Serial_Data_Poll
    movlb 1           ; PIR is in bank 1
PollTxmt   btfss PIR,1      ; chech the TBMT bit of PIR register in bank1
           goto PollTxmt    ; loop until char received, assume WDT is off
           movlb 0         ; RCREG is in bank 0
           movfp SerOutData,TXREG
           return         ; Received 8-bits are in SerInData
;*****

```

Serial Port Utilities

Transmitting & Receiving A Block Of Data (Interrupt Driven)

A general purpose routine which is interrupt driven that transmits and receives a block of data is provided. The reception or transmission of the block is ended when an end of block character is detected. As an example, the

end of block is identified by a 0. The block of data to be transmitted is stored in the program memory and `TABLRD` instruction is used to transfer this example data to the file registers and serial port. The user may modify this code to a more general purpose routine that suits his application.

EXAMPLE 5

MPASM B0.54

PAGE 1

```

;          TITLE   `Serial Interface Routines
;          LIST P=17C42, C=80. I=ON, R=DEC

;This is a short program to demonstrate how to transmit and
;serial data using the PIC17C42.
;
;A message will be transmitted and routed right back to the
;and read. The read information will be saved in an interna
;
;          include "p17reg.h"

0080          TX_BUFFER      equ    0x80
00B0          RX_BUFFER      equ    0xB0
0020          RXPTR          equ    0x20
0021          TXPTR          equ    0x21
0022          SERFLAG        equ    0x22
0023          RTINUM         equ    0x23
0001          RXDONE         equ    1
0000          TXDONE         equ    0
0002          HILOB          equ    2
;
;
;          ORG    0
0000 C072          goto    start
;
;          ORG    0x0010          ;vector for rtcc interrupt
;rtcc_int          ;not used here
;
;          ORG    0x0020          ;vector for peripheral inte
perf_int
0020 C04D          goto    service_perf ;service the interrupts
;
;          ORG    0x0030
;
;initialize the serial port: baud rate interrupts etc.
init_serial
0030 2922          clr     SERFLAG          ;clear all flags
0031 B800          movlb  0                ;select bank 0
0032 B007          movlw  0x07             ;select 9600 baud
0033 770A          movfp  W,SPBRG         ; /
0034 B090          movlw  0x90             ;set up serial pins
0035 730A          movfp  W,RCSTA         ; /
```

```

0036 2915          clrfs  TXSTA          ;setup transmit status
0037 B801          movlb  1              ;select bank 1
0038 2916          clrfs  PIR           ;clear all interrupts
0039 2917          clrfs  PIE           ;clear all enables
003A 8017          bsf    PIE,RCIE      ;enable receive interrupt
003B B0B0          movlw  RX_BUFFER     ;set pointer to rx buffer
003C 4A20          movpfs W,RXPTR      ; /
003D 2907          clrfs  INTSTA       ;clear all interrupts
003E 8307          bsf    INTSTA,PEIE   ;enable peripheral ints
003F 0005          retfies

;
;start transmission of first two bytes
start_xmit
0040 B800          movlb  0              ;select bank 0
0041 8515          bsf    TXSTA,TXEN    ;enable transmit
0042 ABOA          tablrds 1,1,W        ;load latch
0043 A216          tlrds  1,TXREG      ;load high byte
0044 B801          movlb  1              ;select bank 1
empty_chk
0045 9116          btffs  PIR,TBMT      ;TXBUF empty?
0046 C045          goto   empty_chk      ;no then keep checking
0047 B800          movlb  0              ;select bank 0
0048 A916          tablrds 0,1,TXREG    ;load lo byte
0049 B801          movlb  1              ;select bank 1
004A 8117          bsf    PIE,TXIE     ;enable transmit interrupts

004B 8222          bsf    SERFLAG,HILOB ;set up next for high byte

004C 0002          return

;
;
;
service_perfs
;check for transmit or receive interrupts only
004D 9816          btffs  PIR,RBFL      ;RX buffer full?
004E C062          goto   service_recvs ;yes then service
004F 9116          btffs  PIR,TBMT      ;TX buffer empty?
0050 C060          goto   exit_perfs ;no, ignore other int.
service_xmits
0051 9822          btffs  SERFLAG,TXDONE ;all done?
0052 C060          goto   exit_perfs ;yes then quit
0053 9A22          btffs  SERFLAG,HILOB ;if clr, do low byte
0054 C057          goto   rd_hi         ;else read high byte
0055 A90A          tablrds 0,1,W        ;read lo
0056 C058          goto   sx_cont      ;continue
rd_hi
0057 A20A          tlrds  1,W          ;read high byte
sx_cont
0058 3A22          btg    SERFLAG,HILOB ;toggle flag
0059 B800          movlb  0              ;bsr=0
005A 4A16          movpfs W,TXREG      ;load tx reg
005B 330A          tstfss W ;last byte?
005C C060          goto   exit_perfs ;no then cont
;else end transmit
end_xmits
005D B801          movlb  1              ;select bank 1
005E 8917          bcf    PIE,TXIE     ;disable tx interrupt
005F 8022          bsf    SERFLAG,TXDONE ;set done flag
exit_perfs
0060 8F07          bcf    INTSTA,PEIR   ;clear peripheral int
0061 0005          retfies

;
service_recvs
0062 9922          btffs  SERFLAG,RXDONE ;RX complete?
0063 C060          goto   exit_perfs ;exit int
0064 6120          movpfs RXPTR,FSR0 ;get pointer
0065 B800          movlb  0              ;select bank 0
0066 6014          movpfs RCREG,F0 ;load received value
0067 290A          clrfs  W              ;clr W
0068 3200          cpfsgt F0 ;value = 0?

```

Serial Port Utilities

```
0069 C06D          goto    end_recv    ;yes then end
006A 1501          incf    FSR0        ;inc pointer
006B 4120          movpfp FSR0,RXPTR   ;save pointer
006C C060          goto    exit_perf   ;return from int

end_recv
006D 8122          bsf    SERFLAG,RXDONE ;set flag
006E 2907          clrf   INTSTA       ;clear all int
006F B801          movlb  1            ;select bank 1
0070 8817          bcf    PIE,RCIE     ;disable rx interrupts
0071 C060          goto    exit_perf   ;return
;
start
0072 2909          clrf   FSR1         ;assign FSR1 as S.P.
0073 0709          decf   FSR1         ; /
0074 B020          movlw  0x20         ;clear ram space
0075 610A          movfip W,FSR0       ;do indirect addressing

start1
0076 2900          clrf   F0           ;clear ram
0077 1F01          incfsz FSR0         ;inc and skip if done
0078 C076          goto   start1
0079 E030          call  init_serial   ;initialize serial port
007A B000          movlw HIGH MESSAGE ;load table pointer
007B 4A0D          movpfp W,TBLPTRL    ; /
007C B001          movlw LOW MESSAGE  ; /
007D 4A0E          movpfp W,TBLPTRH    ; /
007E E040          call  start_xmit    ;start transmission

chk_end
007F 9122          btfs  SERFLAG,RXDONE ;receive all?
0080 C07F          goto  chk_end       ;no then keep checking
;
loop
0081 C081          loop  goto    loop   ;spin wheel
;
ORG    0x100
MESSAGE
0100 5468 6520 636F DATA  "The code is: Tea for the Tillerman"
0103 6465 2069 733A
0106 2054 6561 2066
0109 6F72 2074 6865
010C 2054 696C 6C65
010F 726D 616E
0111 0000          DATA  0
;
;
END

Errors   :    0
Warnings :    0
```

PARITY GENERATION

Since the serial port of PIC17C42 does not have an on chip parity generator, parity is generated using software. It takes only 10 program memory words and executes in 10 instruction cycles to generate parity. Since the serial port of PIC17C42 can operate in a 9-bit mode, the parity bit can be generated in software and transmitted as the 9th bit or be compared with the received 9th bit.

In case of transmission, set TX8/9 to 1 (< 6> of TXSTA) to enable 9-bit transmission and write the computed parity bit to TXD8 (TXSTA<0>). The 9th bit (parity bit) must be written prior to writing the 8 data bits to TXREG.

In case of reception, first of all enable 9-bit reception by setting RC8/9 to 1 (RCSTA<6>). Upon successful reception, the 9 bit is received in RCD8 (RCSTA<0>). Parity of the 8 bits of received data is computed using the routine listed below and compared with the 9-bit received.

EXAMPLE 6

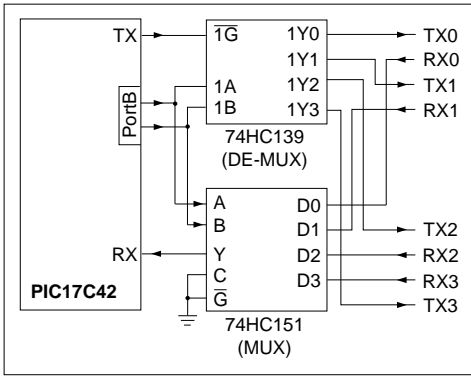
```
*****  
; Generate Parity Bit for the 8 bit register 'data'  
; The parity bit is stored in Bit 0 of 'parity'  
;  
*****  
  
#define ODD_PARITY FALSE  
  
    swapf    data,w  
    xorwf   data,w  
    movwf   parity  
    rrrncf  parity  
    rrrncf  parity  
    xorwf   parity,w  
    andlw   0x03  
    addlw   0x01  
    rrrncf  wreg  
    movwf   parity  
#if ODD_PARITY  
    btg    parity,0  
#endif
```

Serial Port Utilities

SERIAL PORT EXPANSION

The PIC17C42 has only one serial port. For applications that require the PIC17C42 to communicate with multiple serial ports, a scheme that multiplexes and demultiplexes the RX and TX pins is provided below. This method is suited only if no more than one UART is needed at any one time. This is the case in many applications where the microcontroller drives several output devices serially. Figure 1 shown below suggests a way to expand the on-chip serial port to 4 serial ports. To use the scheme as shown in Figure 1, The PIC17C42 must select the desired serial port by appropriately setting the two pins of PORT-B. The same scheme may be used to further expand the serial ports by using more I/O Ports.

FIGURE 1 - MULTIPLEXING THE ON-CHIP UART



RS-232 INTERFACE

Two circuits are provided to interface the CMOS levels of PIC17C42 to RS-232 levels. Figure 2 provides interface to MAX232 (MAXIM's RS-232 Driver/Receiver) with a single +5V power supply. Figure 3 provides a low cost 2 chip solution for RS-232 level translation using a single +5V supply (Note that V- of MC14C88 is connected to DTR of RS-232 Interface. By asserting DTR to low, V- gets the negative voltage from the RS-232 line). An alternative single chip low cost solution is provided in Figure 4. However 3 voltage sources (+5, +12, -12) are necessary.

FIGURE 2 - RS-232 INTERFACE TO MAX232

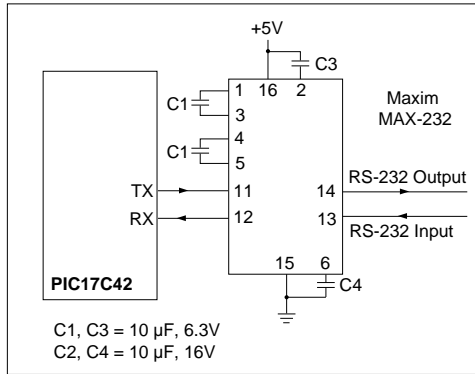


FIGURE 3 - LOW COST 2 CHIP SOLUTION USING SINGLE POWER SOURCE

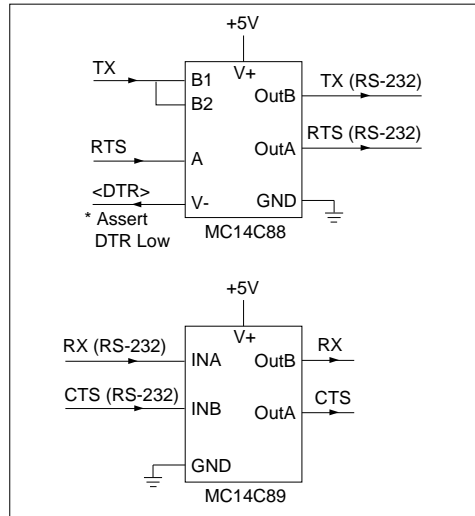


FIGURE 4 - LOW COST SINGLE CHIP SOLUTION USING 3 POWER SOURCES

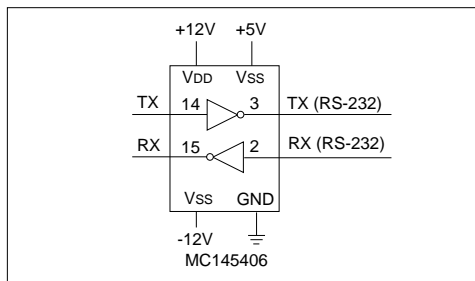


Table 1 provides the summary of RS-232 and V.28 Electrical Specifications.

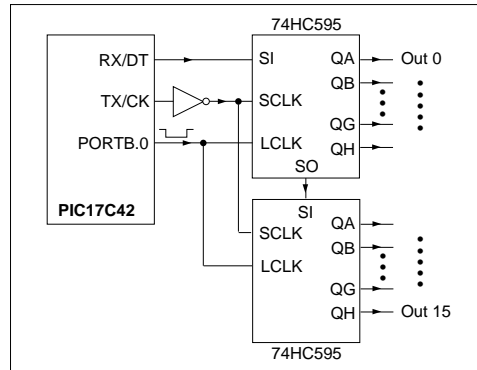
TABLE 1 - SUMMARY OF RS-232C AND V.28 ELECTRICAL SPECIFICATIONS

Parameter	Specification	Comments
Driver Output Voltage		
0 level	+5V to +15V	With 3-7k Ω load with 3-7k Ω load No load
1 level	-5V to -15V	
Max. output	$\pm 25V$	
Receiver Input Thresholds (Data and clock signals)		
0 level	+3V to +25V	
1 level	-3V to -25V	
Receiver Thresholds RTS, DSR, DTR		
On level	+3V to +25V	Detects power Off condition at driver
Off level	Open circuit or -3V to -25V	
Receiver input resistance	3k Ω to 7k Ω	
Driver output resistance.		
Power off condition	300 Ω Min.	Vout < $\pm 2V$
Driver slew rate	30V/ μ s max.	3k Ω < RL < 7k Ω ; 0pF < CL < 2500pF
Signalling rate	Up to 20k bits/sec.	
Cable length		
	50'/15m. Recommended max. length	Longer cables permissible, if CLoad \leq 2500pF

I/O PORT EXPANSION USING SYNCHRONOUS MODE

Although the PIC17C42 has 33 I/O pins, most of these are multiplexed with other peripheral functions. In case more I/O ports are needed, the scheme provided below expands the I/O port using the synchronous mode of serial port by serially shifting the data. Figure 5 shows a scheme to expand the output ports to 16-bits using 2 standard logic chips (74HC595). The PIC17C42's serial port is configured in synchronous mode and set to be the MASTER. Thus serial data is available on DT (pin 22) and the clock is available on CK (pin 21). The following code will transmit 16-bits serially and clock all the 16-bits at the same time.

FIGURE 5 - OUTPUT PORT EXPANSION USING SYNCHRONOUS MODE



4

EXAMPLE 1

```

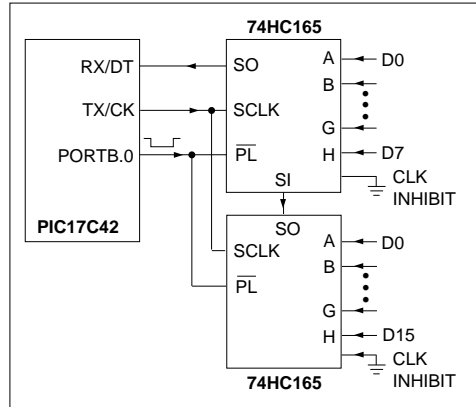
InitSerialPortTxmt
    movlb    0
    clrf          SPBRG          ; set to highest baud rate = CLKOUT = CLKIN/4
    movlw    0x80
    movwf   RCSTA          ; enable serial port
    movlw    0xB0
    movwf   TXSTA          ; 8 bit synchronous master mode
    bcf     DDRB,0          ; set bit 0 of PortB as output, to be used as Latch Clk
    return

SendSerialData          ; shift out DataLo & DataHi serially
    movlb    0
    movfp   DataLo,TXREG
    nop
    ; wait for TXREG transfer : if slower baud rate is
    ; used check for TBMT reset
    movfp   DataHi,TXREG
    wait    btfss   TXSTA,TRMT    ; wait until all 16 bits shifted out
    goto    wait
    bcf     PortB,0
    bsf     PortB,0          ; clock in the serial data to parallel output of HC595
    return
    
```

Serial Port Utilities

A similar scheme as shown above may be implemented in the reverse way to expand Input Ports. For this it is necessary to have a Parallel In and Serial Out device. Using a standard logic chip (74HC165), the scheme is shown in Figure 6. In order to read the 16-bit input data, the serial port of PIC17C42 is configured to be in Synchronous Mode Reception (MASTER mode). An I/O port (PortB<0>) is used to parallel load the 16 inputs for reading serially. A sample code to read the 16 inputs is shown below.

FIGURE 6 - INPUT PORT EXPANSION USING SYNCHRONOUS MODE



*Author: Amar Palacherla
Logic Products Division*

EXAMPLE 2

```

InitSerialPortRcv
    movlb    0
    clrf    SPBRG           ; set to highest baud rate = CLKOUT = CLKIN/4
    movlw   0x80           ; enable serial port
    movwf   RCSTA
    movlw   0x90           ; 8-bit synchronous master mode
    movwf   TXSTA
    bcf     DDRB,0        ; bit 0 of PortB is output, to be used as Parallel
                        ; Load
    bsf     PortB,0       ; disable parallel Load
    return

ReadSerialData           ; shift out DataLo & DataHi serially
    movlb    0
    bcf     PortB,0       ; Parallel Load The Inputs into 74HC165
    bsf     PortB,0       ; disable parallel Load
    bsf     RCSTA,SREN    ; enable single byte reception and wait for data
    movlb    1
wait1    btfss  PIR,RBFL  ; check until 8-bits are received
    goto    wait1
    movlb    0
    movpf   RCREG,DataLo  ; 1st byte is read
    movlb    0
    bsf     RCSTA,SREN    ; enable another byte of reception and wait for data
    movlb    1
wait2    btfss  PIR,RBFL  ; check until 8-bits are received
    goto    wait2
    movlb    0
    movpf   RCREG,DataHi  ; 2nd byte is read
    return
    
```

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

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

Atlanta

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

Boston

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

Chicago

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

Dallas

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

Dayton

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

Los Angeles

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

New York

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

AMERICAS (continued)

San Jose

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

ASIA/PACIFIC

Hong Kong

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

Korea

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

Singapore

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

Taiwan

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

EUROPE

United Kingdom

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

France

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

Germany

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

Italy

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

JAPAN

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

9/22/95

All rights reserved. © 1995, Microchip Technology Incorporated, USA.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. 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. The Microchip logo and name are registered trademarks of Microchip Technology Inc. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.