Frequency and Resolution Options for PWM Outputs

INTRODUCTION

The PIC17C42 is equipped with two high frequency Pulse Width Modulation (PWM) outputs. In a pulse width modulated signal the period of the signal is (usually) kept fixed, while the duty cycle is varied. In this application note, we will discuss options in selecting their frequency and resolution.

This application brief assumes that internal clock is used for the time-base, which is typically the preferred set-up. Also, throughout this application brief, PWM1 output is used in examples, timer1 is assumed to be the time-base.

Definition of terms:

Period of a PWM output is the duration after which the PWM pattern will repeat itself.

Frequency of a PWM output is = 1/Period.

Resolution of a PWM output is the granularity with which the duty cycle can be modulated.

In the case of the PIC17C42, when using PWM1 with timer1 as time-base the:

PWM1 period = [(PR1) + 1] x 4tosc

PWM1 duty cycle = (DC1) x tosc

where PR1 = period register for timer1

DC1 = PW1DCH, PW2DCL concantenated (10-bit value)

tosc = oscillator period

At 16 MHz oscillator frequency, tosc = 62.5 ns. The user can control the frequency of the PWM output by altering the 'period' value of the time-base. For example, if period is chosen to be 100 tosc (PR1 = 18h), then PWM frequency is 1/(100 x 62.5) ns = 160 KHz. Note however that duty cycle resolution is a little less than 7-bits.

Useful and Common PWM Modes

While a variety of period values can be selected, the following modes would be most commonly used:

10-Bit Mode: In this mode PWM duty cycle has full 10-bit resolution (maximum offered by the PIC17C42). The period register PR1 is set at FFh. PWM period is 1024tosc = 64 µs. PWM frequency is 15.625 KHz. The user must write both PW1DCH and PW1DCL to update PWM output. See Appendix A for an example that code modules 10-bit resolution PWM output (PWM10.LST).

8-Bit Hi-Resolution Mode: In this mode, the user has only an 8-bit quantity to write to the duty-cycle register. Period register is set at 3Fh (63 decimal), such that PWM period is 256 tosc. To write the 8-bit duty-cycle value, first the 8-bit is right shifted two bits. The upper six bits are written to PW1DCH and the lower two bits are written to PW1DCL as follows:

```
;8-bit duty-cycle value is in W reg
CLRF  TEMP        ;
RRCF  WREG        ;
RRCF  TEMP        ;
RRCF  WREG        ;
RRCF  TEMP        ;Shift right twice
ANDLW 00111111b   ;Mask off two-high bits
MOVPF WREG,PW1DCH ;Write duty-cycle val-
ues
MOVFP TEMP,PW1DCL ;
```

Note that in 8-bit, hi-resolution mode, maximum PWM frequency is attained. For example, at 16 MHz clock, PWM period = 256 tosc = 16 µs; PWM frequency = 62.5 KHz. See appendix B for an example code that generates 8-bit low high resolution PWM output (PWM8HI.LST).

FIGURE 1 - PWM OUTPUT

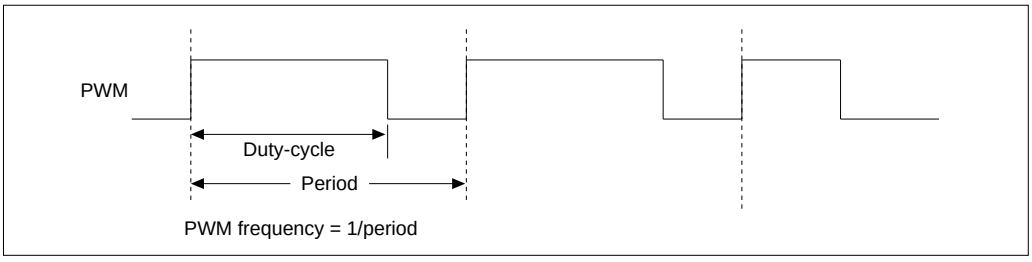
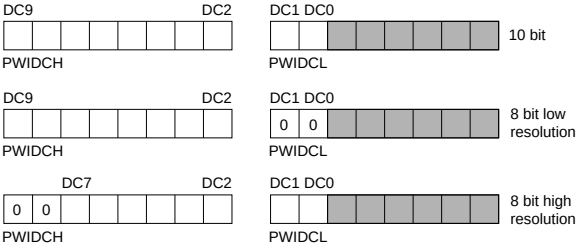


FIGURE 2 - VARIOUS PWM MODES



Note: DC registers suited for 16 bit computation; utilizing 10 MSB only.

8-Bit Low Resolution Mode

In this mode, the user still has only an 8-bit quantity to write to duty cycle register. However, the desired frequency of the PWM output is less, due to the nature of the application. For example, if the PWM output is being used to drive a motor through a power stage, the power transistors (or devices) due to their switching time will prefer PWM frequency not to exceed certain frequency. In the previous section, we derived an 8-bit resolution PWM output at 62.5 KHz.

To attain a low-resolution PWM output, the PW1DCL is always kept at zero. The 8-bit value is written to PW1DCH. The period (PR1) is set at FFh, i.e. 256 Tcy equals 1024 tosc (15.625 KHz). See Appendix C for an example code that produces 8-bit low resolution PWM output (PWM8LO.LST).

Choosing Resolution and Frequency of PWM Output

Actually, the resolution and the frequency of the PWM output is selectable within certain limits. The user will need to first define the requirements based on the application. There may be an upper limit to the frequency if the PWM is being used to drive motors. On the other hand, if the PWM is being filtered to generate an analog signal, higher frequency may be desirable. In any case, the lowest frequency achievable (using internal clock for the timer) is (OSC freq/1024). At 16 MHz oscillator input, the lowest PWM frequency possible is 15.625 KHz. At resolutions less than 10-bit, higher frequencies are possible (see Figure 3). For example, if 7-bit resolution is chosen, then the PWM frequencies can be 15-625 KHz, 31.25 KHz, 62.5 KHz or 125 KHz. The reader will note that it's how the 7-bits are placed within the 10-bit possible duty cycle value.

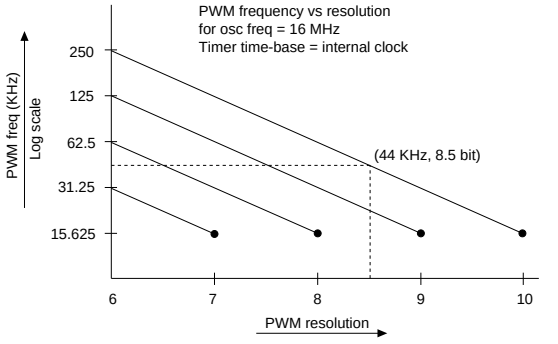
Conversely, if a certain frequency is desired, such as 44 KHz, then referring to Figure 1, resolution can be 8.5-bit or 7.5-bit or 6.5-bit etc.

Summary

The frequency and resolution of the PWM outputs of the PIC17C42 can be traded off against each other to best suit the application. The oscillator frequency can also be varied to adjust PWM frequency, if necessary. External clock should be used as timer time-base to generate very low frequency PWM output.

Authors: Stan D'Souza
Sumit Mitra
Logic Products Division

FIGURE 3 - PWM FREQ VS RESOLUTION



At a given oscillator frequency (16 MHz in this figure), a family of curves represents the PWM freq/resolution combinations possible.

APPENDIX A: PWM10.LST

```
MPASM B0.54                                                          PAGE  1
PULSE WIDTH MODULATION 10 BIT RESOLUTION


                        TITLE   “PULSE WIDTH MODULATION 10 BIT RESOLUTION”
                        LIST     P=17C42, C=80, T=ON


                        include        “17c42.h”


0021            PWM_HI  equ     0x21
0020            PWM_LO  equ     0x20
0022            TEMP    equ     0x22

                ;The user would generate a 16 bit value which is saved in r
                ;locations PWM_HI and PWM_LO byte. In 10 bit mode, the prog
                ;transfers these values directly to the Duty Cycle (DC) reg
                ;generate the required 10 bit PWM.
                ;
                ;
                ;This is a short program to demonstrate how to generate PWM
                ;10 bit resolution. Since a 10Mhz crystal was used in the t
                ;The max. period = 1024x100nS = 102.4 uS or 9.8 KHz. This p
                ;keeps the period constant and varies the duty cycle (which
                ;to the most significant 10 bits of the 16 bit value PWM_LO
                ;This program is interrupt driven, i.e. the update to the D
                ;is done in the rtcc interrupt, which then enables the pwm
                ;The period update is done during the pwm interrupt. The pw
                ;ramps up from 0% to 100% duty cycle and then repeats. The
                ;sweep takes approx. 52 secs.
                ;
                ;
                        ORG     0
0000 C058               goto    start
                ;
                        ORG     0x10            ;vector for rtcc interrupt
                rtcc_int
0010 C04C               goto    service_rtcc    ;service rtcc
                ;
                        ORG     0x0020          ;vector for pwm interrupt
                pwm_int
0020 C03E               goto    service_pwm     ;service pwm only
                ;
                        ORG     0x0030
                ;
                ;initialize internal hardware to generate the output
                ;for 10 bit resolution pwm.
                init_pwm10
0030 B802               movlb   2
0031 2910               clrf    tmr1            ;clear timer 1
                                                ;used to “drive” pwm1
0032 2B14               setf    pr1             ;set period=9.8 khz
0033 B803               movlb   3
0034 7221               movfp   PWM_HI,pw1dch   ;load duty cyl. hi byte
0035 7020               movfp   PWM_LO,pw1dcl   ;load duty cycle lo byte
0036 2916               clrf    tcon1           ;tmr1 inc. internally
                                                ;as 8 bit counter
0037 B01B               movlw   00010001B       ;start tmr1 and
0038 4A17               movpf   wreg,tcon2      ;enable pwm1
0039 B801               movlb   1
003A 2917               clrf    pie             ;clr all int. enables
003B 2916               clrf    pir             ;clear all interrupts
003C 8307               bsf     _peie           ;except peripheral int.
003D 0005               retfie
                ;
                ;
```

```
                        ;everytime a new value is written to the PWM_HI, PWM_LO reg
                        ;tmr1 interrupts is enabled. The DC value are written just
                        ;the “pwm interrupt” is enabled. Here the new period regist
                        ;updated. In this example, period is kept constant at 0xff
                        service_pwm
                                ;if the period changed, write new value here.
003E B802                       movlb   2               ;select bank 2
003F 2B14                       setf    pr1             ;period <— 0xff
0040 B801                       movlb   1               ;disable tmr1 int
0041 8C17                       bcf     _tm1ie          ;       /
0042 0005                       retfie
                        ;
                        ;This part of the program is basically used to simmulate a
                        ;which would be used to drive the pwm output.
                        ;
                        ;the rtcc is set up to interrupt every 52 mS.
                        init_rtcc
0043 B00B                       movlw   00100000b       ;set up rtcc timer
0044 650A                       movfp   wreg,rtcsta     ;       /
0045 290B                       clrf    rtccl           ;clear rtcc
0046 290C                       clrf    rtcch           ;       /
0047 B080                       movlw   0x80            ;start pwm at 50%
0048 0121                       movwf   PWM_HI          ;       /
0049 2920                       clrf    PWM_LO          ;       /
004A 8107                       bsf     _rtcie          ;enable rtcc int.
004B 0002                       return
                        ;
                        ;Every rtcc interrupt, the PWM_HI&PWM_LO bytes are incremen
                        ;Only the 10 most significant bits are incremented. Once th
                        ;
                        service_rtcc
004C 8D07                       bcf     _rtcir          ;reset int flag
                                ;do a pseudo inc of the 10 bit PWM_HI, PWM_LO.
004D 8804                       bcf     _carry          ;clear carry
004E B00B                       movlw   01000000b       ;load lsb for 10 bit
004F 0F20                       addwf   PWM_LO,1        ;add to LSB
0050 9804                       btfsc   _carry          ;carry?
0051 1521                       incf    PWM_HI          ;yes then inc PWM_HI
                                ;now load the values into the Duty Cycle registers

0052 B803                       movlb   3               ;bank 3
0053 7020                       movfp   PWM_LO,pw1dcl   ;load lo value
0054 7221                       movfp   PWM_HI,pw1dch   ;load hi value
0055 B801                       movlb   1
0056 8417                       bsf     _tm1ie          ;enable tmr1 int
0057 0005                       retfie
                        ;
                        ;
                        start
0058 8406                       bsf     _glintd         ;disable interrupts
0059 E043                       call    init_rtcc       ;initailize the RTCC tmr
                                                        ;for test purposes
005A E030                       call    init_pwm10      ;initialize pwm
005B C05B               loop    goto    loop            ;spin wheels
                        ;

                                END


Errors   :    0
Warnings :    0
```

AppendixB:PWM8HI.LST

```
MPASM B0.54                                                          PAGE  1
PULSE WIDTH MODULATION 8 BIT HIGH RESOLUTION


                        TITLE   “PULSE WIDTH MODULATION 8 BIT HIGH RESOLUTION”
                        LIST            P=17C42, C=80, T=ON

                        include         “17c42.h”


0021                PWM_HI  equ     0x21
0020                PWM_LO  equ     0x20
0022                TEMP    equ     0x22
                    ;The user would generate a 16 bit value which is saved in r
                    ;locations PWM_HI and PWM_LO byte. In 8 bit hi-res mode, th
                    ;transfers the 8 bit values to the lo Duty Cycle (DC) regis
                    ;generate the required 8 bit hi-res PWM.
                    ;
                    ;
                    ;This is a short program to demonstrate how to generate PWM
                    ;8 bit resolution. Since a 10Mhz crystal was used in the te
                    ;The max. period = 256x100nS = 25.6uS or 39KHz. This progra
                    ;keeps the period constant and varies the duty cycle (which
                    ;to the most significant 10 bits of the 16 bit value PWM_LO
                    ;This program is interrupt driven, i.e. the update to the D
                    ;is done in the rtcc interrupt, which then enables the pwm
                    ;The period update is done during the pwm interrupt. The pw
                    ;ramps up from 0% to 100% duty cycle and then repeats. The
                    ;sweep takes approx. 13.3 secs.
                    ;
                    ;
                            ORG     0
0000 C063                   goto    start
                    ;
                            ORG     0x10            ;vector for rtcc interrupt

                    rtcc_int
0010 C054                   goto    service_rtcc    ;service rtcc
                    ;
                            ORG     0x0020          ;vector for pwm interrupt
                    pwm_int
0020 C046                   goto    service_pwm     ;service pwm only
                    ;
                            ORG     0x0030
                    ;
                    ;initialize internal hardware to generate the pwm output
                    init_pwm8hi
0030 B802                   movlb   2
0031 2910                   clrf    tmr1            ;clear timer 1
                                                    ;used to “drive” pwm1
0032 B062                   movlw   62              ;set period=39khz
0033 0114                   movwf   pr1             ;       /
0034 B803                   movlb   3
0035 2922                   clrf    TEMP            ;TEMP = mask for pw1dcl
0036 6A21                   movfp   PWM_HI,wreg     ;get duty cyl. hi byte
0037 190A                   rrcf    wreg            ;rotate hi through carry
0038 1922                   rrcf    TEMP            ;rotate into lo byte
0039 190A                   rrcf    wreg            ;repeat for 2nd lsb
003A 1922                   rrcf    TEMP            ;       /
003B B53F                   andlw   b’00111111'     ;mask hi bits
003C 4012                   movpf   W,pw1dch        ;save in high
003D 7022                   movfp   TEMP,pw1dcl     ;save in low
003E 2916                   clrf    tcon1           ;tmr1 inc. internally
                                                    ;as 8 bit counter
003F B011                   movlw   b’00010001'     ;start tmr1 and
0040 4017                   movpf   W,tcon2         ;enable pwm1
0041 B801                   movlb   1
```

```
0042 2917                       clrf    pie             ;clr all int. enables
0043 2916                       clrf    pir             ;clear all interrupts
0044 8307                       bsf     _peie           ;except peripheral int.
0045 0005                       retfie
                        ;
                        ;
                        ;everytime a new value is written to the PWM_HI, PWM_LO reg
                        ;tmr1 interrupts is enabled. The DC value are written just
                        ;the “pwm interrupt” is enabled. Here the new period regist
                        ;updated. In this example, period is kept constant at 62 Tc
                        service_pwm
                                ;if the period changed, write new value here.
0046 B802                       movlb   2               ;select bank 2
0047 B062                       movlw   62              ;period = 62 Tcyl.
0048 0114                       movwf   pr1             ;       /
0049 B801                       movlb   1               ;disable tmr1 int
004A 8C17                       bcf     _tm1ie          ;       /
004B 0005                       retfie
                        ;
                        ;This part of the program is basically used to simmulate a

                        ;which would be used to drive the pwm output.
                        ;
                        ;the rtcc is set up to interrupt every 52 mS.
                        init_rtcc
004C B020                       movlw   b’00100000'     ;set up rtcc timer
004D 650A                       movfp   wreg,rtcsta     ;       /
004E 290B                       clrf    rtccl           ;clear rtcc
004F 290C                       clrf    rtcch           ;       /
0050 B031                       movlw   31              ;init pwm at 50%
0051 0121                       movwf   PWM_HI          ;save in high
0052 8107                       bsf     _rtcie          ;enable rtcc int.
0053 0002                       return
                        ;
                        ;Every rtcc interrupt, the PWM_HI&PWM_LO bytes are incremen

                        ;Only the 8 most significant bits are incremented.
                        ;
                        service_rtcc
0054 8D07                       bcf     _rtcir          ;reset int flag
                                                        ;do a pseudo inc of the 8 bit PWM_HI.
0055 1521                       incf    PWM_HI          ;inc PWM_HI
                                                        ;now load the values into the Duty Cycle

0056 B803                       movlb   3               ;bank 3
0057 2922                       clrf    TEMP            ;TEMP = mask for pw1dcl
0058 6A21                       movfp   PWM_HI,wreg     ;get duty cyl. hi byte
0059 190A                       rrcf    wreg            ;rotate hi through carry
005A 1922                       rrcf    TEMP            ;rotate into lo byte
005B 190A                       rrcf    wreg            ;repeat for 2nd lsb
005C 1922                       rrcf    TEMP            ;       /
005D B53F                       andlw   b’00111111'     ;mask hi bits
005E 4A12                       movpf   wreg,pw1dch     ;save in high
005F 7022                       movfp   TEMP,pw1dcl     ;save in low
0060 B801                       movlb   1
0061 8417                       bsf     _tm1ie          ;enable tmr1 int
0062 0005                       retfie
                        ;
                        ;
                        start
0063 8406                       bsf     _glintd         ;disable interrupts
0064 E04C                       call    init_rtcc       ;initailize the RTCC tmr
                                                        ;for test purposes
0065 E030                       call    init_pwm8hi     ;initialize 8 bit pwm
0066 C066               loop    goto    loop            ;spin wheels.
                        ;
                                END
Errors   :    0
Warnings :    0
```

AppendixC:PWM8LO.LST

```
MPASM B0.54                                                          PAGE  1
PULSE WIDTH MODULATION 8 BIT LOW RESOLUTION

                        TITLE   “PULSE WIDTH MODULATION 8 BIT LOW RESOLUTION”
                        LIST    P=17C42, T=ON, C=80

                      include “17c42.h”

0021                   PWM_HI  equ     0x21
0020                   PWM_LO  equ     0x20
0022                   TEMP    equ     0x22
                    ;The user would generate a 16 bit value which is saved in r
                    ;locations PWM_HI and PWM_LO byte. In 8 bit lo-res mode, th
                    ;transfers the 8 hi-byte value directly to the PW1DCH regis
                    ;
                    ;
                    ;This is a short program to demonstrate how to generate PWM
                    ;8 bit low resolution. Since a 5.068Mhz crystal was used in
                    ;The max. period = 1024x100nS = 102.4 uS or 9.8 KHz. This p
                    ;keeps the period constant and varies the duty cycle (which
                    ;to the most significant 8 bits of the 16 bit value PWM_LO&
                    ;This program is interrupt driven, i.e. the update to the D
                    ;is done in the rtcc interrupt, which then enables the pwm
                    ;The period update is done during the pwm interrupt. The pw
                    ;ramps up from 0% to 100% duty cycle and then repeats. The
                    ;sweep takes approx. 52 secs.
                    ;
                    ;
                             ORG     0
0000 C053                    goto    start
                    ;
                             ORG     0x10            ;vector for rtcc interrupt

                    rtcc_int
0010 C04C                    goto    service_rtcc    ;service rtcc
                    ;
                             ORG     0x0020          ;vector for pwm interrupt
                    pwm_int
0020 C03E                    goto    service_pwm     ;service pwm only
                    ;
                             ORG     0x0030
                    ;
                    ;initialize internal hardware to generate the output
                    ;for 8 bit low resolution pwm
                    init_pwm8lo
0030 B802                    movlb   2
0031 2910                    clrf    tmr1            ;clear timer 1
                                                     ;used to “drive” pwm1
0032 2B14                    setf    pr1             ;set period=9.8 khz
0033 B803                    movlb   3
0034 7221                    movfp   PWM_HI,pw1dch   ;load duty cyl. hi byte
0035 2910                    clrf    pw1dcl          ;clear lo byte
0036 2916                    clrf    tcon1           ;tmr1 inc. internally
                                                     ;as 8 bit counter
0037 B011                    movlw   b’00010001'     ;start tmr1 and
0038 4A17                    movpf   wreg,tcon2      ;enable pwm1
0039 B801                    movlb   1
003A 2917                    clrf    pie             ;clr all int. enables
003B 2916                    clrf    pir             ;clear all interrupts
003C 8307                    bsf     _peie           ;except peripheral int.
003D 0005                    retfie
                    ;
                    ;
```

```
                    ;everytime a new value is written to the PWM_HI, PWM_LO reg
                    ;tmr1 interrupts is enabled. The DC value are written just
                    ;the “pwm interrupt” is enabled. Here the new period regist
                    ;updated. In this example, period is kept constant at 0xff
                    service_pwm
                           ;if the period changed, write new value here.
003E B802                    movlb   2               ;select bank 2
003F 2B14                    setf    pr1             ;period <— 0xff
0040 B801                    movlb   1               ;disable tmr1 int
0041 8C17                    bcf     _tm1ie          ;       /
0042 0005                    retfie
                    ;
                    ;This part of the program is basically used to simmulate a
                    ;which would be used to drive the pwm output.
                    ;
                    ;the rtcc is set up to interrupt every 52 mS.
                    init_rtcc
0043 B020                    movlw   b’00100000'     ;set up rtcc timer
0044 6500                    movfp   W,rtcsta        ;       /
0045 290B                    clrf    rtccl           ;clear rtcc
0046 290C                    clrf    rtcch           ;       /
0047 B080                    movlw   0x80            ;begin PWM at 50% dc
0048 0121                    movwf   PWM_HI          ;       /
0049 2920                    clrf    PWM_LO          ;       /
004A 8107                    bsf     _rtcie          ;enable rtcc int.
004B 0002                    return
                    ;
                    ;Every rtcc interrupt, the PWM_HI&PWM_LO bytes are incremen
                    ;Only the 8 most significant bits are incremented.
                    ;
                    service_rtcc
004C 8D07                    bcf     _rtcir          ;reset int flag
                                                     ;do a inc of the 8 bit PWM_HI
004D 1521                    incf    PWM_HI
                                                     ;now load the values into the Duty Cycle
004E B803                    movlb   3
004F 7221                    movfp   PWM_HI,pw1dch   ;load hi byte
0050 B801                    movlb   1
0051 8417                    bsf     _tm1ie          ;enable tmr1 int
0052 0005                    retfie
                    ;
                    ;
                    start
0053 8406                    bsf     _glintd         ;disable interrupts
0054 E043                    call    init_rtcc       ;initailize the RTCC tmr
                                                     ;for test purposes
0055 E030                    call    init_pwm8lo     ;initialize pwm
0056 C056             loop   goto    loop            ;spin wheels.
                    ;

                           END

Errors   :    0
Warnings :    0
```

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