MICROCHIP

TB025

Downloading HEX Files to PIC16F87X PICmicro® Microcontrollers

Author: Rodger Richey
Microchip Technology Inc.

INTRODUCTION

The release of the PIC16F87X devices introduces the first mid-range family of devices from Microchip Technology that has the capability to read and write to internal program memory. This family has FLASH-based program memory, SRAM data memory and EEPROM data memory. The FLASH program memory allows for a truly reprogrammable system. Table 1 shows the features of the PIC16F87X family of devices.

ACCESSING MEMORY

The read and write operations are controlled by a set of Special Function Registers (SFRs). There are six SFRs required to access the FLASH program memory:

- EECON1
- EECON2
- EEDATA
- EEDATH
- EEADR
- EEADRH

The registers EEADRH:EEADR holds the 12-bit address required to access a location in the 8K words of program memory. The registers EEDATH:EEDATA are used to hold the data values. When reading program memory, the EEPGD bit (EECON1<7>) must be set to indicate to the microcontroller that the operation is going to be on program memory. If the bit is cleared, the operation will be performed on data memory at the address pointed to by EEADR. The EEDATA register will hold the data. The EECON1 register also has bits for write enable and to initiate the read or write operation. There is also a bit to indicate a write error has occurred, possibly due to a reset condition happening while a write operation is in progress. Figure 1 shows the register map for EECON1.

The EECON2 register is not a physical register. Reading it will result in all '0's. This register is used exclusively in the EEPROM and FLASH write sequences. Listing 1 shows the code snippet to initiate a write operation on the PIC16F87X devices.

TABLE 1 PIC16F87X FAMILY FEATURES

Key Features	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
Resets	POR, BOR	POR, BOR	POR, BOR	POR, BOR
Flash Prog Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels

FIGURE 1: EECON1 REGISTER

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	—	—	—	WRERR	WREN	WR	RD
bit7							bit0

R= Readable bit
W= Writable bit
S= Settable bit
U= Unimplemented bit, read as ‘0’
- n= Value at POR reset

bit 7: **EEPGD**: Program / Data EEPROM Select bit
1 = Accesses Program memory
0 = Accesses data memory
Note: This bit cannot be changed while a write operation is in progress.

bit 6:4: **Unimplemented:** Read as '0'

bit 3: **WRERR**: EEPROM Error Flag bit
1 = A write operation is prematurely terminated
(any $\overline{\text{MCLR}}$ reset or any WDT reset during normal operation)
0 = The write operation completed

bit 2: **WREN**: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1: **WR**: Write Control bit
1 = initiates a write cycle.
The bit is cleared by hardware once write is complete.
The WR bit can only be set (not cleared) in software.
0 = Write cycle to the EEPROM is complete

bit 0: **RD**: Read Control bit
1 = Initiates an EEPROM read (read takes one cycle)
RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
0 = Does not initiate an EEPROM read

HEX FILE FORMAT

The data to be programmed into program memory will be read into the microcontroller using one of its standard interface modules: SPI, I^2C™, USART, or PSP. Probably the simplest format to send the data to the microcontroller is in the standard HEX format used by the Microchip development tools. The formats supported are the Intel HEX Format (INHX8M), Intel Split HEX Format (INHX8S), and the Intel HEX 32 Format (INHX32). The most commonly used formats are the INHX8M and INHX32 and therefore are the only formats discussed in this document. Please refer to Appendix A in the MPASM User's Guide (DS33014) for more information about HEX file formats. The difference between INHX8M and INHX32 is that INHX32 supports 32-bit addresses using a linear address record. The basic format of the hex file is the same between both formats as shown below:

```
:BBAAAATTHHHH...HHHHCC
```

Each data record begins with a 9 character prefix and always ends with a 2 character checksum. All records begin with a '**:**' regardless of the format. The individual elements are described below.

- **BB** - is a two digit hexadecimal byte count representing the number of data bytes that will appear on the line. Divide this number by two to get the number of words per line.
- **AAAA** - is a four digit hexadecimal address representing the starting address of the data record. Format is high byte first followed by low byte. The address is doubled because this format only supports 8-bits (to find the real PICmicro address, simply divide the value **AAAA** by 2).
- **TT** - is a two digit record type that will be '00' for data records, '01' for end of file records and '04' for extended address record (INHX32 only).
- **HHHH** - is a four digit hexadecimal data word. Format is low byte followed by high byte. There will be **BB**/2 data words following **TT**.
- **CC** - is a two digit hexadecimal checksum that is the two's complement of the sum of all the preceding bytes in the line record.

Since the PIC16F87X devices only have a maximum of 8K words, the linear address record '04' is ignored by the routine. The HEX file is composed of ASCII characters 0 thorough 9 and A to F and the end of each line has a carriage return and linefeed. The downloader code in the PICmicro microcontrollers must convert the ASCII characters to binary numbers to be used for programming.

PICmicro Code

The sample downloader code does not specifically use one of the interface modules on the PIC16F87X device. Instead, a routine called GetByte retrieves a single character from the HEX file over the desired interface. It is up to the engineer to write this routine around the desired interface. Another routine GetHEX8 calls GetByte twice to form a two digit hexadecimal number.

One issue that arises is how many times to reprogram a location that does not program correctly. The sample code provided simply exits the downloader routine and stores a value of 0xFF in the WREG if a program memory location does not properly program on the first attempt. The engineer may optionally add code to loop several times if this event occurs.

Still another issue that is not specifically addressed in the sample code is to prevent the downloader from overwriting its own program memory address locations. The designer must add an address check to prevent this situation from happening.

Finally, the designer must account for situations where the download of new code into the microcontroller is interrupted by an external event such as power failure or reset. The system must be able to recover from such an event. This is not a trivial task, is very system dependent, and is therefore left up to the designer to provide the safeguards and recovery mechanisms.

Another error that could happen is a line checksum error. If the calculated line checksum does not match the line checksum from the HEX file, a value of 1 is returned in WREG. The part of the routine that calls the downloader should check for the errors 0xFF (could not program a memory location) and 1. If program memory is programmed correctly and no errors have been encountered, the downloader routine returns a 0 in WREG to indicate success to the calling routine. Figure 2 shows the flowchart for the downloader routines. Listing 2 shows the complete listing for the downloader code.

The routine ASCII2HEX converts the input character to a binary number. The routine does not provide any out of range error checking for incoming characters. Since the only valid characters in a HEX file are the colon (:), the numbers 0 through 9 and the letters A through F, the routine can be highly optimized. It first subtracts 48 from the character value. For the ASCII numbers 0 through 9, this results in a value from 0 to 9. If the character is A through F, the result is a number greater than 15. The routine checks to see if the upper nibble of the result is 0. If not 0, then the original value was A through F and the routine now subtracts an additional 43 from the character resulting in the binary values 10 through 15. The colon is not accounted for in this routine because the main part of the downloader code uses it as a line sync.

LISTING 1: FLASH WRITE SEQUENCE

```
bsfSTATUS,RP1     ; Bank2
bcfSTATUS,RP0
movfAddrH,W        ; Load address into
movwfEEADRH         ; EEADRH:EEADR
movfAddrL,W
movwfEEADR
bsfSTATUS,RP0     ; Bank3
bsfEECON1,EEPGD   ; Set for Prog Mem
bsfEECON1,RD      ; read operation
bcfSTATUS,RP0     ; Bank2
nop
movfEEDATA,W       ; Data is read
…                      ; user can now
movfEEDATH,W       ; access memory
…
```

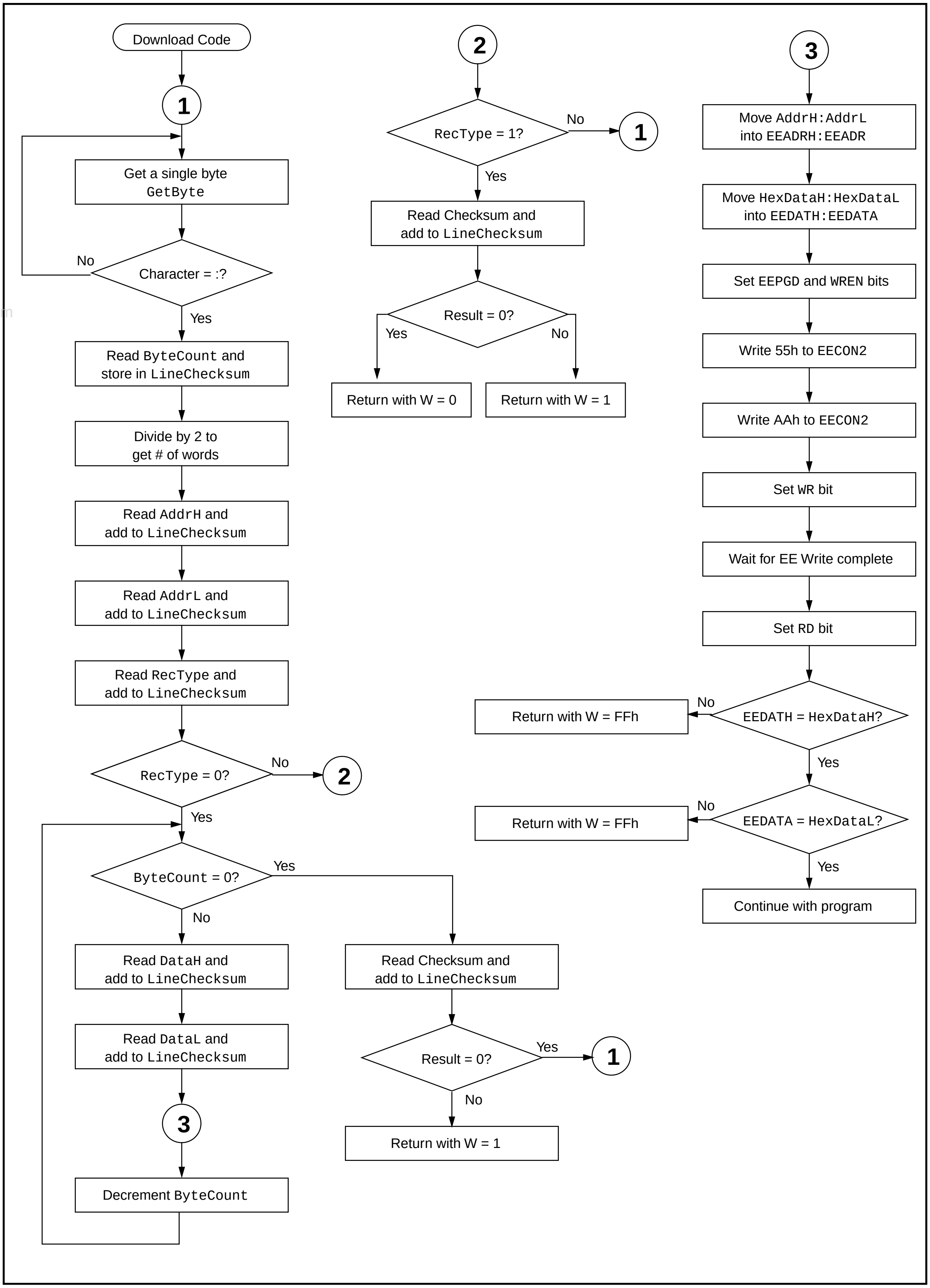
LISTING 2: HEX DOWNLOAD CODE WRITTEN FOR MPASM

```
        list p=16f877
        #include "c:\progra~1\mplab\p16f877.inc"


DownloadCode                                ;Uses USART to receive data from PC
        banksel     RCREG
DCStart
        call        GetByte
        movlw       ':'                     ;Wait for colon
        subwf       RCREG,W
        btfss       STATUS,Z
        goto        DCStart

        call        GetHex8                 ;Read byte count
        movwf       ByteCount               ;Store in register
        movwf       LineChecksum            ;Store in line checksum
        bcf         STATUS,C
        rrf         ByteCount,F             ;Divide byte counter by 2 to get words

        call        GetHex8                 ;Read high byte of 16-bit address
        movwf       AddrH
        addwf       LineChecksum,F          ;Add high byte to line checksum
        call        GetHex8                 ;Read low byte of 16-bit address
        movwf       AddrL
        addwf       LineChecksum,F          ;Add low byte to line checksum

        call        GetHex8                 ;Read record type
        movwf       RecType
        addwf       LineChecksum,F          ;Add to line checksum

DataRec                                     ;Data reception
        movf        RecType,F               ;Check for data record (0h)
        btfss       STATUS,Z
        goto        EndOfFileRec            ;Otherwise check for EOF
DRLoop
        movf        ByteCount,F             ;Check for bytecount = 0
        btfsc       STATUS,Z
        goto        DRCkChecksum            ;If zero, goto checksum validation
        call        GetHex8                 ;Read lower byte of data (2 characters)
        movwf       HexDataL                ;Add received data to checksum
        addwf       LineChecksum,F
        call        GetHex8                 ;Read upper byte of data (2 characters)
        movwf       HexDataH                ;Add received data to checksum
        addwf       LineChecksum,F

WriteDataSequence                           ;Write sequence to internal prog. mem FLASH
        banksel     EEADRH
        movf        AddrH,W                 ;Write address to EEADRH:EEADR registers
        movwf       EEADRH
        movf        AddrL,W
        movwf       EEADR
        movf        HexDataH,W              ;Write data to EEDATH:EEDATA registers
        movwf       EEDATH
        movf        HexDataL,W
        movwf       EEDATA
        banksel     EECON1                  ;Write sequence
        bsf         EECON1,EEPGD            ;Set EEPGD to indicate program memory
        bsf         EECON1,WREN             ;Enable writes to memory
        bcf         INTCON,GIE              ;Make sure interrupts are disabled
        movlw       0x55                    ;Required write sequence
        movwf       EECON2
        movlw       0xaa
        movwf       EECON2
        bsf         EECON1,WR               ;Start internal write cycle
        nop
```

```
        nop
        bcf         EECON1,WREN        ;Disable writes

        banksel     EECON1             ;Read sequence
        bsf         EECON1,EEPGD       ;Set EEPGD to indicate program memory
        bsf         EECON1,RD          ;Enable reads from memory
        bcf         STATUS,RP0
        nop
        movf        EEDATH,W           ;Compare memory value to HexDataH:HexDataL
        subwf       HexDataH,W
        btfss       STATUS,Z
        retlw       0xff               ;If upper byte not equal, return FFh
        movf        EEDATA,W           ; to indicate programming failure
        subwf       HexDataL,W
        btfss       STATUS,Z
        retlw       0xff               ;If lower byte not equal, return FFh
                                       ; to indicate programming failure
        incf        AddrL,F            ;Increment address for next iteration
        btfsc       STATUS,Z
        incf        AddrH,F
        decf        ByteCount,F        ;Decrement byte count
        goto        DRLoop             ;Go back to check for ByteCount = 0

DRCkChecksum                           ;Checksum verification
        call        GetHex8            ;Read in checksum
        addwf       LineChecksum,W     ;Add to calculated checksum
        btfss       STATUS,Z           ;Result should be 0
        retlw       1                  ; If not return 1 to indicate checksum fail
        goto        DCStart            ;Do it again

EndOfFileRec                          ;End of File record (01h)
        decf        RecType,W         ;If EOF record, decrement should = 0
        btfss       STATUS,Z
        goto        DCStart           ;Not valid record type, wait for next :
        call        GetHex8           ;Read in checksum
        addwf       LineChecksum,W    ;Add to calculated checksum
        btfss       STATUS,Z          ;Result should be 0
        retlw       1                 ; If not return 1 to indicate checksum fail
        retlw       0                 ;Otherwise return 0 to indicate success


GetByte
; Insert your code here to retrieve a byte of data from
; the desired interface. In this case it is the USART on F877.
;clear CTS
;       banksel    PIR1
;GH4Waitbtfss      PIR1,RCIF
;       goto       GH4Wait
;set CTS
        nop
        banksel     RCREG
        movf        RCREG,W
        return

GetHex8                               ;This function uses the USART
        call        GetByte           ;Read a character from the USART
        call        ASCII2Hex         ;Convert the character to binary
        movwf       Temp              ;Store result in high nibble
        swapf       Temp,F

        call        GetByte           ;Read a character from the USART
        call        ASCII2Hex         ;Convert the character to binary
        iorwf       Temp,F            ;Store result in low nibble
        movf        Temp,W            ;Move result into WREG
        return
```

```
ASCII2Hex                                    ;Convert value to binary
        movwf       Temp1                    ;Subtract ASCII 0 from number
        movlw       '0'
        subwf       Temp1,F
        movlw       0xf0                     ;If number is 0-9 result, upper nibble
        andwf       Temp1,W                  ; should be zero
        btfsc       STATUS,Z
        goto        ASCIIOut
        movlw       'A'-'0'-0x0a             ;Otherwise, number is A - F, so
        subwf       Temp1,F                  ;subtract off additional amount
ASCIIOut
        movf        Temp1,W                  ;Value should be 0 - 15
        return

        end
```

FIGURE 2: FLOWCHART



MICROCHIP

WORLDWIDE SALES AND 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.microchip.com

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: 630-285-0071 Fax: 630-285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit
Microchip Technology Inc.
42705 Grand River, Suite 201
Novi, MI 48375-1727
Tel: 248-374-1888 Fax: 248-374-2874

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

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

AMERICAS (continued)

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

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

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

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700 Fax: 86 21-6275-5060

ASIA/PACIFIC (continued)

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

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-1189-21-5858 Fax: 44-1189-21-5835

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
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 Müchen, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

9/29/98

DNV Certification, Inc. USA
ACCREDITED REGISTRAR ANSI-RAB
DNV
DNV MSC The Netherlands Accredited by the RvA
ISO 9001 REGISTERED FIRM

Microchip received ISO 9001 Quality System certification for its worldwide headquarters, design, and wafer fabrication facilities in January, 1997. Our field-programmable PICmicro® 8-bit MCUs, Serial EEPROMs, related specialty memory products and development systems conform to the stringent quality standards of the International Standard Organization (ISO).