

## Bootloader for dsPIC30F/33F and PIC24F/24H Devices

*Author: Leonard Elevich and Veena Kudva  
Microchip Technology, Inc.*

### INTRODUCTION

The bootloader for dsPIC30F/33F and PIC24H/24F devices is used to load and run your application on the target device. The bootloader consists of two applications:

- Target side bootloader application which must be programmed into dsPIC30F/33F or PIC24F/24H program memory prior to bootloader operation.
- Host PC bootloader application which communicates with the target side bootloader.

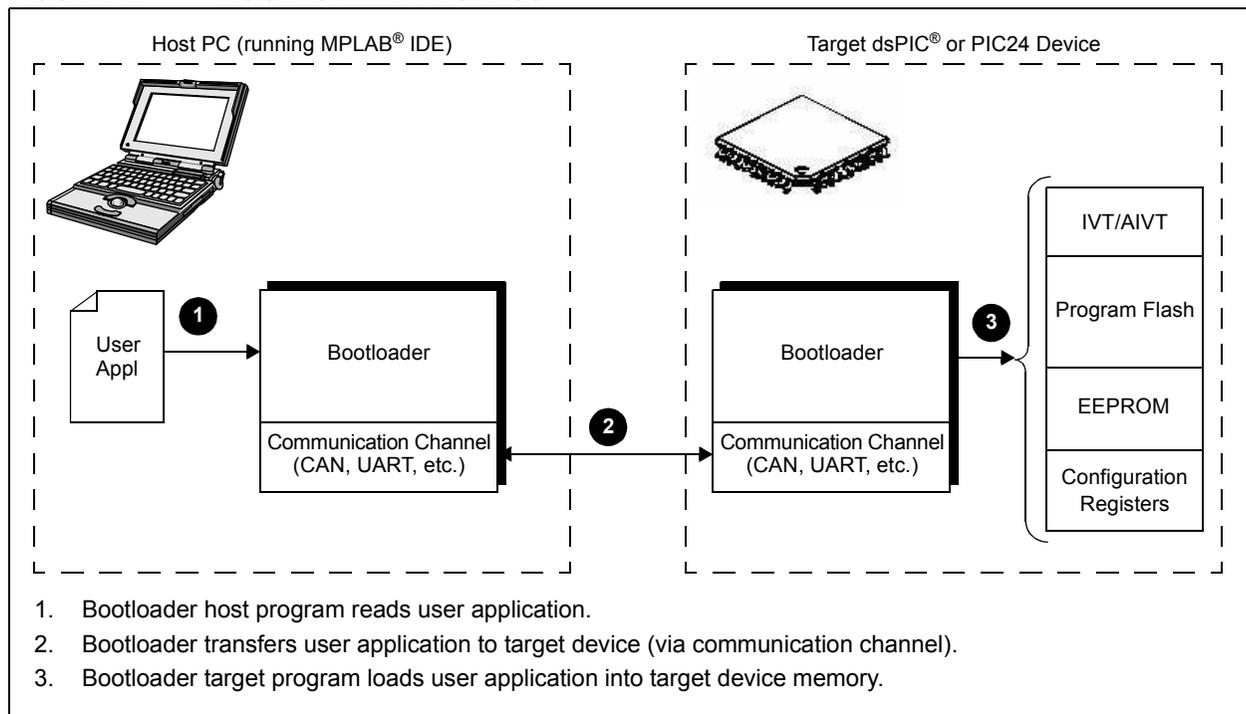
The bootloader parses the program HEX file and then copies it into the appropriate program and EEPROM memory (if present) on the target device via the communication channel (UART, CAN, etc.). Figure 1 illustrates this process.

### SYSTEM CONCEPT

The bootloader target application is located in the dedicated program memory region, starting at address 0x100 (for dsPIC30F devices) or 0x400 (for all other device families). On start-up, the bootloader reads program memory address 0x600 (for dsPIC30F family) or address 0xC00 (for all other device families), which contains a bootloader delay value. If the bootloader fails to detect UART activity within the time period specified by the delay value, it suspends itself and transfers execution to the user application located at program memory address 0x602 (for dsPIC30F family) or address 0xC02 (for all other families). On the other hand, if the bootloader detects UART activity before it suspends itself, it programs both EEPROM (if present) and program memories with the data it receives from the bootloader host application via UART interface.

The bootloader host application parses the HEX file containing the user application (generated by MPLAB® IDE) and sends this data to the bootloader target application via UART. The bootloader host application also supports additional features, such as read of program and EEPROM memories.

**FIGURE 1: BOOTLOADER PROCESS**



# AN1094

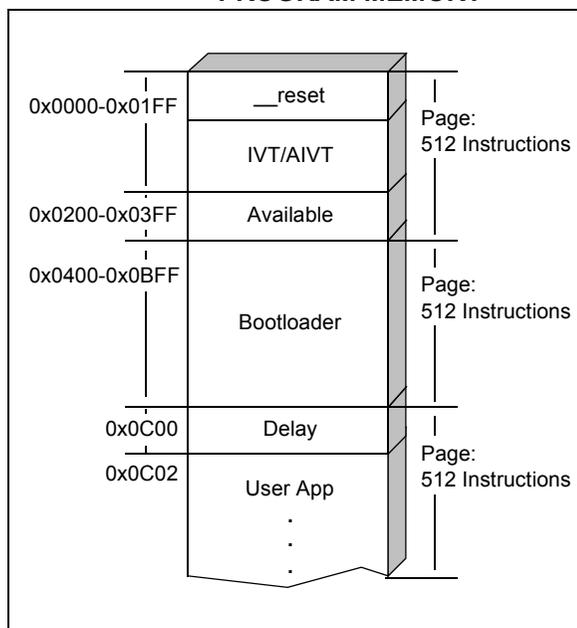
## DEVICE MEMORY USAGE

Although the target side bootloader application requires minimal memory, target side architecture constrains how the bootloader and user application fit into memory.

Before Flash memory can be programmed, the bootloader must first erase it. The bootloader can erase Flash memory, either by mass erasing all of Flash memory at once, or by erasing individual memory pages (which are 512 instructions long).

Figure 2 illustrates memory organization for dsPIC33F, PIC24H and PIC24F targets.

**FIGURE 2: dsPIC33F AND PIC24F/24H PROGRAM MEMORY**

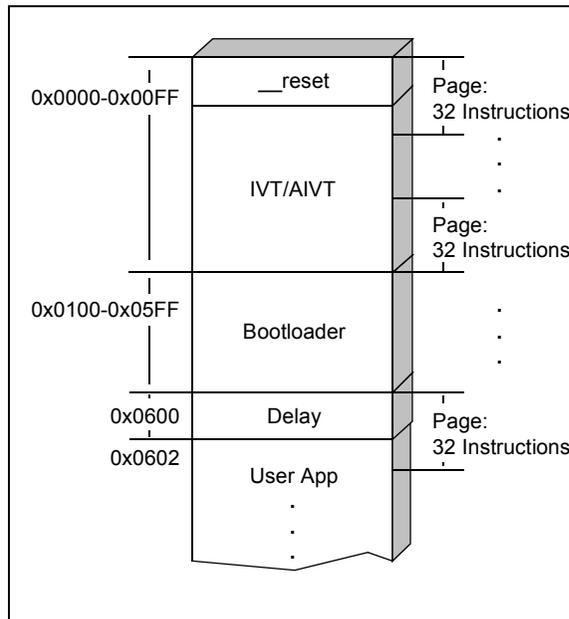


The interrupt tables (IVT/AIWT) use memory space up to address 0x1FE. The bootloader can not be placed immediately following this address because erasing the first Flash memory page also erases the bootloader. Therefore, the bootloader must start at address 0x400, which leaves a "hole" of unused memory from 0x200 through 0x3FE. However, this available memory can be used for your user application.

Also, because of this Flash memory page restriction, the user application can not be placed immediately after the bootloader. It must be pushed to the beginning of the next Flash memory page (address 0xC00). Starting at that address, the application specifies the bootloader delay value, followed by the actual application code at address 0xC02.

Flash memory pages are smaller on dsPIC30F devices, so the bootloader and user application locations are different, as shown in Figure 3.

**FIGURE 3: dsPIC30F PROGRAM MEMORY**



## DEVICE PERIPHERAL USAGE

The target side bootloader application uses program memory from address 0x100 to 0x600 (inclusive) on dsPIC30F devices and 0x400 to 0xC00 (inclusive) on dsPIC33F and PIC24H/24F devices. It also uses Reset vectors from the Interrupt Vector Table (IVT). The target side bootloader application uses these peripherals:

- UART
- Timer

## PERFORMANCE

Maximum measured performance for the bootloader on the dsPIC33F target is 8.1 Kbytes/second.

## FILES

The bootloader application is organized into two subdirectories:

- Target Side: ...\`Bootloader\target`
- Host Side: ...\`Bootloader\host`

The target side bootloader application is developed with MPLAB IDE tools and consists of the following files:

- Project Files:
  - `16-bit Flash Programmer.mcp`
  - `16-bit Flash Programmer.mcw`
- Main Program (performs all main tasks, such as initialization and communication):
  - `main.c`
- Support File (contains memory routines, such as erase and write):
  - `memory.s`
- Test Application Files (located in the `.\test` directory)

The bootloader host application is developed with Visual C++<sup>®</sup> development system tools and consists of the following files:

- Project Files:
  - `16-Bit Flash Programmer.vcproj`
  - `16-Bit Flash Programmer.suo`
  - `16-Bit Flash Programmer.sln`
  - `16-Bit Flash Programmer.ncb`
- Main Program:
  - `16-Bit Flash Programmer.cpp`
  - `16-Bit Flash Programmer.h`
- Command Line Parsing Class (used to parse and validate command line arguments):
  - `cmd.cpp`
  - `cmd.h`
- Memory Class (used to hold parsed HEX data):
  - `mem.cpp`
  - `mem.h`
- Executable Files (located in the `.\Debug` directory)

## BUILDING AND LOADING THE TARGET SIDE BOOTLOADER

The target side bootloader can be built in two modes:

- Debug mode
- Stand-Alone mode

## Debug Mode

If you want to debug the bootloader code with MPLAB<sup>®</sup> ICD 2 tools, you should use the Debug mode.

To build and load the target side bootloader in Debug mode:

**Note:** See “Target Bootloader Configuration” on page 4 for jumper settings.

1. Open the project file:
  - `16-bit Flash Programmer.mcp`
2. From the **Project** menu, select the **Build** command.
3. Connect the target board to the host computer via MPLAB ICD 2 (see the development board user’s guide for more detailed instructions).
4. From the **Debugger** menu, choose **Select Tool**, then click on **ICD2**.
5. From the **Debugger** menu, select **Program**, then click on **Run**.

At this point, the progress bar should be present, indicating that the target is running.

## Stand-Alone Mode

If you don’t want to debug the bootloader code, you should build and load the bootloader in Stand-Alone mode.

To build and load the target side bootloader in Stand- Alone mode:

**Note:** See “Target Bootloader Configuration” on page 4 for jumper settings.

1. Open the project file:
  - `16-bit Flash Programmer.mcp`
2. From the **Project** menu, select the **Build** command.
3. Connect the target board to the host computer via MPLAB ICD 2 (see development board user’s guide for more detailed instructions).
4. From the **Programmer** menu, choose **Select Tool**, then click on **ICD2**.
5. From the **Programmer** menu, select **Program**.
6. Reset the target board.

At this point, the bootloader reads the delay value of 0xFF (since Flash was erased by the MPLAB IDE tools), and waits for UART activity.

The bootloader host application can be rebuilt with the project file, `16-Bit Flash Programmer.vcproj`; however, this is not necessary as an executable file is provided in the `.\Debug` directory.

# AN1094

## TARGET BOOTLOADER CONFIGURATION

The bootloader target side application was developed and tested on the following hardware:

**TABLE 1: DEVELOPMENT HARDWARE**

Device Family	Development Board Used	CPU/PIM Used	dsPIC®/PIC® Device Peripherals Used	dsPIC®/PIC® Device Memory Used
dsPIC30F	dsPICDEM™ 2	dsPIC30F4011	UART1 with ALT pins, Timer2, Timer3	0x100-0x600
dsPIC33F	Explorer 16	dsPIC33FJ256GP710	UART2, Timer2, Timer3	0x100-0xC00
PIC24H	Explorer 16	PIC24HJ256GP610	UART2, Timer2, Timer3	0x100-0xC00
PIC24F	Explorer 16	PIC24FJ128GA010	UART2, Timer2, Timer3	0x100-0xC00

The bootloader can be reconfigured to use different sets of UART and timer peripherals by modifying initialization code.

**TABLE 2: JUMPER CONFIGURATIONS FOR dsPICDEM™ 2 DEVELOPMENT BOARD**

Component	Header		Socket U2A1	
	No.	Setting	Pin	Device Functions
Jumper JP1	JP1	Jumper	—	5 VDC jumper installed
Jumper JP2	JP2	Jumper	—	5 VDC jumper installed
Selector Switch	S2	M ALL ON	—	—
Selector Switch	S3	M ALL ON	—	—
Selector Switch	S4	OFF	—	Not used in this configuration
PGM U3	H11	Open	—	—
CAN Tx	H2	Open	—	—
CAN Rx	H2	Open	—	—
UART1 Tx	H3	Open	—	—
UART1 Rx	H3	Open	—	—
Alternate UART1 Tx	H4	M ALL	—	EMUD1/SOSCI/T2CK/U1ATX/CN1/RC13
Alternate UART1 Rx	H4	M ALL	—	EMUC1/SOSCO/T1CK/U1ARX/CN0/RC14
UART2 Tx	H5	Open	—	—
UART2 Rx	H5	Open	—	—
Temperature Sensor	H10	M ALL	5	AN3/INDX/CN5/RB3
Potentiometer	H13	M ALL	4	AN2/SS1/CN4/RB2
Switch S5	H6	M ALL	17	FLTA/INT0/SCK1/OCFA/RE8
Switch S6	H7	M ALL	23	EMUC2/OC1/IC1/INT1/RD0
LED D3	H12	M D3	2	EMUD3/AN0/VREF+/CN2/RB1
LED D4	H12	M D4	3	EMUC3/AN1/VREF-/CN3/RB1
LCD – SPI Clock	H1	M40	24	FLTA/INT0/SCK1/OCFA/RE8
LCD – SPI Data	H1	M40	25	PGD/EMUD/U1TX/SDO1/SCL/RF3

**TABLE 3: JUMPER CONFIGURATION FOR EXPLORER 16 DEVELOPMENT BOARD**

Component	Setting
S2	PIM
J7	PIC24
JP2	On

The bootloader delay period is configured by loading a value (in seconds) into the program memory location shown in Table 4.

**TABLE 4: BOOTLOAD DELAY CONFIGURATIONS**

Device Family	Program Memory Address for Delay Value
dsPIC30F	0x600
dsPIC33F	0xC00
PIC24H	0xC00
PIC24F	0xC00

Valid bootloader delay values are shown in Table 5.

**TABLE 5: VALID DELAY VALUES**

Delay Value (Seconds)	Result
0	Suspend bootloader and transfer execution to the user application.
1-254	Wait specified number of seconds for HEX file transfer. If no serial communication is detected before the delay time has expired, suspend bootloader and transfer execution to the user application.
255	Wait forever for HEX file transfer.

# AN1094

## REQUIREMENTS FOR A USER APPLICATION

The following requirements apply to any application intended to be loaded by the bootloader:

- Application can not place code into memory space reserved by the bootloader.
- The user's application must fit within memory space of the target device.
- Bootloader delay must be specified for subsequent bootloader executions.

To satisfy these requirements, the corresponding user application's linker script (.GLD file) must be modified to specify the application address and designate the bootloader delay period.

### .GLD File Modifications for dsPIC30F Devices

For dsPIC30F devices, the .GLD file is modified to place the user application at address 0x602 and provide a time-out value for the bootloader.

#### EXAMPLE 1:

```
program (xr) : ORIGIN = 0x600, LENGTH = ((16K * 2) - 0x600)
__CODE_BASE = 0x600; /* Handles, User Code, Library Code */

/*
** User Code and Library Code
*/
.text __CODE_BASE :
{
    SHORT(0x0A); /* Bootloader timeout in sec */
    *(.handle);
    *(.libc) *(.libm) *(.libdsp);
    *(.lib*);
    *(.text);
} >program
```

### .GLD File Modifications for dsPIC33F and PIC24F/24H Devices

For dsPIC33F and PIC24F/24H devices, the .GLD file is modified to place the user's application at address 0xC02 and provide a time-out value for the bootloader.

#### EXAMPLE 2:

```
program (xr) : ORIGIN = 0xC00, LENGTH = 0x29E00
__CODE_BASE = 0xC00; /* Handles, User Code, Library Code */

/*
** User Code and Library Code
*/
.text __CODE_BASE :
{
    SHORT(0x0A); /* Bootloader timeout in sec */
    *(.handle);
    *(.libc) *(.libm) *(.libdsp);
    *(.lib*);
    *(.text);
} >program
```

## PC HOST BOOTLOADER EXECUTION

PC host bootloader application can be executed from the command line. It has the following interface:

### EXAMPLE 3:

```
Usage: "16-Bit Flash Programmer.exe" -i interface [-bpe] hexfile
Options:
  -i
      specifies serial interface name such as COM1, COM2, etc
  -b
      specifies baud rate for serial interface. Default is 115200 bps
  -p
      read program Flash. Must provide address to read in HEX format:
                                          -p 0x000100
  -e
      read EEPROM. Must provide address to read in HEX format:
                                          -e 0x7FFC00
```

**Note:** Both the PC and target side bootloader use a default baud rate of 115200 bps for applications on dsPIC30F, dsPIC33F and PIC24H families, so the “-b” option does not need to be explicitly specified for those devices. However, the target side of PIC24F applications is configured for 38400 bps, so the PC host bootloader must be executed with the “-b 38400” option.

The following example illustrates how to program the `app.hex` file into the target device.

### EXAMPLE 4:

```
16-Bit Flash Programmer.exe -i COM1 apps.hex
```

# AN1094

---

## LOADING USER APPLICATION WITH THE BOOTLOADER

Use the following procedure to load the user application into the target device with the 16-bit bootloader:

1. Configure the target side bootloader as described in “**Target Bootloader Configuration**” on page 4.
2. Build and load the target side bootloader as described in “**Building and Loading the Target Side Bootloader**” on page 3.

3. Connect the serial cable between the PC host and the target hardware.
4. Press the Reset button on the target hardware.
5. Use the PC host bootloader as described in “**PC Host Bootloader Execution**” on page 7.

If loading is successful, the command line window will display results similar to this:

### EXAMPLE 5:

```
Reading Target Device ID... Found dsPIC30F4011 (ID: 0x0101)

Reading HexFile.
Reading Target

Programming Device ..... Done.
```

If any errors are discovered during execution, the bootloader will stop and display an error message (see **Appendix A: “Error Messages”** on page 9).

## APPENDIX A: ERROR MESSAGES

The PC host bootloader detects and displays the following errors:

```
assert(pReadPMAAddress[0] == '0' && pReadPMAAddress[1] == 'x')
```

This error indicates that the program memory read address specified with option `-p` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadPMAAddress[2]))
```

This error indicates that the program memory read address specified with option `-p` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadPMAAddress[3]))
```

This error indicates that the program memory read address specified with option `-p` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadPMAAddress[4]))
```

This error indicates that the program memory read address specified with option `-p` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadPMAAddress[5]))
```

This error indicates that the program memory read address specified with option `-p` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadPMAAddress[6]))
```

This error indicates that the program memory read address specified with option `-p` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadPMAAddress[7]))
```

This error indicates that the program memory read address specified with option `-p` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(pReadEEAddress[0] == '0' && pReadEEAddress[1] == 'x')
```

This error indicates that the EEPROM memory read address specified with option `-e` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadEEAddress[2]));
```

This error indicates that the EEPROM memory read address specified with option `-e` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadEEAddress[3]))
```

This error indicates that the EEPROM memory read address specified with option `-e` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadEEAddress[4]))
```

This error indicates that the EEPROM memory read address specified with option `-e` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadEEAddress[5]))
```

This error indicates that the EEPROM memory read address specified with option `-e` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadEEAddress[6]))
```

This error indicates that the EEPROM memory read address specified with option `-e` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(isxdigit(pReadEEAddress[7]))
```

This error indicates that the EEPROM memory read address specified with option `-e` did not conform to the format, `0xAAAAAA`, where `AAAAAA` is the program memory address to read.

```
assert(OpenConnection(&ComDev, pInterfaceName, pBaudRate) != NULL)
```

This error indicates that the bootloader couldn't initialize the serial device on the host PC.

# AN1094

---

## `assert(0)`

This error will be preceded by the following error message, which indicates that the HEX file to be loaded contains a bad address:

```
"Bad Hex file: 0xAAAAAA out of range"
```

## `assert(!"Unknown hex record type\n")`

This error indicates that the HEX file to be loaded contains a bad record type (see **Appendix B: "Hex File Format"** for a detailed description of the HEX file format).

## `assert(bDeviceFound == TRUE)`

This error indicates that the bootloader didn't recognize the device ID, which was read from the target CPU.

## `assert(Family != dsPIC33F)`

This error indicates that the user-executed EEPROM read command via the `-e` option is on an architecture that doesn't have EEPROM.

## `assert(Family != PIC24H)`

This error indicates that the user-executed EEPROM read command via the `-e` option is on an architecture that doesn't have EEPROM.

## `assert(GetLastError() == ERROR_IO_PENDING)`

This error indicates that the write to serial port has failed.

## `assert(ErrorFlags == 0);`

This error indicates that the read from serial port has failed.

## `assert(SetCommTimeouts(*pComDev, &CommTimeOuts) == TRUE)`

This error indicates that the bootloader couldn't initialize the serial device on the host PC.

## `assert(GetCommState(*pComDev, &Dcb) == TRUE)`

This error indicates that the bootloader couldn't initialize the serial device on the host PC.

## `assert(SetCommState(*pComDev, &Dcb) == TRUE)`

This error indicates that the bootloader couldn't initialize the serial device on the host PC.

## `assert (!"Unknown memory type");`

This error indicates that the data to be sent to the target belongs to an unknown memory type.

---

## APPENDIX B: HEX FILE FORMAT

The bootloader processes the standard HEX format used by the Microchip development tools. The formats supported are the Intel HEX 32 format (INHX32). Please refer to **Appendix A “Instruction Sets”** in the “*MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User’s Guide*” (DS33014) for more information about HEX file formats.

The basic format of the HEX file is:

```
:BBAAAATTHHHH...HHHCC
```

Each data record begins with a 9-character prefix and always ends with a 2-character checksum. All records begin with ':' 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 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. The format is high byte first, followed by low byte. The address is doubled because this format only supports 8 bits. Divide the value by two to find the real PIC® device address.
- **TT** – is a two-digit record type that will be '00' for data records, '01' for End-of-File (EOF) records and '04' for extended address records (INHX32 only).
- **HHHH** – is a four-digit hexadecimal data word. The 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.

# AN1094

---

NOTES:

---

---

**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 provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

**Trademarks**

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, PS logo, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2007, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, 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.*

**QUALITY MANAGEMENT SYSTEM  
CERTIFIED BY DNV  
== ISO/TS 16949:2002 ==**



---

---

## WORLDWIDE SALES AND SERVICE

---

---

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://support.microchip.com>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Kokomo**  
Kokomo, IN  
Tel: 765-864-8360  
Fax: 765-864-8387

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**  
Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**  
Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Fuzhou**  
Tel: 86-591-8750-3506  
Fax: 86-591-8750-3521

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Shunde**  
Tel: 86-757-2839-5507  
Fax: 86-757-2839-5571

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xian**  
Tel: 86-29-8833-7250  
Fax: 86-29-8833-7256

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-4182-8400  
Fax: 91-80-4182-8422

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Yokohama**  
Tel: 81-45-471-6166  
Fax: 81-45-471-6122

**Korea - Gumi**  
Tel: 82-54-473-4301  
Fax: 82-54-473-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Penang**  
Tel: 60-4-646-8870  
Fax: 60-4-646-5086

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-572-9526  
Fax: 886-3-572-6459

**Taiwan - Kaohsiung**  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820