
An Introduction to USB Descriptors with a Game Port to USB Game Pad Translator Example

*Author: Reston Condit
Microchip Technology Inc.*

INTRODUCTION

This Technical Brief demonstrates the translation of a game port game pad to a USB game pad using the PIC16C765, Microchip's low-speed USB PICmicro[®] microcontroller (MCU). The purpose of this Technical Brief is not only to show the translation of a game pad to USB, but also to show how to successfully develop a USB peripheral using the PIC16C765. An understanding of USB descriptors is the foundation for successful USB peripheral development.

Note: This Technical Brief is the first in a series of five technical briefs. This series is meant to familiarize developers with USB. For the best understanding of USB, read the briefs in order: TB054, TB055, TB056, TB057 and TB058.

USB Basics

All USB communication takes the form of frames sent over the USB bus. Frames are one-millisecond increments in which the host schedules what device endpoints it will communicate with. At the scheduled time, the host and device send one another requests, or data, in the form of packets, which are limited in length to eight bytes for low-speed devices. All packets are sent to and received by the device via endpoints. Endpoints are buffers where a device either puts data to wait for a chance to be sent to the host, or where data received from the host is stored until the device has a chance to access the data and utilize it. Low-speed devices have two endpoints that can be configured by the device as IN or OUT (with respect to the host). These are Endpoints 1 and 2. Endpoint 0 is reserved for control transfers between the host and device. This is the bidirectional avenue through which the host and device share administrative data. Enumeration is the process in which a peripheral describes what type of device it is to the host, and occurs via Endpoint 0.

USB is a master-slave protocol. In other words, the host (i.e., PC) directs all communications to and from the peripherals. Peripherals do not send information to the host unless the host requests the information. How does a peripheral tell the host what type of device it is when it is first connected to the host? The answer lies in the use of "descriptors," as described in the following section.

Descriptors

GENERAL

USB descriptors tend to be the biggest stumbling block for developers that have been recently introduced to USB. The purpose of descriptors is to communicate the identity of a particular peripheral to the host. For instance, the game pad in this technical brief communicates to the host that it is a two axis, six-button game pad that sends data to the host via Endpoint 1. A device does not volunteer this information to the host, rather the host requests this information when it detects that a new device has been attached to the USB bus. As mentioned before, the process in which the host requests and receives a device's descriptors is called "enumeration."

Peripherals have more than one descriptor. Each new descriptor progressively provides the host more information about the peripheral or about other descriptors to follow. Descriptors can be thought of as a hierarchy. The first descriptor, the device descriptor, is very general and conveys the most basic information about the device. The next descriptor is more specific, and so on, until the host finally gains all the information it needs to communicate effectively with the device.

Game pads fall under the Human Interface Device (HID) class. Mice, keyboards and LED displays are all examples of other devices that fall into the HID class. The HID class is unique in that driver support is supplied automatically by Windows[®] (Windows 98 second edition and newer) and the Macintosh[®] operating systems. These operating systems will custom build a driver whenever they detect a new HID peripheral. The driver is constructed from the data format conveyed in the report descriptor received from that device. Developing within the HID class specification is the easiest way to learn how USB works because the developer doesn't have to be concerned with writing a driver at the host end.

TYPES

Descriptors general to USB are discussed in Chapter 9 of the USB Specification v1.1. The general descriptors that pertain to all USB devices are the following:

- **Device descriptor** – describes the most general information about a USB device. For instance, it communicates to the host the product and vendor ID numbers. It tells the host how many configurations the device has, and how many configuration descriptors the host must request from the device. A device can have only one device descriptor.
- **Configuration descriptor** – describes information about a specific device configuration. Included in the configuration descriptor, is the number of interfaces under the specific configuration. In essence, an interface is a feature, therefore, a configuration can be viewed as a collection of peripheral features. Multiple configurations can exist for one device. For instance, if a Power Save mode is desired for a part, it may have two configurations, Normal mode and Power Save mode.
- **Interface descriptor** – tells how many endpoints a device feature uses. It also declares the class identity of a device. For the game pad, the class identity is HID.
- **Endpoint descriptor** – describes the properties of an endpoint. These properties are namely whether the endpoint is an IN or OUT endpoint and what the endpoint number is. Every endpoint specified in an interface has its own endpoint descriptor.

Descriptors specific to the HID class are discussed in the “*Device Class Definition for HID*” document. The HID descriptors that pertain to describing the game pad are the following:

- **HID descriptor** – for a Human Interface Device, a HID descriptor immediately follows the interface descriptor and precedes the endpoint descriptor(s). This is done because the HID descriptor may be associated with more than one endpoint, and as a result, is higher in the “descriptor hierarchy” than endpoint descriptors. A HID descriptor identifies additional descriptors specific to the HID class, namely report descriptors and physical descriptors.

Note: The game pad example does not have a physical descriptor associated with it. Physical descriptors are never necessary, they are used to describe the physical aspects and appearance of a device mainly for an engineer's reference.

Report descriptor – specifies the data format for a device. The information that each bit represents in a data packet is defined in the report descriptor. For the game pad, certain bits correspond to the logic output of its buttons. Other bit groupings correspond to the X and Y-axis positions of the directional pad.

Note: The *USB Implementers Forum* (www.usb.org) is the main web location for obtaining the USB Specification, Device Class Definition for HID, HID Usage Tables, and all other defining documents pertaining to USB. “*USB Complete*”, written by Jan Axelson, presents the USB protocol in a easy to understand format.

The Game Pad Report

The device, configuration, interface, HID and endpoint descriptors are relatively straightforward to create. For these descriptors, all that is necessary is to create the descriptors according to the definitions in the USB or HID specifications. The report descriptor, however, is not straightforward. There is no set format for a report descriptor since it is describing the data format for a peripheral, which can be very different from one peripheral to another. The best way to learn how to create a report descriptor is to walk through example report descriptors that have been proven to work. Example 1 shows the report descriptor for the game pad.

EXAMPLE 1: USB GAME PAD REPORT DESCRIPTOR

```

0x05, 0x01      ;USAGE_PAGE (Generic Desktop)
0x09, 0x05      ;USAGE (Gamepad)
0xA1, 0x01      ;COLLECTION (Application)
0x09, 0x01      ;  USAGE (Pointer)
0xA1, 0x00      ;  COLLECTION (Physical)
0x09, 0x30      ;    USAGE (X)
0x09, 0x31      ;    USAGE (Y)
0x15, 0x00      ;    LOGICAL_MINIMUM (0)
0x26, 0xFF, 0x00 ;    LOGICAL_MAXIMUM (255)
0x75, 0x08      ;    REPORT_SIZE (8)
0x95, 0x02      ;    REPORT_COUNT (2)
0x81, 0x02      ;    INPUT (Data,Var,Abs)
0xC0            ;  END_COLLECTION
0x05, 0x09      ;  USAGE_PAGE (Button)
0x19, 0x01      ;  USAGE_MINIMUM (Button 1)
0x29, 0x06      ;  USAGE_MAXIMUM (Button 6)
0x15, 0x00      ;  LOGICAL_MINIMUM (0)
0x25, 0x01      ;  LOGICAL_MAXIMUM (1)
0x75, 0x01      ;  REPORT_SIZE (1)
0x95, 0x06      ;  REPORT_COUNT (6)
0x81, 0x02      ;  INPUT (Data,Var,Abs)
0x95, 0x02      ;  REPORT_COUNT (2)
0x81, 0x03      ;  INPUT (Constant,Var,Abs)
0xC5            ;END_COLLECTION

```

THOUGHT PROCESS FOR DEVELOPING THIS DESCRIPTOR

The following is the thought process for developing this descriptor. Please refer to the “*HID Usage Tables*” for a complete list of Usage Pages and Usages. (See “**References**” at the end of this Technical Brief).

The first thing to do is tell the host that the device is a game pad. To do this, it is necessary to declare “USAGE (Game pad)” or “0x09, 0x05” numerically. But, before doing this, the appropriate page that “Game pad” is listed on as a Usage must be referenced. Looking game pad up in the HID Usage Tables reveals that the appropriate Usage Page is Generic Desktop. Once this Usage Page is declared, Usage Game Pad is specified. When the host has this information, it knows to add this device to the game controller category of its device listing. As a result, the game pad can be calibrated using the game controller calibration program provided with the host's operating system.

All report descriptors must have an Applications Collection. This Collection specifies the sources of I/O for the device and the format that the I/O is reported to the host. The Usage item that follows the Applications Collection specifies the general function of the collection. In this case, the Usage that follows is Pointer. Pointer is a Usage on the Generic Desktop Usage Page. It is not necessary to specify the Generic Desktop Usage Page again.

Note: The Usage Page must be specified only if the Usage that follows is not on the existing Usage Page.

The D-pad on the game pad controls two axis, X and Y. Because each axis has the same characteristics in terms of data report format, these axes can be grouped into a Collection. The X-axis and Y-axis are physical parameters; therefore, the Collection is of type Physical.

After the Collection is initialized, Usages X and Y are declared. Each axis of the D-pad is connected to a potentiometer inside the game pad. When the D-pad is pressed, the potentiometer varies the voltage on a wire connected to one of the A-to-D ports on the PIC16C765. Therefore, the PICmicro will see a digital value coming from each axis in the range of 0 to 255. The Logical Minimum and Logical Maximum correspond to this range. Eight bits are needed to report values in this range, the Report Size reflects this number. Two reports are needed, one for the X-axis and one for the Y-axis. The Report Count reflects this number. These two bytes are then designated as data inputs to the host with Input (Data, Variable, Absolute). Finally, the Collection is closed with the item End Collection.

The game pad has six buttons. A Physical Collection could be made and all the buttons listed as Usages (i.e., Usage {Button 1}, Usage {Button 2}, etc.) similar to what was done with the X and Y-axis. However, an easier way to make a list of buttons is to use the Usage Minimum and Usage Maximum items on the Buttons Usage Page. Each button accounts for a logical bit as reflected by the:

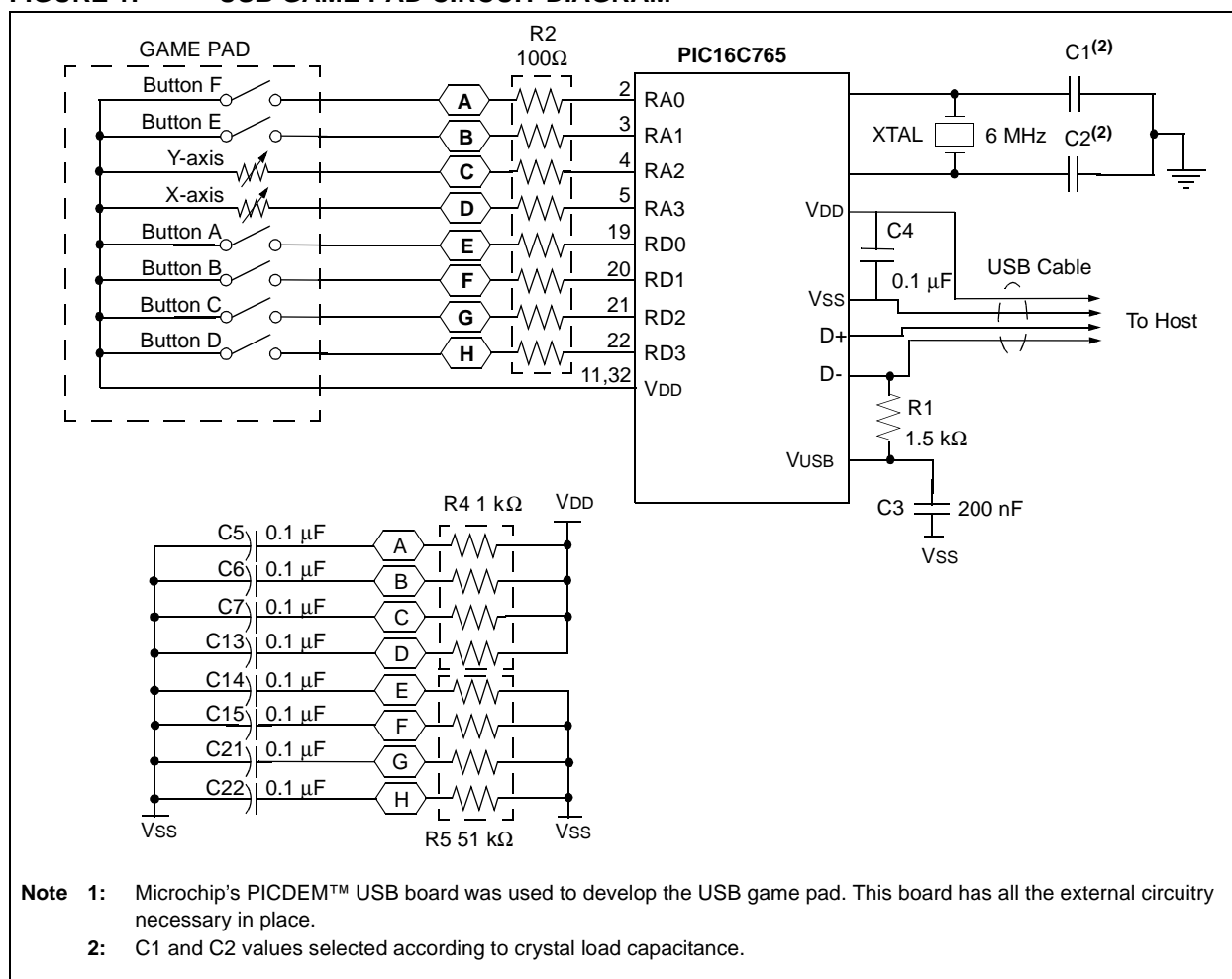
- Logical Minimum,
- Logical Maximum, and
- Report Size

The Report Count is six, corresponding to the number of buttons. These six bits are then input as data with the input item. Data sent over the USB bus must be sent as blocks of whole bytes. The six button report bits are two bits short of a byte. Therefore, a Report Count of two is specified and these bits are input as constants with the input item. The entire Application Collection is closed with the End Collection item.

Implementation

HARDWARE

The specific game pad used in this Technical Brief is the Dexxa™ eight-button (six actual as seen by the host) game pad. This game pad connects to the standard game port on a PC. A PC game port has four pins designated for analog inputs and four pins for digital inputs. Each axis of the D-pad uses an analog pin to send its position to the host. The remaining analog and digital pins are utilized by the six game pad buttons. Although two of the buttons are connected to analog pins their inputs are digital, either high or low. The following circuit diagram shows how the PICmicro is wired as a translator between the game port and PC.

FIGURE 1: USB GAME PAD CIRCUIT DIAGRAM⁽¹⁾

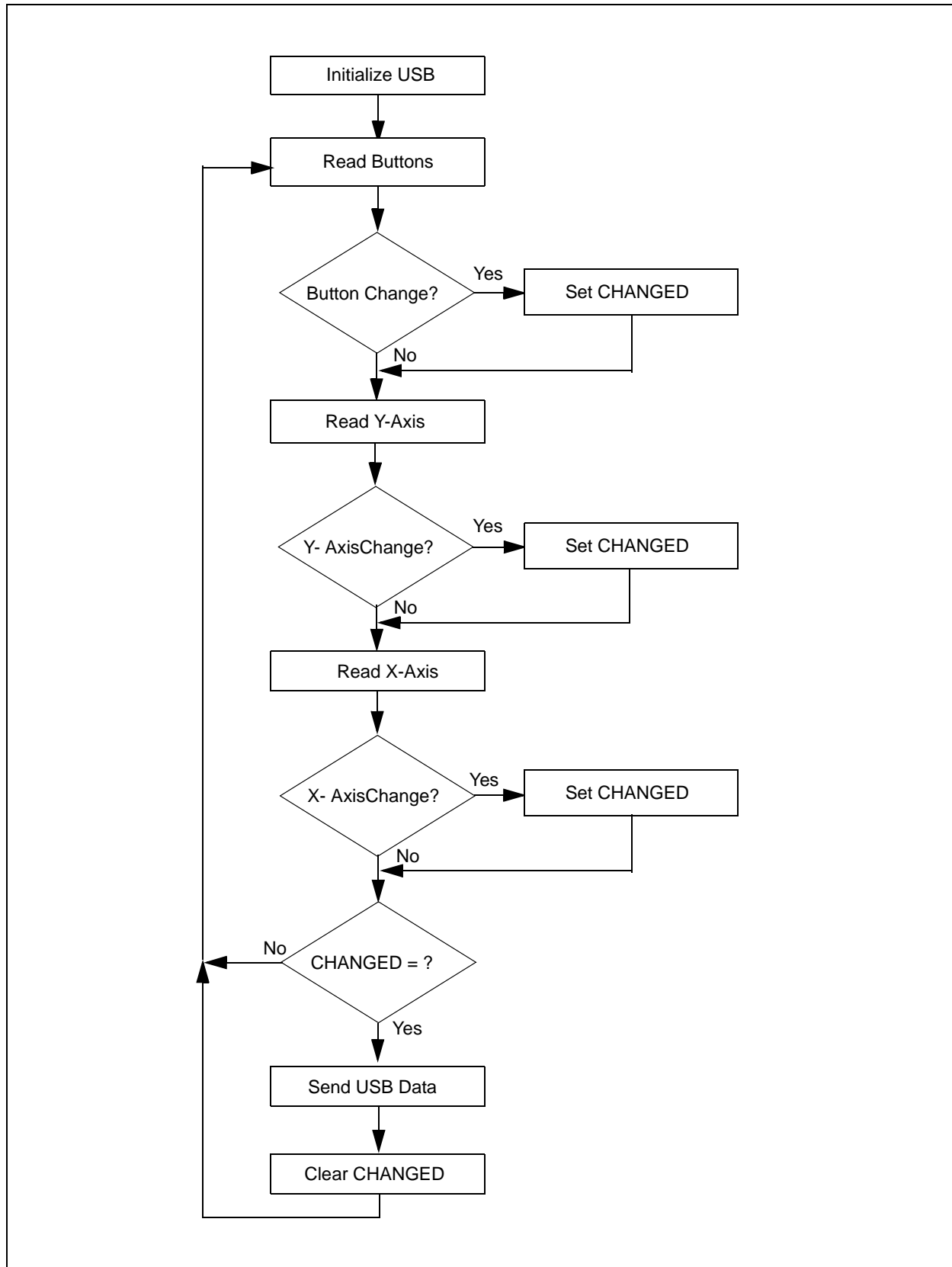
From Figure 1 it can be seen that the analog pins have several external components added between the game pad and PICmicro, namely two resistors and one capacitor per pin. The reason for this circuitry stems from the way analog pins were originally read by the PC. The PC would clear a capacitor tied to ground at its end and then time how long it took the capacitor to charge up. The capacitor would charge at a rate proportional to the resistance varied by one axis of the joystick, for instance. This is legacy technology and is not needed when using a PICmicro with an analog-to-digital converter. As a result, the before-mentioned circuitry was put into place in order to obtain a clean analog output from the D-pad. This circuitry is not needed for buttons E and F since they are digital, but was put in place so that this diagram will support other game pads.

SOFTWARE

Microchip's USB support firmware provided a foundation for creating the USB game pad firmware. The firmware provides a linkable file named `ch9_usb.asm`, which includes most of the functions described in Chapter 9 of the USB Specification, v1.1. Also included with this firmware is the file `hidclass.asm`. This file implements functions described in the "HID Device Class Definitions". These two files were linked without any changes to the USB game pad project. The two remaining assembly files included with the Chapter 9 USB firmware, `descript.asm` and `main.asm`, were altered to implement the game pad conversion.

The USB support firmware has a cursor demonstration incorporated into it. The file `descript.asm` contains the descriptors necessary for the demonstration to enumerate properly. To convert this file for the game pad application, the cursor demonstration descriptors were removed and replaced with the game pad descriptors. `Main.asm` was rewritten to contain the main routine for the game pad conversion. The main program flow is shown in Figure 2.

FIGURE 2: MAIN ROUTINE BLOCK DIAGRAM



CONCLUSION

For engineers just introduced to USB, descriptors pose a hurdle that is time-consuming to overcome. Walking through the conversion of the game pad from the game port interface to USB is a very effective way of overcoming this hurdle. Although most of the PIC16C765's resources are unused for the USB game pad, this Technical Brief serves as a starting place for developing more complex USB peripherals using the PIC16C745 or PIC16C765.

MEMORY USAGE

In the PIC16C765, the following memory was used:

Data Memory: 50 bytes
Program Memory: 1.9 Kbytes

REFERENCES

1. *USB Specification*, Version 1.1: Chapter 9 (located at www.usb.org)
2. *Device Class Definition for Human Interface Devices* (located at www.usb.org)
3. *HID Usage Tables* (located at www.usb.org)
4. *USB Firmware User's Guide* (located in USB Support Firmware zip file at www.microchip.com)
5. *USB Complete, Second Edition*, Jan Axelson; Lakeview Research, 2001 (www.lvr.com)
6. TB055: *PS/2® to USB Mouse Translator*
7. TB056: *Demonstrating the Set_Report Request with a PS/2® to USB Keyboard Translator Example*
8. TB057: *USB Combination Devices Demonstrated by a Combination Mouse and Game Pad Device*
9. TB058: *Demonstrating the Soft Detach Function with a PS/2® to USB Translator Example*

APPENDIX A: SOURCE CODE

Due to the length of the source code for the Game Port to USB Game Pad Translator example, the source code is available separately. The complete source code is available as a single WinZip archive file, tb054sc.zip, which may be downloaded from the Microchip corporate Web site at www.microchip.com.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

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


AmpLab, FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL, SmartShunt and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Application Maestro, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICkit, PICDEM, PICDEM.net, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, rfLAB, Select Mode, SmartSensor, SmartTel and Total Endurance 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.

© 2004, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

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

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



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: 480-792-7627
Web Address: <http://www.microchip.com>

Atlanta

3780 Mansell Road, Suite 130
Alpharetta, GA 30022
Tel: 770-640-0034
Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848
Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071
Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423
Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, IN 46902
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888
Fax: 949-263-1338

San Jose

1300 Terra Bella Avenue
Mountain View, CA 94043
Tel: 650-215-1444
Fax: 650-961-0286

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia

Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing

Unit 706B
Wan Tai Bei Hai Bldg.
No. 6 Chaoyangmen Bei Str.
Beijing, 100027, China
Tel: 86-10-85282100
Fax: 86-10-85282104

China - Chengdu

Rm. 2401-2402, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200
Fax: 86-28-86766599

China - Fuzhou

Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506
Fax: 86-591-7503521

China - Hong Kong SAR

Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai

Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700
Fax: 86-21-6275-5060

China - Shenzhen

Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380
Fax: 86-755-8295-1393

China - Shunde

Room 401, Hongjian Building, No. 2
Fengxiangnan Road, Ronggui Town, Shunde
District, Foshan City, Guangdong 528303, China
Tel: 86-757-28395507 Fax: 86-757-28395571

China - Qingdao

Rm. B505A, Fullhope Plaza,
No. 12 Hong Kong Central Rd.
Qingdao 266071, China
Tel: 86-532-5027355 Fax: 86-532-5027205

India

Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaughnessy Road
Bangalore, 560 025, India
Tel: 91-80-22290061 Fax: 91-80-22290062

Japan

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

168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5932 or
82-2-558-5934

Singapore

200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan

Kaohsiung Branch
30F - 1 No. 8
Min Chuan 2nd Road
Kaohsiung 806, Taiwan
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan

Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria

Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark

Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45-4420-9895 Fax: 45-4420-9910

France

Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany

Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy

Via Quasimodo, 12
20025 Legnano (MI)
Milan, Italy
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands

P. A. De Biesbosch 14
NL-5152 SC Drunen, Netherlands
Tel: 31-416-690399
Fax: 31-416-690340

United Kingdom

505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-118-921-5869
Fax: 44-118-921-5820

02/17/04