AN2681

ASF-USB Device Stack

Features

- USB 2.0 Compliance
  - USB chapter 9 certified
  - Control, bulk, isochronous and interrupt transfer types
  - Low speed (1.5Mbit/s), full speed (12Mbit/s), high speed (480Mbit/s) data rates
- Small Stack Size Frees Space for Main Application
- Real Time (OS Compliance, No Latency)
- Supports 8-bit and 32-bit AVR® Platforms
- USB DMA Support Increases Speed Performance
- Supports Most USB Classes and Ready to Use

Introduction

This document introduces the USB device stack. This stack is included in the Advanced Software Framework (ASF), and aims to provide the customer with the quickest and easiest way to build a USB application. A full description of this stack is available in this document. Only basic knowledge of USB is required to use this stack.

Table of Contents

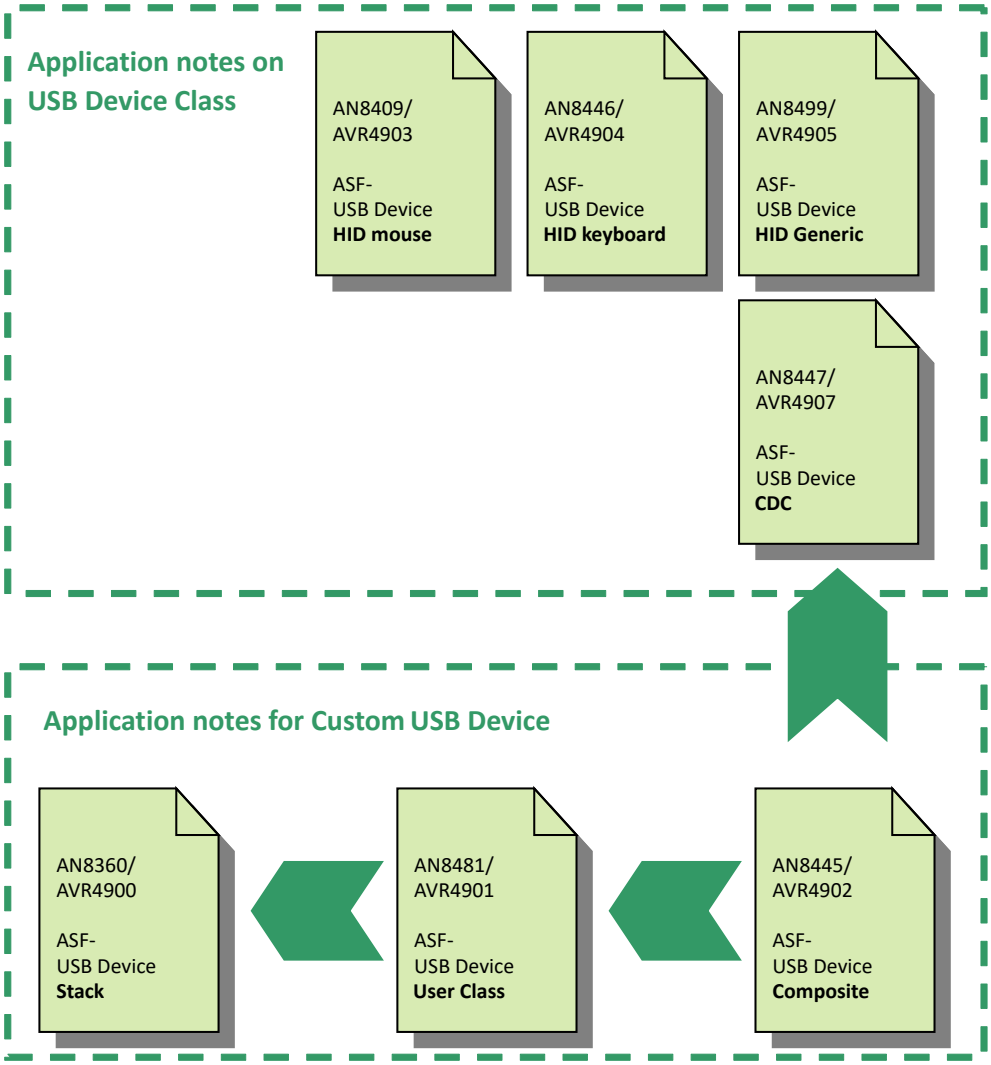
1. Abbreviations

- APP: User Application
- ASF: Advanced Software Framework
- CBW: Command Block Wrapper (from Mass Storage Class)
- CDC: Communication Device Class
- CSW: Command Status Wrapper (from Mass Storage Class)
- DP or D+ Data Plus Differential Line
- DM or D- Data Minus Differential Line
- FS: USB Full Speed
- HID: Human Interface Device
- HS: USB High Speed
- UDC: USB Device Controller
- UDD: USB Device Descriptor
- UDI: USB Device Interface
- USB: Universal Serial Bus
- MSC: Mass Storage Class
- PHDC: Peripheral Health Device Class
- sleepmgr: Sleep Management Service from ASF
- ZLP: Zero Length Packet

2. USB Device Application Notes

Several USB device examples are provided by Microchip. Some of these examples are covered by their own application note.

Figure 2-1. USB Device Application Notes



Basic USB knowledge is necessary to understand the USB device class application notes (Classes: HID, and CDC).

To create a USB device with one of the ASF provided classes, refer directly to the related application note for this USB class.

The new class and composite USB device application notes are designed for advanced USB developers.

Other examples are also available in Atmel Studio. To list these, select "New Example Project..." from the start screen or the File menu (File → New → Example Project...) in Atmel Studio 7. The list of examples can be reduced to list only USB by either searching for USB in the search field or select USB from the technology tab.

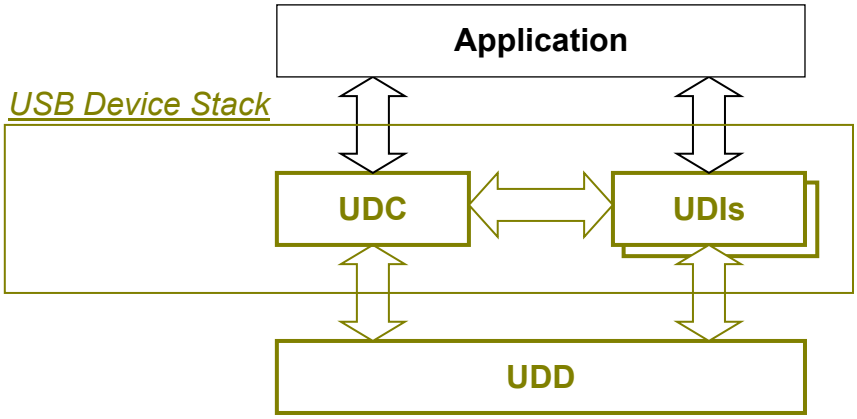
3. Organization

3.1 Overview

The USB device stack is divided into three parts:

- USB Device Controller (UDC) Provides USB Chapter 9 Compliance
- USB Device Interface (UDI) Provides USB Class Compliance
- USB Device Driver (UDD) Provides the USB Interface for Each AVR Product

Figure 3-1. USB Device Stack Architecture



3.2 Memory Footprint

The USB device stack memory footprint depends on:

- AVR Core (XMEGA®, megaAVR®, UC3)
- USB Hardware Version
- USB Class Used
- Compiler and Optimization Level

In average the USB device stack does not exceed 10KB of Flash and 1KB of RAM compiling with high optimization level.

3.3 USB Device Stack Files

The USB device stack files are available as part of ASF in Atmel® Studio.

To list relevant USB examples in Atmel Studio 7, select "New Example Project..." from the start screen or the File menu (File → New → Example Project...). The list of examples can be reduced to list only USB by either searching for USB in the search field or select USB from the Technology tab.

Note: This USB device stack does not apply to ASF examples with names containing "from ASF V1".

Table 3-1. Common Files for all AVR Products

	Files	Paths
Defines USB constant	usb_protocol.h (from usb.org)	common/services/usb/
	usb_atmel.h (from Microchip)	
UDC files	udc.c/h	common/services/usb/udc/
	udc_desc.h	
	udi.h	
	udd.h	
Classes protocols files	usb_protocol_foo.h	common/services/usb/class/foo/
UDI files	udi_foo.c/h	common/services/usb/class/foo/device/
	udi_foo_desc.c	
	udi_foo_conf.h	

Table 3-2. UDD Files Depending on Selected AVR Products

	Files	Paths
AVR32	usbb_device.c/h	avr32/drivers/usbb/
	usbb_otg.h	avr32/drivers/usbb/
	usbc_device.c/h	avr32/drivers/usbc/
	usbc_otg.h	avr32/drivers/usbc/
XMEGA	usb.c/h	xmega/drivers/usb/
MEGA AVR	usb.c/h	mega/drivers/usb/

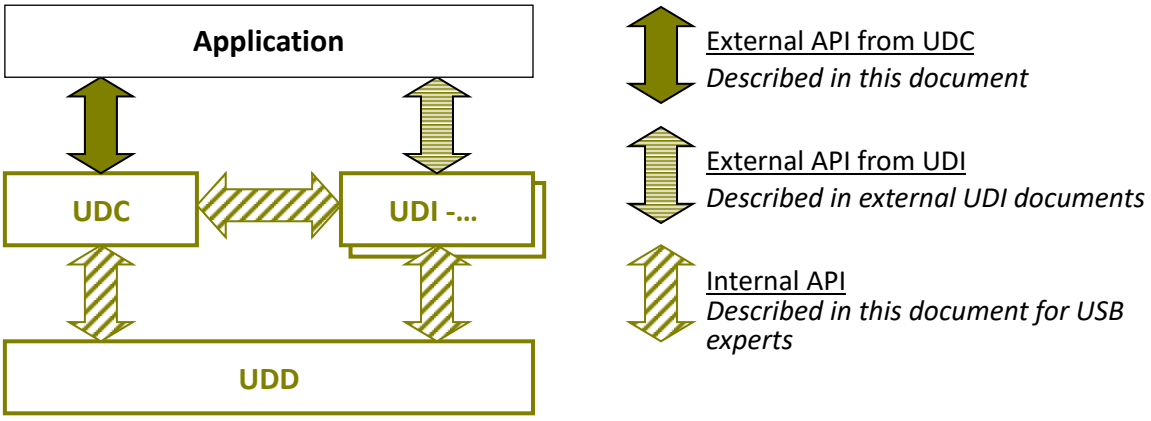
Table 3-3. Specific File for Each Application

	Files	Paths
Application file (This configuration file is mandatory)	usb_conf.h	user/application defined location, e.g. application config folder

4. Application Programming Interface

This section describes all USB APIs except the UDI API, which is described directly in the ASF documentation (http://asf.atmel.com).

Figure 4-1. USB Blocks



4.1 External API from UDC

The external UDC API allows the application to manage common USB device behavior and receive common USB device events. These controls and events are common to any USB application.

Figure 4-2. External API from UDC

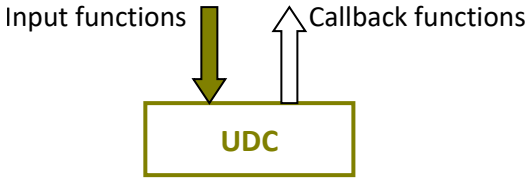


Table 4-1. External API from UDC – Input

Declaration	Description
udc_start()	Start USB device stack
udc_stop()	Stop USB device stack
udc_attach() udc_detach()	Authorize the device enumeration or not. Enable pull-up on DM or DP.
udc_remotewakeup()	Wake-up the USB device

All UDC callbacks are optional and defined by user in usb_conf.h for each application. When defined, the callbacks will be called by the UDD ISR.

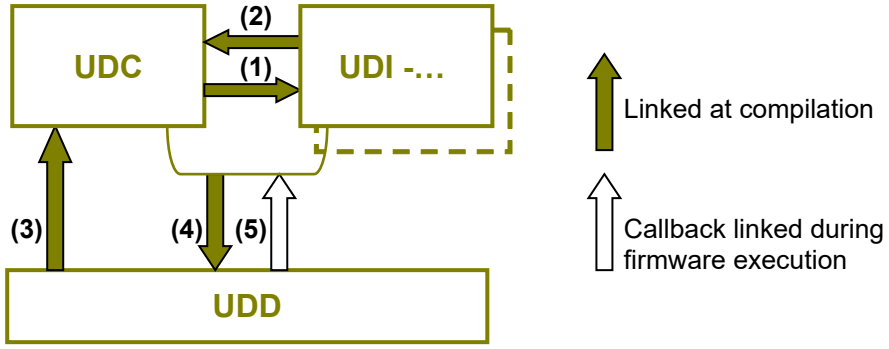
Table 4-2. External API from UDC – Callback.

Define name	Description
UDC_VBUS_EVENT(bool b_present)	To notify VBUS level change (only if USB hardware includes VBUS monitoring)
UDC_SUSPEND_EVENT()	Called when USB enters in Suspend mode
UDC_RESUME_EVENT()	Called when USB wakes-up
UDC_SOF_EVENT()	Called for each received SOF each 1ms Note: Available in High and Full Speed mode
UDC_REMOTEWAKEUP_ENABLE()	Called when USB host requests to enable/disable remote wake-up feature when the device supports it
UDC_REMOTEWAKEUP_DISABLE()	
UDC_GET_EXTRA_STRING()	When an extra string descriptor must be supported (other than manufacturer, a product and serial string)
UDC_SPECIFIC_REQUEST()	When a specific device setup request must be supported

4.2 Internal APIs

The following definitions are for advanced USB users who intend to develop a specific USB device not provided in ASF.

Figure 4-3. Internal USB Device API Overview



(*) See tables for number references.

Table 4-3. UDI Input from UDC (1)

Declaration	Description
bool (*enable)()	Called by UDC to enable/disable a USB interface
void (*disable)()	
bool (*setup)()	Called when a USB setup interface request is received
uint8_t (*getsetting)()	Called by UDC to obtain the current alternate setting of a USB interface
uint8_t (*sof_notify)()	Called by UDC to notify a SOF event at USB interface enabled

Note: The UDI API is linked with the UDC module via the UDC descriptor configuration file.

Table 4-4. UDC Input from UDI (2)

Declaration	Description
usb_iface_desc_t* udc_getiface()	Gives the USB interface descriptor selected by UDC when UDI is called (Table 4-3)

Table 4-5. UDC Input from UDD (3)

Declaration	Description
void udc_reset()	Called when Reset bus state occurs
bool udc_process_setup()	Called when a setup packet is received

Table 4-6. UDD Input (4)

Declaration	Caller	Description
void udd_enable()	UDC	Enables/disables the USB Device mode
void udd_disable()	UDC	
void udd_attach() void udd_dettach()	UDC	Inserts or removes pull-up on USB line
void udd_set_address(uint8_t add)	UDC	Changes/returns the USB device address
uint8_t udd_getaddress()	UDC	
bool udd_is_high_speed()	UDC/UDI	In case of USB HS device, then checks speed chosen during enumeration
uint16_t udd_get_frame_number()	APP	Returns the current start of frame number
udd_send_wake_up()	APP	The USB driver sends a resume signal called “Upstream Resume”
bool udd_ep_alloc(usb_ep_id_t ep, uint8_t bmAttributes, uint16_t wMaxPacketSize)	UDC	Enables/disables endpoints
udd_ep_free(usb_ep_id_t)	UDC	
bool udd_ep_clear_halt(usb_ep_id_t)	UDC/UDI	Clears/sets/gets the endpoint state (halted or not)
bool udd_ep_set_halt(usb_ep_id_t)		
bool udd_ep_is_halted(usb_ep_id_t)		
bool udd_ep_wait_stall_clear(udd_ep_id_t endp, udd_callback_nohalt_t callback)		Registers a callback to call when endpoint halt is removed
bool udd_ep_run(usb_ep_id_t endp, bool b_shortpacket, uint8_t *buf, uint32_t u32_size_buf, udd_callback_trans_t callback)	UDI	Starts/stops a data transfer in or out on an endpoint Note: The control endpoint is not authorized here
udd_ep_abort(usb_ep_id_t endp)	UDI	

Table 4-7. UDD Callback (5)

Declaration	Description
typedef void (*udd_callback_nohalt_t) (void);	Called when the halt on endpoint is removed. This one is registered via udd_ep_wait_stall_clear().
typedef void (*udd_callback_trans_t) (udd_ep_status_t status, iram_size_t nb_transfered)	Called when a transfer request is finished or canceled. This one is registered via udcdrv_ep_run().

Table 4-8. UDD Input for High Speed Application Only (4)

Declaration	Caller	Description
uint16_t udd_get_microframe_number()	APP	Returns the current micro start of frame number
udd_test_mode_j()	UDC	Features to test the USB HS device. These are requested to run a USB certification.
udd_test_mode_k()	UDC	
udd_test_mode_se0_nak()	UDC	
udd_test_mode_packet ()	UDC	

The global variable `udd_g_ctrlreq` is declared by UDD and contains two parts:

- Values updated by UDD and used by UDC and UDIs (Table 4-9)
- Values updated by UDC and UDIs and used by UDD (Table 4-10)

Outside the UDD, this variable is processed by `udc_process_setup()` for UDC and `*setup()` for UDI.

Table 4-9. udd_g_ctrlreq Field Updated by UDD

Declaration	Description
usb_setup_req_t req	Values included in SETUP packet and used to decode request.
uint8_t *payload	The content of the buffer is sent or filled by UDD. Can be NULL if u16_size is equal to 0.

Table 4-10. udd_g_ctrlreq Updated by UDC or UDI

Declaration	Description
uint8_t *payload	Pointer value of the buffer to send or fill
uint16_t u16_size	Buffer size to send or fill It can be 0 when no DATA phase is needed
bool over_under_run(void)	Called by UDD when the buffer given (.payload) is full or empty. Can be NULL
void *callback(void)	Called by UDD when the setup request is finished (setup+data+ZLP). Can be NULL

5. Behavior

This UDC stack implementation is based on an interrupt driven scheme. This solution ensures low latency, does not require any wait loop, and ensures OS compatibility.

Depending on the USB interrupt routine priority, the USB interrupt can be blocked by other interrupt routines with higher priority, or a critical code section. The USB hardware and software does not have any timing requirements except for the USB “Set Address” request (performed during USB enumeration phase to assign the USB address). The user must take care that the USB interrupt is not blocked during the "Set Address" request for longer than the maximum delay given in the table below.

Table 5-1. Set Address Timing

USB Host	Maximum Delay(1)
Specification	2ms
USB org certification tools	12ms
Windows® XP	48ms
Windows 7, Vista	32ms
Mac Mini OSX 10.5.8	77ms
Ubuntu 8.04, Ubuntu 9, Open Suse 11.1	29ms
Fedora 9, Fedora 10	24ms

Note: (1) These numbers will depend on USB host hardware, and is only a ballpark number for reference. These numbers include the time for *setup retry*.

USB hosts uses a timeout to reset a non-answering USB device (this time is not specified by the USB specification). The table below lists examples of operating system's timeout:

Table 5-2. OS Timeout

USB host	Timeout				
	Control Endpoint		Mass Storage		
	Data Phase	ZLP Phase	CBW	Data Read	CSW
Specification	No timeout				
Windows XP	5.3s	5.3s	19s	9.3s	9.3s
Windows 7, Vista	5.3s	5.3s	19s	160s/60s	160s/60s
Mac Mini OSX 10.5.8	5.9s	5.6s	11s	31s	22s
Ubuntu 8.04, Ubuntu 9, Open Suse 11.1	5s	5s	30s	30s	30s
Fedora 9, Fedora 10	5s	5s	30s	60s	30s

The following figures describe the interaction between the different layers.

Figure 5-1. USB Device Start-up and Stop

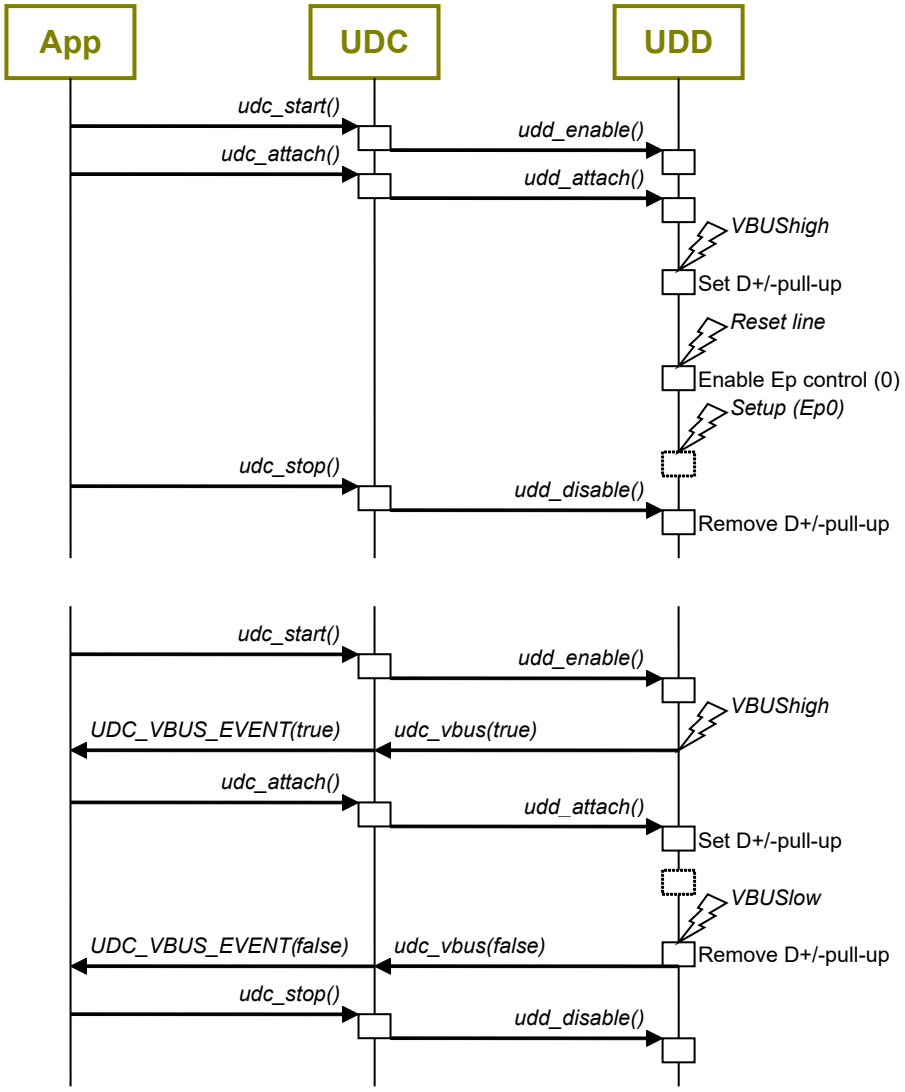
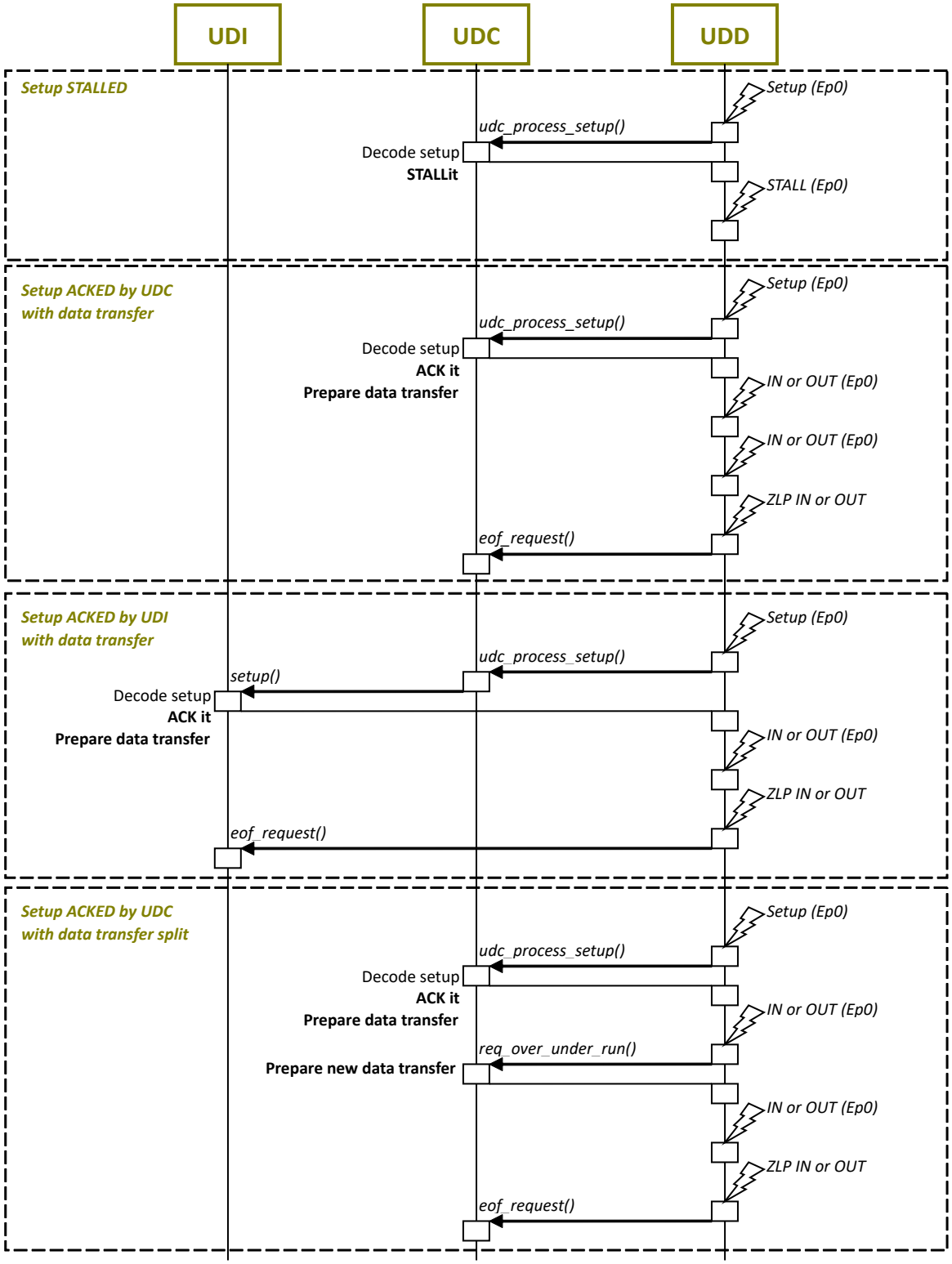
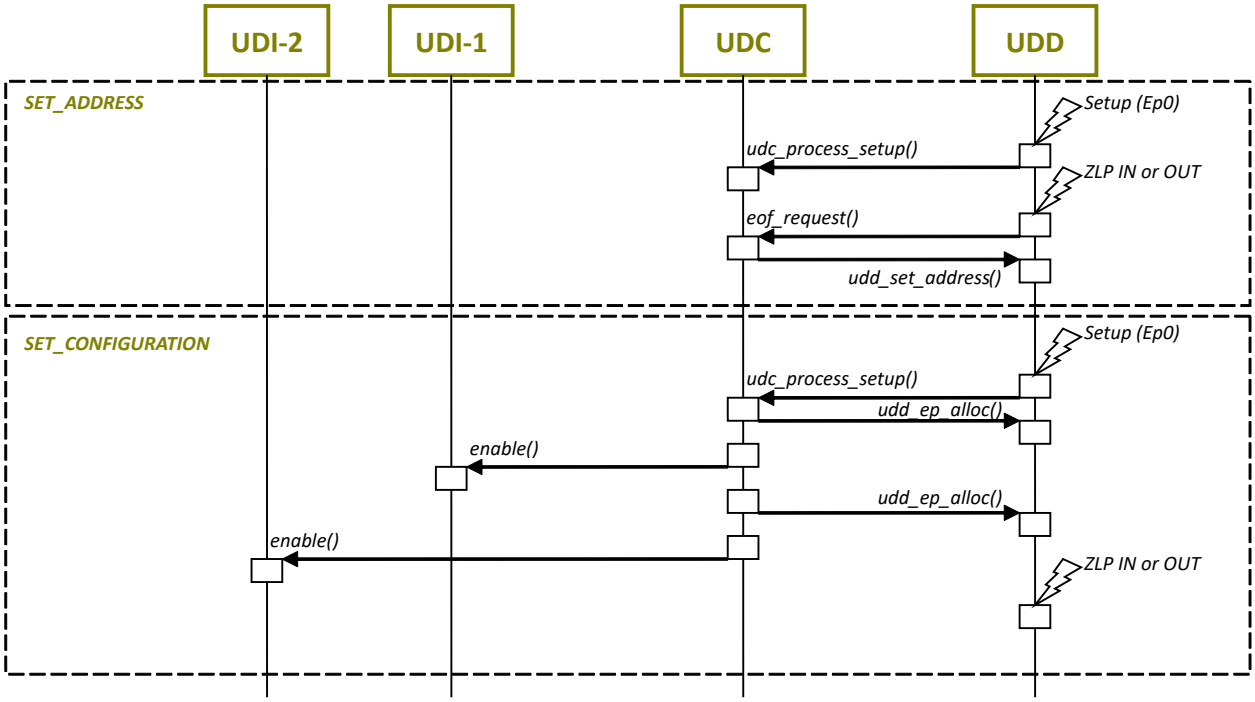


Figure 5-2. Management of Control Endpoint



Note: The `udd_g_ctrlreq` variable is used to communicate between UDD and UDC/UDIs.

Figure 5-3. Typical Enumeration



6. Configuration

The configuration is divided into two sections: application and USB descriptors.

The application's configuration is defined in the conf_usb.h file. This file must be created for each application. To create/edit this file only basic USB knowledge is required.

The conf_usb.h file must define the following configuration:

- USB Device Configuration
- USB Interface Configuration
- USB Driver Configuration

The USB descriptor configuration is required when the default configuration provided by Microchip is not used. This configuration information requires more in-depth knowledge of USB.

6.1 USB Configuration

USB Device Configuration

The following configuration must be included in the conf_usb.h file of the application, which is the main USB device configuration.

Table 6-1. USB Device Configuration

Define name	Type	Description
USB_DEVICE_VENDOR_ID	Word	Vendor ID provided by USB org (ATMEL 0x03EB)
USB_DEVICE_PRODUCT_ID	Word	Product ID (referenced in usb_atmel.h)
USB_DEVICE_MAJOR_VERSION	Byte	Major version of the device
USB_DEVICE_MINOR_VERSION	Byte	Minor version of the device
USB_DEVICE_MANUFACTURE_NAME(1)	String(2)	Static ASCII name for the manufacture
USB_DEVICE_PRODUCT_NAME(1)	String(2)	Static ASCII name for the product
USB_DEVICE_SERIAL_NAME(1)	String(2)	Static ASCII name to enable and set a serial number
USB_DEVICE_GET_SERIAL_NAME_POINTER()(1)	const uint8_t* function(void)	Give a pointer on a dynamic ASCII name to enable and set a serial number. Require USB_DEVICE_GET_SERIAL_NAME_LENGTH and ignore USB_DEVICE_SERIAL_NAME.
USB_DEVICE_GET_SERIAL_NAME_LENGTH()(1)	uint8_t function(void)	Give the length of dynamic ASCII name used to enable a serial number.
USB_DEVICE_POWER	Numeric	Maximum device power (mA)
USB_DEVICE_ATTR	Byte	USB attributes to add to enable feature: • USB_CONFIG_ATTR_SELF_POWERED

Define name	Type	Description
		• USB_CONFIG_ATTR_REMOTE_WAKEUP (3)
USB_DEVICE_LOW_SPEED(1)	Only defined	Force the USB device to run in Low Speed
USB_DEVICE_HS_SUPPORT(1)	Only defined	Authorize the USB device to run in High Speed
USB_DEVICE_MAX_EP	Byte	Define the maximum endpoint number used by the device (don't include control endpoint)

Note: (1) Optional configuration. Comment the define statement to disable it (ex: // #define USB_DEVICE_X).

Note: (2) Examples of String syntax: #define USB_DEVICE_MANUFACTURE_NAME "ATMEL". The Define can be omitted, thus the string is removed of USB enumeration.

Note: (3) If the remote wake feature is enabled, remote wake-up callbacks must be implemented.

USB Interface Configuration
The UDI configurations are described in USB device class application notes.

USB Drivers Configuration
The following configuration must be included in the conf_usb.h file of the application.

The AVR products provide specific hardware features that can be enabled here.

Table 6-2. USB Device Driver Configuration

Define name	Values	UDD	Description
UDD_NO_SLEEP_MGR	Only defined	All	Remove the management of sleepmgr service
UDD_ISOCHRONOUS_NB_BANK	1, 2, 3	AVR32 - USBB	Reduces or increases isochronous endpoint buffering. Default value: 2
UDD_BULK_NB_BANK	1, 2, 3	AVR32 - USBB	Reduces or increases bulk endpoint buffering. Default value: 2
UDD_INTERRUPT_NB_BANK	1, 2, 3	AVR32 - USBB	Reduces or increases interrupt endpoint buffering. Default value: 1
UDD_USB_INT_LEVEL	0 to 3	AVR32 - USBB AVR32 - USBC	Sets the USB interrupt level on AVR32 core. Default value: 0 (recommended)
UDD_USB_INT_LEVEL	USB_INTLVL_LO_gc USB_INTLVL_...	XMEGA - USB	Sets the USB interrupt level on Xmega core.

Define name	Values	UDD	Description
			Default value: USB_INTLVL_LO_gc (recommended)

6.2 USB Descriptors

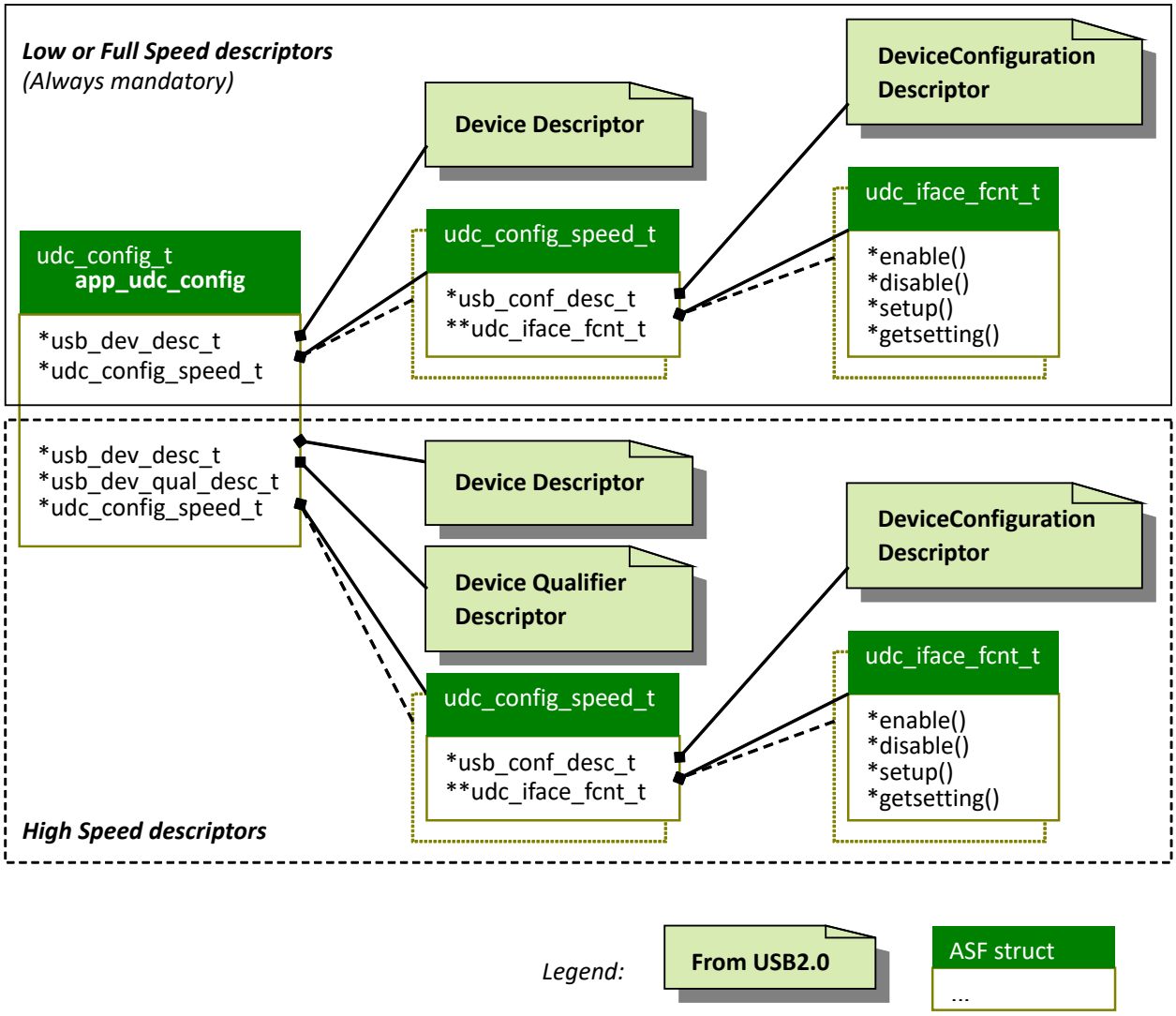
This section is oriented to USB developers who want to create a new UDI or a composite USB device.

The USB classes that are already provided by ASF include default USB Device descriptors. These descriptors are defined in the UDI files udi_foo_desc.c and udi_foo_conf.h, and allow an easy implementation described in all UDI application notes.

The descriptor file declares the global variable `app_udc_config` that includes:

- A device descriptor for each speed possible (`usb_dev_desc_t`)
- One device qualifier in case of High Speed device (`usb_dev_qual_desc_t`)
- A configuration descriptor for each configuration (`usb_conf_desc_x_t`)
- A link between UDI and configuration descriptor (`udc_iface_fcnt_t`)

Figure 6-1. USB Descriptors



7. Power Consumption

The different Power or Sleep modes available on AVR products is supported by the USB hardware according to USB line state. The USB drivers use the *sleepmgr* service to manage these Power or Sleep modes. Any USB application must include the *sleepmgr* service, and the *sleepmgr* must be initialized by calling `sleepmgr_init()`.

7.1 UC3 USBB and USBC Sleep Modes

All UC3 Sleep modes are described in the *Power Manager* chapter in the specific UC3 data sheet. Sleep modes supported by USBB and USBC drivers are:

- USB IDLE state: the driver requires the USB clock, hence IDLE Sleep mode is the lowest supported Sleep mode.
- USB SUSPEND state: the driver does not require the USB clock, but will request the clock on USB wake-up. Hence, STATIC and STANDBY is supported.
- VBUS monitoring: used in USB Self-Power mode, allows the UC3 to go down to STOP mode.

Table 7-1. Sleep Modes Supported in USB SUSPEND State on UC3

USB Power Mode	USB Speed Mode	USB Clock Start-up(1)	Sleep Mode Authorized
Bus and self-powered	LS, FS	>10ms	STANDBY
Bus and self-powered	HS	>3ms	STANDBY
Self-powered	LS, FS	<=10ms	STOP
Self-powered	HS	<=3ms	STOP
Bus powered	LS, FS	<=10ms	STATIC
Bus powered	HS	<=3ms	STATIC

Note: (1) Often external oscillators are used to generate the USB clock. The USB clock start-up timing will depend on the oscillator start-up timing.

The UC3 family support the specified power limit (2.5mA) in USB SUSPEND mode.

For any bus power device application, it is required to:

- Remove USB_CONFIG_ATTR_SELF_POWERED bit in USB_DEVICE_ATTR define from conf_usb.h file.
- Use an external oscillator with a low start-up time.
  This value is specified in the board header by the BOARD_OSC0_STARTUP_US define. Be aware of start-up time configuration options, specifically see the OSCCTRL0 register description in the UC3 data sheets.
- Use an external Low Dropout (LDO) voltage regulator or similar, to generate 3.3V for the UC3. When selecting this regulator, be sure its quiescent current does not consume a too large of a proportion of the required 2.5mA maximum suspend current.

7.2 XMEGA Sleep Modes

All XMEGA Sleep modes are described in the XMEGA data sheet chapter *Power Management and Sleep Modes*. The Sleep modes supported by the USB stack are:

- USB IDLE state: the driver requires the USB clock; hence IDLE Sleep mode is the lowest supported Sleep mode.
- USB SUSPEND state: the driver does not require the USB clock, but will request the clock on USB wake-up. Hence, POWER DOWN and STANDBY is supported.

Table 7-2. Sleep Modes Supported in USB SUSPEND State on XMEGA

USB Power Mode	USB Speed Mode	USB Clock Start-up (1)	Sleep Mode Authorized
Bus and self-powered	LS, FS	>10ms	STANDBY
Bus and self-powered	LS, FS	<=10ms	POWER DOWN

Note: (1)The internal oscillator can be used to generate the USB clock. Hence, USB clock start-up timing is < 10ms.

The XMEGA family support the specified power limit (2.5mA) in USB Suspend mode.

For any bus power device application, it is required to:

- Remove USB_CONFIG_ATTR_SELF_POWERED bit in USB_DEVICE_ATTR define from conf_usb.h file.
- Use the internal oscillator to reduce the start-up time.
- Use an external Low Dropout (LDO) voltage regulator, or similar, to generate 3.3V for the XMEGA. When selecting this regulator, be sure its quiescent current does not consume a too large of a proportion of the required 2.5mA maximum suspend current.

8. Revision History

Doc. Rev.	Date	Comments
A	04/2018	This document, Microchip DS00002681, replaces Atmel document 8360E-08/13. Updates include: the template, language, typos, references to the latest release of Atmel Studio, the list of relevant examples and app notes, as well as updated referenced names and versions to point to the latest items.
8360E	08/2013	• Updated UDC API2. • Added clarification about callbacks.
8360D	XX/XX	NOTE: Non-released version. • Add missing ATxmega product information (§4.3.2, Table 7-2).
8360C	08/2012	• Add new option to implement a dynamic serial number in Table 7-1. • Add ATxmega product information. • In features list, fix the High Speed 48Mbit/s by 480Mbit/s.
8360B	04/2011	• Updated all section concerning Power consumption. • Updated UDI and UDD APIs. See “sof_notify()” and udd_get_microframe_number() description.
8360A	12/2010	Initial revision

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

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.

Legal Notice

Trademarks

ISBN: 978-1-5224-2869-5

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. 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.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **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** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-7289-7561 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820