Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 001-67154

Spec Title: EZ LOADER CUSTOM USB FIRMWARE LOADER DRIVER - AN041

Replaced by: NONE

AN041

EZ Loader Custom USB Firmware Loader Driver

Author: Samir Joshi
Associated Project: Yes
Associated Part Family: CY7C68013/14/15/16
Software Version: NA
Related Application Notes: AN50963

This Application note describes building the EZ Loader firmware loader driver. A unique feature of the Cypress EZ-USB, EZ-USB FX1™, and EZ-USB FX2LP™ family of microcontrollers is the ability to change device personality through firmware download and ReNumeration™. A typical EZ-USB-based device has only enough nonvolatile storage in an EEPROM to store a unique Vendor ID (VID) and Product ID (PID). This VID/PID combination is bound to a specific device driver on the host system. This VID/PID combination can also be bound to a driver whose only function is to download firmware to the device. This application note describes such a driver. The EZ Loader Driver described in a template that peripheral designers can be used to create a custom driver that knows how to download their firmware to their specific device.

Contents

Introduction

A unique feature of the Cypress EZ-USB FX1™ and EZ-USB™ FX2LP™ family of microcontrollers is the ability to change device personality through firmware download and ReNumeration™. An EZ-USB-based device may have only enough nonvolatile storage in an EEPROM to store a unique Vendor ID (VID) and Product ID (PID). This VID/PID combination can be bound to a specific device driver on the host system. This VID/PID combination can also be bound to a driver whose only function is to download firmware to the device. This application note describes such a driver. The EZ Loader Driver discussed in this application note is a template that peripheral designers can use to create a custom driver that knows how to download their firmware to their specific device. The loader driver can be downloaded.

Note By using scripting capabilities of new tool 'CyConsole', similar firmware download and ReNumeration™ functionality is possible. In this method 'tool generated script' is used for firmware downloading instead of 'user created driver' as described above. This method is easier than the firmware loader driver method described in this document. For more details of this method, refer AN50963 'Downloading FX2LP Firmware Using CyConsole Script Capabilities'.

Getting Started

During your development cycle you uses the EZ-USB Control Panel via the USB cable to download an Intel® Hex file or you uses the Keil tools to download your firmware via the serial cable. While these methods are fine for development, eventually you want to automate the process of firmware download and ReNumeration. This is where the EZ Loader driver comes in.

The EZ Loader driver provided with the Cypress Semiconductor Cy3681 EZ-USB FX2 Development kit requires very little modification to support a specific device, so an extensive knowledge of Windows® driver programming is not required. However, you should be familiar with Windows Plug and Play, Windows INF files, the Windows Registry and USB.

The EZ Loader driver is a device driver and therefore requires a windows device driver kit (DDK). Before attempting to customize the EZ Loader driver, you should verify that you are able to successfully build a device driver using the DDK. Later in this application note you verify that you can successfully build a device driver following step-by-step procedures.

EZ Loader Driver Files

The source files needed to build the EZ Loader driver are included with the EZ-USB FX2 Development kit and can be found in C:\Cypress\USB\Drivers\ezloader after the development tools have been installed. The driver files and a brief description follow.

1. ezloader.c – This is the main EZ Loader source file. Contains the DriverEntry() and other standard USB driver entry points along with the code to perform the firmware download.
2. ezloader.h – Header file for the EZ Loader driver.
3. firmware.c – Contains an array of Intel Hex records for the device firmware. In the step by step procedures described later, you will use the hex2c.exe utility to convert your firmware from Intel Hex Record format into C code that can be placed into this file. When the EZ Loader driver is compiled, your firmware is included in the driver image.
4. loader.c – Contains an array of Intel Hex Records for device firmware that enables download to external RAM. If your firmware extends into external RAM, this firmware is downloaded to the EZ-USB device first to enable download to external RAM. This file was created using the Cypress Hex2c utility from the a3load.hex file found in the C:\Cypress\USB\Examples\FX2\a3load directory. The a3load.hex file works for the FX1 and FX2LP parts.
5. ezloader.rc – Resource file. Contains driver version information.
6. makefile – Required by the DDK build utility.
7. sources – Required by the DDK build utility. You customize this file in the Step-by-Step Example section to create a driver with a unique name.

Required Tools

Microsoft Visual C++ .NET

Microsoft® Visual C++ 2008 must be installed prior to installing the DDK. The Microsoft C compiler is automatically invoked by the DDK build utility. Microsoft® Visual C++.NET™ also works with the EZ Loader.

Windows Driver Kit (WDK)

A windows driver development kit is required to build the EZ Loader device driver. Visit Microsoft at the following URL for further information concerning Windows Driver Kit: www.microsoft.com/whdc/devtools/wdk/default.mspx

hex2c Utility - Intel Hex Record to C Translator

This Cypress-provided utility converts an input file of Intel Hex Records into C code that can be compiled and linked into the EZ-Loader driver. This utility creates a C file containing an array of INTEL_HEX_RECORD structures (defined in ezloader.h). It is a Win32 console application and is used as follows from a command prompt:

hex2c <intel_hexfile_name> <c_filename> <var_name>

intel_hexfile_name is an input file, and is the hex file created when you compile your firmware using the Keil Tools (that is widget.hex).

c_filename is the output filename that is created/overwritten.

var_name is optional and is the name of the array in the C output file. By default var_name = firmware.

The hex2c utility is located in the C:\Cypress\USB\bin directory. The source code for this utility is provided with the EZ-USB FX2 Development kit, in the C:\Cypress\USB\Util\ Hex2c directory.

Building the EZ Loader Device Driver

Before modifying the EZ Loader driver, you should create a new directory, which is used to build your custom loader driver. This is to prevent you from inadvertently modifying the ezmon.sys driver provided in the development kit. After a new directory has been created, and the EZ Loader source files have been copied into it, the EZ Loader driver can be compiled using the WDK build utility.

The file ‘sources’ tells the WDK ‘build’ utility how to build the driver. It specifies the source files that comprise the driver and the name of the driver output file. As provided, the ‘sources’ file creates a driver called ‘ezloader.sys.’

Depending on the build environment, your completed driver is placed in the \lib\i386\free or checked directory. Verify that these directories have been created prior to building the driver or the 'build' utility reports an error. When customizing the EZ Loader driver, the driver output file name should be changed to something other than ezloader.sys. This is done by changing the "TARGETNAME=" field in the 'sources' file to a new unique name. The "Step-by-Step Example" section illustrates how to change the 'sources' file.

Step-by-Step Example

This step-by-step example uses a hypothetical USB device, the Cypress Widget that you are building a firmware loader driver for.

Devices that use ReNumeration actually need two PIDs. One PID is for the device prior to firmware download, and the other is the PID for the fully functional device after firmware download. We uses PID 0x1004 for the pre-firmware Widget and PID 0x1005 for the fully functional device. The latter is the PID that is embedded in the Widget firmware and is usually designated in the dscr.a51 file.

Cypress's Vendor ID (VID) are 0x0547 and 0x04B4. For the purpose of this exercise, the 0x0547 Vendor ID will be used. Additionally, the Widget uses a Product ID (PID) of 0x1004 and 0x1005. You may use the Cypress VID for development purposes only, when your device is ready for production, you need to obtain a unique VID. For information on obtaining a Vendor ID for your production devices, visit www.usb.org.

The device driver for the Widget is the EZ-USB General-Purpose Device Driver (ezusb.sys) that is provided with the EZ-USB FX2 Development kit. This driver will be bound to VID = 0x547, PID = 0x1005.

This step-by-step example creates a new driver called wdgtldr.sys (Widget Loader). This driver is bound to VID = 0x0547, PID = 0x1004.

Follow these steps when your firmware is complete enough that you want to automate the firmware download process by integrating the firmware into a firmware loader driver. For this example we uses the bulkloop.hex firmware found in C:\Cypress\USB\Examples\FX2\bulkloop directory.

1. Create a new directory called Widget and copy the entire contents (files and subdirectories) of the C:\Cypress\USB\Drivers\ezloader directory into it.
2. Edit the 'sources' file with a text editor in the newly created directory and change the line TARGETNAME=ezloader to TARGETNAME=wdgtldr (Figure 1). You should use no more than eight characters for your loader name. Uncomment the following line in the sources file, when using Windows WDK: TARGETLIBS=$(DDK_LIB_PATH)\usbd.lib. To do this, simply remove the # symbol and also delete the space following the # symbol (Figure 2). By default, the line is commented.
3. Using the Keil tools, open the bulkloop.Uv2 project file in C:\Cypress\USB\Examples\FX2\bulkloop. In the dscr.a51 file change the PID to 1005 as shown in Figure 3, and then recompile the project. Note that these bits are in little endian order as shown below. Place a copy of the bulkloop.hex file into the following directory: C:\Cypress\USB\Bin. While this is not required, doing so stops you from having to type in the full path to your file in the next step. Another option is to place a copy of hex2c.exe in the bulkloop firmware directory.

Figure 1. TARGETNAME in Widget Loader "Sources" File

```
#*****************************************
***
#*
#* File:Sources
#*build
#* information file for the WDM build.exe
utility
#*
#* Date: February 12, 1999
#* Version:1.00.00
#*
#* Notes:
#*Copyright (c) 1997,1998,1999 Anchor Chips,
#*Inc. May not be reproduced without
#*permission. See the license agreement for
#*more details.
#*
#* Environment:
#*    WDM Build utility information file
#*
#* Revision History:
#*
#*****************************************
****

TARGETNAME=wdgtldr
TARGETTYPE=DRIVER
TARGETPATH=.\LIB
DRIVERTYPE=WDM
```

Figure 2. "Sources" File modified for Win XP SP1 DDK

```
# to build this driver using the Windows 2000 DDK, uncomment the following line:
TARGETLIBS=$(DDK_LIB_PATH)\usbd.lib
```

Figure 3. Changing VID in dscr.a51

```
;------------------------------------------------------------------------
;; Global Variables
;;------------------------------------------------------------------------
;; Note: This segment must be located in on-chip memory.
            rseg DSCR  ;; locate the descriptor table anywhere below 8K
DeviceDscr:dbdeviceDscrEnd-DeviceDscr;; Descriptor length
            db    DSCR_DEVICE;; Decriptor type
            dw    0001H;; Specification Version (BCD)
            db    00H   ;; Device class
            db    00H   ;; Device sub-class
            db    00H   ;; Device sub-sub-class
            db    64    ;; Maximum packet size
            dw    4705H;; Vendor ID
            dw    0510H;; Product ID - changed for EZ-Loader example ID
            dw    0100H;; Product version ID
            db    0     ;; Manufacturer string index
            db    0     ;; Product string index
            db    0     ;; Serial number string index
            db    1     ;; Number of configurations
deviceDscrEnd:
```

4. Use the 'hex2c.exe' utility to convert the Intel Hex Record format firmware to C code that can be linked into the loader driver. Open a command prompt window and navigate to C:\Cypress\USB\Bin or your firmware directory, if you copied hex2c into it. At the command prompt type the following: hex2c bulkloop.hex bulkloop.c (the Figure 4 shows ep_pair example instead of bulkloop). This takes the bulkloop.hex file as input and generate bulkloop.c. This new C file now contains a large array of INTEL_HEX_RECORD type structures called firmware. See Figure 5.
5. Place a copy of bulkloop.c into the Widget directory you created earlier. Open the firmware.c file for editing and replace its firmware array with the firmware array from bulkloop.c (Figure 4). After you have modified the firmware.c file, save it and then delete the bulkloop.c file from the directory.
6. Use the 'hex2c.exe' utility to convert the Intel Hex Record format firmware to C code that can be linked into the loader driver. Open a command prompt window and navigate to C:\Cypress\USB\Bin or your firmware directory, if you copied hex2c into it. At the command prompt type the following: hex2c bulkloop.hex bulkloop.c (the Figure 4 shows ep_pair example instead of bulkloop). This takes the bulkloop.hex file as input and generate bulkloop.c. This new C file now contains a large array of INTEL_HEX_RECORD type structures called firmware. See Figure 5.
7. Place a copy of bulkloop.c into the Widget directory you created earlier. Open the firmware.c file for editing and replace its firmware array with the firmware array from bulkloop.c (Figure 4). After you have modified the firmware.c file, save it and then delete the bulkloop.c file from the directory.
8. Now you are ready to build the firmware loader driver. Open your DDK via the Window Start and Programs menus. When a command prompt is opened navigate to C:\Widget\ezloader. At the command prompt type build -c (Figure 6). Your driver will be located in C:\Widget\ezloader\lib\i386. Place the wdgtldr.sys file in your Windows driver directory.

Figure 4. Converting ep_pair.hex Using the Hex2c Utility

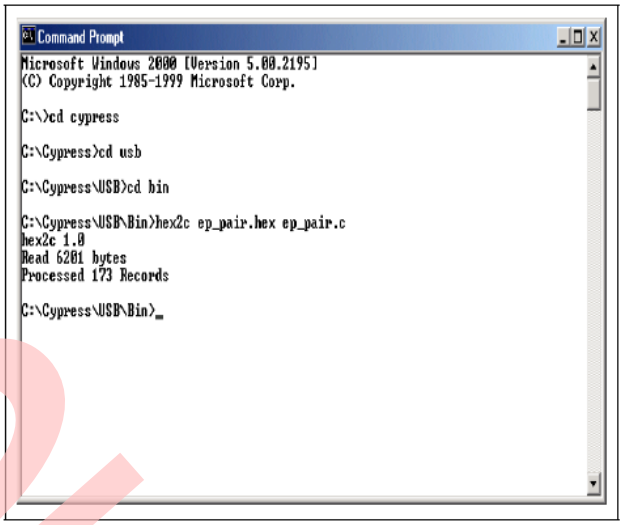
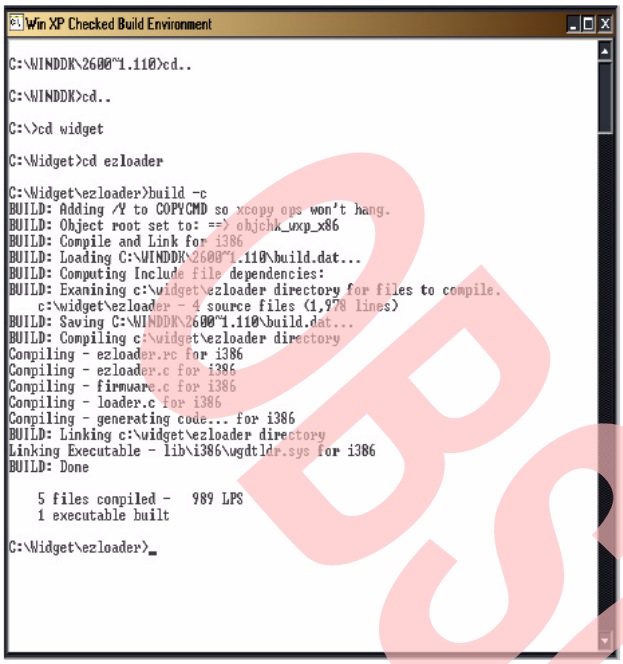


Figure 5. bulkloop.c INTEL_HEX_RECORD Structure

```
INTEL_HEX_RECORD firmware[] = {
  16,
  0x42f6,
  0,
  {0xe4,0xf5,0x2c,0xf5,0x2b,0xf5,0x2a,0xf5,0x29,0xc2,0x03,0xc2,0x00,0xc2,0x02,0xc2},
  16,
  0x4306,
  0,
  {0x01,0x12,0x46,0x15,0x7e,0x44,0x7f,0x50,0x8e,0x08,0x8f,0x09,0x75,0x0a,0x44,0x75},
  16,
  0x4316,
  0,
  {0x0b,0x62,0x75,0x0c,0x44,0x75,0x0d,0x82,0xee,0x54,0xe0,0x70,0x03,0x02,0x44,0x02},
  16,
  0x4326,
  0,
  {0x75,0x2d,0x00,0x75,0x2e,0x80,0x8e,0x2f,0x8f,0x30,0xc3,0x74,0xbc,0x9f,0xff,0x74},
  16,
  0x4336,
  0,
  {0x44,0x9e,0xcf,0x24,0x02,0xcf,0x34,0x00,0xfe,0xe4,0x8f,0x28,0x8e,0x27,0xf5,0x26},
  16,
  0x4346,
  0,
. . . .
```

Figure 6. Building wgdtldr with Windows XP SP1 DDK



9. To bind the new driver with the widget device you need to create an INF file. A sample INF file for the Cypress widget used in this example is provided at the end of this application note. For user convenience, the same sample INF (Widget.inf) is provided as an attachment to this document. Place the completed INF file in the same directory where other INF files are located in your system. This example was written for Windows 2000 systems and is compatible with Windows XP. Additional information on creating INF files can be found at: http://msdn.microsoft.com/en-us/library/ms924764.aspx. While the order you place your sections inside the INF file should not matter, the order of the items inside of a section can be critical.
10. Before you can use your new firmware driver you need to bind it with your device. This is done by loading the serial EEPROM with the VID/PID combination defined in the INF file you create for your custom USB device. This is the first of two VID/PID combinations your device will use. When the VID/PID in the EEPROM is reported to the host, the host invokes the wdgtldr.sys which loads your firmware. To load your VID/PID combination in the serial EEPROM open the EZ-USB Control Panel, ensure the Target Field matches your target device, and download the Vend_ax.hex example. Do not use the following Cypress PIDs: 0080, 0081, 1002, 2122, 2125, 2126, 2131, 2136, 2225, 2226, 2235, 2236 or 8613. Using these PID values can cause conflicts in the Windows Registry and result in the wrong driver being loaded.
    - EZ-USB FX1: Change the Dir field to 0 OUT and the Hex Bytes field to B4 47 05 04 10 01 00 and press the Vend Req button. Next, change the Dir field to 1 IN and press the Vend Req button and verify the same values are reported back to you (Figure 7).
    - EZ-USB FX2LP: Change the Dir field to 0 OUT and the Hex Bytes field to C0 47 05 04 10 01 00 00 and press the Vend Req button. Next, change the Dir field to 1 IN and press the Vend Req button and verify the same values are reported back to you. See section 3.5 of the FX2LP Technical Reference Manual for additional EEPROM configuration byte (byte 8) information.
11. When you disconnect and reconnect your device the Windows New Hardware Wizard appears (Figure 8) and install your device. The first VID/PID combination causes the host to load your firmware via the wdgtldr.sys firmware driver you have created (Figure 9). The device then renumerates to the VID/PID combination in the bulkllop.hex example created in step 3, loading the ezusb.sys device driver (Figure 10).

Figure 7. Programming the EEPROM with EZ-USB Control Panel

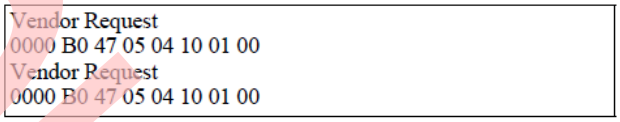
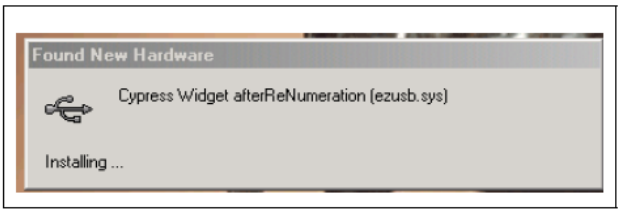


Figure 8. Windows New Hardware Wizard Detects Your Device

Figure 9. wdgtldr.sys Will Load Your Firmware

Figure 10. Device Now ReNumerates and Uses ezusb.sys



Verifying the wdgtldr.sys

To verify the wdgtldr.sys, open the EZ-USB Control Panel. Press the GetDevice button (Figure 11). Press the GetPipes button (Figure 12). If using the FX2LP, you can see additional pipes declared (Figure 13). Finally, pressing the bulkloop button demonstrates the bulkloop example (Figure 14). The device has been configured without any further user interaction.

Figure 11. Pressing GetDevice report

```
Device Descriptor:
bLength: 18
bDescriptorType: 1
bcdUSB: 256
bDeviceClass: 0x0
bDeviceSubClass: 0x0
bDeviceProtocol: 0x0
bMaxPacketSize0: 0x40
idVendor: 0x547
idProduct: 0x1005
bcdDevice: 0x1
iManufacturer: 0x0
iProduct: 0x0
iSerialNumber: 0x0
bNumConfigurations: 0x1
```

Figure 12. Pressing GetPipes report

```
Pipe: 0   Type: BLK Endpoint: 2 IN   MaxPktSize: 0x40
Pipe: 1   Type: BLK Endpoint: 2 OUT MaxPktSize: 0x40
```

Figure 13. Pressing GetPipes report - FX2LP

```
Get PipeInfo
Pipe:  0  Type: BLK Endpoint:  2 OUT  MaxPktSize: 0x200
Pipe:  1  Type: BLK Endpoint:  4 OUT  MaxPktSize: 0x200
Pipe:  2  Type: BLK Endpoint:  6 IN   MaxPktSize: 0x200
Pipe:  3  Type: BLK Endpoint:  8 IN   MaxPktSize: 0x200
```

Figure 14. Pressing BulkLoop Report

```
Write IOCTL passed
0000 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
0010 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
0020 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
0030 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
Read IOCTL passed
0000 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
0010 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
0020 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
0030 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05 05
```

Updating the EZ Loader

Providing firmware and device driver updates can be accomplished by creating an Install Shield program, which is beyond the scope of this application note, or by using the update INF example file shown at the end of this application note. For user convenience, the same example INF (DevDri.inf) is provided as an attachment to this document. Figure 15, Figure 16, Figure 17, and Figure 18 show a summary of the update process from selecting Update Driver in the Windows Device Manager to an EZ-USB Control Panel display of the updated firmware (the bulktest example was used in creating the updated wdgtldr.sys. to replace the wdgtldr.sys built using ep_pair.) The update procedure shown is from a Windows 2000 system but this process works equally well with Windows XP.

Figure 15. Updating Firmware or Device Driver

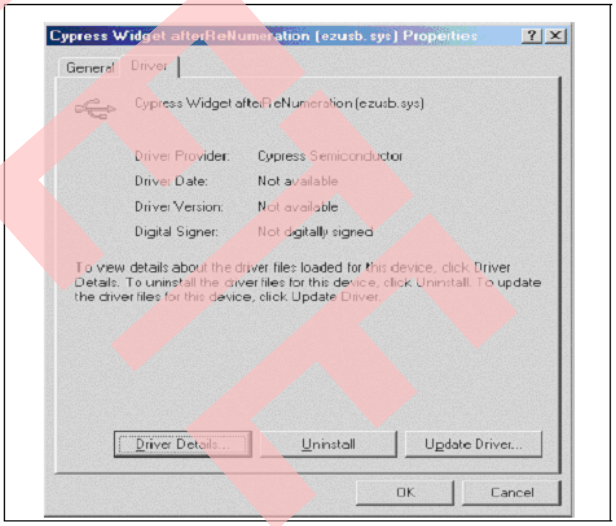


Figure 16. Updating Firmware

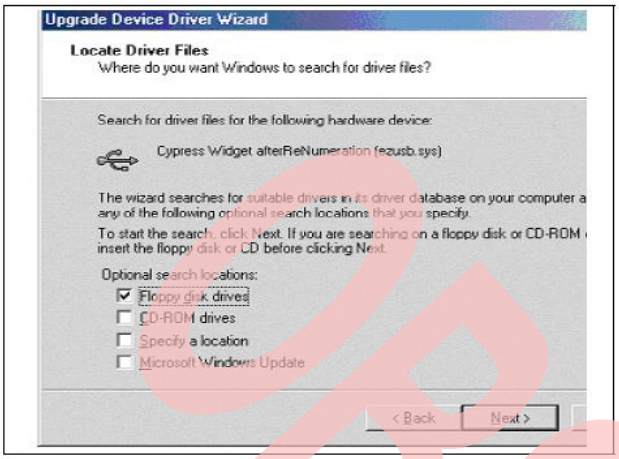


Figure 17. Selecting Updated Driver

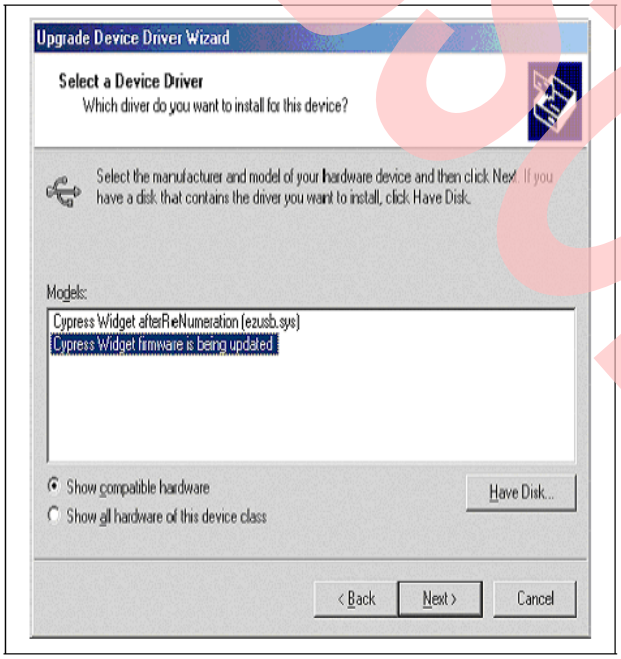
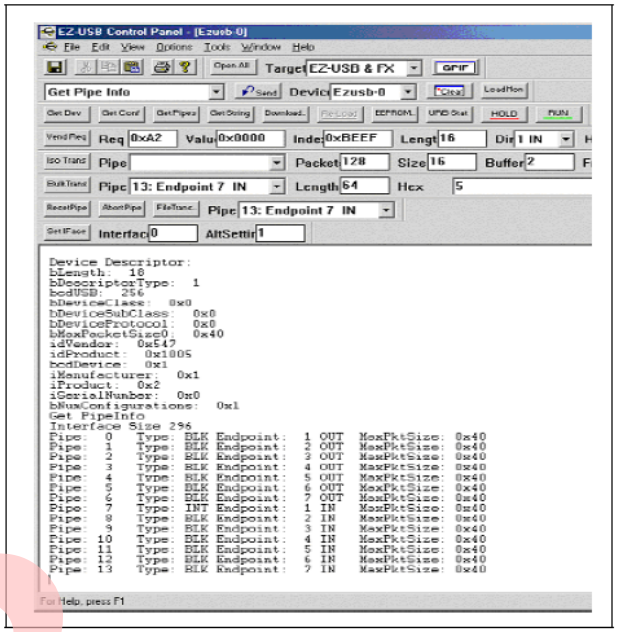


Figure 18. Updated Firmware Load in Control Panel



Widget Troubleshooting

If after building the widget you experience problems, check the following:

- Ensure you do not have a space in the directory name where your driver development kit is located.
- When using windows driver kit WDK, ensure you have uncommented the line and removed the leading space as noted in step 2 of the "Step-by-Step Example" instructions.
- If you lose communications with the development board procedures for regaining communications for each development board can be found in a knowledge base article available on our website, search for Corrupted EEPROM.
- If you need to clean the Windows Registry, procedures for different operating systems can be found in a knowledge base article available on our website, search for Windows Registry.

Additional support is available via our web-based technical support system at www.cypress.com.

;WIDGET INF File

```
; Place your Copyright information here


[Version]
Signature="$CHICAGO$"
Class=USB
ClassGuid={36FC9E60-C465-11CF-8056-
444553540000}
provider=%Cypress%
;CatalogFile=ezusb.cat
;The CatalogFile entry above is an example
and should be uncommented and modified for
your driver
;DriverVer=06/18/2003, 1.0.0.0
;The DriverVer entry above is an example
and should be uncommented and modified for
your driver


[Manufacturer]
%Cypress%=Cypress

;[SourceDisksNames]
;1=%strWidgetSourceDiskName%,,,
;The above section is for example purposes
;Uncomment and modify this section for your
driver

;[SourceDisksFiles]
;wdgtldr.sys=1
;ezusb.sys=1
;The above section is for example purposes
;Uncomment and modify this section for your
driver

[Cypress]
;
; Entry point for the widget before
firmware download and renumeration
; This VID/PID combination will call the
EZ-Loader driver and download
; your firmware. Your finished product
should use your own unique VID
; see www.usb.org for additional
information
%USB\VID_0547&PID_1004.DeviceDesc%=WIDGET.D
ev, USB\VID_0547&PID_1004

; Entry point for the widget after firmware
download and renumeration
; Your firmware has been download, the
device has ReNumerated, now we
; want to use the EZ-USB General Purpose
Device Driver. Your finished
; product should use your own unique VID -
see www.usb.org
%USB\VID_0547&PID_1005.DeviceDesc%=EZUSB.De
v, USB\VID_0547&PID_1005

[DestinationDirs]
EZUSB.Files.Ext = 10,System32\Drivers
WIDGET.Files.Ext = 10,System32\Drivers

[EZUSB.Dev.NT]
CopyFiles=EZUSB.Files.Ext
AddReg=EZUSB.AddReg

[EZUSB.Dev.NT.Services]
Addservice = EZUSB, 0x00000002,
EZUSB.AddService

[EZUSB.AddService]
DisplayName    = %EZUSB.SvcDesc%
ServiceType    = 1                  ;
SERVICE_KERNEL_DRIVER
StartType      = 3                  ;
SERVICE_DEMAND_START
ErrorControl   = 1                  ;
SERVICE_ERROR_NORMAL
ServiceBinary  =
%10%\System32\Drivers\ezusb.sys
LoadOrderGroup = Base

[EZUSB.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,ezusb.sys

[EZUSB.Files.Ext]
ezusb.sys

[WIDGET.Dev.NT]
CopyFiles=WIDGET.Files.Ext
AddReg=WIDGET.AddReg

[WIDGET.Dev.NT.Services]
Addservice = WIDGET, 0x00000002,
WIDGET.AddService

[WIDGET.AddService]
DisplayName = %WIDGET.SvcDesc%
ServiceType = 1                  ;
SERVICE_KERNEL_DRIVER
StartType = 3                    ;
SERVICE_DEMAND_START
ErrorControl   = 1                  ;
SERVICE_ERROR_NORMAL
ServiceBinary  =
%10%\System32\Drivers\wdgtldr.sys
LoadOrderGroup = Base

[WIDGET.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,wdgtldr.sys

[WIDGET.Files.Ext]
wdgtldr.sys
```

```
;-----------------------------------------
---------------------;

[Strings]
;strWidgetSourceDiskName = "Cypress USB
Drivers"
;The string above and the strings below
should be modified to meet your needs
Cypress="Cypress Semiconductor"
USB\VID_0547&PID_1004.DeviceDesc="Using
widget.inf for EZ-Loader before
ReNumeration (wdgtldr.sys)"
USB\VID_0547&PID_1005.DeviceDesc="Cypress
Widget afterReNumeration (ezusb.sys)"

EZUSB.SvcDesc="Cypress General Purpose USB
Driver (ezusb.sys)"
WIDGET.SvcDesc="Cypress General Purpose USB
Driver w/ Widget Loader (wdgtldr.sys)"

;The Windows DDK provides a chkinf tool
that can be used to check your inf for
errors.
;The chkinf tools requires Perl which can
be downloaded, at the time of publication,
;at
http://www.activestate.com/Products/Downloa
d/Download.plex?id=ActivePerl

;end of widget inf file
```

;Widget Firmware/Device Driver Update INF File

```
; Place your Copyright information here

[Version]
Signature="$CHICAGO$"
Class=USB
ClassGuid={36FC9E60-C465-11CF-8056-
444553540000}
provider=%Cypress%
;CatalogFile=ezusb.cat
;The CatalogFile entry above is an example
and should be uncommented and modified for
your driver
;DriverVer=06/18/2003, 2.0.0.0
;The DriverVer entry above is an example
and should be uncommented and modified for
your driver


[Manufacturer]
%Cypress%=Cypress

;[SourceDisksNames]
;1=%strWidgetSourceDiskName%,,,
;The above section is for example purposes
;Uncomment and modify this section for your
driver

;[SourceDisksFiles]
;wdgtldr.sys=1
;ezusb.sys=1
;The above section is for example purposes
;Uncomment and modify this section for your
driver


[Cypress]
; Windows will only recognize the active
VID/PID
; for updates so we will use the already
renumerated device as our update entry
point.
%USB\VID_0547&PID_1005.DeviceDesc%=EZUSB.De
v, USB\VID_0547&PID_1005


[DestinationDirs]
EZUSB.Files.Ext = 10,System32\Drivers

[EZUSB.Dev.NT]
CopyFiles=EZUSB.Files.Ext
AddReg=EZUSB.AddReg

[EZUSB.Dev.NT.Services]
Addservice = EZUSB, 0x00000002,
EZUSB.AddService

[EZUSB.AddService]
DisplayName    = %EZUSB.SvcDesc%
ServiceType    = 1                  ;
SERVICE_KERNEL_DRIVER
StartType      = 3                  ;
SERVICE_DEMAND_START
ErrorControl   = 1                  ;
SERVICE_ERROR_NORMAL
ServiceBinary  =
%10%\System32\Drivers\ezusb.sys
LoadOrderGroup = Base

[EZUSB.Files.Ext]
;This section will cause the wdgtldr to be
updated, which in turn will call for the
ezusb.sys
;These files should be located on the
install medium. You must include ezusb.sys
or the system
;will continue to renumerate, even if the
device driver does not have an update.
wdgtldr.sys
ezusb.sys

;-----------------------------------------
---------------------;

[Strings]
Cypress="Cypress Semiconductor"
```

```
USB\VID_0547&PID_1005.DeviceDesc="Cypress
Widget firmware is being updated."

EZUSB.SvcDesc="Cypress General Purpose USB
Driver (ezusb.sys)"

;The Windows DDK provides a chkinf tool
that can be used to check your inf for
errors.

;The chkinf tools requires Perl which can
be downloaded, at the time of publication,
;at
http://www.activestate.com/Products/Downloa
d/Download.plex?id=ActivePerl

;end of firmware/device driver update inf
file
```

Summary

In this application note the Cypress general-purpose USB driver, ezusb.sys was used as the device driver after renumeration. Developers should note that this driver is provided as a starting point for driver development and it is not intended to serve as a production device driver. A list of Cypress recommended consultants to assist you with your driver development is available at: www.cypress.com/support/cypros.cfm

The fw.c file included in the bulkloop example and other examples provided in our development kits, contains code that ensures descriptor and setup information is loaded into the chip. Do not comment out this code when developing your firmware, unless you are positive that the descriptors resides in the on-chip RAM.

As you are creating your custom INF file and you need to make changes during the development process it is a good idea to clean the Windows Registry of the VID/PID combinations for your widget. When you clean the Window Registry you should also delete the wgdtldr PNF file that will be located in the Windows/INF directory. We recommend making a back up copy of your Windows Registry prior to making any modifications.

After following the process outlined in this application note you are now able to create a firmware loader driver to automate the firmware download for your custom USB device. You have also learned of one method available for issuing firmware loader driver updates for your custom USB device.

Document History

Document Title: EZ Loader Custom USB Firmware Loader Driver - AN041

Document Number: 001-67154

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3162876	SSJO	02/09/2011	This spec is not available in spec system but it is available in web. Created a new spec number for this application note. Updated as per application note template. Added Abstract.
*A	4271402	NIKL	02/04/2014	Updated in new template. Completing Sunset Review.
*B	5660069	HPPC	03/15/2017	Obsolete document

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709