

CY4611 – FX2 USB to ATA Reference Design Notes

Note

Cypress offers several mass storage reference designs. This document describes the CY4611 software, which runs **only** on the EZ-USB FX2 (CY7C68013). See the Cypress website for information on other designs.

Introduction

Cypress has two USB 2.0 High Speed Mass Storage solutions. The ISD-300 is a fixed function Mass Storage solution for ATA devices. The EZ-USB FX2 is a flexible bridge solution that enables additional features to be added to a USB 2.0 bridge device. It has a downloadable memory interface to incorporate last minute specification changes, and it supports the full USB 2.0 throughput (480Mbits/sec). Since not all mass storage devices have exactly the same interface, Cypress has developed a flexible reference design compliant with the USB mass storage specification, so that mass storage customers can quickly test and adapt the design to their individual requirements. The Cypress EZ-USB FX2 Mass Storage reference design demonstrates how to connect the EZ-USB FX2 to the following device types:

- IDE devices
 - hard disk drives
- ATAPI devices
 - ZIP drives
 - CD-ROM/R/RW drives
 - DVD-ROM/RAM/RW drives
 - Other ATAPI devices

Reference Design Elements

- FX2 Mass Storage Reference Design PCBA
- Reference Design Schematic in both PDF and OrCAD source files
- Reference Design BOM
- Firmware source and object code
- Reference Design Notes (this document)
- UDMA White Paper
- Operating Instructions
- Release Notes
- Errata

Background Information

This document assumes the reader to be familiar with the USB Mass Storage Class specification and general operation of the Cypress' EZ-USB FX2. For more information please refer to these specifications or Cypress's EZ-USB FX2 Technical Reference Manual.

Mass Storage Class Specification

The USB Mass Storage Class specification contains two subclasses, the CBI (Command, Bulk, Interrupt), and the newer Bulk Only Transport. This reference design complies with the Bulk Only subclass of the USB Mass Storage Specification. The Bulk Only subclass is supported by the Windows XP, 2000 and Me drivers as well as MacOS 9 and X.

Boot Support

The current level of boot functionality will allow you to boot to DOS or Win9x Safe Mode from a Hard Drive or CDROM. You cannot currently boot to Windows due to issues with the way Windows attempts to access a boot drive directly. Boot functionality has been tested with both Phoenix and AMI BIOS.

48 bit LBA Addressing

The proposed ATA-6 spec contains support for large drives with 48 bit Logical Block Addresses (LBAs). Version 2.10 of the CY4611 adds support for this specification. However, the SCSI commands passed by the Mass Storage Class Specification only support 32 bit LBAs, which limits support to 2⁴¹ (2Tera) bytes on a 512 byte sector device.

Firmware Overview

Note: CBW, CSW, dataTransferLength, and "Persistent Stall" are defined in the "USB Mass Storage Class, Bulk Only Transport" document referenced below.

The firmware for the device is a straightforward implementation of a USB Bulk Only Mass Storage Device. After reset, it waits for a CBW packet, checks it and then executes the data phase of the command (if any). Once the data phase is complete, the firmware sends a CSW packet to the host. The only commands that the firmware generates on its

own are SCSI Identify Device (to get the device name) and ATA Identify Device (to get the device serial number).

The current FX2 firmware supports both high speed (480Mbps) and full speed (12Mbps) devices. IDE and ATAPI devices are supported by a single firmware image.

Firmware Details

Main()

Upon reset, the firmware begins execution at main(). This routine checks to see if this is a hard or soft reset. Soft resets do not trigger a disconnect/reconnect sequence, while hard resets do.

resetATAPIDevice()

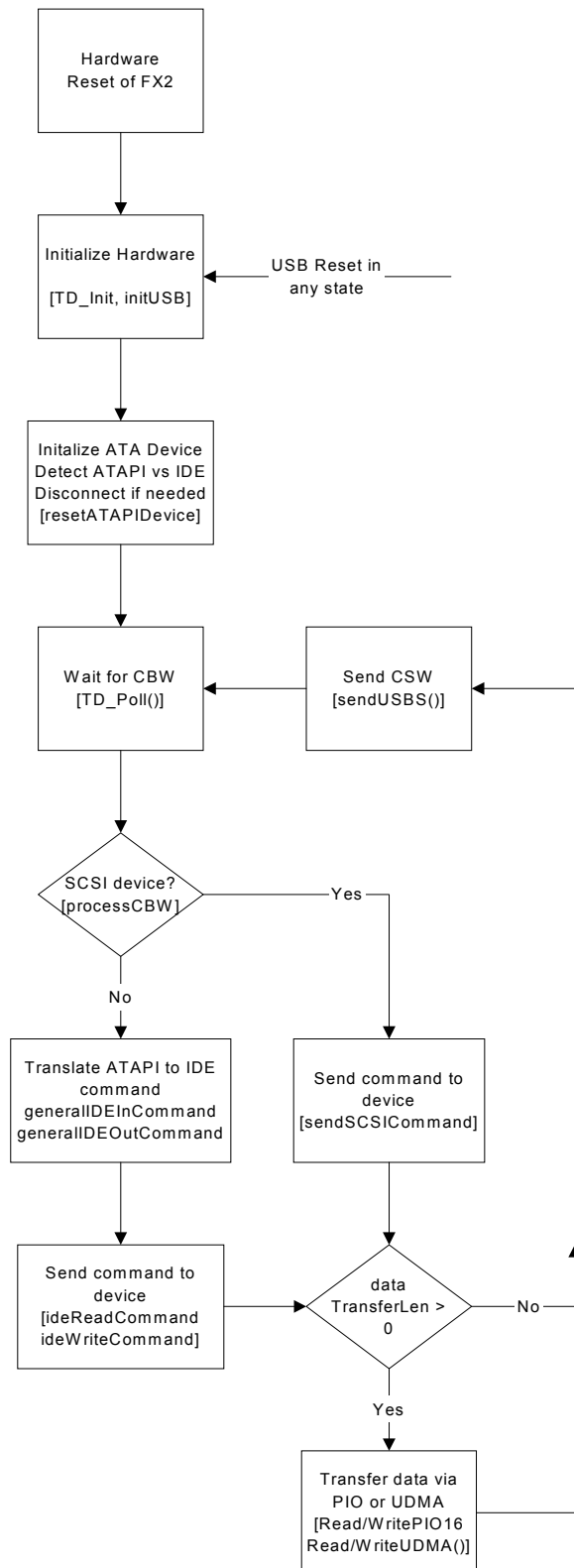
On a hard reset, TD_init() is called, which initializes the hardware using initUSB() and initPorts(). The firmware will then attempt to reset the drive and determine its command set by calling resetATAPIDevice(). If no drive is detected, the firmware will continue pulsing the ATA reset line until one is detected. Once a drive is detected, the firmware determines whether it is an IDE or ATAPI device by reading the byte count registers. The scsi flag is set to 1 to indicate an ATAPI device, scsi is set to 0 on an IDE device.

ATAPIIdDevice()

This routine is called to collect information from the drive into internal data structures. This information includes the max PIO speed supported and the serial number of the drive. If the device supports PIO-3, PIO-4 or UDMA, this routine will program the drive to run at the new speed. The serial number must be read before the device connects to USB, since it is one of the descriptors read during USB enumeration.

EZUSB_Discon()

Once the serial number is read, EZUSB_Discon is called to connect to USB (the default condition of "disconnected" is set at reset). EZUSB_Discon() may be called repeatedly if the host doesn't send a SETUP after the device has connected. Once the device is connected, the main while() loop of TD_Poll starts.



TD_Poll()

As in all Cypress Frameworks based code, the main code loop is called TD_Poll(). This routine is called repeatedly until it detects a packet in the OUT buffer. TD_poll() checks the received packet for a valid CBW signature. If one is found, it calls processCBW(). If the packet is not a valid CBW, the device enters a "persistent stall" condition awaiting a device reset. ProcessCBW() calls generalIDEInCommand() or generalIDEOutCommand() depending on the direction flag in the CBW. If the dataTransferLength is non-zero, the readPIO16() or writePIO16() routines are called to pass data directly from the USB buffers to the drive using the GPIF.

SETUP messages are handled in an ISR, so they may be received and responded to at any time. The entire SETUP message will be handled within the ISR, therefore long SETUP traffic will adversely affect disk performance. This is not expected to be an issue since Windows does not use SETUP

packets after enumeration except to clear STALL conditions.

ReadPIO16(), WritePIO16()

These data transfer routines activate the GPIF to move data to/from the FIFO memory to/from the ATA bus. The data is read from the drive to the EP8 buffer. Write data moves from the host through the EP2 buffer.

Resets

The firmware performs a hard reset of the drive on a USB Reset condition. The firmware performs a soft reset of the 8051 on a USB Reset condition

File Descriptions

The FX2 firmware is stored in its own directory. All of the FX2 firmware is contained in the FX2 software directory on the CD.

The purpose of the source files in the **software** directory is shown in the following table:

Filename	Purpose
Dscr.a51	Descriptor table containing product/vendor ID, endpoint descriptions and other information reported to the host on startup.
reset.a51	Assembly routine used to branch to 0 on USB reset.
Startup.a51	Modified Keil startup file that does not initialize any variables.
USBJumpTb.a51	INT2/INT4 vector table. Used by all images.
iic_ata.bat	Batch file used to create the EEPROM image.
atareset.c	Contains hard reset routine, selection of IDE vs ATAPI protocol. Identifies device characteristics, including serial number, capacity and transfer rate. Selects transfer rate by loading new GPIF waveforms.
fw.c	Frameworks based main routine. This fw.c has major differences from the fw.c released with the dev kit, since several implementation-specific functions have been merged with the general startup code in this file.
gpif.c	EZ-USB FX2 low level i/o routines. Waveform descriptors. Routines for loading the GPIF memory with the waveform descriptors.
gpifpio0.c	Output file from the GPIF tool. The GPIF table from this file is manually inserted into the gpif.c file.
gpifpio4.c	Output file from the GPIF tool. The GPIF table from this file is manually inserted into the gpif.c file.
ide.c	Translates SCSI (ATAPI) commands sent by the host driver into IDE commands.
periph.c	TD_Init and TD_Poll(), misc init routines, misc util routines including our smaller version of memmove.
scsi.c	High level data transfer routines for ATAPI devices. (Named SCSI.c because ATAPI devices use the SCSI command set.) Calls low level transfer routines in gpif.c.
atapi.h	Header file containing application specific items.
gpif.h	Header file containing hardware specific items.
scsi.h	SCSI command set
fx2_ata	Debuggable object code image
fx2_ata.hex	Output image (Intel hex format)
fx2_ata.iic	Output image (binary image, ready to be loaded into an EEPROM)

Filename	Purpose
fx2_ata.M51	Map file
fx2_ata.Opt	Options for UV2 project
fx2_ata.Uv2	UV2 project file

Building the Software

Since the software is distributed on a CD, many operating systems will set the read-only flag when copying the data to your local directory. This flag must be turned off before uVision2 will properly build the .hex file. To do this, use "attrib -r *.*" at the DOS command line or select all of the files in Explorer, select "properties" and turn off the "read-only" checkbox in the "general" tab.

This Reference Design has only been tested with the release of the Dev Kit contained on the release CD. Please install the current Dev Kit before building.

The Mass Storage design will NOT compile with the 4K demo version of the Keil tools.

Once the files are no longer read-only, start the full uVision2 environment (available separately from www.keil.com) and click the "build all" button. This will generate an image that can be loaded with the control panel or the debugger. Use the debugger with caution. Once the OS has detected this device as a Mass Storage device, bad or missed responses from the device (breakpoints) will result in a USB Reset in the best case, and a total OS lockup in the worst case.

The code image is currently about 0x1F00 bytes code / 0x240 xdata. The FX2 has 0x2200 (8.5k) of internal code/xdata memory.

Debugging the Software

The CY4611 software will run with the Keil debugger on a CY3681 development board. This is a useful environment for debugging startup issues by single stepping the firmware.

Debugging specific commands requires a different approach because the Mass Storage driver will timeout while you are single-stepping and may lock up or reboot the host machine. The CY4611 firmware can be bound to the Cypress General Purpose Driver by modifying the descriptors in dscr.a51 as follows:

Old

```
70  dw    1146H    ;; Product ID (4611)
104 db    08H     ;; IF class -- Mass
      Storage
```

New

```
70  dw    0210h    ;; PID = Sample dev
104; db    0ffh    ;; Class -- NOT Mass
      Storage -- use for testing
```

Once the firmware is bound to the General Purpose Driver, commands can be sent to the device using the control panel. An easy way to do this is to construct a file containing the command and use the FileTrans button to send it.

- 1) Start the Keil debugger, download your firmware.
- 2) Run the firmware, it will enumerate and bind to the general purpose driver
- 3) Start the control panel.
- 4) Do a "get pipes" on the control panel. This will fill in the pipe fields.
- 5) Select the OUT pipe and hit the FileTrans button.
- 6) Select your command file.
- 7) Manually transfer the IN or OUT data required by the command
- 8) Do a final IN to collect the CSW.

Difference between ATAPI and IDE devices

Although both ATAPI and IDE devices attach to the same 40 pin cable, they operate using different protocols, much like TCP/IP and NetBEUI share the same Ethernet wire, but cannot talk to each other. ATAPI commands are basically SCSI commands sent over an ATA interface.

This firmware will support **both** ATAPI and IDE task file commands. It will detect the type of device after reset. If the device is an IDE device, the ATAPI commands received over USB will be translated into IDE task file commands. One way to gain additional code space is to eliminate one of the supported protocols.

Serial numbers

The USB Mass Storage specification requires that each device has a unique serial number. The board manufacturer will have to initialize this serial number during the manufacturing process. This reference design initializes the serial number string to the drive's serial number, which is frequently not compliant with the spec requirement that the serial number string be hex digits (0-9 A-F). In some cases, this serial number may not even be unique in the range selected by the firmware.

Hardware Notes – EZ-USB FX2 design

The FX2 design takes advantage of its internal GPIF (General Programmable InterFace) to move data from the endpoint buffers to the mass storage device. For more details on the EZ-USB FX2 and GPIF, see the EZ-USB FX2 Technical Reference Manual and the UDMA white paper on this CD.

The GPIF is used to create several different PIO waveforms. There are two waveforms for word (16 bit) data transfers, one for read and one for write. These waveforms transfer data between the FIFOs and the drive. They generally operate on full 512 byte packets. There are also two waveforms for byte (8 bit) GPIF waveforms, one for read and one for write. These waveforms are used for register reading and writing.

The firmware initially uses PIO mode 0 to identify the device. The firmware uses this data to determine the maximum transfer speed (PIO or UDMA mode). After the mode is selected, a new set of GPIF waveforms is loaded into the GPIF waveform memory to increase the read/write waveform speed.

The gpifpio0.c and gpifpio4.c files can be loaded into the GPIF tool to modify or examine the PIO waveforms that are used in this design.

Power

The reference design can be powered from the USB bus for convenience (with JP2 inserted), but the only configuration that is compliant with the USB spec is to power both the USB interface and the drive electronics from an external supply with JP2 removed.

References

USB Mass Storage Class – Bulk Only Transport,
USB Mass Storage DWG. (www.usb.org)
USB Mass Storage Class – Overview Specification,
USB Mass Storage DWG. (www.usb.org)
USB Specification – Revision 2.0, USB
Implementers Forum. (www.usb.org)
EZ-USB FX2 Technical Reference Manual, Revision
2.1, Cypress (www.cypress.com)
ATA/ATAPI-6 Specification, Proposed ANSI
Standard (www.t13.org).
SCSI-3 Specification (www.t10.org)

Document Revision History

Revision #	Date	Comments
2.10	2/1/01	Updated for new board, added flowchart
2.09	12/1/01	Updated for final release.
2.0B8	8/15/01	Minor typographical and technical corrections.
2.0B7	7/1/01	Added information about unified code image.
2.0B5	5/20/01	Added DVD support info
2.0B1	3/26/01	Revised for Beta release. Added more file descriptions Added build instructions
2.0	11/29/00	Initial Release