Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

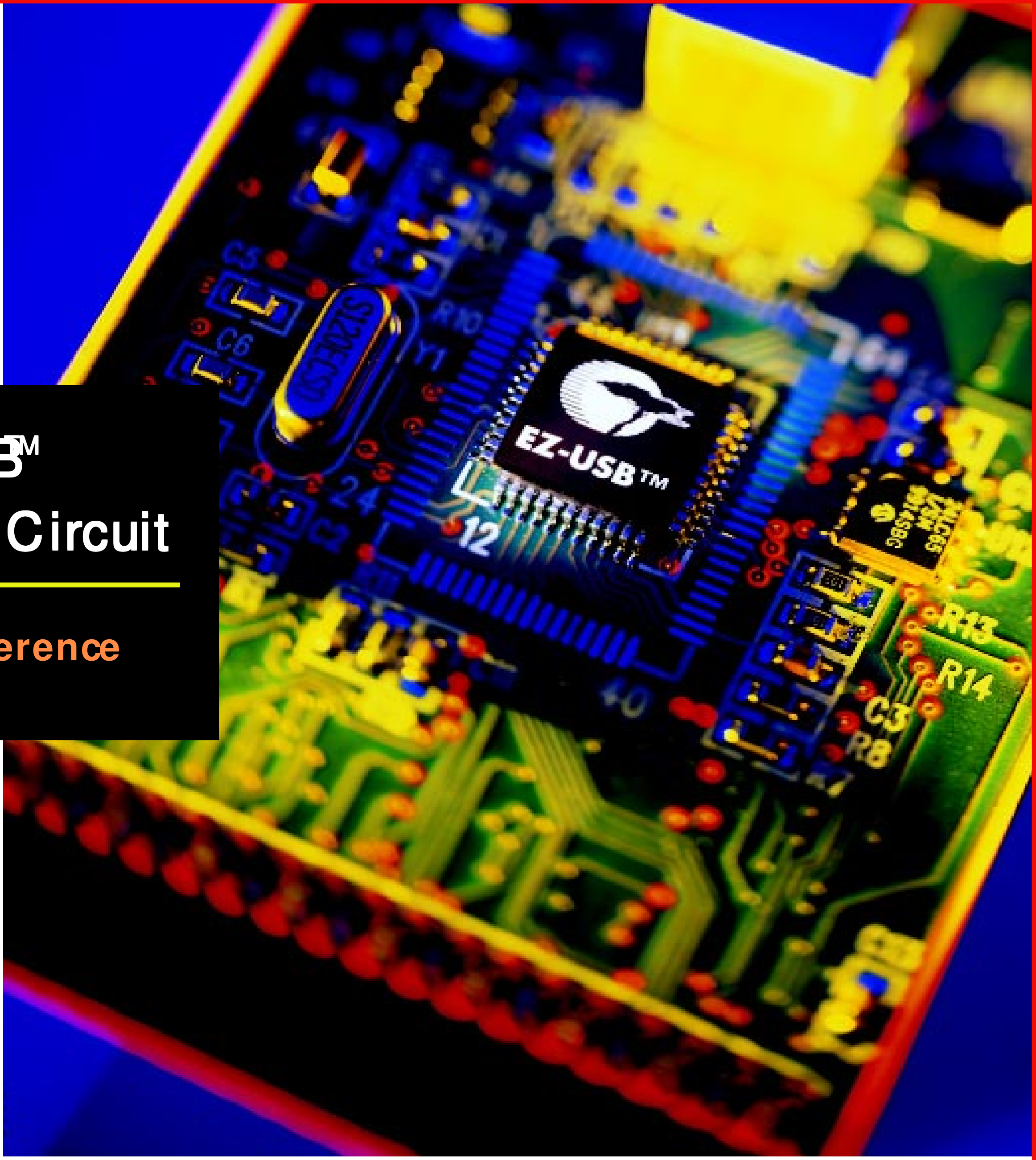
We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

The EZ-USB™ Integrated Circuit

Technical Reference



CYPRESS

Development Kit — Getting Started

Documentation for the EZ-USB™ Xcelerator™ Development it. Includes an overview of the kit, descriptions of kit components with installation instructions, and details about the development board.

Technical Reference

Documentation of the EZ-USB controller. Includes details about the CPU, memory, input/output, ReNumeration™, bulk transfers, endpoint zero, isochronous transfers, interrupts, resets, power management, registers, AC/DC parameters, and packages.

Appendices

Documentation for the 8051 enhanced core. Includes an introduction, an architectural overview, and a hardware description.

Registers

EZ-USB register maps.

Technical Support:
Phone: (858) 613-7929
E-mail: usbapps@cypress.com
Website:
www.cypress.com

EZ-USB
Technical Reference Manual
Version 1.9
May 2000

EZ-USB
Technical Reference Manual

Table of Contents

Table of Contents i
Figures vii
Tables xi

1 Introducing EZ-USB 1-1

1.1 Introduction 1-1
1.2 EZ-USB Block Diagrams 1-2
1.3 The USB Specification 1-3
1.4 Tokens and PIDs 1-4
1.5 Host is Master 1-5
1.5.1 Receiving Data from the Host 1-6
1.5.2 Sending Data to the Host 1-6
1.6 USB Direction 1-6
1.7 Frame 1-6
1.7.1 Bulk Transfers 1-7
1.7.2 Interrupt Transfers 1-7
1.8 EZ-USB Transfer Types 1-7
1.8.1 Isochronous Transfers 1-8
1.8.2 Control Transfers 1-8
1.9 Enumeration 1-9
1.10 The USB Core 1-10
1.11 EZ-USB Microprocessor 1-11
1.12 ReNumeration‘ 1-12
1.13 EZ-USB Endpoints 1-12
1.13.1 EZ-USB Bulk Endpoints 1-13
1.13.2 EZ-USB Control Endpoint Zero 1-13
1.13.3 EZ-USB Interrupt Endpoints 1-14
1.13.4 EZ-USB Isochronous Endpoints 1-14
1.14 Fast Transfer Modes 1-14
1.15 Interrupts 1-15

1.16 Reset and Power Management 1-15
1.17 EZ-USB Product Family 1-16
1.18 Summary of AN2122, AN2126 Features 1-16
1.19 Revision ID 1-17
1.20 Pin Descriptions 1-18

2 EZ-USB CPU 2-1

2.1 Introduction 2-1
2.2 8051 Enhancements 2-1
2.3 EZ-USB Enhancements 2-2
2.4 EZ-USB Register Interface 2-2
2.5 EZ-USB Internal RAM 2-3
2.6 I/O Ports 2-3
2.7 Interrupts 2-4
2.8 Power Control 2-5
2.9 SFRs 2-6
2.10 Internal Bus 2-7
2.11 Reset 2-7

3 EZ-USB Memory 3-1

3.1 Introduction 3-1
3.2 8051 Memory 3-2
3.3 Expanding EZ-USB Memory 3-4
3.4 CS# and OE# Signals 3-5
3.5 EZ-USB ROM Versions 3-7

4 EZ-USB Input/Output 4-1

4.1 Introduction 4-1
4.2 IO Ports 4-2
4.3 IO Port Registers 4-5
4.4 I2C Controller 4-6
4.5 8051 I2C Controller 4-6
4.5.1 START 4-8
4.5.2 STOP 4-8
4.6 Control Bits 4-8
4.6.1 LASTRD 4-9
4.6.2 DONE 4-9
4.6.3 ACK 4-9
4.7 Status Bits 4-9
4.7.1 BERR 4-10

4.7.2 ID1, ID0 4-10
4.8 Sending I2C Data 4-10
4.9 Receiving I2C Data 4-11
4.10 I2C Boot Loader 4-12

5 EZ-USB Enumeration and ReNumeration 5-1

5.1 Introduction 5-1
5.2 The Default USB Device 5-2
5.3 EZ-USB Core Response to EP0 Device Requests 5-4
5.4 Firmware Load 5-5
5.5 Enumeration Modes 5-7
5.6 No Serial EEPROM 5-8
5.7 Serial EEPROM Present, First Byte is 0xB0 5-9
5.8 Serial EEPROM Present, First Byte is 0xB2 5-10
5.9 ReNumeration‘ 5-11
5.10 Multiple ReNumerations‘ 5-13
5.11 Default Descriptor 5-13

6 EZ-USB Bulk Transfers 6-1

6.1 Introduction 6-1
6.2 Bulk IN Transfers 6-4
6.3 Interrupt Transfers 6-5
6.4 EZ-USB Bulk IN Example 6-5
6.5 Bulk OUT Transfers 6-6
6.6 Endpoint Pairing 6-8
6.7 Paired IN Endpoint Status 6-9
6.8 Paired OUT Endpoint Status 6-10
6.9 Using Bulk Buffer Memory 6-10
6.10 Data Toggle Control 6-11
6.11 Polled Bulk Transfer Example 6-13
6.12 Enumeration Note 6-14
6.13 Bulk Endpoint Interrupts 6-15
6.14 Interrupt Bulk Transfer Example 6-16
6.15 Enumeration Note 6-21
6.16 The Autopointer 6-22

7 EZ-USB Endpoint Zero 7-1

7.1 Introduction 7-1
7.2 Control Endpoint EP0 7-2
7.3 USB Requests 7-5

7.3.1 Get Status 7-7
7.3.2 Set Feature 7-10
7.3.3 Clear Feature 7-12
7.3.4 Get Descriptor 7-12
7.3.4.1 Get Descriptor-Device 7-14
7.3.4.2 Get Descriptor-Configuration 7-15
7.3.4.3 Get Descriptor-String 7-16
7.3.5 Set Descriptor 7-16
7.3.6 Set Configuration 7-19
7.3.7 Get Configuration 7-19
7.3.8 Set Interface 7-20
7.3.9 Get Interface 7-21
7.3.10 Set Address 7-21
7.3.11 Sync Frame 7-22
7.3.12 Firmware Load 7-23

8 EZ-USB Isochronous Transfers 8-1

8.1 Introduction 8-1
8.1.1 Initialization 8-2
8.2 Isochronous IN Transfers 8-2
8.2.1 IN Data Transfers 8-3
8.3 Isochronous OUT Transfers 8-3
8.3.1 Initialization 8-4
8.3.2 OUT Data Transfer 8-4
8.4 Setting Isochronous FIFO Sizes 8-5
8.5 Isochronous Transfer Speed 8-8
8.6 Fast Transfers 8-9
8.6.1 Fast Writes 8-10
8.6.2 Fast Reads 8-11
8.7 Fast Transfer Timing 8-11
8.7.1 Fast Write Waveforms 8-12
8.7.2 Fast Read Waveforms 8-13
8.8 Fast Transfer Speed 8-14
8.8.1 Disable ISO 8-15
8.9 Other Isochronous Registers 8-15
8.9.1 Zero Byte Count Bits 8-16
8.10 ISO IN Response with No Data 8-17
8.11 Using the Isochronous FIFOs 8-17

9 EZ-USB Interrupts 9-1

9.1 Introduction 9-1

9.2 USB Core Interrupts 9-1
9.3 Wakeup Interrupt 9-2
9.4 USB Signaling Interrupts 9-4
9.5 SUTOK, SUDAV Interrupts 9-8
9.6 SOF Interrupt 9-9
9.7 Suspend Interrupt 9-9
9.8 USB RESET Interrupt 9-9
9.9 Bulk Endpoint Interrupts 9-9
9.10 USB Autovectors 9-10
9.11 Autovector Coding 9-11
9.12 I^{2}C Interrupt 9-13
9.13 In Bulk NAK Interrupt - (AN2122/AN2126 only) 9-13
9.14 I^{2}C STOP Complete Interrupt - (AN2122/AN2126 only) 9-15

10 EZ-USB Resets 10-1

10.1 Introduction 10-1
10.2 EZ-USB Power-On Reset (POR) 10-1
10.3 Releasing the 8051 Reset 10-3
10.3.1 RAM Download 10-4
10.3.2 EEPROM Load 10-4
10.3.3 External ROM 10-4
10.4 8051 Reset Effects 10-4
10.5 USB Bus Reset 10-5
10.6 EZ-USB Disconnect 10-7
10.7 Reset Summary 10-8

11 EZ-USB Power Management 11-1

11.1 Introduction 11-1
11.2 Suspend 11-2
11.3 Resume 11-3
11.4 Remote Wakeup 11-4

12 EZ-USB Registers 12-1

12.1 Introduction 12-1
12.2 Bulk Data Buffers 12-3
12.3 Isochronous Data FIFOs 12-4
12.4 Isochronous Byte Counts 12-6
12.5 CPU Registers 12-8
12.6 Port Configuration 12-9
12.7 Input-Output Port Registers 12-11

12.8 230-Kbaud UART Operation - AN2122, AN2126 12-14
12.9 Isochronous Control/Status Registers 12-14
12.10 I^{2}C Registers 12-16
12.11 Interrupts 12-19
12.12 Endpoint 0 Control and Status Registers 12-29
12.13 Endpoint 1-7 Control and Status Registers 12-31
12.14 Global USB Registers 12-37
12.15 Fast Transfers 12-46
12.16 SETUP Data 12-49
12.17 Isochronous FIFO Sizes 12-50

13 EZ-USB AC/DC Parameters 13-1

13.0.1 Absolute Maximum Ratings 13-1
13.0.2 Operating Conditions 13-1
13.0.3 DC Characteristics 13-1

13.1 Electrical Characteristics 13-1

13.1.1 AC Electrical Characteristics 13-2
13.1.2 General Memory Timing 13-2
13.1.3 Program Memory Read 13-2
13.1.4 Data Memory Read 13-2
13.1.5 Data Memory Write 13-3
13.1.6 Fast Data Write 13-3
13.1.7 Fast Data Read 13-3

14 EZ-USB Packaging 14-1

14.1 44-Pin PQFP Package 14-1
14.2 80-Pin PQFP Package 14-3
14.3 48-Pin TQFP Package 14-5

Figures

Figure 1-1. AN2131S (44 pin) Simplified Block Diagram 1-2
Figure 1-2. AN2131Q (80 pin) Simplified Block Diagram 1-3
Figure 1-3. USB Packets 1-4
Figure 1-4. Two Bulk Transfers, IN and OUT 1-7
Figure 1-5. An Interrupt Transfer 1-7
Figure 1-6. An Isochronous Transfer 1-8
Figure 1-7. A Control Transfer 1-8
Figure 1-8. What the SIE Does 1-10
Figure 1-9. 80-pin PQFP Package (AN2131Q) 1-18
Figure 1-10. 44-pin PQFP Package with Port B (AN2121S, AN2122S, and AN2131S) 1-19
Figure 1-11. 44-pin Package with Data Bus (AN2125S, AN2126S, AN2135S, and AN2136) 1-20
Figure 1-12. 48-pin TQFP Package (AN2122T) 1-21
Figure 1-13. 48-pin TQFP Package (AN2126T) 1-22
Figure 2-1. 8051 Registers 2-3
Figure 3-1. EZ-USB 8-KB Memory Map - Addresses are in Hexadecimal 3-1
Figure 3-2. EZ-USB 4-KB Memory Map - Addresses are in Hexadecimal 3-1
Figure 3-3. Unused Bulk Endpoint Buffers (Shaded) Used as Data Memory 3-3
Figure 3-4. EZ-USB Memory Map with EA=0 3-4
Figure 3-5. EZ-USB Memory Map with EA=1 3-6
Figure 3-6. 8-KB ROM, 2-KB RAM Version 3-7
Figure 3-7. 32-KB ROM, 4-KB RAM Version 3-8
Figure 4-1. EZ-USB Input/Output Pin 4-2
Figure 4-2. Alternate Function is an OUTPUT 4-4
Figure 4-3. Alternate Function is an INPUT 4-4
Figure 4-4. Registers Associated with PORTS A, B, and C 4-5
Figure 4-5. General I2C Transfer 4-6
Figure 4-6. General FC Transfer 4-7
Figure 4-7. FC Registers 4-8
Figure 5-1. USB Control and Status Register 5-11
Figure 5-2. Disconnect Pin Logic 5-12
Figure 5-3. Typical Disconnect Circuit (DISCOE=1) 5-12
Figure 6-1. Two BULK Transfers, IN and OUT 6-1
Figure 6-2. Registers Associated with Bulk Endpoints 6-3
Figure 6-3. Anatomy of a Bulk IN Transfer 6-4
Figure 6-4. Anatomy of a Bulk OUT Transfer 6-7
Figure 6-5. Bulk Endpoint Toggle Control 6-11
Figure 6-6. Example Code for a Simple (Polled) BULK Transfer 6-14
Figure 6-7. Interrupt Jump Table 6-18
Figure 6-8. INT2 Interrupt Vector 6-19

Figure 6-9. Interrupt Service Routine (ISR) for Endpoint 6-OUT 6-19
Figure 6-10. Background Program Transfers Endpoint 6-OUT Data to Endpoint 6-IN 6-20
Figure 6-11. Initialization Routine 6-21
Figure 6-12. Autopointer Registers 6-23
Figure 6-13. Use of the Autopointer 6-24
Figure 6-14. 8051 Code to Transfer External Data to a Bulk IN Buffer 6-25
Figure 7-1. A USB Control Transfer (This One Has a Data Stage) 7-2
Figure 7-2. The Two Interrupts Associated with EP0 CONTROL Transfers 7-3
Figure 7-3. Registers Associated with EP0 Control Transfers 7-4
Figure 7-4. Data Flow for a Get_Status Request 7-7
Figure 7-5. Using the Setup Data Pointer (SUDPTR) for Get_Descriptor Requests 7-13
Figure 8-1. EZ-USB Isochronous Endpoints 8-15 8-1
Figure 8-2. Isochronous IN Endpoint Registers 8-2
Figure 8-3. Isochronous OUT Registers 8-4
Figure 8-4. FIFO Start Address Format 8-5
Figure 8-5. Assembler Translates FIFO Sizes to Addresses 8-7
Figure 8-6. 8051 Code to Transfer Data to an Isochronous FIFO (IN8DATA) 8-8
Figure 8-7. 8051 MOVX Instructions 8-9
Figure 8-8. Fast Transfer, EZ-USB to Outside Memory 8-10
Figure 8-9. Fast Transfer, Outside Memory to EZ-USB 8-11
Figure 8-10. The FASTXFR Register Controls FRD# and FWR# Strobes 8-11
Figure 8-11. Fast Write Timing 8-12
Figure 8-12. Fast Read Timing 8-13
Figure 8-13. 8051 Code to Transfer 640 Bytes of External Data to an Isochronous
IN FIFO 8-14
Figure 8-14. ISOCTL Register 8-15
Figure 8-15. ZBCOUT Register 8-16
Figure 9-1. EZ-USB Wakeup Interrupt 9-2
Figure 9-2. USB Interrupts 9-4
Figure 9-3. The Order of Clearing Interrupt Requests is Important 9-6
Figure 9-4. EZ-USB Interrupt Registers 9-7
Figure 9-5. SUTOK and SUDAV Interrupts 9-8
Figure 9-6. A Start Of Frame (SOF) Packet 9-9
Figure 9-7. The Autovector Mechanism in Action 9-12
Figure 9-8. I^{2}C Interrupt Enable Bits and Registers 9-13
Figure 9-9. IN Bulk NAK Interrupt Request Register 9-14
Figure 9-10. IN Bulk NAK Interrupt Enable Register 9-14
Figure 9-11. I^{2}C Mode Register 9-15
Figure 9-12. I^{2}C Control and Status Register 9-15
Figure 9-13. I^{2}C Data 9-15
Figure 10-1. EZ-USB Resets 10-1
Figure 11-1. Suspend-Resume Control 11-1
Figure 11-2. EZ-USB Suspend Sequence 11-2
Figure 11-3. EZ-USB Resume Sequence 11-3
Figure 11-4. USB Control and Status Register 11-4

Figure 12-1. Register Description Format 12-2
Figure 12-2. Bulk Data Buffers 12-3
Figure 12-3. Isochronous Data FIFOs 12-4
Figure 12-4. Isochronous Byte Counts 12-6
Figure 12-5. CPU Control and Status Register 12-8
Figure 12-6. IO Port Configuration Registers 12-9
Figure 12-7. Output Port Configuration Registers 12-11
Figure 12-8. PINSn Registers 12-12
Figure 12-9. Output Enable Registers 12-13
Figure 12-10. 230-Kbaud UART Operation Register 12-14
Figure 12-11. Isochronous OUT Endpoint Error Register 12-14
Figure 12-12. Isochronous Control Register 12-15
Figure 12-13. Zero Byte Count Register 12-15
Figure 12-14. I^2C Transfer Registers 12-16
Figure 12-15. I^2C Mode Register 12-18
Figure 12-16. Interrupt Vector Register 12-19
Figure 12-17. IN/OUT Interrupt Request (IRQ) Registers 12-20
Figure 12-18. USB Interrupt Request (IRQ) Registers 12-21
Figure 12-19. IN/OUT Interrupt Enable Registers 12-23
Figure 12-20. USB Interrupt Enable Register 12-24
Figure 12-21. Breakpoint and Autovector Register 12-26
Figure 12-22. IN Bulk NAK Interrupt Request Register 12-27
Figure 12-23. IN Bulk NAK Interrupt Enable Register 12-27
Figure 12-24. IN/OUT Interrupt Enable Registers 12-28
Figure 12-25. Port Configuration Registers 12-29
Figure 12-26. IN Control and Status Registers 12-32
Figure 12-27. IN Byte Count Registers 12-34
Figure 12-28. OUT Control and Status Registers 12-35
Figure 12-29. OUT Byte Count Registers 12-36
Figure 12-30. Setup Data Pointer High/Low Registers 12-37
Figure 12-31. USB Control and Status Registers 12-38
Figure 12-32. Data Toggle Control Register 12-40
Figure 12-33. USB Frame Count High/Low Registers 12-41
Figure 12-34. Function Address Register 12-42
Figure 12-35. USB Endpoint Pairing Register 12-43
Figure 12-36. IN/OUT Valid Bits Register 12-44
Figure 12-37. Isochronous IN/OUT Endpoint Valid Bits Register 12-45
Figure 12-38. Fast Transfer Control Register 12-46
Figure 12-39. Auto Pointer Registers 12-48
Figure 12-40. SETUP Data Buffer 12-49
Figure 12-41. SETUP Data Buffer 12-50
Figure 13-1. External Memory Timing 13-4
Figure 13-2. Program Memory Read Timing 13-4
Figure 13-3. Data Memory Read Timing 13-5
Figure 13-4. Data Memory Write Timing 13-5

Figure 13-5. Fast Transfer Mode Block Diagram 13-6
Figure 13-6. Fast Transfer Read Timing [Mode 00] 13-7
Figure 13-7. Fast Transfer Write Timing [Mode 00] 13-7
Figure 13-8. Fast Transfer Read Timing [Mode 01] 13-8
Figure 13-9. Fast Transfer Write Timing [MODE 01] 13-8
Figure 13-10. Fast Transfer Read Timing [Mode 10] 13-9
Figure 13-11. Fast Transfer Write Timing [Mode 10] 13-9
Figure 13-12. Fast Transfer Read Timing [Mode 11] 13-10
Figure 13-13. Fast Transfer Write Timing [Mode 11] 13-10
Figure 14-1. 44-Pin PQFP Package (Top View) 14-1
Figure 14-2. 44-Pin PQFP Package (Side View) 14-1
Figure 14-3. 44-Pin PQFP Package (Detail View) 14-2
Figure 14-4. 80-Pin PQFP Package (Top View) 14-3
Figure 14-5. 80-Pin PQFP Package (Side View) 14-3
Figure 14-6. 80-Pin PQFP Package (Detail View) 14-4
Figure 14-7. 48-Pin TQFP Package (Side View) 14-5
Figure 14-8. 48-Pin TQFP Package (Top View) 14-5
Figure 14-9. 48-Pin TQFP Package (Detail View) 14-6

Tables

Table 1-1. USB PIDs 1-4
Table 1-2. EZ-USB Series 2100 Family 1-16
Table 1-3. EZ-USB Series 2100 Pinouts by Pin Function 1-23
Table 2-1. EZ-USB Interrupts 2-4
Table 2-2. Added Registers and Bits 2-6
Table 4-1. IO Pin Functions for PORTxCFG=0 and PORTxCFG=1 4-3
Table 4-2. Strap Boot EEPROM Address Lines to These Values 4-13
Table 4-3. Results of Power-On I2C Test 4-14
Table 5-1. EZ-USB Default Endpoints 5-2
Table 5-2. How the EZ-USB Core Handles EP0 Requests When ReNum=0 5-4
Table 5-3. Firmware Download 5-5
Table 5-4. Firmware Upload 5-6
Table 5-5. EZ-USB Core Action at Power-Up 5-7
Table 5-6. EZ-USB Device Characteristics, No Serial EEPROM 5-8
Table 5-7. EEPROM Data Format for “B0” Load 5-9
Table 5-8. EEPROM Data Format for “B2” Load 5-10
Table 5-9. USB Default Device Descriptor 5-13
Table 5-10. USB Default Configuration Descriptor 5-14
Table 5-11. USB Default Interface 0, Alternate Setting 0 Descriptor 5-14
Table 5-12. USB Default Interface 0, Alternate Setting 1 Descriptor 5-15
Table 5-13. USB Default Interface 0, Alternate Setting 1, Interrupt Endpoint Descriptor.. 5-15
Table 5-14. USB Default Interface 0, Alternate Setting 1, Bulk Endpoint Descriptors 5-16
Table 5-14. USB Default Interface 0, Alternate Setting 1, Bulk Endpoint Descriptors 5-17
Table 5-15. USB Default Interface 0, Alternate Setting 1, Isochronous Endpoint Descriptors 5-18
Table 5-16. USB Default Interface 0, Alternate Setting 2 Descriptor 5-19
Table 5-17. USB Default Interface 0, Alternate Setting 1, Interrupt Endpoint Descriptor.. 5-19
Table 5-18. USB Default Interface 0, Alternate Setting 2, Bulk Endpoint Descriptors 5-20
Table 5-19. USB Default Interface 0, Alternate Setting 2, Isochronous Endpoint Descriptors 5-21
Table 6-1. EZ-USB Bulk, Control, and Interrupt Endpoints 6-1
Table 6-2. Endpoint Pairing Bits (in the USB PAIR Register) 6-8
Table 6-3. EZ-USB Endpoint 0-7 Buffer Addresses 6-10
Table 6-4. 8051 INT2 Interrupt Vector 6-16
Table 6-5. Byte Inserted by EZ-USB Core at Location 0x45 if AVEN=1 6-16

Table 7-1. The Eight Bytes in a USB SETUP Packet 7-5
Table 7-2. How the 8051 Handles USB Device Requests (ReNum=1) 7-6
Table 7-3. Get Status-Device (Remote Wakeup and Self-Powered Bits) 7-8
Table 7-4. Get Status-Endpoint (Stall Bits) 7-8
Table 7-5. Get Status-Interface 7-10
Table 7-6. Set Feature-Device (Set Remote Wakeup Bit) 7-10
Table 7-7. Set Feature-Endpoint (Stall) 7-11
Table 7-8. Clear Feature-Device (Clear Remote Wakeup Bit) 7-12
Table 7-9. Clear Feature-Endpoint (Clear Stall) 7-12
Table 7-10. Get Descriptor-Device 7-14
Table 7-11. Get Descriptor-Configuration 7-15
Table 7-12. Get Descriptor-String 7-16
Table 7-13. Set Descriptor-Device 7-16
Table 7-14. Set Descriptor-Configuration 7-17
Table 7-15. Set Descriptor-String 7-17
Table 7-16. Set Configuration 7-19
Table 7-17. Get Configuration 7-19
Table 7-18. Set Interface (Actually, Set Alternate Setting AS for Interface IF) 7-20
Table 7-19. Get Interface (Actually, Get Alternate Setting AS for interface IF) 7-21
Table 7-20. Sync Frame 7-22
Table 7-21. Firmware Download 7-23
Table 7-22. Firmware Upload 7-23
Table 8-1. Isochronous Endpoint FIFO Starting Address Registers 8-6
Table 8-2. Addresses for RD# and WR# vs. ISODISAB bit 8-15
Table 9-1. EZ-USB Interrupts 9-1
Table 9-2. 8051 JUMP Instruction 9-10
Table 9-3. A Typical USB Jump Table 9-11
Table 10-1. EZ-USB States After Power-On Reset (POR) 10-2
Table 10-2. EZ-USB States After a USB Bus Reset 10-6
Table 10-3. Effects of an EZ-USB Disconnect and Re-connect 10-7
Table 10-4. Effects of Various EZ-USB Resets (“U” Means “Unaffected”) 10-8
Table 12-1. Bulk Endpoint Buffer Memory Addresses 12-3
Table 12-2. Isochronous Endpoint FIFO Register Addresses 12-4
Table 12-3. Isochronous Endpoint Byte Count Register Addresses 12-6
Table 12-4. IO Pin Alternate Functions 12-10
Table 12-5. Control and Status Register Addresses for Endpoints 0-7 12-31
Table 12-6. Isochronous FIFO Start Address Registers 12-51
Table 13-1. DC Characteristics 13-1
Table 13-2. General Memory Timing 13-2

Table 13-3. Program Memory Read 13-2
Table 13-4. Data Memory Read 13-2
Table 13-5. Data Memory Write 13-3
Table 13-6. Fast Data Write 13-3
Table 13-7. Fast Data Read 13-3

1 Introducing EZ-USB

1.1 Introduction

Like a well designed automobile or appliance, a USB peripheral's outward simplicity hides internal complexity. There's a lot going on "under the hood" of a USB device, which gives the user a new level of convenience. For example:

- A USB device can be plugged in anytime, even when the PC is turned on.
- When the PC detects that a USB device has been plugged in, it automatically interrogates the device to learn its capabilities and requirements. From this information, the PC automatically loads the device's driver into the operating system. When the device is unplugged, the operating system automatically logs it off and unloads its driver.
- USB devices do not use DIP switches, jumpers, or configuration programs. There is never an IRQ, DMA, MEMORY, or IO conflict with a USB device.
- USB expansion hubs make the bus available to dozens of devices.
- USB is fast enough for printers, CD-quality audio, and scanners.

USB is defined in the *Universal Serial Bus Specification Version 1.1* (**http://usb.org**), a 268-page document that describes all aspects of a USB device in elaborate detail. This EZ-USB Technical Reference Manual describes the EZ-USB chip along with USB topics that should provide help in understanding the Specification.

The Cypress Semiconductor EZ-USB is a compact integrated circuit that provides a highly integrated solution for a USB peripheral device. Three key EZ-USB features are:

- The EZ-USB family provides a *soft* (RAM-based) solution that allows unlimited configuration and upgrades.
- The EZ-USB family delivers full USB throughput. Designs that use EZ-USB are not limited by number of endpoints, buffer sizes, or transfer speeds.
- The EZ-USB family does much of the USB housekeeping in the EZ-USB core, simplifying code and accelerating the USB learning curve.

This chapter introduces some key USB concepts and terminology that should make reading the rest of this Technical Reference Manual easier.

1.2 EZ-USB Block Diagrams

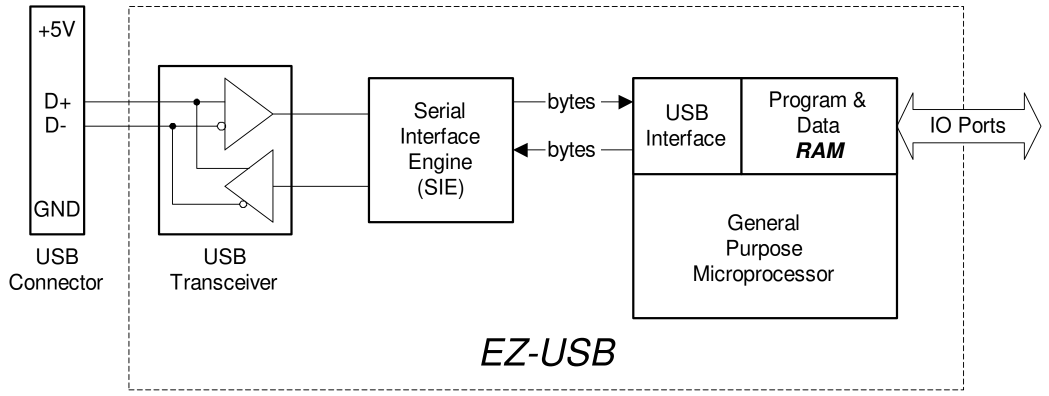


Figure 1-1. AN2131S (44 pin) Simplified Block Diagram

The Cypress Semiconductor EZ-USB chip packs the intelligence required by a USB peripheral interface into a compact integrated circuit. As Figure 1-1 illustrates, an integrated USB transceiver connects to the USB bus pins D+ and D-. A Serial Interface Engine (SIE) decodes and encodes the serial data and performs error correction, bit stuffing, and other signaling-level details required by USB, and ultimately transfers data bytes to and from the USB interface.

The internal microprocessor is enhanced 8051 with fast execution time and added features. It uses internal RAM for program and data storage, making the EZ-USB family a *soft* solution. The USB host downloads 8051 program code and device personality into RAM over the USB bus, and then the EZ-USB chip re-connects as the custom device as defined by the loaded code.

The EZ-USB family uses an enhanced SIE/USB interface (called the “USB Core”) which has the intelligence to function as a full USB device even before the 8051. The enhanced core simplifies 8051 code by implementing much of the USB protocol itself.

EZ-USB chips operate at 3.3V. This simplifies the design of bus-powered USB devices, since the 5V power available in the USB connector (which the USB specification allows to be as low as 4.4V) can drive a 3.3V regulator to deliver clean isolated power to the EZ-USB chip.

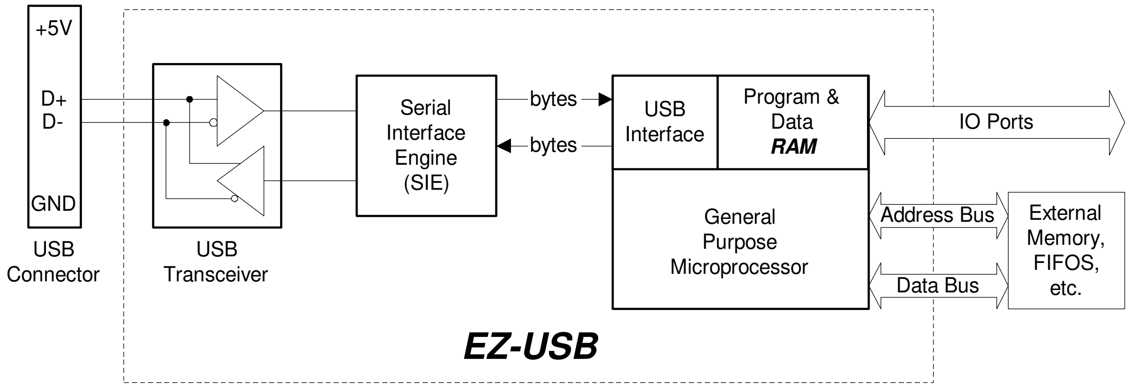


Figure 1-2. AN2131Q (80 pin) Simplified Block Diagram

Figure 1-2 illustrates the An2131Q, an 80-pin version of the EZ-USB family. In addition to the 24 IO pins, it contains a 16-bit address bus and an 8-bit data bus for external memory expansion.

A special *fast transfer* mode moves data directly between external logic and internal USB FIFOs. The fast transfer mode, along with abundant endpoint resources, allows the EZ-USB family to support transfer bandwidths beyond the maximum required by the *Universal Serial Bus Specification Version 1.1.*

1.3 The USB Specification

The *Universal Serial Bus Specification Version 1.1* is available on the Internet at **http://usb.org**. Published in January 1998, the specification is the work of a founding committee of seven industry heavyweights: Compaq, DEC, IBM, Intel, Microsoft, NEC, and Northern Telecom. This impressive list of implementers secures USB as the low to medium speed PC connection method of the future.

A glance at the USB Specification makes it immediately apparent that USB is not nearly as simple as the customary serial or parallel port. The specification uses new terms like "endpoint," isochronous," and "enumeration," and finds new uses for old terms like "configuration," "interface," and "interrupt." Woven into the USB fabric is a software abstraction model that deals with things such as "pipes." The specification also contains detail about the connector types and wire colors.

1.4 Tokens and PIDs

In this manual, you will read statements like, “When the host sends an IN token...” or “The device responds with an ACK.” What do these terms mean? A USB transaction consists of data packets identified by special codes called Packet IDs or PIDs. A PID signifies what kind of packet is being transmitted. There are four PID types, as shown in Table 1-1.

Table 1-1. USB PIDs

PID Type	PID Name
Token Data	IN, OUT, SOF, SETUP, DATA0, DATA1
Handshake	ACK, NAK, STALL
Special	PRE

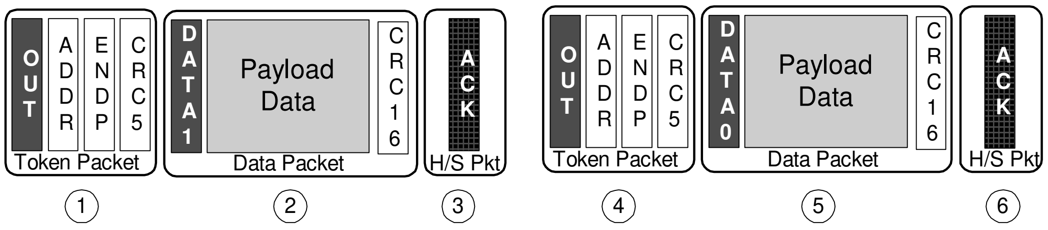


Figure 1-3. USB Packets

Figure 1-3 illustrates a USB transfer. Packet j is an OUT token, indicated by the OUT PID. The OUT token signifies that data from the host is about to be transmitted over the bus. Packet !k contains data, as indicated by the DATA1 PID. Packet l is a handshake packet, sent by the device using the ACK (acknowledge) PID to signify to the host that the device received the data error-free.

Continuing with Figure 1-3, a second transaction begins with another OUT token m, followed by more data n, this time using the DATA0 PID. Finally, the device again indicates success by transmitting the ACK PID in a handshake packet o.

Why two DATA PIDs, DATA0 and DATA1? It’s because the USB architects took error correction very seriously. As mentioned previously, the ACK handshake is a signal to the host that the peripheral received data without error (the CRC portion of the packet is used to detect errors). But what if a handshake packet itself is garbled in transmission? To detect this, each side, host and device maintains a *data toggle* bit, which is toggled between data packet transfers. The state of this internal toggle bit is compared with the

PID that arrives with the data, either DATA0 or DATA1. When sending data, the host or device sends alternating DATA0-DATA1 PIDs. By comparing the Data PID with the state of the internal toggle bit, the host or device can detect a corrupted handshake packet.

SETUP tokens are unique to CONTROL transfers. They preface eight bytes of data from which the peripheral decodes host Device Requests.

SOF tokens occur once per millisecond, denoting a USB *frame*.

There are three handshake PIDs: ACK, NAK, and STALL.

- ACK means "success;" the data was received error-free.
- NAK means "busy, try again." It's tempting to assume that NAK means "error," but it doesn't. A USB device indicates an error by *not responding*.
- STALL means that something unforeseen went wrong (probably as a result of mis-communication or lack of cooperation between the software and firmware writers). A device sends the STALL handshake to indicate that it doesn't understand a device request, that something went wrong on the peripheral end, or that the host tried to access a resource that isn't there. It's like "halt," but better, because USB provides a way to recover from a stall.

A PRE (Preamble) PID precedes a low-speed (1.5 Mbps) USB transmission. The EZ-USB family supports high-speed (12 Mbps) USB transfers only, so it ignores PRE packets and the subsequent low-speed transfer.

1.5 Host is Master

This is a fundamental USB concept. There is exactly one master in a USB system: the host computer. *USB devices respond to host requests*. USB devices cannot send information between themselves, as they could if USB were a peer-to-peer topology.

Actually, there is one case where a USB device can initiate signaling without prompting from the host. After being put into a low-power suspend mode by the host, a device can signal a remote wakeup. But that's the only way to "yank the host's chain." Everything else happens because the host makes device requests and the device responds to them.

There's an excellent reason for this host-centric model. The USB architects were keenly mindful of cost, and the best way to make low-cost peripherals is to put most of the smarts into the host side, the PC. If USB had been defined as peer-to-peer, every USB device would have required more intelligence, raising cost.

Here are two important consequences of the “host is master” concept:

1.5.1 Receiving Data from the Host

To send data to a USB peripheral, the host issues an OUT token followed by the data. If the peripheral has space for the data, and accepts it without error, it returns an ACK to the host. If it is busy, it instead sends a NAK. If it finds an error, it sends nothing back. For the latter two cases, the host re-sends the data at a later time.

1.5.2 Sending Data to the Host

A USB device never spontaneously sends data to the host. Nevertheless, in the EZ-USB chip, there’s nothing to stop the 8051 from loading data for the host into an endpoint buffer (Section 1.13, "EZ-USB Endpoints") and *arming* it for transfer. But the data will sit in the buffer *until the host sends an IN token to that particular endpoint*. If the host never sends the IN token, the data sits there indefinitely.

1.6 USB Direction

Once you accept that the host is the bus master, it’s easy to remember USB direction: OUT means from the host to the device, and IN means from the device to the host. EZ-USB nomenclature uses this naming convention. For example, an endpoint that sends data to the host is an IN endpoint. This can be confusing at first, because the 8051 *sends* data by loading an IN endpoint buffer, but keeping in mind that an 8051 *out* is IN to the host, it makes sense.

1.7 Frame

The USB host provides a time base to all USB devices by transmitting a SOF (Start Of Frame) packet every millisecond. The SOF packet includes an incrementing, 11-bit frame count. The 8051 can read this frame count from two EZ-USB registers. SOF-time has significance for isochronous endpoints; it’s the time that the *ping-ponging* buffers switch places. The EZ-USB core provides the 8051 with an SOF interrupt request for servicing isochronous endpoint data.

1.8 EZ-USB Transfer Types

USB defines four transfer types. These match the requirements of different data types delivered over the bus. (Section 1.13, "EZ-USB Endpoints" explains how the EZ-USB family supports the four transfer types.)

1.8.1 Bulk Transfers

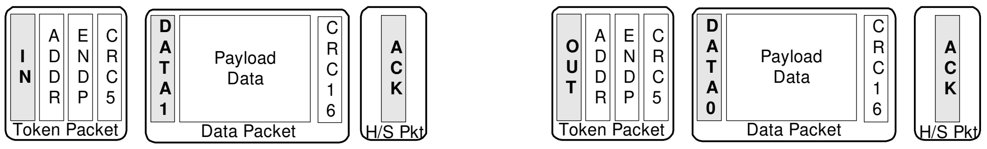


Figure 1-4. Two Bulk Transfers, IN and OUT

Bulk data is *bursty*, traveling in packets of 8, 16, 32, or 64 bytes. Bulk data has guaranteed accuracy, due to an automatic re-try mechanism for erroneous data. The host schedules bulk packets when there is available bus time. Bulk transfers are typically used for printer, scanner, or modem data. Bulk data has built-in flow control provided by handshake packets.

1.8.2 Interrupt Transfers

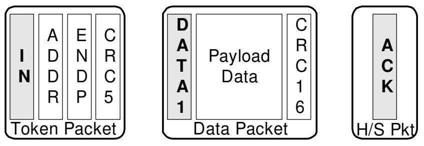


Figure 1-5. An Interrupt Transfer

Interrupt data is like bulk data, but exists only for IN endpoints in the “Universal Serial Bus Specification Version 1.1.” Interrupt data can have packet sizes of 1-64 bytes. Interrupt endpoints have an associated polling interval that ensures that they will be *pinged* (will receive an IN token) by the host on a regular basis.

1.8.3 Isochronous Transfers

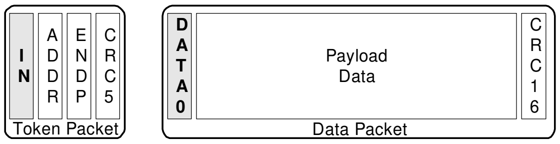


Figure 1-6. An Isochronous Transfer

Isochronous data is time-critical and used for *streaming* data like audio and video. Time of delivery is the most important requirement for isochronous data. In every USB frame, a certain amount of USB bandwidth is allocated to isochronous transfers. To lighten the overhead, isochronous transfers have no handshake (ACK/NAK/STALL), and no retries. Error detection is limited to a 16-bit CRC. Isochronous transfers do not use the data toggle mechanism; isochronous data uses only the DATA0 PID.

1.8.4 Control Transfers

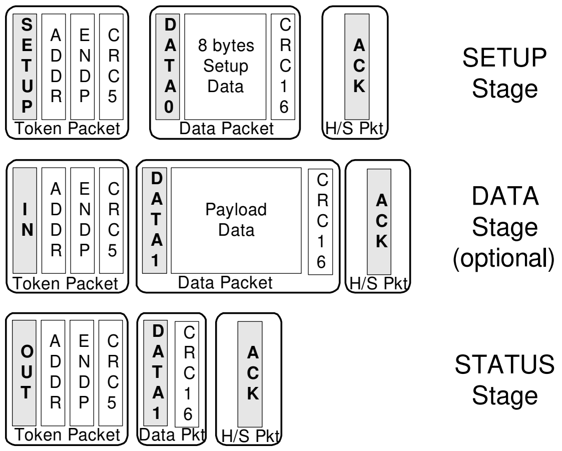


Figure 1-7. A Control Transfer

Control transfers are used to configure and send commands to a device. Being *mission critical*, they employ the most extensive error checking USB offers. Control transfers are delivered on a *best effort* basis by the host (*best effort* is defined by a six-step process in the *Universal Serial Bus Specification Version 1.1, "Section 5.5.4"*). The host reserves a part of each USB frame time for Control transfers.