CY3663
Hardware User's Manual

Version 1.5

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
Tel.: (800) 858-1810 (toll-free in the U.S.)
(408) 943-2600
www.cypress.com

Warranty Disclaimer and Limited Liability

Cypress Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Cypress's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Cypress are granted by the Company in connection with the sale of Cypress products, expressly or by implication. Cypress's products are not authorized for use as critical components in life support devices or systems.

CY3663 Hardware User's Manual,
Version 1.5.

Table of Contents

Chapter 4. Cypress EZ-OTG Development Board

Chapter 5. Developing with the EZ-Host and EZ-OTG Boards

Chapter 6. Developing with the Cypress StrongArm SBC

Appendix A

List of Figures

Chapter 1. Introduction

1.1 Introduction

This manual describes the hardware in the Cypress CY3663 Development Kit (DVK). The three individual boards included in the kit are the Cypress StrongARM Single Board Computer (SBC), the EZ-Host DVK board, and the EZ-OTG DVK board. The next three chapters provide a description of the hardware circuitry on each of the boards, and the last two chapters give the developer details on using the boards.

Please refer as needed to the Appendix for a definition of abbreviations and terms used throughout the manual.

Additional hardware documentation that should be used in conjunction with this document can be found in the "Hardware" subdirectory that was installed from the kit CD. This directory contains schematics, layout files, Bill-of-Materials, and CPLD source for all three boards in this kit.

Chapter 2. Cypress StrongARM Single Board Computer

2.1 Introduction

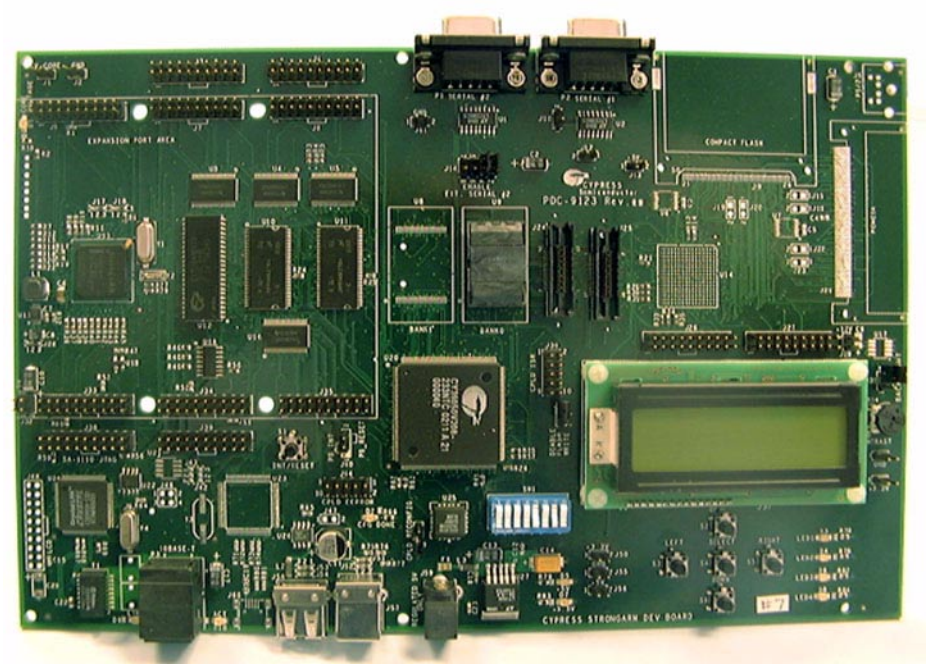


Figure 2-1. Cypress StrongARM Single Board Computer

The Cypress StrongARM Single Board Computer (SBC) is a versatile development platform that currently supports the Linux operation system. Other features include:

- 133 MHz Intel StrongARM SA-1110 microprocessor
- 32 Mbytes SDRAM, 16 Mbytes FLASH, 512 Kbytes SRAM
- User interface including two-line LCD, LEDs, and pushbuttons

- High density, flexible in-circuit programmable CPLD
- Ethernet port
- Multiple synchronous and asynchronous serial ports
- High signal visibility with quick connect logic analyzer support
- Robust buffered expansion port

2.2 System Block Diagram

The following is a block diagram of the Cypress StrongARM SBC.

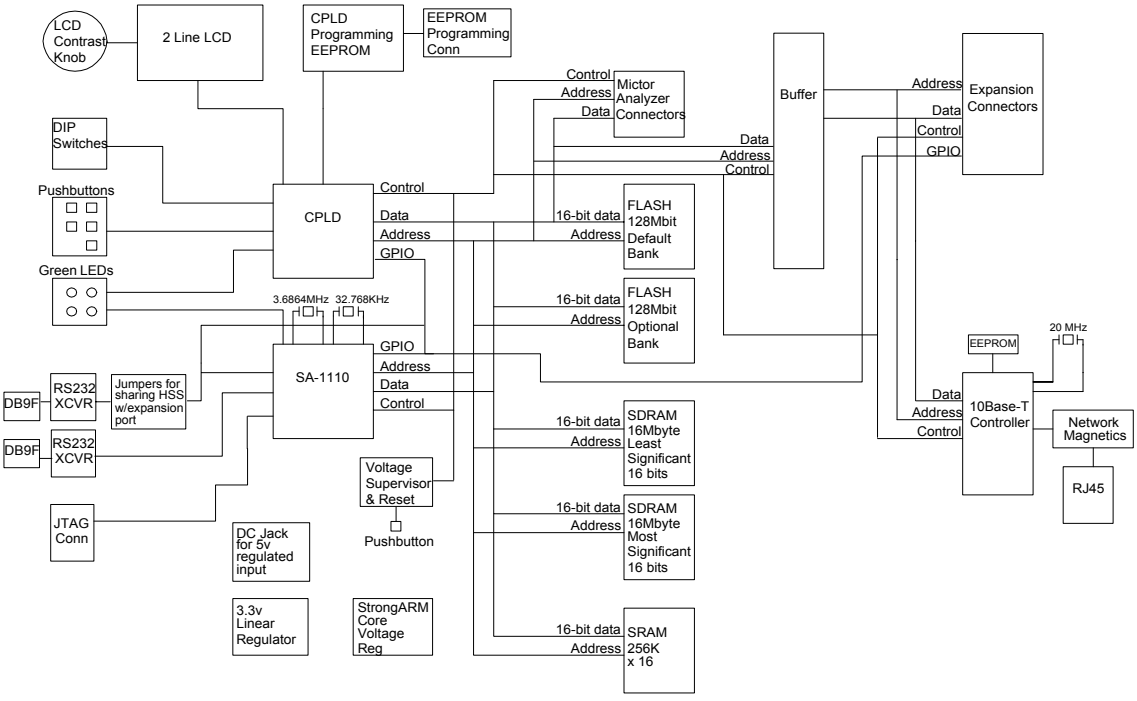


Figure 2-2. Cypress StrongARM SBC - Block Diagram

2.3 Processor and Memory

The Cypress StrongARM SBC uses the Intel StrongARM SA-1110 microprocessor running at 133 MHz. The SA-1110 has built-in serial ports, memory management unit, programmable chip select unit, interrupt controller, JTAG interface, instruction cache, data cache, and programmable memory controller. The Cypress StrongARM SBC has the following memory:

- 32 Mbytes SDRAM (two 8 Meg x 16-bit parts used to make a 32-bit bus)
- 16 Mbytes FLASH (one 8 Meg x 16-bit part loaded, a second 8Meg x 16-bit part unpopulated)
- 512 Kbytes SRAM (one 256K x 16-bit part)

2.4 Serial Ports

2.4.1 Serial Port #1

This asynchronous serial port provides communication with a terminal for control of the SBC. It does not support hardware handshaking. The port has an external DB9F connector and is labeled in silkscreen “P2 Serial #1”. This port uses the on-chip StrongARM SA-1110 Channel 1 UART. Its default settings are 115200 baud, no parity, 8 data bits, and 1 stop bit. Refer to *Figure 2-3* for port location.

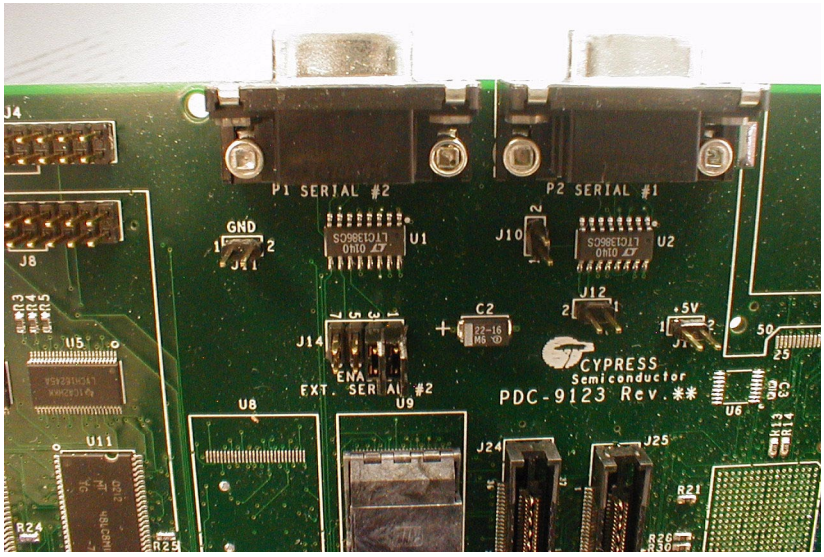


Figure 2-3. Serial Port Location

2.4.2 Serial Port #2

This asynchronous serial port provides communication with a host PC running the DVK software application. By default, it does not support hardware handshaking, but hardware handshaking can be configured by loading shunts on the appropriate pins of J14 and modifying the Linux kernel to support it. Note that the hardware handshaking signals are shared with serial port #3 and are mutually exclusive with that port. Serial port #2 has an external DB9F connector and is labeled in silkscreen "P1 Serial #2". This port uses the on-chip StrongARM SA-1110 Channel 3 UART. Refer to *Figure 2-3* for port location.

2.4.3 Serial Port #3

This asynchronous serial port provides communication with a serial interface on the expansion port. By default, it does not support hardware handshaking, but hardware handshaking can be used via the CTS_3 and RTS_3 signals on J34. Note that the hardware handshaking signals are shared with serial port #2 and are mutually exclusive with that port. The port does not have an external connector or serial transceiver but is directly routed to the expansion port on J7 (signal names GPIO_IR1 as receive and GPIO_IR2 as transmit). This port uses the on-chip StrongARM SA-1110 Channel 2 UART.

2.4.4 Serial Port #4

This synchronous serial port provides communication with a synchronous serial interface on the expansion port. The port does not have an external connector or serial transceiver but is directly routed to the expansion port on J33 (signal names SSP_SFRM, SSP_SCLK, SSP_TXD, SSP_RXD). The serial port supports various synchronous serial protocols including SPI. This port uses the on-chip StrongARM SA-1110 Channel 4 Synchronous Serial Controller.

2.5 Expansion Port

The expansion port is a group of connectors (J3, J4, J6, J7, J8, J32, J33, J34, J35) that allows expansion cards to be connected to the SBC. The EZ-Host DVK board and EZ-OTG DVK board are examples of two expansion boards that fit this form factor. The expansion port offers the following features:

- Buffered 16-bit or 32-bit data bus, buffered address bus, buffered bus control, and buffered master active low reset
- Interrupt support
- Direct connection to several GPIO pins on StrongARM SA-1110
- Direct connection to UART and synchronous serial port on SA-1110

- Direct connection to separate programmable clocks from SA-1110
- Connection of expansion port clock to SBC's CPLD
- Direct connection to several GPIO signals from the CPLD
- Two chip selects with programmable wait states. One chip select supports the use of RDY for variable latency peripherals.
- 3.3 volt and 5 volt power from SBC.
- Expansion port connectors in 20-pin dual row connectors, which are compatible with Agilent logic analyzer terminator adapters.

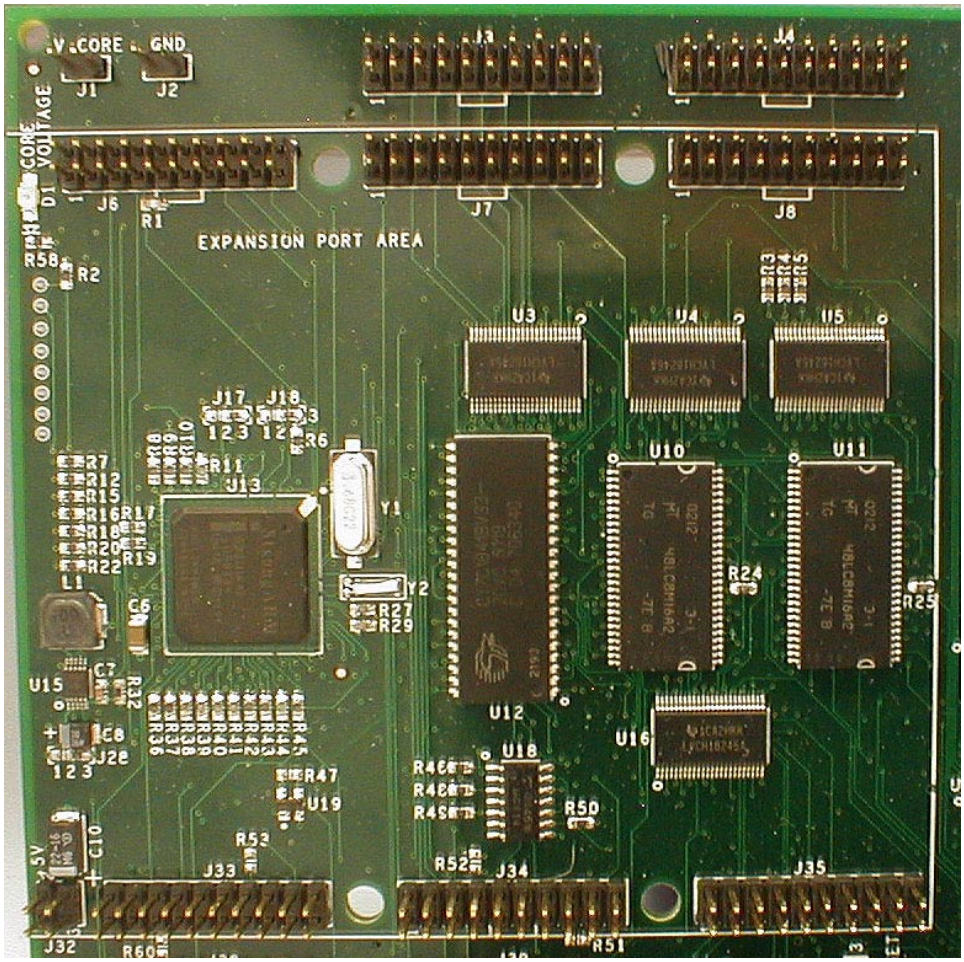


Figure 2-4. SBC Expansion Port Location

Note: Be careful when removing expansion boards from the expansion port. The high number of 0.1" spacing dual row connectors in the expansion port creates so much static friction that removal of expansion cards can cause PCA flex and possible damage.

2.6 CPLD

The CPLD is a Cypress Delta39K series part offering high density, great flexibility, and easy in-system programming. The configuration for the CPLD is stored in U25, which is a serial EEPROM. At power-up, the CPLD loads the configuration from the EEPROM, then allows the board to come out of reset. D2 is an LED that is lit when the configuration is finished loading in the CPLD. The CPLD has the following functions and features:

- LCD controller
- Chip select decode for network controller chip, FLASH, and LCD
- Expansion port buffer control
- DIP switch, push button, and LED control
- GPIO controller
- Expansion port reset control
- Over 86% of resources free for custom logic

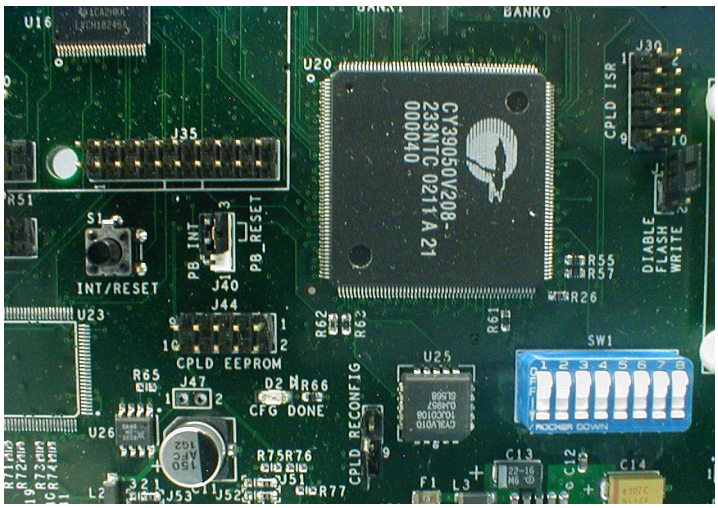


Figure 2-5. CPLD

2.7 User Interface

Figure 2-6 shows the location of some LEDs, LCD, DIP switches, and pushbuttons that are part of the User Interface.

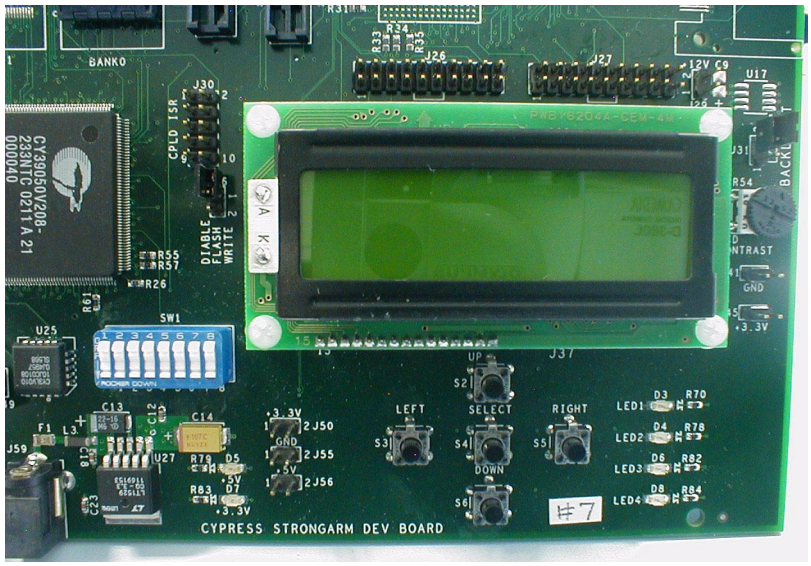


Figure 2-6. User Interface

2.7.1 LCD

The SBC has a two line by 16 characters LCD that is controlled by the CPLD and is used as the primary on-board display. There is a contrast dial located on the right side of the PCA to adjust the LCD contrast. The jumper located at J31 turns on the LCD backlight if shunted.

2.7.2 LEDs

The SBC has several LEDs on-board to allow the user feedback about board operation.

2.7.2.1 Power LEDs

- D5 – Green LED lit when 5 volts is available
- D7 – Green LED lit when 3.3 volts is available
- D1 – Green LED lit when 1.55 volts is available (used for the SA-1110 core)

2.7.2.2 General Purpose LEDs

- D3 – Green LED connected to GPIO of SA-1110
- D4 – Green LED connected to GPIO of SA-1110
- D6 – Green LED connected to CPLD
- D8 – Green LED connected to CPLD

2.7.2.3 Network LEDs

- D9 – Green LED active when network link is established
- D10 – Green LED that blinks during network activity

2.7.2.4 Other LEDs

- D2 – Green LED active when CPLD has successfully loaded its configuration from EEPROM

2.7.3 DIP Switches

The SBC has a bank of eight DIP switches for configuration. Unless indicated otherwise by other documentation, the DIP switches should be set to all OFF for normal operation.

2.7.4 Pushbuttons

The SBC has six pushbuttons on board. Pushbutton S1 is the master hard reset for the board. Pushing it will result in the entire board being reset. Alternatively, it can be connected to a GPIO on the SA-1110 by moving the shunt on J40. Pushbuttons S2 through S6 provide a menu navigation interface if such a menu is implemented in software; otherwise they provide a bank of general purpose pushbuttons.

2.8 Ethernet

The SBC has support for a 10Base-T network connection. The network cable is connected to the RJ45 connector located at P3, and LEDs D9 and D10 provide link and activity indication.

2.9 Power

The SBC receives power from a 5 volt regulated wall transformer. Power is fused on board by the fast acting non-resetable fuse at F1. 5 volts is distributed around the board and to the expansion

port. A 3.3 volt regulator provides 3.3 volts to the board and expansion port. A 1.55 volt regulator provides voltage to the SA-1110 core. Several two-pin jumpers are sprinkled around the board providing convenient ground, 3.3 volt, and 5 volt connection.

2.10 Miscellaneous Jumpers

Other jumpers are present on the PCA as follows:

- J14 – Isolation jumper for Serial port #2. For normal operation, pins 1-2 and 2-3 should be shunted.
- J24 and J25 – These are Mictor footprints for connection of a logic analyzer supporting the Mictor form factor. If development with the SBC results in the need for logic analyzer connection to the main memory bus, Mictor connectors can be soldered in these two locations for analyzer support.
- J26, J27, and J39 – These are debug headers for easy connection to an Agilent logic analyzer with a termination adapter.
- J38 – This header is the JTAG interface for the SA-1110.
- J36 – This jumper allows all FLASH to be protected from writes. Note that the FLASH file system in Linux needs FLASH to be writable.
- J44 – This header connects to an in-circuit EEPROM programmer for CPLD configuration updates.
- Unloaded components and connections – There are several unloaded component and connector footprints on the PCB. These are for future use and have not been tested.

Chapter 3. Cypress EZ-Host Development Board

3.1 Introduction

The Cypress EZ-Host development board is a versatile development platform used to demonstrate and develop with the Cypress EZ-Host USB host/peripheral controller (CY7C67300).

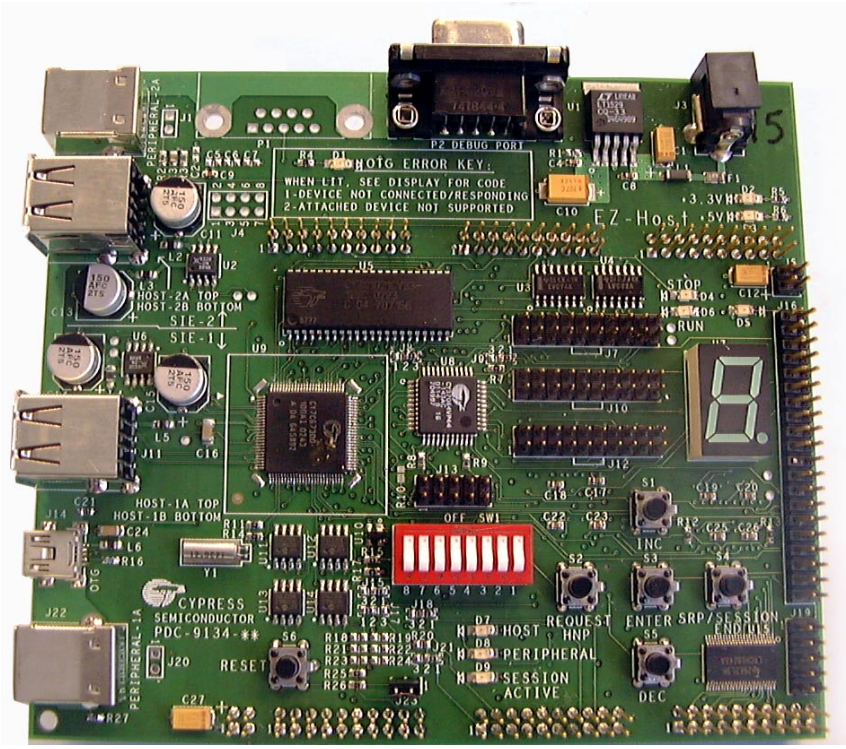


Figure 3-1. Cypress EZ-Host Development Board

Other features include:

- 48 MHz Cypress CY16 microprocessor in the Cypress EZ-Host USB host/peripheral controller (CY7C67300)

- 64Kybtes external SRAM (one 64K x 16 part, with 32K x 16 accessible)
- 8-bit FLASH footprint (not loaded)
- User interface including seven segment display, LEDs, and pushbuttons
- In-circuit programmable CPLD
- Serial port
- Multiple host and peripheral USB ports and USB On-The-Go (OTG) support
- Charge pump for up to 10mA of five volt power for OTG VBUS
- High signal visibility with quick connect logic analyzer support
- 40-pin IDE connector
- User selectable operation either in standalone mode or co-processor mode
- Support for co-processor mode connection to the Cypress StrongARM SBC
- Support for power from external wall transformer or co-processor

3.2 System Block Diagram

The following is a block diagram of the Cypress EZ-Host development board containing the Cypress EZ-Host USB host/peripheral controller (CY7C67300).

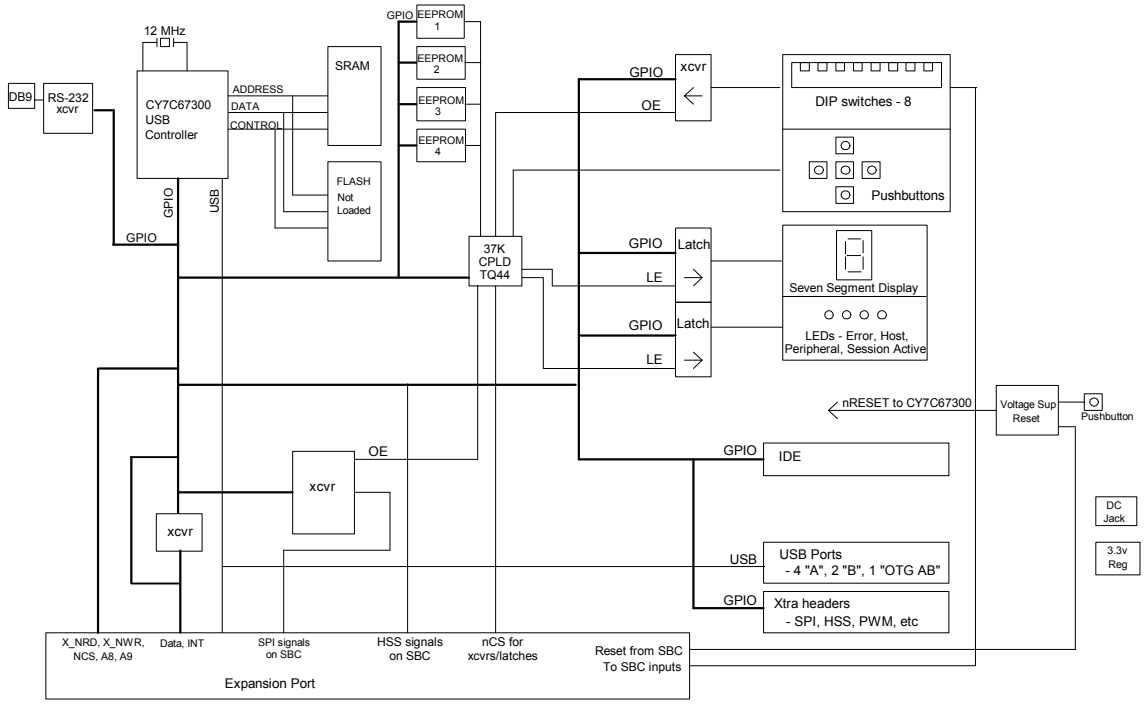


Figure 3-2. Cypress EZ-Host Development Board - Block Diagram

3.3 Processor and Memory

The Cypress EZ-Host development board is centered around the Cypress EZ-Host USB host/peripheral controller (CY7C67300). This USB controller has several on-chip features as listed below:

- 16-bit RISC Processor
- 8 Kbytes Internal ROM BIOS
- 16 Kbytes Internal SRAM
- Two UARTs
- Up to four Full-speed USB 2.0 compliant host ports

- Up to two Full-speed USB 2.0 compliant peripheral ports
- One Full-speed USB 2.0 compliant On-The-Go (OTG) port
- Watch-dog Timer
- External SRAM and DRAM controllers
- IDE Controller
- HPI Controller
- SPI Master/Slave Controller
- Timers

3.4 Asynchronous Serial Ports

3.4.1 UART

This serial port provides a debug and development interface for the EZ-Host chip. It does not support hardware or software handshaking. This port has an external DB9F connector with reference designator P2 and is labeled in silkscreen "DEBUG PORT".

3.4.2 High Speed Serial (HSS)

This serial port provides a high speed asynchronous serial interface to a co-processor. It is routed to the expansion connectors on J25 and J30 (signal names GPIO_16, GPIO_17, GPIO_18, GPIO_26). If the EZ-Host board is connected as a co-processor to the Cypress StrongARM SBC and HSS operation is selected by the appropriate DIP switch settings, the HSS port communicates with one of the serial ports on the SBC. The HSS port supports hardware handshaking. An optional connection for HSS would be to load the components for J4, U16 (and its surrounding capacitors), and P1 for an RS232 compliant port. Note that if these optional components are used, the baud rate may be limited by the maximum switching frequency of the serial transceiver at U16.

3.5 SPI

The CY7C67300 contains a master/slave SPI controller. When the EZ-Host board is connected to the Cypress StrongARM SBC and SPI operation is selected by the appropriate DIP switch settings, the SPI port operates in slave mode only and is buffered before connecting to the SBC. The buffered SPI signals are available on J31 (SBC_NSSI, SBC_SCK, SBC_MOSI, SBC_MISO). The DIP switches configure the EZ-Host board for SPI operation.

3.6 Co-processor Connection

The EZ-Host board has a group of connectors (J24, J25, J26, J29, J30, J31, J32) which allow connection of it to the Cypress StrongARM SBC. The DIP switches configure the EZ-Host board for operation in one of the three co-processor modes: HPI, SPI, or HSS.

Note: Be careful when removing this board from the expansion port of the Cypress StrongARM SBC. The high number of 0.1" spacing dual row connectors in the expansion port creates so much static friction that removal of expansion cards can cause PCA flex and possible damage.

3.7 CPLD

The CPLD is a Cypress 37K series part offering flexibility and in-circuit programming. The CPLD controls the mode the CY7C67300 powers up in by reading the DIP switch settings and driving the boot pins (GPIO_30, GPIO_31) appropriately. It also controls serial EEPROM selection, buffer control, and pushbutton control. The header in position J13 allows an in-circuit programmer to update the CPLD.

3.8 User Interface

Figure 3-3 shows the location of LEDs, DIP switches, Pushbuttons, and SSD for the EZ-Host User Interface.

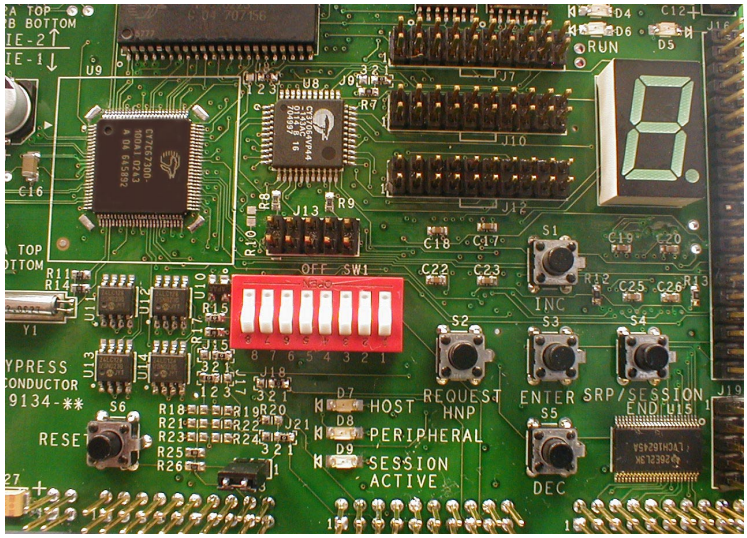


Figure 3-3. EZ-Host User Interface

3.8.1 Seven Segment Display

The Seven Segment Display (SSD) allows hex digits to be displayed to the user. When in stand-alone mode, the SSD is driven by the CY7C67300. When in co-processor mode, the SSD is driven by the Cypress StrongARM SBC.

3.8.2 DIP Switches

A bank of eight DIP switches controls the operation of the EZ-Host board. Based on the switch settings, the EZ-Host board powers up in standalone mode with one of four serial EEPROMs active, or in one of the three co-processor modes. The DIP switches also select any of several demonstration examples. Refer to Chapter 5. "Developing with the EZ-Host and EZ-OTG Boards" for a table of DIP switch settings and their definitions.

3.8.3 LEDs

The EZ-Host board has several LEDs on-board to allow the user feedback about board operation.

3.8.3.1 Power LEDs

- D3 – Green LED lit when 5 volts is available
- D2 – Green LED lit when 3.3 volts is available

3.8.3.2 General Purpose LEDs

- D1 – Red LED lit when an OTG error is encountered during the OTG design example
- D7 – Green LED lit when the board is operating in the host position during the OTG design example
- D8 – Green LED lit when the board is operating in the peripheral position during the OTG design example
- D9 – Green LED lit when the board has an active USB session during the OTG design example
- D4 – Red general purpose LED.
- D6 – Green general purpose LED.

Note: When in stand-alone mode, D1, D7, D8, and D9 are driven by the CY7C67300. When in co-processor mode, these four LEDs are driven by the Cypress StrongARM SBC.

3.8.3.3 IDE LED

- D5 – Green LED lit when a connected IDE device drives the nDASP line.

3.8.4 Pushbuttons

The EZ-Host board has six pushbuttons on-board. Pushbutton S6 causes a hard reset to the board when in standalone mode. It has no effect in co-processor mode. Note that this pushbutton can be disabled by removing the shunt on J23. Pushbuttons S1 through S5 are general purpose input buttons that are used during the operation of the design examples. Note that the pushbuttons are read and cleared by the CY7C67300 in stand-alone mode and by the Cypress StrongARM SBC in co-processor mode.

3.9 Power

The EZ-Host board can receive power either from an external five volt wall transformer or through the co-processor connectors. When in standalone mode, the power from the external five volt wall transformer is fused on board by a fast acting non-resetable fuse at F1. A 3.3 volt regulator provides 3.3 volts to the logic on the board. When in co-processor mode, power is provided by the Cypress StrongARM SBC; five volts through J32 and 3.3 volts through J24 and J30.

3.10 USB Ports

The EZ-Host board has four USB host ports, two USB peripheral ports, and one USB OTG port. Not all ports can be connected at the same time. See the CY7C67300 datasheet for a table of possible simultaneous active ports. Some details for the USB section follow:

- USB host VBUS current limiting – Each of the four USB host ports has a current limiting device on VBUS to prevent a USB device from drawing so much current over VBUS that the host would be damaged. These current limiting devices automatically reset after the overcurrent situation is remedied.
- USB host VBUS power – Each of the four USB host ports can supply up to 500mA current to a device, but not all four ports can simultaneously supply 500mA to peripherals. No more than two USB host ports should be connected to devices consuming greater than 100mA current or the five volt power on the board will be drawn down.
- VBUS detect for USB peripheral ports – The board supports detection of valid VBUS on both of its peripheral ports. Peripheral port one VBUS is detected via the OTG VBUS pin that connects to the CY7C67300. Peripheral port two VBUS can be detected in one of several ways:

1. In standalone or co-processor mode, a register in the CPLD can be read that shows VBUS level. Refer to Chapter 5. "Developing with the EZ-Host and EZ-OTG Boards" for register settings.
2. In standalone or co-processor mode the VBUS level can be output on GPIO_30 by setting a register in the CPLD. Refer to Chapter 5. "Developing with the EZ-Host and EZ-OTG Boards" for register settings.
3. In standalone mode, R10 can be loaded to put VBUS on GPIO_20.

3.11 Miscellaneous

The EZ-Host board has several features to allow easy development and debug as listed below.

- Voltage supervisor – An on-board voltage supervisor ensures clean system resets under all conditions including voltage brownout.
- EEPROM support – The board has support for four separate EEPROMs. These EEPROMs are shipped as 256-Kbit devices, but any smaller size can also be used for custom development. If 16Kbit or smaller EEPROMs are loaded, the resistors at J15 and J17 must be moved to the opposite position (see schematic).
- Boot pin support – The board automatically sets the boot pins for the CY7C67300 in the correct configuration based on the DIP switches. But the board can be forced into a single boot configuration regardless of DIP switch settings if the appropriate combination of R18, R19, R21, and R22 are loaded and R47 and R49 are removed.
- Debug headers – All the 20-pin headers on the board are compatible with Agilent logic analyzer terminator adapters. This feature allows quick and simple debug connection.

Chapter 4. Cypress EZ-OTG Development Board

4.1 Introduction

The Cypress EZ-OTG development board is a versatile development platform used to demonstrate and develop with the Cypress EZ-OTG USB host/peripheral controller (CY7C67200). Other features include:

- 48 MHz Cypress CY16 microprocessor in the Cypress EZ-OTG USB Host/peripheral controller (CY7C67200)
- User interface including seven segment display, LEDs, and pushbuttons
- In-circuit Programmable CPLD
- Serial Port
- Multiple Host and peripheral USB ports and USB On-The-Go (OTG) support
- Charge pump for up to 10mA of five volt power for OTG VBUS
- High signal visibility with quick connect logic analyzer support
- User selectable operation either in standalone mode or co-processor mode
- Support for co-processor mode connection to the Cypress StrongARM SBC
- Support for power from external wall transformer or co-processor

4.2 System Block Diagram

The following is a block diagram of the Cypress EZ-OTG development board containing the Cypress EZ-OTG USB host/peripheral controller (CY7C67200).

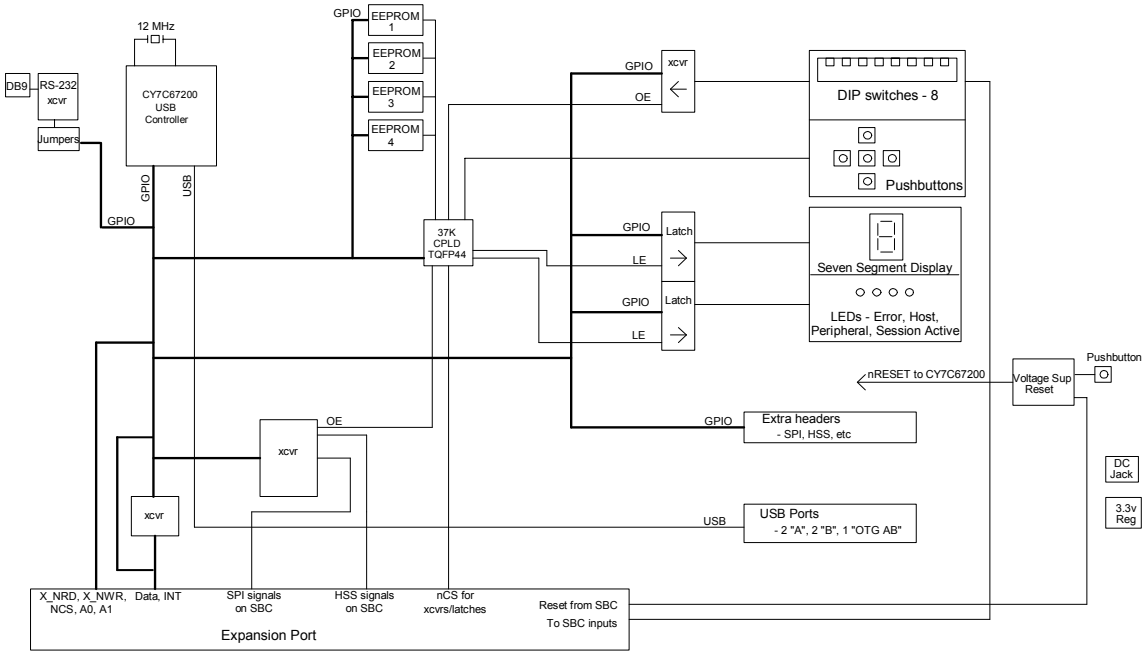


Figure 4-1. Cypress EZ-OTG Development Board - Block Diagram

4.3 Processor and Memory

The Cypress EZ-OTG development board is centered around the Cypress EZ-OTG USB host/ peripheral controller (CY7C67200). This USB controller has several on-chip features as listed below:

- 16-bit RISC Processor
- 8 Kbytes Internal ROM BIOS
- 16 Kbytes Internal SRAM
- Two UARTs
- Up to two Full-speed USB 2.0 compliant host ports
- Up to two Full-speed USB 2.0 compliant peripheral ports
- One Full-speed USB 2.0 compliant On-The-Go (OTG) port
- Watch-dog Timer
- HPI Controller
- SPI Master/Slave Controller
- Timers

4.4 Asynchronous Serial Ports

4.4.1 UART

This serial port provides a debug and development interface for the EZ-OTG chip. It does not support hardware or software handshaking. This port has an external DB9F connector at P2 and is labeled in silkscreen “DEBUG PORT”. Note that the serial debug port on the EZ-OTG board cannot be used when in co-processor mode. To use the serial port in standalone mode, the header at J33 must have shunts applied between pins 1-2 and 3-4. In addition, the UART should not be used if the firmware uses the DIP switches or SSD. The shunts on J33 must be removed for co-processor mode.

4.4.2 High Speed Serial (HSS)

This serial port provides a high speed asynchronous serial interface to a co-processor. It is routed through a buffer to the expansion connectors on J25 and J30 (signal names GPIO_12, GPIO_13, GPIO_14, GPIO_15). If the EZ-OTG board is connected as a co-processor to the Cypress StrongARM SBC and HSS operation is selected by the appropriate DIP switch settings, the HSS port communicates with one of the serial ports on the SBC. The HSS port supports hardware handshaking. An optional connection for HSS would be to load the components for J4, U16 (and its surrounding capacitors), and P1 for an RS232 compliant port. Note that if these optional components are used, the baud rate may be limited by the maximum switching frequency of the serial transceiver at U16.

4.5 SPI

The CY7C67200 contains a master/slave SPI controller. When the EZ-OTG board is connected to the Cypress StrongARM SBC and SPI operation is selected by the appropriate DIP switch settings, the SPI port operates in slave mode only and is buffered before connecting to the SBC. The buffered SPI signals are available on J31 (SBC_NSSI, SBC_SCK, SBC_MOSI, SBC_MISO). The DIP switches configure the EZ-OTG board for SPI operation.

4.6 Co-processor Connection

The EZ-OTG board has a group of connectors (J24, J25, J26, J29, J30, J31, J32) which allow connection of it to the Cypress StrongARM SBC. But even in standalone mode, the UART should not be used if the firmware accesses the DIP switches or SSD on the board. The DIP switches configure the EZ-OTG board for operation in one of the three co-processor modes: HPI, SPI, or HSS.

Note: Be careful when removing this board from the expansion port of the Cypress StrongARM SBC. The high number of 0.1" spacing dual row connectors in the expansion port creates so much static friction that removal of expansion cards can cause PCA flex and possible damage.

4.7 CPLD

The CPLD is a Cypress 37K series part offering flexibility and in-circuit programming. The CPLD controls the mode the CY7C67200 powers up in by reading the DIP switch settings and driving the boot pins (GPIO_30, GPIO_31) appropriately. It also controls serial EEPROM selection, buffer control, and pushbutton control. The header in position J13 allows an in-circuit programmer to update the CPLD.

4.8 User Interface

4.8.1 Seven Segment Display

The Seven Segment Display (SSD) allows hex digits to be displayed to the user. When in stand-alone mode, the SSD is driven by the CY7C67200. When in co-processor mode, the SSD is driven by the Cypress StrongARM SBC.

4.8.2 DIP Switches

A bank of eight DIP switches controls the operation of the EZ-OTG board. Based on the switch settings, the EZ-OTG board powers up in standalone mode with one of four serial EEPROMs active, or in one of the three co-processor modes. The DIP switches also select any of several demonstration examples. Refer to Chapter 5. "Developing with the EZ-Host and EZ-OTG Boards" for a table of DIP switch settings and their definitions.

4.8.3 LEDs

The EZ-OTG board has several LEDs on-board to allow the user feedback about board operation.

4.8.3.1 Power LEDs

- D3 – Green LED lit when 5 volts is available
- D2 – Green LED lit when 3.3 volts is available

4.8.3.2 General Purpose LEDs

- D1 – Red LED lit when an OTG error is encountered during the OTG design example
- D7 – Green LED lit when the board is operating in the host position during the OTG design example
- D8 – Green LED lit when the board is operating in the peripheral position during the OTG design example
- D9 – Green LED lit when the board has an active session during the OTG design example

Note: When in stand-alone mode, D1, D7, D8, and D9 are driven by the CY7C67200. When in co-processor mode, these four LEDs are driven by the Cypress StrongARM SBC.

4.8.4 Pushbuttons

The EZ-OTG board has six pushbuttons on-board. Pushbutton S6 causes a hard reset to the board when in standalone mode. It has no effect in co-processor mode. Note that this pushbutton can be disabled by removing the shunt on J23. Pushbuttons S1 through S5 are general purpose input buttons that are used during the operation of the design examples. Note that the pushbuttons are read and cleared by the CY7C67200 in stand-alone mode and by the Cypress StrongARM SBC in co-processor mode.

4.9 Power

The EZ-OTG board can receive power either from an external five volt wall transformer or through the co-processor connectors. When in standalone mode, the power from the external five volt wall transformer is fused on board by a fast acting non-resetable fuse at F1. A 3.3 volt regulator provides 3.3 volts to the logic on the board. When in co-processor mode power is provided by the Cypress StrongARM SBC; five volts through J32 and 3.3 volts through J24 and J30.

4.10 USB Ports

The EZ-OTG board has two USB host ports, two USB peripheral ports, and one USB OTG port. Not all ports can be connected at the same time. See the CY7C67200 datasheet for a table of possible simultaneous active ports. Some details for the USB section follow:

- USB host VBUS current limiting – Each of the two USB host ports has a current limiting device on VBUS to prevent a USB device from drawing so much current over VBUS that the host would be damaged. These current limiting devices automatically reset after the overcurrent situation is remedied.

- USB host VBUS power – Each of the two USB host ports can supply up to 500mA current to a device.
- VBUS detect for USB peripheral ports – The board supports detection of valid VBUS on both of its peripheral ports. Peripheral port one VBUS is detected via the OTG VBUS pin that connects to the CY7C67200. Peripheral port two VBUS can be detected in one of several ways:
    1. In standalone or co-processor mode, a register in the CPLD can be read that shows VBUS level. Refer to Chapter 5. "Developing with the EZ-Host and EZ-OTG Boards" for register settings.
    2. In standalone or co-processor mode the VBUS level can be output on GPIO_30 by setting a register in the CPLD. Refer to Chapter 5. "Developing with the EZ-Host and EZ-OTG Boards" for register settings.
    3. In standalone mode, R10 can be loaded to put VBUS on GPIO_20.

4.11 Miscellaneous

The EZ-OTG board has several features to allow easy development and debug as listed below.

- Voltage supervisor – An on-board voltage supervisor ensures clean system resets under all conditions including voltage brownout.
- EEPROM support – The board has support for four separate EEPROMs. These EEPROMs are shipped as 128Kbit devices, but any smaller size can also be used for custom development. If 16Kbit or smaller EEPROMs are loaded, then the resistors at J15 and J17 must be moved to the opposite position (see schematic).
- Boot pin support – The board automatically sets the boot pins for the CY7C67200 in the correct configuration based on the DIP switches. But the board can be forced into a single boot configuration regardless of DIP switch settings if the appropriate combination of R18, R19, R21, and R22 are loaded and R47 and R49 are removed.
- Debug headers – All the 20-pin headers on the board are compatible with Agilent logic analyzer terminator adapters. This feature allows quick and simple debug connection.

Chapter 5. Developing with the EZ-Host and EZ-OTG Boards

5.1 Standalone Mode Operation

5.1.1 Power-up EEPROM Selection

Since no signal is present on the CP_PRESENT (co-processor present) line, the CPLD looks at the DIP switch settings and decides which EEPROM should be enabled.

5.1.2 DIP Switches

- Read via the CPLD
- Note on EZ-OTG board, if Debug UART is being used, then the upper two DIP positions cannot be read.

5.1.3 Pushbutton Inputs

- PB inputs are latched in CPLD
- Read through CPLD
- Cleared through CPLD write, one write per button

5.1.4 LEDs: OTG ERROR, SESSION ACTIVE, HOST, PERIPHERAL

Written through the CPLD.

5.1.5 LEDs: RUN/STOP

Only available on EZ-Host board. Any write to external ROM space affects these two LEDs based on the lowest two data bus bits as follows:

Table 5-1. Run/Stop LEDs

CY7C67300 memory bus data bit 1 (D1)	CY7C67300 memory bus data bit 0 (D0)	STOP LED (red)	RUN LED (green)
0	0	ON	ON
0	1	ON	OFF
1	0	OFF	ON
1	1	OFF	OFF

NOTE: LEDs reset on power-up to OFF

5.1.6 Seven Segment Display

- Similar to LEDs. Written through CPLD
- Note that if the UART is enabled on the EZ-OTG board, the upper two segments cannot be written and will be driven to whatever happens to be on the UART pins at the time of the write.

5.1.7 Debugging Through the UART

- The UART will work in all situations with the EZ-Host board.
- The UART can be used in standalone mode with the EZ-OTG board. The limitation is that whenever the CPLD is accessed, GPIO_6 must not be changed from its UART configuration as an input. And because the UART uses GPIO_6 and GPIO_7, the upper two DIP switch positions cannot be reliably read, and the upper two segments cannot be reliably written on the Seven Segment Display.

5.1.8 CPLD Accesses

In stand-alone mode, the CY7C67300 or CY7C67200 accesses the peripherals through the CPLD by creating a typical memory access type cycle using the following GPIO signals:

- Address: A0 (GPIO_19)
- Chip Select: nCS (GPIO_21)
- Read Strobe: nRD (GPIO_23)
- Write Strobe: nWR (GPIO_22)
- Data: D[7:0] (GPIO_[7:0], where GPIO_7 is the most significant bit)

See the waveform diagram below. Note that nRD and nWR are mutually exclusive; that is, only one should be asserted (low) per access. Timing should not be an issues since f/w is bit-banging these signal bits creating plenty of timing margin between signal transitions.

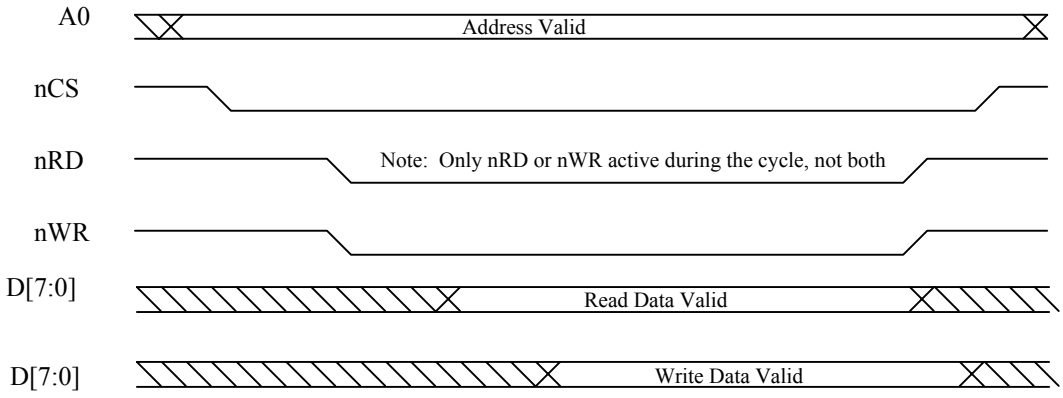


Figure 5-1. Timing Waveforms

5.2 Co-processor Mode Operation

5.2.1 Power-up Port Selection

Since the CP_PRESENT pin will be driven by the SBC, the EZ-Host/OTG CPLD will strap the boot pins to whatever the DIP inputs select (HPI, HSS, or SPI), assuming it is a legitimate combination of demo and communication method. The EZ-Host/OTG controller will power up into that mode.

NOTE: In SPI and HSS mode, the EZ-Host/OTG controller must be configured to tri-state GPIO[24:19] and GPIO[7:0] or the SBC cannot read and write the DIP switches, pushbuttons, LEDS, and SSD.

5.2.2 DIP Switches

- HPI mode: SBC can read them through the EZ-Host/OTG CPLD
- SPI mode: SBC can read them through the EZ-Host/OTG CPLD
- HSS mode: SBC can read them through the EZ-Host/OTG CPLD
- Additionally, SBC can read them through its own CPLD (the SBC CPLD)

5.2.3 Push Button Inputs

- HPI mode: SBC can read them through EZ-Host/OTG CPLD and clear them individually through EZ-Host/OTG CPLD
- SPI mode: SBC can read them through the EZ-Host/OTG CPLD and clear them individually through EZ-Host/OTG CPLD
- HSS mode: SBC can read them through the EZ-Host/OTG CPLD and clear them individually through EZ-Host/OTG CPLD

5.2.4 LEDs

- HPI mode: SBC can write them via the EZ-Host/OTG CPLD
- SPI mode: SBC can write them via the EZ-Host/OTG CPLD
- HSS mode: SBC can write them via the EZ-Host/OTG CPLD

5.2.5 Seven Segment Display

- Similar to LEDs in all three co-processor modes

5.2.6 CPLD Accesses

- The co-processor will access the EZ-Host/OTG CPLD with a normal asynchronous memory-type access. Refer to Section 5.3. "Addressing of EZ-Host/OTG CPLD" for more details.

5.2.7 Debugging Through the UART

- The UART will work in all situations with the EZ-Host board.
- The UART cannot be reliably used in co-processor mode with the EZ-OTG board. Unrelated memory cycles on the SBC will clobber both transmit and receive data randomly.

5.3 Addressing of EZ-Host/OTG CPLD

All EZ-Host/OTG CPLD activity is controlled by indirect addressing. There are two direct registers in the CPLD: the pointer register (called ADD_Reg), and the data register (called Data_Reg). Writing to the CPLD with A0 = 0 (this is the GPIO_19 pin) will write to the ADD_Reg, which makes the Data_Reg register point to the new indirect register. Reading or writing with A0 = 1 will read or write to that indirect register.

Table 5-2. Memory Map for Direct CPLD Registers

Direct Address	Register Name	Description	
A0 = 0	ADD_Reg	Address Pointer Register. Writes to pointer register	Write Only (Read value can be accessed indirectly)
A0 = 1	Data_Reg	Read or write from/to indirect register pointed to by Address Pointer Register (ADD_Reg)	Read/Write

5.3.1 ADD_Reg

Register that contains the address pointer to the indirect registers

Address A0 = 0x0 (Direct address)
Write only
Resets to XXX00000

Bit	Description
4:0	Address pointer to indirect registers
7:5	Unused

5.3.2 Data_Reg

Register that writes to or reads from an indirect register. Controlled by the address in the ADD_Reg register

Address A0 = 0x1 (Direct address)
Read/Write
Resets to XXXXXXXX

Bit	Description
7:0	Data in indirect register pointed to by ADD_Reg

5.4 Memory Map for Indirect CPLD Registers

Table 5-3. Memory Map for Indirect CPLD Registers

Register Name	Indirect Address			
PB_Read	0x0	Read only		
PB_UP_Clr	0x1	Write only		
PB_LEFT_Clr	0x2	Write only		
PB_RIGHT_Clr	0x3	Write only		
PB_DOWN_Clr	0x4	Write only		
PB_ENTER_Clr	0x5	Write only		
DIP_Read	0x6	Read only		
LED_Write	0x7	Write only		
SSD_Write	0x8	Write only		
VBUS_Level	0x9	Read only		
VBUS_On_GPIO_30	0xA	Read/Write		
ADD_Reg_Read	0xB	Read only		
EEPROM_MFG_CTL	0xC	Read/Write		

5.5 EZ-Host/OTG CPLD Indirect Register Descriptions

5.5.1 PB_Read

Register that holds latched value of push-buttons

Address 0x0
Read only
Resets to XXX11111

Bit	Description
0	PB_UP latched value (0 is latched, 1 is unlatched)
1	PB_LEFT latched value
2	PB_RIGHT latched value
3	PB_DOWN latched value
4	PB_ENTER latched value
7:5	Unused

5.5.2 PB_UP_Clr

Register that clears the latch of a pushbutton

Address 0x1
Write only

Any write of any value to this register will result in the PB_UP latch being cleared.

5.5.3 PB_LEFT_Clr

Register that clears the latch of a pushbutton

Address 0x2
Write only

Any write of any value to this register will result in the PB_LEFT latch being cleared.

5.5.4 PB_RIGHT_Clr

Register that clears the latch of a pushbutton

Address 0x3
Write only

Any write of any value to this register will result in the PB_RIGHT latch being cleared.

5.5.5 PB_DOWN_Clr

Register that clears the latch of a pushbutton

Address 0x4
Write only

Any write of any value to this register will result in the PB_DOWN latch being cleared.

5.5.6 PB_ENTER_Clr

Register that clears the latch of a pushbutton

Address 0x5
Write only

Any write of any value to this register will result in the PB_ENTER latch being cleared.

5.5.7 DIP_Read

Register that allows the read of the bank of 8 DIP switches

Address 0x6
Read only

Bit	Description
0	DIP_1 switch value
1	DIP_2 switch value
2	DIP_3 switch value
3	DIP_4 switch value
4	DIP_5 switch value
5	DIP_6 switch value
6	DIP_7 switch value
7	DIP_8 switch value

5.5.8 LED_Write

Register that allows the four individual LEDs to be written

Address 0x7
Write only
Resets to XXXXXXXX

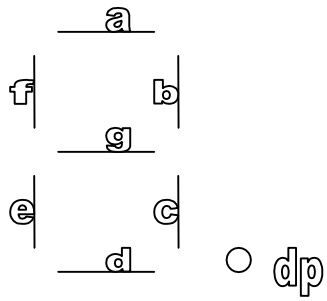
Bit	Description
0	ERROR LED – Writing a "0" turns it on
1	HOST LED
2	PERIPHERAL LED
3	SESSION ACTIVE LED
7:4	Not connected. Write any value

5.5.9 SSD_Write

Register that allows the Seven-Segment-Display to be written

Address 0x8
Write only
Resets to XXXXXXXX

Bit	Description
0	A Segment – Writing a “0” turns it on
1	B Segment
2	C Segment
3	D Segment
4	E Segment
5	F Segment
6	G Segment
7	DP Segment



5.5.10 VBUS_Level

Register that contains the VBUS level of the peripheral port on SIE2 of the EZ-Host/OTG controller.

Note: *This is a TTL input reading an analog voltage. If the actual VBUS voltage is between about 0.8 volts and 2.7 volts, the register may read low or high.*

Address 0x9
Read only

Bit	Description
0	VBUS level on peripheral port of SIE2
7:1	Unused

5.5.11 VBUS_On_GPIO_30

Register that drives VBUS from the peripheral port of SIE2 of the EZ-Host/OTG controller onto signal pin GPIO_30.

NOTE: *In standalone mode GPIO_30 is used as the EEPROM clock (SCL), so make sure EEPROM activity is done before configuring GPIO_30 to reflect the SIE2 VBUS level.*

Address 0xA

Read/Write

Resets to XXXXXXX0

Bit	Description
0	If set, drive the level of VBUS on peripheral port of SIE2 onto GPIO_30. If clear, GPIO_30 is driven to a level dictated by the DIP switches
7:1	Not used

5.5.12 Add_Reg_Read

Register that contains the indirect register address. This register is probably not very useful.

Address 0xB

Read Only

Resets to XXXXXXXX

Bit	Description
3:0	Value of indirect register address
7:4	Not used

5.5.13 EEPROM_MFG_CTL

Register that allows a co-processor to drive the EEPROM address lines, clock and data lines. It also allows the EZ-Host/OTG controller to select any EEPROM regardless of the DIP switch settings. This register is meant for manufacturing use and is probably not useful for development.

NOTE: *Control of the EEPROM using this register does not allow EEPROM reads through the CPLD. The actual data pin of the EEPROM (SDA) is read through one of the SBC's directly connected general purpose pins. Also, when reading an EEPROM, the user must make sure to drive*

the SCL pin high (set bit 0 of this register) because that drives a pull-up, which is required for that signal.

Address 0xC

Read/Write

Resets to XXX00000

Bit	Description
0	Value to drive SCL if bit 2 is set
1	Value to drive SDA if bit 2 is set
2	Setting this bit makes the other bits of this register drive the EEPROM address lines and control lines.
3	EEPROM address select bit 0 if bit 2 is set. See table below.
4	EEPROM address select bit 1 if bit 2 is set. See table below.
7:5	Not used

Table 5-4. EEPROM Address Select Bits 3 and 4

EEPROM_MFG_CTL[4]	EEPROM_MFG_CTL[3]	EEPROM Selected
0	0	EEPROM 1
0	1	EEPROM 2
1	0	EEPROM 3
1	1	EEPROM 4

5.6 EZ-Host/OTG Board Pushbutton and DIP Switch Definitions

5.6.1 DIP Switch Settings

There is a bank of eight DIP switches on the EZ-Host/OTG board. Note that only the DIP switches on the EZ-Host/OTG board will be used to configure the demos, not the DIP switches on the SBC. The DIP switches are labeled 1-8 on the part and also have an “OFF” or “OPEN” label for one direction. Note: “OFF” is a logical 1 when read by hardware, and “ON” is logical 0 when read by hardware. In addition not all combinations of Communication Mode (DIP switches 1 and 2) are valid with all combinations of Design Example Number (DIP switches 3, 4, 5, and 6).

Table 5-5. Switch Settings for Switch 1 and 2

Switch 2	Switch 1	Communication Mode
OFF	OFF	Standalone
OFF	ON	SPI
ON	OFF	HPI
ON	ON	HSS

Table 5-6. Switch Settings for Switch 3, 4, 5, and 6

Switch 6	Switch 5	Switch 4	Switch 3	Design Example #
OFF	OFF	OFF	OFF	Default *
OFF	OFF	OFF	ON	Design Example 1 – OTG Standalone mode - Valid, EEPROM 1 enabled Coprocessor mode - Valid, HPI only
OFF	OFF	ON	OFF	Design Example 2 – Dual Slave Standalone mode - Invalid Coprocessor mode - Valid
OFF	OFF	ON	ON	Design Example 3 – Peripheral-Host Passthrough Standalone mode- Valid, EEPROM 2 enabled Coprocessor mode - Valid, HPI only
OFF	ON	OFF	OFF	Design Example 4 - Dual Host Standalone mode - Valid, EEPROM 3 enabled Coprocessor mode - Valid, HPI only
OFF	ON	X	ON	Invalid
OFF	ON	ON	X	Invalid
ON	OFF	X	X	Invalid
ON	ON	OFF	OFF	Standalone mode - Valid, IDE enabled, CPLD inactive, EEPROM 4 Enabled Coprocessor mode - Invalid
ON	ON	OFF	ON	Invalid
ON	ON	ON	OFF	Invalid
ON	ON	ON	ON	Standalone mode - Valid, EEPROM 4 enabled Coprocessor mode - Invalid

* Default means the following:
- Standalone mode: Power up with no EEPROM (BIOS configures EZ-Host/OTG controller)
- Coprocessor mode: Power up in HPI mode.

Note: *The four switches assigned to selecting the Design Example # are more than is currently necessary, but they allow more selections as demos are added in the future.*

Note: If an invalid switch combination is set and the EZ-Host/OTG board is connected to a co-processor, the EZ-Host/OTG controller will power up in HPI mode, but the SBC will not be able to talk to the EZ-Host/OTG controller. The SBC can, however, talk to the rest of the DVK board. If an invalid switch combination is set and the EZ-Host/OTG board is NOT connected to a co-processor, it will power up in standalone mode with no boot EEPROM.

Table 5-7. Switch Settings for Switch 7 and 8

Switch 7	Switch 8	Function
OFF	OFF	Default to allow demos to run
X	ON	Undefined
ON	X	Undefined

Note: The EZ-Host/OTG board does not look at switches 7 and 8 for power-up h/w configuration.

Table 5-8. Table of Valid Switch Combinations

DIP[6:1]	Description
000000	"User" setting. Standalone mode has no EEPROM selected. Co-processor mode has HPI enabled.
000100	Demo 1, OTG, Standalone mode, EEPROM 1 Active
000110	Demo 1, OTG, Co-processor mode, HPI communication
001001	Demo 2, Dual Slave, Co-processor mode, SPI communication
001010	Demo 2, Dual Slave, Co-processor mode, HPI communication
001011	Demo 2, Dual Slave, Co-processor mode, HSS communication
001100	Demo 3, Peripheral Host Passthrough, Standalone mode, EEPROM 2 Active
001110	Demo 3, Peripheral Host Passthrough, Co-processor mode, HPI communication
010000	Demo 4, Dual Host, Standalone mode, EEPROM 3 Active
010010	Demo 4, Dual Host, Co-processor mode, HPI communication
1100XX	Standalone mode, EEPROM 4 active, IDE mode, CPLD disabled
111100	Standalone mode, EEPROM 4 active

Note: In this table, OFF=0 and ON=1

5.6.2 Pushbuttons

The pushbuttons necessary on the EZ-Host/OTG board for the demo functionality are as follows. They are laid out in a form factor resembling the SBC board as shown in the diagram below.

```
    X
X   X   X
    X
```

- Top button — S1: PB_UP, labeled "INC" (increment or "+")
- Bottom button — S5: PB_DOWN, labeled "DEC" (decrement or "-")
- Left Button — S2: PB_LEFT, labeled "Request HNP" (request a Host Negotiation Protocol if I am an OTG device at the moment)
- Right Button — S4: PB_RIGHT, labeled "SRP/Session End" (Request a Session Request Protocol if I am an OTG device and there is no active session. Force an end of session if I am an OTG host)
- Center button — S3: PB_ENTER, labeled "ENTER". Not used in the demos.

Note: *There is a pushbutton (S6) available that causes a hard reset to the EZ-Host/OTG board. Note that it would be a bad idea to push this button when connected to the SBC because communication with the SBC would be broken and likely not recoverable without a system power cycle.*

5.6.3 LEDs

- There is an "OTG ERROR" LED to indicate that one of the few OTG reportable errors has occurred, such as over current on the VBUS line or device not supported. This LED will work in conjunction with the hex display. In the case of an error, the "OTG ERROR" LED will be lit and the appropriate error code will be displayed on the hex display. Silkscreen on the board will be a key to the errors. See the LED_Write register for the "OTG ERROR" bit position.
- There are a "HOST" LED and a separate "PERIPHERAL" LED on the board that will be displayed when the board is acting in the appropriate function. These LEDs will only be useful when running the OTG demo because some of the other demos require the EZ-Host/OTG board to be a host and peripheral simultaneously. See the LED_Write register for the "HOST" and "Peripheral" bit position.
- There is a "SESSION ACTIVE" LED. Once again, it is only useful in the OTG demo. See the LED_Write register for the "SESSION ACTIVE" bit position.
- There is a "+5v" LED and a "+3.3v" LED. These will be lit when the appropriate power is present.

5.6.4 Hex Display

A seven segment hex display is available. It will be used in the OTG demo to show an incrementing or decrementing value or to display an error code. This display may also be used in other demos as a debugging aid. See the SSD_Write register for its bit positions.

5.7 Co-processor Mode Hints

When a EZ-Host/OTG board is in co-processor mode, it will:

1. Power up and be idle until the SBC toggles its hardware reset.
2. Read its DIP switch settings.
3. Detect it is in co-processor mode.
4. Detect what its method of communication with the SBC is based on DIP switch settings.
5. Configure itself for that mode.
6. Wait for communication from the SBC.

The SBC has the ability to:

- Determine that an EZ-Host/OTG board is attached.
- Control the hardware reset line of the EZ-Host/OTG board.
- Read the DIP switch settings of the EZ-Host/OTG board no matter the what mode the EZ-Host/OTG board thinks it is in. The ability of the SBC to read the DIP switches gives the opportunity to report back to the controlling software if the user has not configured the EZ-Host/OTG board correctly.

A couple of things to be aware of are:

- The SBC should reset the EZ-Host/OTG board.
- The SBC must wait a reasonable amount of time for the EZ-Host/OTG board BIOS to configure itself (see datasheet for minimum time delay).
- When the EZ-Host/OTG board is powered up correctly, the SBC will then have access to the pushbuttons and DIP switches and must drive the LEDs and seven-segment display on the EZ-Host/OTG board appropriately.

5.8 Recommended GPIO Settings for the EZ-Host/OTG Boards

The following table gives recommended GPIO settings for the EZ-Host/OTG controller when used on either the EZ-Host or EZ-OTG DVK board. Note the differences in parenthesis for the EZ-OTG controller.

Table Key:

I - Input
O - Output
I/O - Bidirectional
NA - BIOS sets it for you

Table 5-9. GPIO Settings

GPIO #	Standalone	Co-processor HPI	Co-processor SPI	Co-processor HSS
0	I/O	NA	I	I
1	I/O	NA	I	I
2	I/O	NA	I	I
3	I/O	NA	I	I
4	I/O	NA	I	I
5	I/O	NA	I	I
6	I/O	NA	I	I
7	I/O	NA	I	I
8	O	NA	NA	O
9	O	NA	NA	I
10	O	NA	NA	I
11	O	NA	NA	I
12	O	NA	O	O (NA for EZ-OTG)
13	O	NA	O (I for EZ-OTG)	O (NA for EZ-OTG)
14	O	NA	O	O (NA for EZ-OTG)
15	O	NA	O (I for EZ-OTG)	O (NA for EZ-OTG)
16	O	O	O	NA
17	I	I	I	NA
18	O	O	O	NA
19	O	NA	I	I
20	O	NA	I	I
21	O	NA	I	I
22	O	NA	I	I

Table 5-9. GPIO Settings

GPIO #	Standalone	Co-processor HPI	Co-processor SPI	Co-processor HSS
23	O	NA	I	I
24	I	NA	I	I
25	I	I	I	I
26	I	I	I	NA
27	I	I	I	I
28	O	O	O	O
29	I	I	I	I
30	NA	I	I	I
31	NA	I	I	I

5.9 Restoring the EZ-Host and EZ-OTG Boards to Factory Defaults

The EZ-Host board and the EZ-OTG board can be restored to factory defaults by doing the following things:

1. If the EZ-Host or EZ-OTG board is connected to the expansion port of a SBC, remove it.
2. Set all DIP switches to the OFF or OPEN position
3. Plug a USB cable from the PC to either of the board's USB peripheral ports. Note the PC must have the CY3663 DVK software installed.
4. Power on the EZ-Host or EZ-OTG board. The board will power up in a debug mode.
5. Open a DOS window. Alternatively, you can open a BASH shell.
6. Change your DOS prompt or BASH shell prompt to the Tools\Utilities subdirectory that was installed from the CD that originally shipped with the kit.
7. Set the DIP switches to only DIP switch 3 ON.
8. Type in the following at the DOS prompt. This runs a utility that downloads the de1_scan.bin file over USB to the board and programs it into the EEPROM that is selected by the DIP switch settings (EEPROM 1 in this case). Note that if you are using a BASH shell you must replace the backslash characters with forward slash characters.

   ```
   qtui2c ..\..\cy3663~1\stand-alone\de1_scan.bin f
   ```

9. Wait for the prompt to come back. The OTG design example is now restored to factory default. NOTE: This programming step is only valid on the EZ-Host board since the EZ-OTG board does not support the OTG design example in standalone mode.
10. Set the DIP switches to only 3 and 4 ON.

11. Type in the following at the DOS prompt. This runs a utility that downloads the de3_scan.bin file over USB to the board and programs it into the EEPROM that is selected by the DIP switch settings (EEPROM 2 in this case). Note that if you are using a BASH shell you must replace the backslash characters with forward slash characters.

```
qtui2c ..\..\cy3663~1\stand-alone\de3_scan.bin f
```

12. Wait for the prompt to come back. The Peripheral-Host design example is now restored to factory default
13. Set the DIP switches to only 5 ON.
14. Type in the following at the DOS prompt. This runs a utility that downloads the de4_scan.bin file over USB to the board and programs it into the EEPROM that is selected by the DIP switch settings (EEPROM 3 in this case). Note that if you are using a BASH shell you must replace the backslash characters with forward slash characters.

```
qtui2c ..\..\cy3663~1\stand-alone\de4_scan.bin f
```

15. Wait for the prompt to come back. The Dual Host design example is now restored to factory default. NOTE: This programming step is only valid on the EZ-Host board since the EZ-OTG board does not support the Dual Host design example in standalone mode.

Chapter 6. Developing with the Cypress StrongArm SBC

6.1 Normal Operation of the Cypress StrongARM SBC

6.1.1 Running Design Examples

The Cypress StrongARM SBC can be used for normal operation to run the Design Examples right out of the box. Do the following to connect it:

1. Connect the desired mezzanine card (either the EZ-Host board or the EZ-OTG board) to the expansion port.
2. Connect a PC running the OTG-Host Navigator software to the serial port on the SBC marked “P1 Serial #2”.
3. Verify that the bank of DIP switches on the SBC near the LCD has all switches in the off position.
4. Connect the power supply to the SBC.
5. Follow the instructions in the Design Example tutorial software.

6.1.2 Operation as a Linux USB Muti-port Host

The Cypress StrongARM SBC can be used as a multi-port USB host system right out of the box. The Linux kernel that ships on the board has USB class driver support for hub, mass storage, and audio. Do the following to boot the board into the multi-port USB host system mode:

1. If not already attached, connect the desired mezzanine card (either the EZ-Host board or the EZ-OTG board) to the expansion port.
2. Connect a PC running a serial terminal program such as HyperTerminal to the serial port on the SBC marked "P2 Serial #1". The terminal software should be set up for 115200 baud, no parity, 8 data bits, 1 stop bit, and no flow control.
3. Verify that the bank of DIP switches on the SBC near the LCD has all switches in the OFF or OPEN position.
4. Verify that the bank of DIP switches on the EZ-Host or EZ-OTG card (which is connected to the expansion port of the SBC) are in the OFF or OPEN position.
5. Connect the power supply to the SBC.

6. Watch the terminal window to see it boot to the Linux login prompt. Login username is "root", and there is no password.

6.2 Linux Development with the Cypress StrongARM SBC

6.2.1 Boot Sequence

The normal boot sequence of Cypress StrongARM SBC is as follows:

1. At power-on-reset, the ARMBoot boot loader executes.
2. Boot loader configures all the board's hardware.
3. The boot loader starts a countdown.
4. If uninterrupted by the user, the countdown to zero is reached and the Linux image resident in FLASH is then booted.
5. If desired, the user can follow the boot progress and login to Linux by doing the following: Connect a PC running a serial terminal program such as HyperTerminal to the serial port on the SBC marked "P2 Serial #1". The terminal software should be set up for 115200 baud, no parity, 8 data bits, 1 stop bit, and no flow control. Watch the terminal window to see it boot to the Linux login prompt. Login username is "root", and there is no password. Note that the terminal connection should be set up prior to powering on the board.

If development of the Linux kernel is desired, the user will generally want to have the boot loader download a kernel from a network TFTP server on each bootup. The following boot sequence should then be followed:

1. Upon power-on-reset, the ARMBoot boot loader executes.
2. Boot loader configures all the board's hardware, then requests an IP address from a network DHCP server if DHCP is enabled and a static IP address is not assigned.
3. Once the IP address is obtained, boot loader starts a countdown.
4. If uninterrupted by the user, the countdown to zero is reached and the boot loader looks for the TFTP server named in its "serverip" environmental setting.
5. Once the TFTP server is found, the boot loader looks for the Linux image named in its "bootcmd" environmental setting and downloads the image into RAM.
6. The IP address is relinquished to the DHCP server (if DHCP is enabled) and Linux boots from the downloaded image.
7. If desired, the user can follow the boot progress and login to Linux by doing the following: Connect a PC running a serial terminal program such as HyperTerminal to the serial port on the SBC marked "P2 Serial #1". The terminal software should be set up for 115200 baud, no parity, 8 data bits, 1 stop bit, and no flow control. Watch the terminal window to see it boot to the Linux login prompt. Login username is "root", and there is no password. Note that the terminal connection should be set up prior to powering on the board.

6.2.2 Using Ethernet Downloads During Development

A few changes are necessary for the boot loader to download an image from a TFTP server instead of its FLASH resident image. To make these changes, the boot loader must be interrupted by the user during its countdown by the user typing any key on the terminal (i.e. Windows HyperTerminal with baudrate set for 115200) connected to the serial port labeled “P2 Serial #1”. Once at the ARMBoot prompt, the user can view the environmental settings by typing “printenv” and make the following changes:

1. Tell the boot loader to use a different TFTP server by updating the “serverip” environmental setting. An example change is:

   ```
   <ARMBoot prompt>setenv serverip 172.19.3.137
   ```

   The above command tells the boot loader to look for a TFTP server at that IP address.

2. Tell the boot loader to download a Linux image from a TFTP server instead of using the on-board FLASH resident image by updating the “bootcmd” environmental setting. An example change is:

   ```
   <ARMBoot prompt>setenv bootcmd tftp c0000000 vmlinux.img\; bootm
   ```

   The above command tells the boot loader to download the file vmlinux.img from the tftp server, then boot the image.

   NOTE: To set the boot loader’s bootcmd environmental settings back to using the resident FLASH Linux image do the following:

   ```
   <ARMBoot prompt>setenv bootcmd bootm 80000
   ```

   This tells the boot loader to load the image located at 0x80000 in FLASH.

3. If a dynamic IP address is desired from a network DHCP server, enable DHCP in the boot loader by setting the "dhcpen" environmental setting as follows:

   ```
   <ARMBoot prompt>setenv dhcpen 1
   ```
   NOTE: To disable DHCP support in the boot loader, remove the "dhcpen" environmental setting as follows:

   ```
   <ARMBoot prompt>setenv dhcpen
   ```

   NOTE: If a static IP address is desired, make sure the "dhcpen" environmental setting is removed as shown above. Then assign a static IP address to the boot loader as follows (your actual IP address will be different than the example shown below):

   ```
   <ARMBoot prompt>setenv ipaddr 172.19.3.151
   ```

4. After all environmental settings have been made, update them permanently in FLASH by doing the following:

```
<ARMBoot prompt>saveenv
```

The above command tells ARMBoot to write the new environmental settings FLASH, so they will be used at each power-up.

5. After the environment is saved, reboot the board so any network setting changes can take effect. You can reboot by either pushing the button S1 on the SBC or by typing the following:

```
<ARMBoot prompt>reset
```

6.2.2.1 Updating FLASH contents over Ethernet

At some point, the user may want to write a new Linux image to FLASH. If that is the case, the following sequence of ARMBoot commands should be followed. Note: It is assumed that an Ethernet network is available and the board is configured properly for Ethernet as instructed in the previous section. Note: Overwriting the resident Linux image in FLASH will most likely prevent the Design Examples shipped with the kit from operating.

```
<ARMBoot prompt>tftp c0000000 vmlinux.img
<ARMBoot prompt>protect off 1:4-31
<ARMBoot prompt>erase 1:4-31
<ARMBoot prompt>cp.b c0000000 80000 37FFFF
```

The first command copies an image over the network from a TFTP server to RAM. The second command turns the write protect off the FLASH sectors to be erased. The third command erases the FLASH sectors. The fourth command copies the image from RAM into FLASH.

At some point, the user may want to write a new JFFS FLASH file system to FLASH. If that is the case, the following sequence of ARMBoot commands should be followed. NOTE: Overwriting the resident FLASH file system will most likely prevent the Design Examples shipped with the kit from operating.

```
<ARMBoot prompt>mw.b c0000000 FF c00000
<ARMBoot prompt>tftp c0000000 rootfs.jffs
<ARMBoot prompt>protect off 1:32-127
<ARMBoot prompt>erase 1:32-127
<ARMBoot prompt>cp.b c0000000 400000 BFFFFF
```

The first command fills memory with an FF pattern. The second command copies an image over the network from a TFTP server to RAM. The third command turns the write protect off the FLASH sectors to be erased. The fourth command erases the FLASH sectors. The fifth command copies the image from RAM into FLASH. NOTE: Because a FLASH file system is very large, the time needed to write it to FLASH may be several minutes.

6.2.3 Using Serial Downloads During Development

If development using an Ethernet network is not possible, a new kernel may be downloaded over the serial terminal port and booted. Of course, serial download is much slower than network down-

load, but an Ethernet infrastructure is not needed. The following shows an example download using Windows HyperTerminal.

1. Run HyperTerminal set up correctly (115200, n, 8, 1) and connected via a serial cable to the serial port labeled "P2 Serial #1".
2. Stop boot process at ARMBoot prompt.
3. At ARMBoot prompt, type the following. This tells ARMBoot you are going to download a file over serial using the kermit transfer protocol to address 0xc0000000.

```
<ARMBoot prompt>loadb c0000000
```

4. In HyperTerminal, choose the Transfer menu. Choose Send File. Browse to your file (probably vmlinux.img), and make sure the Protocol is Kermit. Choose Send. The file transfer process will probably take several minutes.
5. When the file is fully transferred, it is located starting at address 0xc0000000. At the ARMBoot prompt, type the following to boot the kernel:

```
<ARMBoot prompt>bootm c0000000
```

6.2.3.1 Updating FLASH contents over Serial

The kernel and filesystem residing in FLASH can be updated over the serial port if updating using an Ethernet network is not possible. The following shows an example download of the kernel and a write to FLASH using Windows HyperTerminal.

1. Run HyperTerminal set up correctly (115200, n, 8, 1) and connected via a serial cable to the serial port labeled "P2 Serial #1".
2. Stop boot process at ARMBoot prompt
3. At ARMBoot prompt, type the following. This tells ARMBoot you are going to download a file over serial using the kermit transfer protocol to address 0xc0000000.

```
<ARMBoot prompt>loadb c0000000
```

4. In HyperTerminal, choose the Transfer menu. Choose Send File. Browse to your file (probably vmlinux.img), and make sure the Protocol is Kermit. Choose Send. The file transfer process will probably take several minutes.
5. Copy the file now residing in RAM to FLASH with the following commands.

```
<ARMBoot prompt>protect off 1:4-31
<ARMBoot prompt>erase 1:4-31
<ARMBoot prompt>cp.b c0000000 80000 37FFFF
```

The following shows an example download of the jffs filesystem and a write to FLASH using Windows HyperTerminal.

1. Run HyperTerminal set up correctly (115200, n, 8, 1) and connected via a serial cable to the serial port labeled "P2 Serial #1".
2. Stop boot process at ARMBoot prompt

3. At ARMBoot prompt, type the following. This fills memory with a default value, then it tells ARMBoot you are going to download a file over serial using the kermit transfer protocol to address 0xc0000000.

```
<ARMBoot prompt>mw.b c0000000 FF c00000
<ARMBoot prompt>loadb c0000000
```

4. In HyperTerminal, choose the Transfer menu. Choose Send File. Browse to your file (probably rootfs.jffs), and make sure the Protocol is Kermit. Choose Send. The file transfer process will probably take several minutes.
5. Copy the file now residing in RAM to FLASH with the following commands.

```
<ARMBoot prompt>protect off 1:32-127
<ARMBoot prompt>erase 1:32-127
<ARMBoot prompt>cp.b c0000000 400000 bFFFFF
```

6.3 Restoring the SBC to Factory Defaults

The Cypress StrongARM SBC can be restored to factory defaults by doing the following things:

1. If the Linux kernel has been modified, follow the procedure in the "Updating FLASH contents over Ethernet" section or the "Updating FLASH contents over Serial" section to restore the kernel to the original. The original kernel binary can be found in the *CY3663 Binaries\coprocessor* subdirectory that was installed from the CD that shipped with the kit.
2. If the jffs FLASH filesystem has been modified, follow the procedure in the "Updating FLASH contents over Ethernet" section or the "Updating FLASH contents over Serial" section to restore the filesystem to the original. The original filesystem binary can be found in the *CY3663 Binaries\coprocessor* subdirectory that was installed from the CD that shipped with the kit.
3. At the ARMBoot prompt, type the following. These commands return the boot loader environment to default settings.

```
<ARMBoot prompt>setenv dhcpen
<ARMBoot prompt>setenv bootcmd bootm 80000
<ARMBoot prompt>saveenv
```

6.4 Cypress StrongARM SBC Memory Map

The memory map will be display in several layers. The first is an overview of the StrongARM memory map, then the FLASH, expansion port, and CPLD regions will be blown up to show more detail.

Table 6-1. StrongARM Memory Map

Physical Address Range	Resource Size	Use	CY 3663 Size: Width
E800,0000-FFFF,FFFF	Reserved 384 Mbyte	Reserved	—
E000,0000-E7FF,FFFF	Zeros Bank 128 Mbyte	Cache flush	Read zeros, no bus cycle
D800,0000-DFFF,FFFF	SDRAM bank 3 128 Mbyte	Empty	—
D000,0000-D7FF,FFFF	SDRAM bank 2 128 Mbyte	Empty	—
C800,0000-CFFF,FFFF	SDRAM bank 1 128 Mbyte	Empty	—
C000,0000-C7FF,FFFF	StrongARM SDRAM bank 0 128 Mbyte	SDRAM bank on SBC	32 Mbyte: 32 bits wide
B000,0000-BFFF,FFFF	LCD and DMA Control 256 Mbyte	Internal	—
A000,0000-AFFF,FFFF	Memory Control 256 Mbyte	Internal	—
9000,0000-9FFF,FFFF	SA-1110 System Control Module Register 256 Mbyte	Internal	—
8000,0000-8FFF,FFFF	Peripheral Control 256 Mbyte	Internal	—
5000,0000-7FFF,FFFF	Reserved 768 Mbyte	Reserved	—
4800,0000-4FFF,FFFF	StrongARM Chip Select 5 (nCS5) 128 Mbyte	Expansion Port	EZ-Host/EZ-OTG HPI access. See detailed Expansion Port Memory Map
4000,0000-47FF,FFFF	StrongARM Chip Select 4 (nCS4) 128 Mbyte	Various decodes: Network, User Interface, etc.	Decode by CPLD. See CPLD Memory Map
3000,0000-3FFF,FFFF	StrongARM PCMCIA/CF Slot B 256 Mbyte	Not Used	
2000,0000-2FFF,FFFF	StrongARM PCMCIA/CF Slot A 256 Mbyte	Not Used	
1800,0000-1FFF,FFFF	StrongARM Chip Select 3 (nCS3) 128 Mbyte	Not Used	
1000,0000-17FF,FFFF	StrongARM Chip Select 2 (nCS2) 128 Mbyte	Expansion Port	EZ-Host/EZ-OTG CPLD access: 16 Bit. See detailed Expansion Port Memory Map
0800,0004-0FFF,FFFF	StrongARM Chip Select 1 (nCS1) 128 Mbyte	SRAM	512 KBytes: 16 bits wide
0000,0000-07FF,FFFF	StrongARM Chip Select 0 (nCS0) 128 Mbyte	Boot/Application flash ROM on SBC	16 MBytes: 16 bits wide. See detailed FLASH Memory Map

Table 6-2. FLASH Memory Map

Physical Address Range	Description	Use	Size
0040,0000-00FF,FFFF	Sectors 32 – 127	JFFS – Flash File System (CFG_JFFS2_FIRST_SECTOR)	12.58M
0008,0000-003F,FFFF	Sectors 4 – 31	Linux Kernel	3670K
0006,0000-0007,FFFF	Sector 3	Spare (Unused)	128K
0004,0000-0005,FFFF	Sector 2	ARMboot Autoscripts	128K
0002,0000-0003,FFFF	Sector 1	Boot Parameters for ARMboot Boot Loader	128K
0000,0000-0001,FFFF	Sector 0	ARMboot Boot Loader Code	128K

Table 6-3. Expansion Port Memory Map

Physical Address Range	Linux Virtual Address	Description	Use
1000,0100	D900,0100	Expansion Port: StrongARM Chip Select 2 (nCS2)	CPLD Data Register on EZ-Host and EZ-OTG mezzanine cards
1000,0000	D900,0000	Expansion Port: StrongARM Chip Select 2 (nCS2)	CPLD Address Register on EZ-Host and EZ-OTG mezzanine cards
4800,0300	D800,0300	Expansion Port: StrongARM Chip Select 5 (nCS5)	HPI Port Status Register on EZ-Host and EZ-OTG mezzanine cards
4800,0200	D800,0200	Expansion Port: StrongARM Chip Select 5 (nCS5)	HPI Mailbox Register on EZ-Host and EZ-OTG mezzanine cards
4800,0100	D800,0100	Expansion Port: StrongARM Chip Select 5 (nCS5)	HPI Address Register on EZ-Host and EZ-OTG mezzanine cards
4800,0000	D800,0000	Expansion Port: StrongARM Chip Select 5 (nCS5)	HPI Data Register on EZ-Host and EZ-OTG mezzanine cards

The CPLD is programmed with control and decode logic for many functions. All CPLD activity uses the memory range assigned to StrongARM Chip Select 4. The memory map for this range is below.

Table 6-4. SBC CPLD Memory Map

Physical Address Range	Linux Virtual Address	Description	Use
4380,0000 - 43FF,FFFF		CPLD: StrongARM Chip Select 4 (nCS4)	Not Used: DEBUG area, not decoded by CPLD
4330,0000 - 437F,FFFF	D730,0000	CPLD: StrongARM Chip Select 4 (nCS4)	Not Used
4320,0000 - 432F,FFFF	D720,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC Expansion Port DIP Switches register (MEZ_CARD_DIP)
4310,0000 - 431F,FFFF	D710,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC Expansion Card Present regiser (MEZ_CARD_PRESENT)
4300,0000 - 430F,FFFF	D700,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC Expansion Port Reset Control register (SBC_EXP_RST)
4280,0000 - 42FF,FFFF	D600,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC CPLD Version register (CPLD_VERSION)
4200,0000 - 427F,FFFF	D500,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC LCD control registers
4180,0000 - 41FF,FFFF	D400,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC LED control register (LED_CTL)
4100,0000 - 417F,FFFF	D300,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC DIP switches and pushbuttons registers (DIP_PB_CTL)
4080,0000 - 40FF,FFFF		CPLD: StrongARM Chip Select 4 (nCS4)	Not Used
4000,0000 - 407F,FFFF	D200,0000	CPLD: StrongARM Chip Select 4 (nCS4)	SBC network chip select decode region (NETWORK_DECODE)

6.5 SBC CPLD Register Descriptions

Note: All accesses to the CPLD memory region must be 16-bit accesses

6.5.1 NETWORK_DECODE

Address range that decodes the Cirrus CS-8900A Ethernet IC

Physical Address range: 0x40000000 – 0x407FFFFF

Virtual Address range: 0xD2000000 – 0xD20FFFFF

Read/Write

6.5.2 DIP_PB_CTL

Register that holds the current SBC DIP switch values and latched value of SBC push-buttons; it also allows clearing of the push-button latches.

Physical Address: 0x41000000

Virtual Address: 0xD3000000

Read/Write

Resets to binary XXX0,0000,XXXX,XXXX

Read

Bit	Description
7:0	Value of SBC DIP switch bank (switch 8 is bit 7, etc.)
8	PB_DOWN latched value (1 is latched, 0 is unlatched)
9	PB_RIGHT latched value (1 is latched, 0 is unlatched)
10	PB_ENTER latched value (1 is latched, 0 is unlatched)
11	PB_LEFT latched value (1 is latched, 0 is unlatched)
12	PB_UP latched value (1 is latched, 0 is unlatched)
15:13	Unused

Write

A write of any value to this register will clear all the pushbutton latches.

6.5.3 LED_CTL

Register that controls the LEDS labeled “LED3” and “LED4” on the SBC

Physical Address: 0x41800000

Virtual Address: 0xD4000000

Read/Write

Resets to binary XXXX,XXXX,XXXX,XX00

Read/Write

Bit	Description
0	LED3 pin value (low turns LED off, high turns LED on)
1	LED4 pin value (low turns LED off, high turns LED on)
15:2	Unknown

6.5.4 LCD_CTL

Register that holds characters to be written to the LCD and allows either automatic or manual control of the LCD. When using the LCD, the user normally reads bit 2 of the LCD_STATUS_3 register, and if clear, writes a character and the proper control bits to this register (user writes the character bits in positions 7:0, a 1 to bit 8, and either the command or data bit value to bit 9. Bits 12:10 should be written as zeros). When this register is written, a state machine in the CPLD automatically transfers the character to the LCD. Alternatively, this register allows manual bit-bang control of the 16-character by 2-line LCD. The LCD used is the Optrex DMC16204NY-LY. The character and control bits can be found in the "Character LCD User's Manual", which can be downloaded from the Optrex web site.

Physical Address: 0x42000000 – 0x427FFFFF

Virtual Address: 0xD5000000

Write only

Resets to binary XXX1,0001,0000,0000

Write

Bit	Description
7:0	LCD data bits. No effect if bit 11 is clear.
8	LCD Read/Write bit. High is read, low is write.

9	LCD Register Select bit. Chooses either command accesses or data accesses to the LCD.
10	LCD enable bit. This bit is the read or write strobe that causes an LCD access in bit-bang mode. No effect if bit 11 is clear.
11	Bit-bang bit. If set, the values of bits 10:0 of LCD_CTL register are driven to the LCD (Note: If bit 8 is set, the LCD drives data to the LCD, so bits 7:0 will have no effect.)
12	Auto Init bit. If set, this bit forces an initialization of the LCD. This bit automatically clears itself.
15:13	Unused

6.5.5 LCD_STATUS_0

Register that allows a read-back of LCD_CTL register.

Physical Address: 0x42000000

Virtual Address: 0xD5000000

Read only

Resets to binary XXX1,0001,0000,0000

Read

Bit	Description
12:0	Read value of LCD_CTL register
15:13	Unknown

6.5.6 LCD_STATUS_1

Register that allows values of LCD pin signals to be read

Physical Address: 0x42000002

Virtual Address: 0xD5000002

Read only

Resets to binary XXXX,XXXX,XXXX,XXXX

Read

Bit	Description
7:0	Value of LCD data lines
8	Value of LCD signal NDRW
9	Value of LCD signal DRS
10	Value of LCD signal DE
15:11	Unknown

6.5.7 LCD_STATUS_3

Register used when CPLD state machine control of LCD is desired. The user monitors bit 2 of this register to keep from writing characters to the LCD too fast. When bit 2 is clear, the user writes the character code to the LCD_CTL register.

Physical Address: 0x42000004

Virtual Address: 0xD5000004

Read only

Resets to binary XXXX,XXXX,XXXX,XXXX

Read

Bit	Description
0	AUTO_LCD_INIT – If high, the LCD is actively being initialized. If low, LCD is initialized.
1	LCD_WRITE_REQUEST – If high, the LCD control state machine is actively writing a character to the LCD.
2	LCD_BUSY - If high, don't write to LCD. IF low, OK to write a character to LCD. This bit is a logical OR of bits 0 and 1 of this register.
15:3	Unknown

6.5.8 CPLD_VERSION

Register used to find the programmed version of the CPLD.

Physical Address: 0x42800000

Virtual Address: 0xD6000000

Read only

Resets to binary XXXX,XXXX,XXXX,XXXX

Read

Bit	Description
3:0	Version of CPLD
15:4	Unknown

6.5.9 SBC_EXP_RST

Register used to control the reset line on the expansion port.

Physical Address: 0x43000000

Virtual Address: 0xD7000000

Read/Write

Resets to binary XXXX,XXXX,XXXX,XXX0

Read/Write

Bit	Description
0	Value driven on CPLD_IO[11] signal which goes to expansion port. Currently this signal is used as the master expansion port reset.
15:1	Unknown. No effect.

6.5.10 MEZ_CARD_PRESENT

Register used to detect the presence of a mezzanine card attached to the expansion port. Both the EZ-Host and EZ-OTG boards will drive this signal low if attached.

Physical Address: 0x43100000

Virtual Address: 0xD7100000

Read Only

Resets to binary XXXX,XXXX,XXXX,XXXX

Read

Bit	Description
0	If high, and expansion card is present. If low, an expansion card is not present. This bit reflects the level of the CPLD_IO[2] bit on the SBC.
15:1	Unknown

6.5.11 MEZ_CARD_DIP

Register used to read the value of the bank of DIP switches residing on the mezzanine card.

Physical Address: 0x43200000

Virtual Address: 0xD7200000

Read Only

Resets to binary XXXX,XXXX,XXXX,XXXX

Read

Bit	Description
0	DIP1 on mezzanine card, which is connected to CPLD_IO[4] pin on SBC
1	DIP2 on mezzanine card, which is connected to CPLD_IO[5] pin on SBC
2	DIP3 on mezzanine card, which is connected to CPLD_IO[6] pin on SBC
3	DIP4 on mezzanine card, which is connected to CPLD_IO[7] pin on SBC
4	DIP5 on mezzanine card, which is connected to CPLD_IO[8] pin on SBC
5	DIP6 on mezzanine card, which is connected to CPLD_IO[9] pin on SBC
6	DIP7 on mezzanine card, which is connected to CPLD_IO[10] pin on SBC
7	DIP8 on mezzanine card, which is connected to CPLD_IO[15] pin on SBC
15:8	Unknown

Appendix A
Definitions

Term	Definition
CY16--	The RISC processor core in the EZ-Host and EZ-OTG controllers.
DVK	Development Kit.
Expansion Card CPLD	A Cypress 37K series CPLD on either the EZ-Host board or EZ-OTG board that controls several functions on those cards, including accesses from the Cypress StrongARM SBC, if attached.
EZ-Host	A development PCA that contains the Cypress CY7C67300 USB Host/Peripheral controller. The EZ-Host board can operate as a stand-alone device or as a mezzanine card connected to the expansion port of the Cypress StrongARM SBC.
EZ-Host/OTG	A general name used to describe either the EZ-Host (CY7C67300) or EZ-OTG (CY7C67200) USB Host/Peripheral controllers.
EZ-Host/OTG CPLD	The CPLD resident on either the EZ-Host or the EZ-OTG board. See Expansion Card CPLD.
EZ-OTG	A development PCA that contains the Cypress CY7C67200 USB Host/Peripheral controller. The EZ-OTG board can operate as a stand-alone device or as a mezzanine card connected to the expansion port of the Cypress StrongARM SBC.
HPI	Host Port Interface. 16-bit interface from external device to EZ-Host/OTG controller.
HSS	High Speed Serial. An asynchronous serial interface on the EZ-Host/OTG controller that can operate at very high baud rates and is one of three available co-processor communication modes.
SBC	Cypress StrongARM Single Board Computer
SBC CPLD	A Cypress 39K series CPLD resident on the Cypress StrongARM SBC that controls several logic and decode functions.
SPI	Serial Peripheral Interface. A synchronous serial interface on the EZ-Host/OTG controller that is one of three available co-processor modes.
SSD	Seven Segment Display. A hex display present on the EZ-Host/OTG boards.