XPS USB Host Controller (v1.01a)

DS734 September 16, 2009 **Product Specification**

Introduction

The XPS USB Host Controller is a soft Xilinx IP core for use with the Xilinx Embedded Development Kit (EDK). The core is designed to support the Enhanced Host Controller Interface (EHCI) for USB 2.0 host controllers. The core incorporates integral DMA function controlled by the EHCI Controller block. This assures the highest performance data transfer topology with the system memory. The Standard EHCI register set is accessed via a PLBV46 Slave interface, affording support by the Xilinx MicroBlaze™, Power®PC 405, and PowerPC 440 microprocessors. The core interfaces to an external USB 2.0 PHY (physical interface) via the industry standard ULPI signal interface (8-bit data).

Features

- USB 2.0 Host Controller (v1_01_a)
  - ♦ USB High speed device support (480 Mbps)
  - ♦ USB Full Speed device support (12 Mbps)
    - Low speed devices are not supported
  - ♦ Supports Intel EHCI 1.0 specification for standardized register and system software (RTOS) interfaces
    - Integrated Transaction Translator enables the use of EHCI RTOS interface for Full Speed device support
    - OHCI and UHCI RTOS support is not required
  - ♦ Incorporates ULPI PHY (8-bit data) interface for external (to FPGA) USB 2.0 PHY connection per ULPI v1.1 specification

LogiCORE™ IP Facts	
Core Specifics	
Supported Device Family	Virtex®-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3A
Resources Used	See Table 21, Table 22, Table 23
Version of core	1.01a
Provided with Core	
Documentation	Product Specification
Design File Formats	VHDL
Constraints File	EDK TCL Generated
Verification	N/A
Instantiation Template	EDK
Design Tool Requirements	
Xilinx Implementation Tools	ISE® 11x
Verification	ModelSim PE/SE 6.4b or later
Simulation	ModelSim PE/SE 6.4b or later
Synthesis	XST
Support	
Provided by Xilinx, Inc.	

Features (contd)

- USB 2.0 Host Controller (v1_01_a)
  - Integrated DMA functionality for efficient data transfers between the Host Controller and System memory
    - PLBV46 Master interface provided for dedicated DMA of USB data and EHCI transfer descriptors to and from the Embedded System main memory
    - DMA functionality is leveraged from the EHCI defined functionality and is autonomous to the Embedded System software
- PLBV46 Slave interface for register access and configuration by the Embedded System microprocessor

Functional Description

Figure 1 illustrates the functional composition of the core. The design is an Xilinx Pay Core and delivered using the Xilinx EDK tool suite. Pay Cores are delivered in EDK as encrypted source files and contain a disable timer. Purchasing the appropriate Xilinx core license will enable the delivery of the core without the disable timer. The core may be used in evaluation mode which allows for approximately 8 hours of operation before being disabled.

The core's design has three primary interfaces, the PLBV46 Slave interface, the PLBV46 Master interface, and the 8-bit ULPI interface. Register access and configuration is provided via the PLBV46 Slave interface. High speed DMA connection to system memory is via the PLBV46 Master interface. The backend connection to an external USB 2.0 PHY is provided with the ULPI interface (8-bit data).

The register interface for the Host Controller is provided by the Register Module. The standard EHCI Specified register set includes the Capability Registers, the Operational Registers, and one PortSC register (single port root hub emulation). In addition, the Linux programming model for use with a Transaction Translator requires an additional non-EHCI R/W register designated the Mode Register. A single system interrupt is generated within the Register Module and is output for routing to the User's System Interrupt Controller. The interrupt is a merged (OR) of the operational interrupts required by the EHCI specification. The Register Module also generates the Port Indicator. These are provided for use by the User to connect to external port indicator LEDs and reflect the EHCI port indicator bit definitions (See the PORTSC register). The Host Controller's Register compliment is detailed in "XPS USB Host Registers".

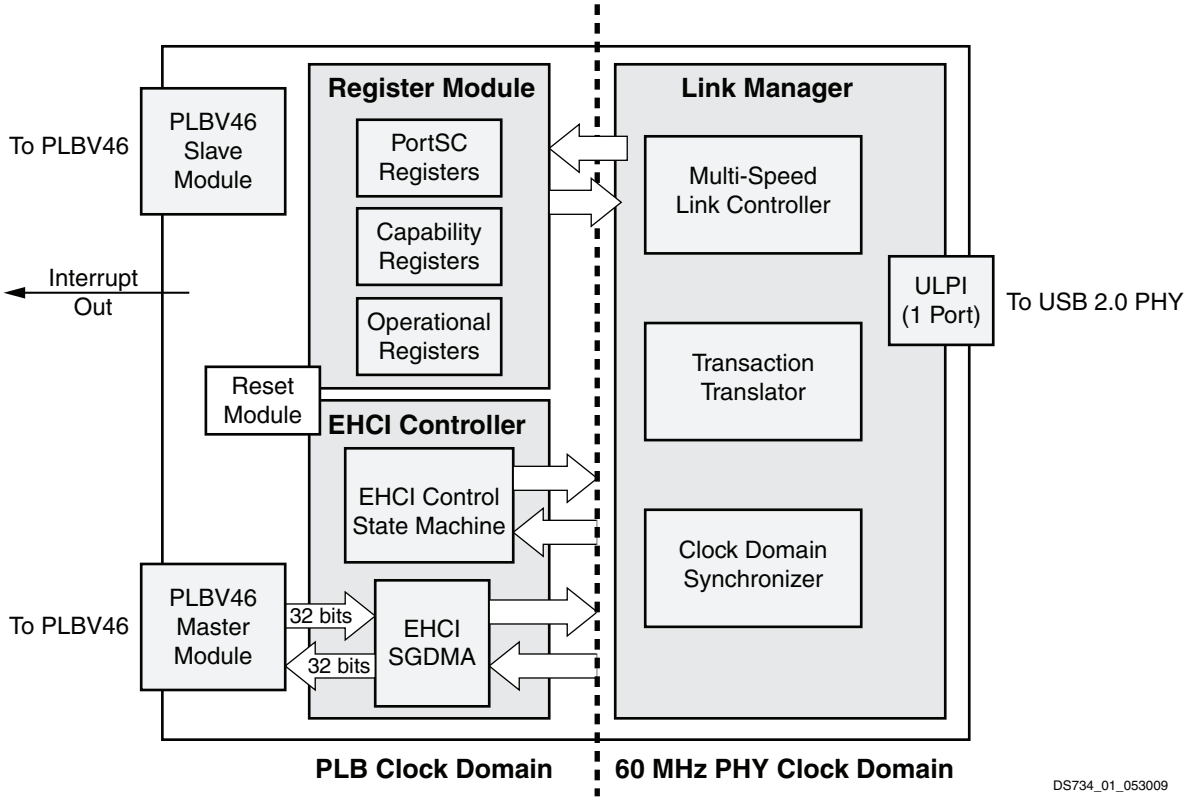

Figure 1: **XPS USB Host Controller Block Diagram**

The EHCI Controller Module is the main control logic for supporting the EHCI Transfer Descriptor (TD) execution and system interface protocol. It also controls the DMA operations for TD fetch and store as well as the USB Data fetch and store with the Embedded System Main Memory. This is via the Host's PLBv46 Master interface.

The Link Manager (LM) is responsible for management of the external USB 2.0 PHY via the ULPI interface. The LM includes 3 major sub-blocks, the Multi-Speed Link Controller (MLC), the Transaction Translator (TT), and the Clock Domain Synchronizer. The MLC is responsible for management of the ULPI interface and the external USB 2.0 PHY. This includes device connect/disconnect detection, speed evaluation, port power management, suspend/resume, and port reset control. The Transaction Translator is active when a Full Speed device is connected directly to the Host Controller's single root hub port. The TT provides the necessary logic and buffering to translate Hi-speed EHCI controlled protocol to Full Speed protocol. The TT is not active if a Full Speed device is connected to the Host via an external HS Hub. The Clock Domain Synchronizer provides the clock domain crossing logic between the applicable PLB clocked logic and the logic synchronized to the ULPI clock (60MHz input from PHY). This synchronizing logic includes various double registered bit synchronizers and 4 asynchronous FIFOs used for the Control, Status, and In/Out Data transfers between the Link Manager and the EHCI Controller.

Typical System Interconnect

The USB Host core is designed to be connected via PLBv46 in the User's System. A typical MicroBlaze processor configuration is shown in Figure 2. A typical PPC440 system is shown in Figure 3. The System's microprocessor has access to the USB Host registers via the PLBV46 Slave Interface. An integral DMA Engine coordinates high speed data transfer between the Host Core and the System Memory via the PLBV46 Master interface. The PLBV46 Master interface may be directly connected to the multi-port memory controller (MPMC5) or PPC440 SPLB using a point to point topology or to a PLBV46 shared bus configuration (not shown). The Host Core is connected to the USB 2.0 PHY (external to the FPGA) via the 8-bit ULPI interface. The Host Core derives the backend clock synchronization from the 60MHz clock input from the PHY device. The PLBV46 Master and Slave interfaces utilize the PLBV46 clock. The single interrupt output of the USB Host Core is routed to the System Interrupt Controller.

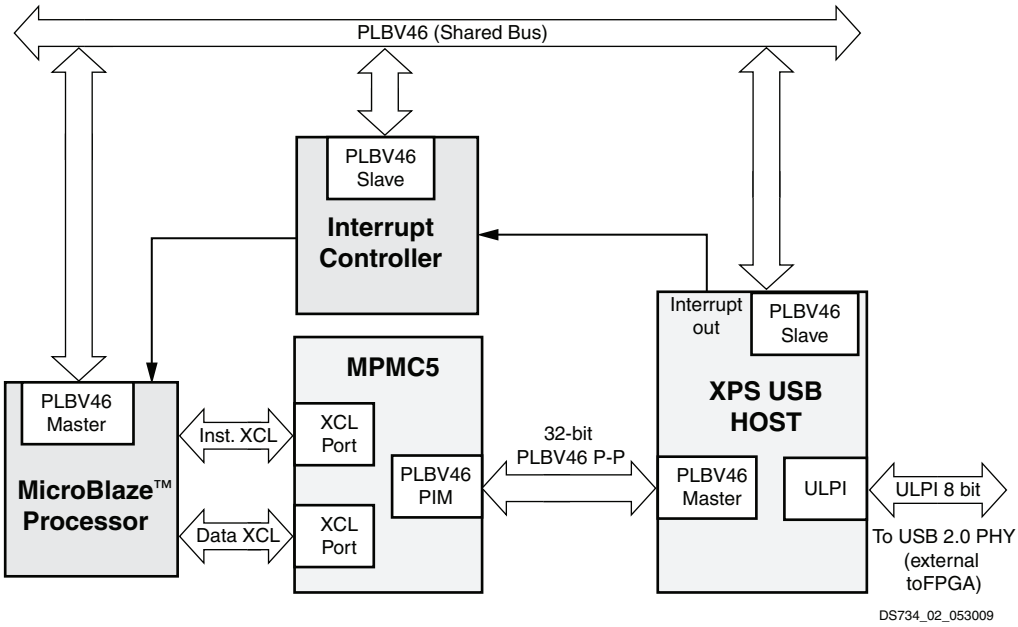

Figure 2: **Typical MicroBlaze Processor System Configuration**

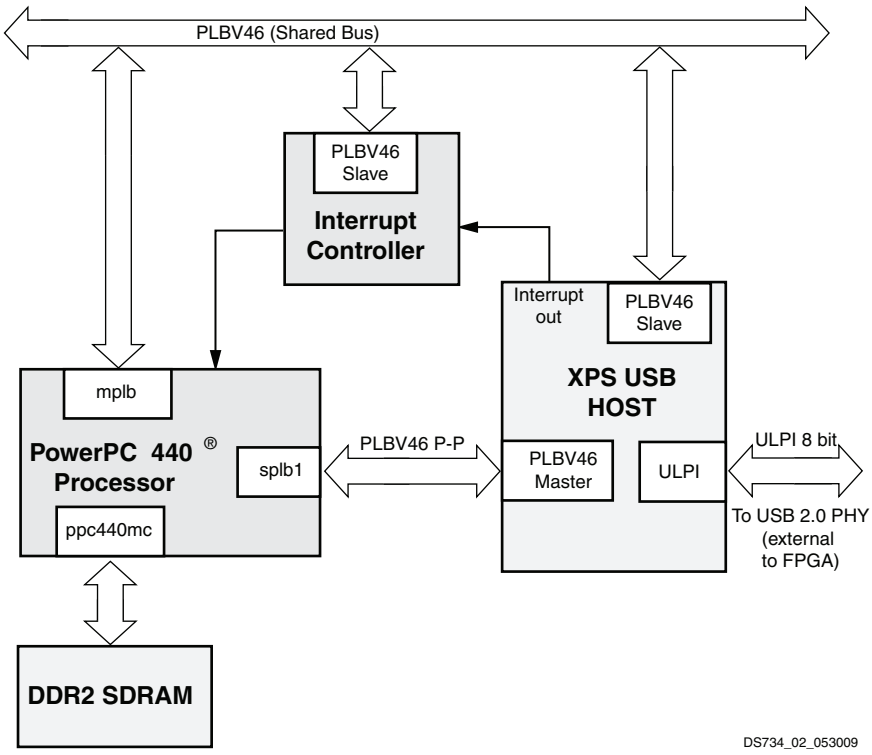

Figure 3: **Typical PPC440 Processor System Configuration**

I/O Signals

The XPS USB High Speed Host Controller signals are listed and described in Table 1.

Table 1: **XPS USB Host Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
System Signals				
Port_Indicator(1:0)	Port Indicator LED	O	"00"	USB Port Indicator LED control bits (per EHCI Spec.) "00" = off "01" = Amber "10" = Green" "11" = Not defined
Host_Intr_out	Interrupt	O	0	Interrupt output to System Interrupt Controller or Processor
PLB Slave Interface Input Signals				
SPLB_Clk	PLB Slave	I		PLB System synchronization Clock
SPLB_Rst	PLB Slave	I		PLB System PLB Reset
PLB_ABus(0:31)	PLB Slave	I		PLB address bus
PLB_UABus(0:31)	PLB Slave	I		PLB upper address bus
PLB_PAValid	PLB Slave	I		PLB primary address valid indicator
PLB_SAValid	PLB Slave	I		PLB secondary address valid indicator

Table 1: **XPS USB Host Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
PLB_rdPrim	PLB Slave	I		PLB secondary to primary read request promotion
PLB_wrPrim	PLB Slave	I		PLB secondary to primary write request promotion
PLB_masterID(0:C_SPLB_MID_WIDTH-1)	PLB Slave	I		PLB requesting master identification
PLB_abort	PLB Slave	I		PLB bus request abort
PLB_busLock	PLB Slave	I		PLB bus lock
PLB_RNW	PLB Slave	I		PLB read not write
PLB_BE(0:(C_SPLB_DWIDTH / 8) -1)	PLB Slave	I		PLB byte enables
PLB_MSize(0:1)	PLB Slave	I		PLB master data bus size
PLB_size(0:3)	PLB Slave	I		PLB transfer size
PLB_type(0:2)	PLB Slave	I		PLB transfer type
PLB_lockErr	PLB Slave	I		PLB lock error
PLB_wrDBus(0:C_SPLB_DWIDTH -1)	PLB Slave	I		PLB write data bus
PLB_wrBurst	PLB Slave	I		PLB burst write transfer
PLB_rdBurst	PLB Slave	I		PLB burst read transfer
PLB_wrPendReq	PLB Slave	I		PLB pending write request
PLB_rdPendReq	PLB Slave	I		PLB pending read request
PLB_wrPendPri(0:1)	PLB Slave	I		PLB pending write request priority
PLB_rdPendPri(0:1)	PLB Slave	I		PLB pending read request priority
PLB_reqPri(0:1)	PLB Slave	I		PLB current request priority
PLB_TAttribute(0:15)	PLB Slave	I		PLB Transfer Attribute bus
PLB Slave Interface Output Signals				
Sl_addrAck	PLB Slave	O	’0’	Slave address acknowledge
Sl_SSize(0:1)	PLB Slave	O	Zeros	Slave data bus size
Sl_wait	PLB Slave	O	’0’	Slave wait indicator
Sl_rearbitrate	PLB Slave	O	’0’	Slave rearbitrate bus indicator
Sl_wrDAck	PLB Slave	O	’0’	Slave write data acknowledge
Sl_wrComp	PLB Slave	O	’0’	Slave write transfer complete indicator
Sl_wrBTerm	PLB Slave	O	’0’	Slave terminate write burst transfer
Sl_rdDBus(0:C_SPLB_DWIDTH -1)	PLB Slave	O	Zeros	Slave read bus
Sl_rdWdAddr(0:3)	PLB Slave	O	Zeros	Slave read word address
Sl_rdDAck	PLB Slave	O	’0’	Slave read data acknowledge
Sl_rdComp	PLB Slave	O	’0’	Slave read transfer complete indicator
Sl_rdBTerm	PLB Slave	O	’0’	Slave read burst terminate indicator

Table 1: **XPS USB Host Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
SI_MBusy(0:C_SPLB_NUM_MASTERS-1)	PLB Slave	O	Zeros	Slave busy indicator (1 bit for each PLB Master)
SI_MWrErr(0:C_SPLB_NUM_MASTERS-1)	PLB Slave	O	Zeros	Slave write error indicator (1 bit for each PLB Master)
SI_MRdErr(0:C_SPLB_NUM_MASTERS-1)	PLB Slave	O	Zeros	Slave read error indicator (1 bit for each PLB Master)
SI_MIRQ(0:C_SPLB_NUM_MASTERS-1)	PLB Slave	O	Zeros	Slave interrupt indication (1 bit for each PLB Master)
PLBV46 Master Interface Output Signals				
MPLB_Clk	PLB Mstr	I		Master PLB System synchronization Clock (not used internally, SPLB_Clk is used as PLB side clocking source in the XPS USB Host)
MPLB_Rst	PLB Mstr	I		Master PLB System PLB Reset
MD_Error	Sideband	O	'0'	Master Detected Error output. This is an active high sticky bit that is set whenever the Master Module encounters an error condition during operation. It can only be cleared by a reset of the master Module either via the PLB reset input or the EHCI Controller requesting a Master reset.
M_request	PLB Mstr	Output	'0'	Master access request
M_priority (0:1)	PLB Mstr	Output	Zeros	Master Request Priority qualifier
M_buslock	PLB Mstr	Output	'0'	Master Bus Lock Request qualifier
M_RNW	PLB Mstr	Output	'0'	Master Read not Write qualifier
M_BE(0 : **C_MPLB_DWIDTH**/8-1)	PLB Mstr	Output	Zeros	Master BE qualifier
M_MSize(0 : 1)	PLB Mstr	Output	Zeros	Master Size qualifier (Native Data Width)
M_size(0 : 3)	PLB Mstr	Output	Zeros	Master Transfer Size qualifier
M_type(0 : 2)	PLB Mstr	Output	Zeros	Master Transfer Type qualifier
M_TAttribute(0:15)	PLB Mstr	Output	Zeros	Master Transfer Attributes qualifier
M_lockErr	PLB Mstr	Output	'0'	Master Lock Error qualifier
M_abort	PLB Mstr	Output	'0'	Master request abort qualifier Never asserted, Always logic '0'
M_UABus(0 : 31)	PLB Mstr	Output	Zeros	Master Upper Address qualifier Always driven to zeros
M_ABus(0 : 31)	PLB Mstr	Output	Zeros	Master Request Lower Address qualifier
M_wrDBus(0 : **C_MPLB_DWIDTH**-1)	PLB Mstr	Output	Zeros	Master Write Data Bus
M_wrBurst	PLB Mstr	Output	'0'	Master Write Burst qualifier
M_rdBurst	PLB Mstr	Output	'0'	Master Read Burst qualifier
PLBV46 Master Interface Input Signals				
PLB_MAddrAck	PLB Mstr	Input		PLB Address Acknowledge
PLB_MSSize(0 : 1)	PLB Mstr	Input		PLB Slave Size indication

Table 1: **XPS USB Host Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
PLB_MRearbitrate	PLB Mstr	Input		PLB Slave Rearbitrate indication
PLB_MTimeout	PLB Mstr	Input		PLB Bus Timeout indication
PLB_MBusy	PLB Mstr	Input		PLB Slave Busy indication
PLB_MRdErr	PLB Mstr	Input		PLB Slave Read Error indication
PLB_MWrErr	PLB Mstr	Input		PLB Slave Write Error indication
PLB_MIRQ	PLB Mstr	Input		PLB Slave Interrupt request Not used by the Master Interface
PLB_MRdDBus(0 : **C_MPLB_DWIDTH**-1)	PLB Mstr	Input		PLB Read Data Bus
PLB_MRdWdAddr(0 : 3)	PLB Mstr	Input		PLB Read Word Address Not used by the Master
PLB_MRdDAck	PLB Mstr	Input		PLB Read Data Acknowledge
PLB_MRdBTerm	PLB Mstr	Input		PLB read Burst Terminate
PLB_MWrDAck	PLB Mstr	Input		PLB Write Data Acknowledge
USB PHY Reset				
USB_PHY_Reset	USB PHY (Not a ULPI defined signal)	O	0	Active high auxiliary reset for external USB PHY connection. It echoes the internal hardware reset used by the XPS USB Host design. Note that is this signal may or may not be used. This is dependent on the User's USB PHY circuit design and PHY selection.
ULPI Interface Signals				
ULPI_Clock	ULPI	I	N/A	60MHz input clock from the ULPI PHY used for synchronization within the XPS USB Host (Link Manager is synchronized with this clock)
ULPI_Dir	ULPI	I	N/A	**Direction:** Direction of data flow between Host and ULPI PHY (See ULPI V1.1 Spec)
ULPI_Stp	ULPI	O	0	**Stop:** Indicator that transmission of last byte is complete (See ULPI V1.1 Spec)
ULPI_Nxt	ULPI	I	N/A	**Next:** Indicator of when the PHY is ready for the next byte (See ULPI V1.1 Spec)
ULPI_Data_I(7:0) (1)	ULPI	I	N/A	**Input Data:** Input data to the core from the ULPI PHY (See ULPI V1.1 Spec)
ULPI_Data_O(7:0)	ULPI	O	Zeros	**Output Data:** Output data from the Host to the ULPI PHY (See ULPI V1.1 Spec)
ULPI_Data_T(7:0)	ULPI	O	0	**Tri-state:** ULPI_Data is a 3-state port, with ULPI_Data_I as the IN port, ULPI_Data_O as the OUT port and ULPI_Data_T as the tristate output. One for one bit match to data bits.

1. The ULPI Data bus is defined in the ULPI specification as a bi-directional bus. Bi-directional buses are not supported within the targeted FPGA families for this core. Therefore bi-dir buses are split into an input bus, an output bus, and a direction control. Using standardize Xilinx notation, these are converted to a true bi-directional bus at the FPGA I/O pin boundary (external to this core design)

XPS USB Host Controller Parameters

To allow the user to obtain an XPS USB HS Host Controller that is uniquely tailored for their system, certain features can be parameterized in the XPS USB HS Host Controller design. This allows the configuration of a design that utilizes only the resources required by the system and that operates with the best possible performance. The features that can be parameterized in Xilinx XPS USB HS Host Controller design are shown in Table 2.

Table 2: **XPS USB Host Controller Design Parameters**

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
General Host Parameters				
Enable Host Support of USB Full Speed devices	C_SUPPORT_ USB_FS	0, 1 0 = High Speed device support only 1= High Speed device support plus Full Speed device Support	1	integer
Indication that Host can control the USB power via User's external PHY circuit	C_HAS_PWR_ CNTL	0,1 0 = Host Power Control is not available 1 = Host Power control is available	1	integer
Indicates if the User's external circuit design requires the PHY to provide VBus power from it's internal charge pump	C_USE_PHY_ BUS_PWR	0 = External PHY not sourcing VBus power 1 = PHY is sourcing VBus Power	0	Integer
Indicates if the User's external PHY circuit needs to provide the VBus valid indicator to the PHY from a source outside of the PHY	C_EXT_VBUS_ VALID	0 = Use internal PHY VBus valid comparator 1 - Use External VBus valid indicator	1	integer
Indicates if the User's external VBus valid indicator to the PHY (if used) needs to be inverted by the PHY for proper interpretation	C_EXT_VBUS_ VALID_INVERT	0 = Don't Invert external Vbus valid input 1 = Invert external Vbus valid input	0	integer
PLBV46 Master Interface Parameters				
Width of the PLB Address Bus to which the Master is attached	C_MPLB_ AWIDTH	32 (1)	32 (2)	integer
Width of the PLB Data Bus to which the Master is attached	C_MPLB_ DWIDTH	32, 64, 128	32	integer
Specifies the connection topology of the PLBV46 attachment to the PLBV46 Master port. This is currently not used in the PLBV46 Master interface.	C_MPLB_P2P	0, 1 0 = Shared Bus PLBV46 (Normal)	0	integer
PLBV46 Slave Interface Parameters				
PLB Slave Register Base Address assignment	C_SPLB_ BASEADDR (3)	System Address value of C_SPLB_AWIDTH bits wide	FFFF_FFFF	std_logi c_vector

Table 2: **XPS USB Host Controller Design Parameters**

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
PLB Slave Register HIGH Address assignment	C_SPLB_ HIGHADDR	System Address value of C_SPLB_AWIDTH bits wide	0000_0000	std_logi c_vector
Width of the PLB Address Bus Width to which the Slave is attached	C_SPLB_ AWIDTH (4)	32	32	integer
Width of the PLB Data bus to which the Slave is attached	C_SPLB_ DWIDTH	32, 64, or 128	32	integer
Number of PLB Masters present on the PLB to which the Slave is attached	C_SPLB_NUM_ MASTERS	1 - 16	1	integer
PLB Width of Master ID Bus of the PLB to which the Slave is attached	C_SPLB_MID_ WIDTH	roundup(log_2(C_SPLB_NUM_MASTERS))	1	integer
Configures the Slave for Point to Point PLB Optimizations	C_SPLB_P2P (5)	0,1 0 = Shared Bus PLBV46 (Normal) 1 = Point to Point PLBV46	0	Integer
Period of the SPLB_Clk in pico-seconds	C_SPLB_CLK_ PERIOD_PS	5000 to 16666 (200MHz to 60MHZ)(6)	10,000 (100MHz)	integer
Target FPGA Family Parameter				
Target FPGA device family	C_FAMILY	"spartan3a", “spartan6, "virtex4", "virtex5", “virtex6”	"virtex5"	String

1. Xilinx EDK limits addressing to 32 bits.
2. Default values are specified for C_BASEADDR and C_HIGHADDR to ensure that they are set by the user. If the value is not set, an implementation error will be generated.
3. C_BASEADDR value must be a power of 2 and a multiple of 512 bytes.
4. These parameters are calculated and automatically assigned by the EDK XPS tools during the system creation process.
5. Point to point optimizations include removal of address decoding. This mode is not usable in a shared bus interconnect environment.
6. The parameter allows for specification of up to 200 MHz, but the core is rated only for those frequencies specified in Table 24, “USB Host Controller System Performance,” on page 45.

Allowable Parameter Combinations

Table 3: **Allowable Parameter Combinations**

Parameter Name	Affects Parameter	Relationship Description
C_HAS_PWR_CNTL	C_USE_PHY_BUS_PWR	Affected Parameter is ignored when C_HAS_PWR_CNTL = 0
C_HAS_PWR_CNTL	C_EXT_VBUS_VALID	Affected Parameter is ignored when C_HAS_PWR_CNTL = 0
C_HAS_PWR_CNTL	C_EXT_VBUS_VALID_INVERT	Affected Parameter is ignored when C_HAS_PWR_CNTL = 0
C_EXT_VBUS_VALID	C_EXT_VBUS_VALID_INVERT	Affected Parameter is ignored when C_EXT_VBUS_VALID = 0
C_SPLB_NUM_MASTERS	C_SPLB_MID_WIDTH	C_SPLB_MID_WIDTH must equal log2(C_SPLB_NUM_MASTERS)
C_SPLB_AWIDTH		Always set to 32
C_MPLB_AWIDTH		Always set to 32
C_SPLB_P2P	C_SPLB_BASEADDR, C_SPLB_HIGHADDR	Affected parameters are ignored when C_SPLB_P2P = 1
C_USE_PHY_BUS_PWR	C_EXT_VBUS_VALID	Affected parameter is ignored when C_USE_PHY_BUS_PWR = 1
C_USE_PHY_BUS_PWR	C_EXT_VBUS_VALID_INVERT	Affected parameter is ignored when C_USE_PHY_BUS_PWR = 1

Parameter - Port Dependencies

Table 4: **XPS USB Host Controller Parameter-Port Dependencies**

Generic ID	Generic Name	Affects Port	Depends on Parameter	Relationship Description
Design Parameters				
G1	C_SPLB_NUM_MASTERS	SI_MBusy, SI_MWrErr, SI_MRdErr, SI_MIRQ		Port widths are set directly from the parameter value
G2	C_SPLB_MID_WIDTH	PLB_masterID		Port widths are set directly from the parameter value
G3	C_SPLB_DWIDTH	PLB_BE, PLB_wrDBus, SI_rdDBus		Port widths are set directly or derived from the parameter value
G3	C_MPLB_DWIDTH	M_BE, M_wrDBus, PLB_MrdDBus		Port widths are set directly or derived from the parameter value

Parameter Descriptions

C_SUPPORT_USB_FS

Type: Integer

Allowed Values: 0,1 (default = 1)

Definition: 0 = HS only devices, 1 = Support HS and FS devices.

Description: This integer parameter is used to control inclusion or omission of Full Speed (FS) device support in the Host. When this parameter is assigned a value of 1, the Transaction Translator is included in the Link Manager and EHCI Split Transfer Support is added to the EHCI Controller. This increases the resource utilization of the core significantly.

C_HAS_PWR_CNTL

Type: Integer

Allowed Values: 0,1 (default = 1)

Definition: 0 = User has not provided for Vbus power control, 1 = Power Control is provided

Description: This integer parameter indicates to the Host that the User has provided the capability for the Host to control the USB Power Bus via a discrete switch or the PHY. The state of this parameter is echoed in the PPC bit of the **HCSPARAMS** register. 0 = Port Power Control not available, 1 = Port Power Control is provided. The parameter is defaulted to 1, which indicates the Host has the ability to control Vbus power.

C_USE_PHY_BUS_PWR

Type: Integer

Allowed Values: 0,1 (default = 0)

Definition: 0 = PHY is not providing Vbus power, 1 = PHY is providing Vbus power

Description: This integer parameter indicates to the Host that the User circuit design is providing USB power on Vbus from the charge pump of the PHY (not an external switch). The parameter is defaulted to 1, indicating that the USB Vbus power will be provided by the PHY device. An assigned value of 0 indicates the User's design is supplying Vbus power from a source other than the PHY.

C_EXT_VBUS_VALID

Type: Integer

Allowed Values: 0,1 (default = 1)

Definition: 0 = Use PHY internal Vbus valid monitor, 1 = Use PHY external input for Vbus Valid indication

Description: This integer parameter indicates to the Host if the User PHY will be using a Vbus Valid input from a monitoring source external of the PHY or if the PHY will be providing the monitoring function. The parameter is defaulted to 1, indicating an external monitoring source is being used (the PHY is using an external Vbus Valid indication).

C_EXT_VBUS_VALID_INVERT

Type: Integer

Allowed Values: 0,1 (default = 0)

Definition: 0 = Use Vbus Valid input non-inverted, 1 = Invert the Vbus Valid input to the PHY

Description: If the PHY is using an external Vbus monitor (**C_EXT_VBUS_VALID** = 1), then this parameter indicates to the Host that the PHY should be programmed to invert the sense of the external power valid input signal. The parameter is defaulted to 0, indicating no inversion. Note that this parameter only has meaning if the **C_EXT_VBUS_VALID** parameter is set to 1.

C_MPLB_AWIDTH

Type: Integer

Allowed Values: 32

Definition: Address bus width of attached PLB on the Master interface

Description: This integer parameter is used by the PLB Master to size the PLB address related components within the Master. The EDK tool suite will assign this parameter a fixed value of 32.

C_MPLB_DWIDTH

Type: Integer

Allowed Values: 32, 64, 128

Definition: Data bus width of attached PLB on the Master interface

Description: This integer parameter is used by the PLB Master to configure PLB data bus attachment logic within the Master. The parameter can be assigned a value of 32, 64 or 128-Bits. The EDK tools will ensure that the attached PLB data width is equal to or greater than the Native data with of any Master or Slave attached to the PLB.

C_MPLB_P2P

Type: Integer

Allowed Values: 0,1 (default = 0)

Definition: 0 = Shared Bus PLBV46 topology, 1 = Point to Point PLBV46 topology

Description: This integer parameter indicates the type of connection topology for the PLBV46 attached to the Master port.

C_SPLB_BASEADDR

Type: std_logic_vector(0 to 31)

Allowed Values: 0000_0000 to FFFF_FE00 (default = FFFF_FFFF)

Definition: 32-bit system address

Description: This 32-bit parameter is a std_logic_vector and is used to assign the start of the system memory mapped address range assigned for the EHCI register set within the USB Host. The address value assigned must start on an even multiple of the address space that the User wishes to assign to the Host's Slave PLB port. Minimum system address space required for the USB Host is 200_{hex} bytes.

C_SPLB_HIGHADDR

Type: std_logic_vector(0 to 31)

Allowed Values: 0000_01FF to FFFF_FFFF (default = 0000_0000)

Definition: 32-bit system address

Description: This 32-bit parameter is a std_logic_vector and is used to assign the end of the assigned system memory mapped address range for the EHCI register set within the USB Host. This parameter must be assigned an address value that reflects the assigned address space to the Registers. If the recommended 200_{hex} byte address space allocation is utilized, then the assigned value for this parameter should be C_SPLB_BASEADDR + $1FF_{hex}$.

C_SPLB_MID_WIDTH

Type: Integer

Allowed Values: 1 to 4

Definition: Sets the width of the PLB_masterID input port

Description: This parameter is set to indicate the bit width of the PLB_masterID input port. The assigned value must be equal to log2(C_SPLB_NUM_MASTERS);

C_SPLB_NUM_MASTERS

Type: Integer

Allowed Values: 1 to 16

Definition: 1 = 1 PLB Master, 2 = 2 PLB Masters, etc.

Description: This parameter is set to indicate the number of Masters connected to the Slave PLB bus. This parameter is used to size the Sl_MBusy and Sl_MErr slave reply buses to the PLB. For example, if eight PLB Masters are connected to the PLB Bus, then this parameter must be set to 8. The Sl_MBusy bus and Sl_MErr bus will be sized to 8 bits wide each.

C_SPLB_AWIDTH

Type: Integer

Allowed Values: 32

Definition: Address bus width of attached PLB on the Slave interface

Description: This integer parameter is used by the PLB Slave to size the PLB address related components within the Slave Attachment. The EDK tool suite will assign this parameter a fixed value of 32.

C_SPLB_DWIDTH

Type: Integer

Allowed Values: 32, 64, 128

Definition: Data bus width of attached PLB on the Slave interface

Description: This integer parameter is used by the PLB Slave to size PLB data bus related components within the Slave Attachment. This value should be set to match the actual width of the PLB bus, 32, 64 or 128-Bits.

C_SPLB_P2P

Type: Integer

Allowed Values: 0,1 (default = 0)

Definition: 0 = Shared Bus PLBV46 topology, 1 = Point to Point PLBV46 topology

Description: This integer parameter indicates the type of connection topology for the PLBV46 attached to the Slave port.

C_SPLB_CLK_PERIOD_PS

Type: Integer

Allowed Values: 16,666 (60MHz) to 5000 (200MHz) (default = 10,000 (100MHz))

Definition: Clock period in picoseconds of the input clock on the PLBV46 Slave interface

Description: This integer parameter specifies the period of the input SPLB_Clk. It is used by internal logic to set timer count values that are clocked off of the SPLB_Clk clock signal. The actual F_{MAX} frequencies obtainable are system dependent and indicated in Table 24.

C_FAMILY

Type: String

Allowed Values: "spartan3", "virtex4", "virtex5"

Definition: Family of target FPGA device

Description: This parameter is defined as a string. It specifies the target FPGA technology for implementation of the PLB Slave. This parameter is required for proper selection of FPGA primitives. The configuration of these primitives can vary from one FPGA technology family to another. FPGA families supported are Spartan-3, Virtex-4, Virtex-5 FPGAs.

XPS USB Host Registers

All User accessible register functions (via the PLBV46 Slave interface) for the USB Host are encapsulated in the Register Module. The module is partitioned into 2 sub-blocks, the EHCI Capability Register, and the EHCI Operational registers. The Register Module Block Diagram is shown in Figure 4.

The Register Module interfaces to the PLBV46 Slave Single module on the System Interface and with the internal USB Host logic. The internal logic side of the Register Module has varying set/reset/access requirements based on the individual register and the bit/field within each register. The XPS USB Host Controller complies with the Intel Enhanced Host Controller Interface (EHCI) version 1.0 with those exceptions listed in section "Specification Exceptions". EHCI calls out two major sets of software accessible hardware registers. Memory mapped Host Controller Registers and optional PCI configuration registers. The USB Host soft core does not have a PCI interface so the PCI configuration registers are omitted from the design.

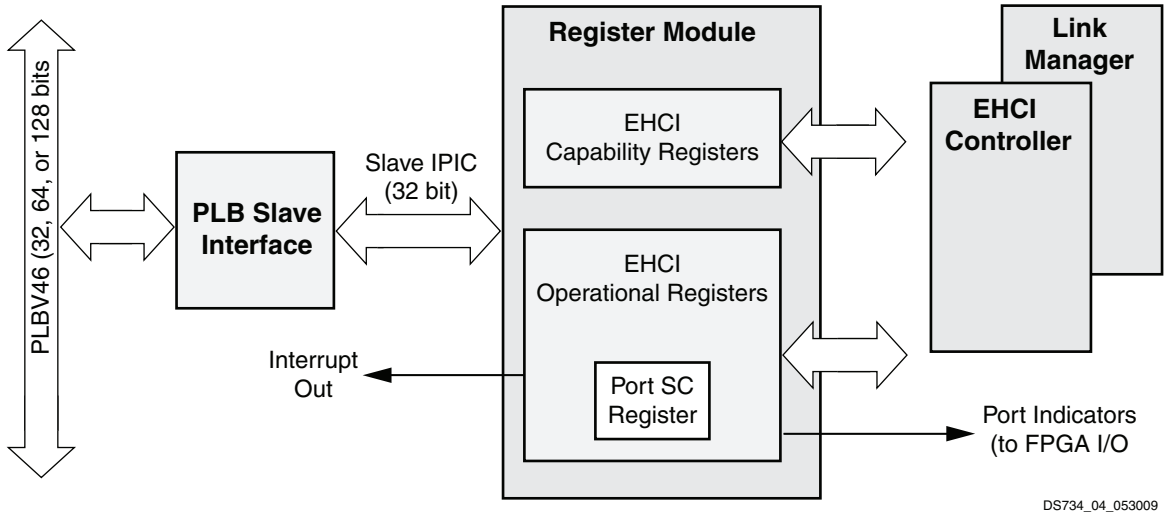

Figure 4: **Register Module Block Diagram**

Register Space Partitioning

The general USB Host soft core register space partitioning is shown in Table 5. The USB Host Controller Registers are memory-mapped into non-cacheable memory space. This memory space must be aligned on a PLB word (32-bit) address boundary.

Table 5: **General XPS USB Host Register Partition Address Mapping**

Address Space Offset (1)	Register Group	Description
000_H - $0FC_H$	Xilinx Reserved (2)	Xilinx Reserved
000_H - $0FC_H$	Xilinx Reserved	Xilinx Reserved
100_H - 124_H	EHCI Capability Registers	EHCI Capability Registers. The Capability registers specify the limits, restrictions, and capabilities of a host controller implementation. These values are used as parameters to the host controller driver.
140_H - $1FC_H$	EHCI Operational Registers	EHCI Operational Registers. The operational registers are used by system software to control and monitor the operational state of the host controller.

1. Address Space Offset is relative to C_SPLB_BASEADDR assignment.
2. Reserved memory space partition for Xilinx internal testing use only.

EHCI Register Definitions

The reserved bits defined in EHCI version 1.0 of the specification may be allocated in later revisions. Software should not assume reserved bits are always zero and should preserve these bits when writing to modifiable registers. Per the EHCI specification, the following notation is used to describe register access attributes:

- **RO (Read Only)**. If a register is read only, writes have no effect.
- **WO (Write Only)**. If a register is write only, reads return a zero for all bit positions.
- **R/W (Read/Write)**. A register with this attribute can be read and written. Note that individual bits in some read/write registers may be read only.

- **R/WC (Read/Write Clear)**. A register bit with this attribute can be read and written. However, a write of a 1 clears (sets to 0) the corresponding bit and a write of a 0 has no effect.

All Registers in the XPS USB Host are powered by the FPGA power which is considered the core power well from the EHCI perspective. There are no auxiliary or PCI power considerations for this core.

EHCI Register Endianess Considerations

The EHCI specification is targeted for a little endian processor environment. However the Xilinx MicroBlaze and PowerPC processor embedded systems are natively big endian due to their basis in the IBM CoreConnect Architecture. The EHCI register interfaces for the Xilinx USB Host are presented in this document in the big endian format. The relative bit significance of the registers are the same as EHCI, Most significant is left-most, Least Significant is right-most. If the registers are accessed as 32-bit values by the system software, then the endiness is transparent. The only exception to this is the CAPLENGTH_HCIVERSION register which is required by the LINUX programming model to allow access to the CAPLENGTH and HCI Version fields as separate entities. This register has been formatted to support that requirement. The XPS USB Host Controller does not provide any endianess conversion features.

Host Controller Capability Registers

These registers specify the limits, restrictions and capabilities of the Xilinx USB Host Controller implementation. Table 6 provides address offset and name information.

Table 6: **Enhanced Host Controller Capability Registers Summary**

Address Offset (From Capabilities Base Address)	Size (bytes)	Mnemonic	Register Name
00h	1	CAPLENGTH	Capability Register Length
01h	1	Reserved	N/A
02h	2	HCIVERSION	Interface Version Number
04h	4	HCSPARAMS	Structural Parameters
08h	4	HCCPARAMS	Capability Parameters
0Ch	8	HCSP-PORTROUTE	Companion Port Route Description (1)

1. This register is required only if companion host controllers are implemented. This host controller does not have companion controllers, therefore this register is not provided as part of this design.

CAPLENGTH_HCIVERSION (Capability Registers Length/Host Controller Version)

This register has two fields as shown in Figure 5. The first field (CAPLENGTH) is a 1 byte value that is used as an offset to add to the register address base to find the beginning of the Operational Register Space. The second field (HCIVERSION) is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this field represents a major revision and the least significant byte is the minor revision. Table 7 provides information about this register.

Big-endian/little-endian compatibility note:

The CAPLENGTH value must be accessed by software as a 1-byte entity and the HCIVERSION must be accessed as a 2-byte (PLB Half-word) entity. The byte offset into the register for these values has been adjusted for this type of access.

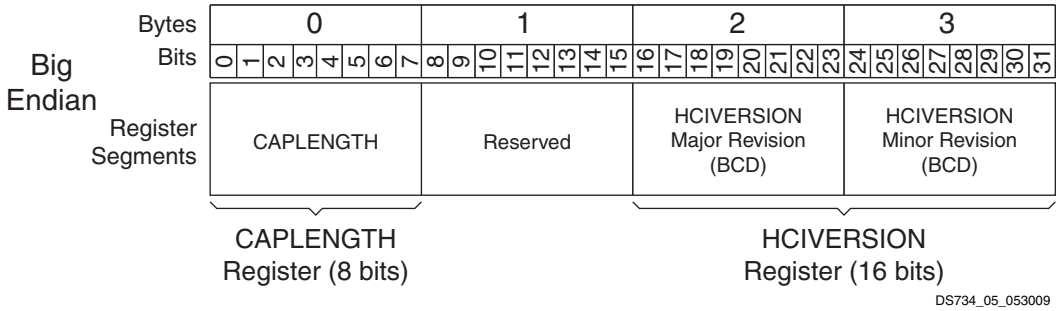

Figure 5: **CAPLENGTH_HCIVERSION Register Layout**

Table 7: **CAPLENGTH_HCIVERSION Details**

Bits	Field Name	Default Value	Access Type	Description
0-7	CAPLENGTH	40_{hex}	RO	The length (in bytes) of the capability register space. This value is added (as an address offset to the Capabilities register base address) to get to the address space for the Operational Registers.
8-15	Reserved	Zeros	RO	These bits are reserved and set to zero.
16-23	HCIVERSION (major)	01_{hex}	RO	This field is a BCD encoding of the major version number of the supported EHCI specification revision.
24-31	HCIVERSION (minor)	00_{hex}	RO	This field is a BCD encoding of the minor version number of the supported EHCI specification revision.

HCSPARAMS (Host Controller Structural Parameters)

This register contains the parameters that provide structural composition of the host controller to the user application or RTOS. The register format is shown in Figure 6 and described in Table 8.

Big-endian/little-endian compatibility note:

The HCSPARAMS value must be accessed as a 4-byte (PLB word) entity.

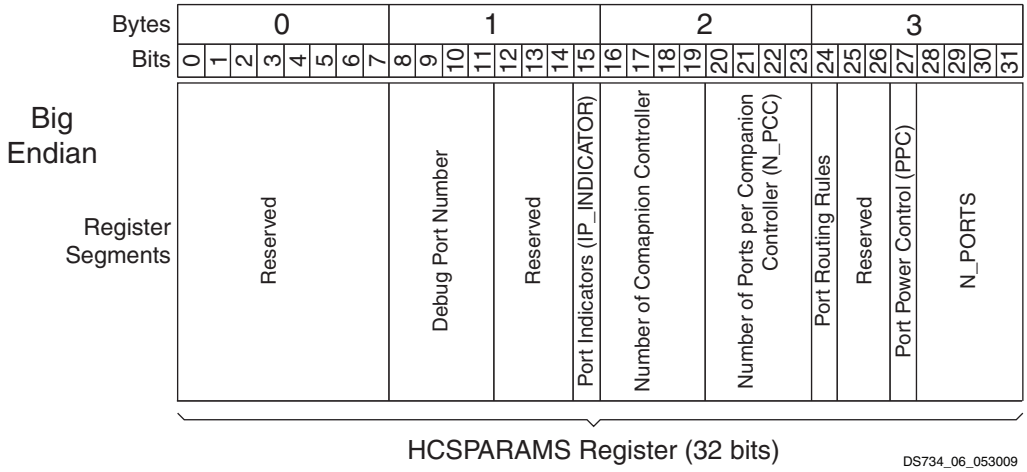

Figure 6: **HCSPARAMS Register Layout**

Table 8: **HCSPARAMS Details**

Bits	Field Name	Default Value	Access Type	Description
0-7	Reserved	Zeros	RO	These bits are reserved and set to zero.
8-11	Debug Port Number	Zeros	RO	Debug port is not supported so this field is set to a constant zero.
12-14	Reserved	Zeros	RO	These bits are reserved and set to zero.
15	Port Indicators	1	RO	Port Indicator support is parameterizable so this field reflects the value of the **C_HAS_PORT_INDICATORS** parameter.
16-19	Number of Companion Controllers	Zeros	RO	There are no companion controllers, so this field is set to constant 0.
20-23	Number of Ports per Companion Controller	Zeros	RO	There are no companion controllers, so this field is set to constant 0.
24	Port Routing Rules	0	RO	Because no companion controllers are used, this value is set to a constant 0.
25-26	Reserved	Zeros	RO	These bits are reserved and set to zero.
27	Port Power Control	1	RO	Port Power Control is parameterizable so this field reflects the value of the **C_HAS_PWR_CNTL** parameter.
28-31	Number of Ports	1	RO	This bit set to a constant 1. This Host Controller implements only 1 port.

HCCPARAMS (Capability Parameters)

This register provides the Host Controller capabilities parameters. The register layout is shown in Figure 7 and the register fields are described in Table 9.

Big-endian/little-endian compatibility note:

The HCSPARAMS value must be accessed by software as a 4-byte (PLB word) entity.

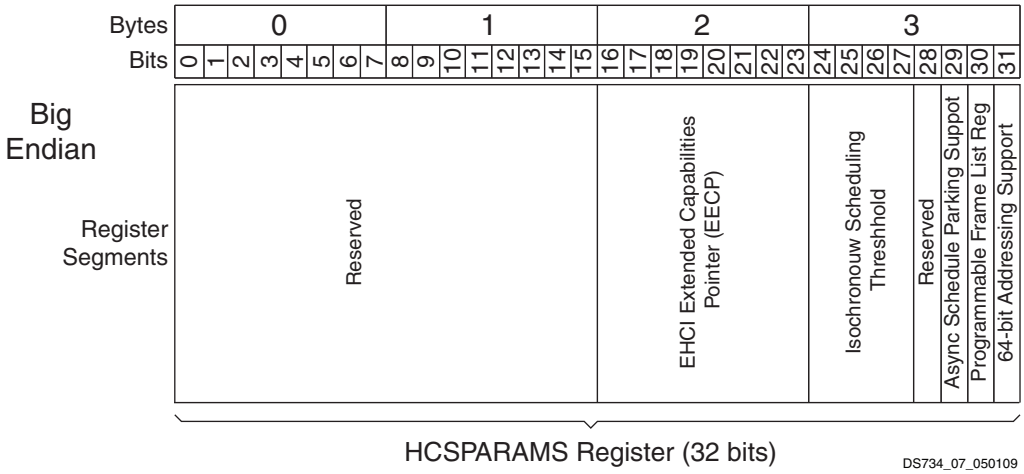

Figure 7: **HCCPARAMS Register**

Table 9: **HCCPARAMS Details**

Bits	Field Name	Default Value	Access Type	Description
0-15	Reserved	Zeros	RO	These bits are reserved and set to zero.
16-23	EHCI Extended Capabilities Pointer	Zeros	RO	This Host does not have Extended Capabilities so this field is set to constant zeros.
24-27	Isochronous Scheduling Threshold	"1000"	RO	Set to constant "1000". The EHCI Controller will cache a full isochronous data structure for an entire Frame.
28	Reserved	0	RO	This bit is reserved and set to 0.
29	Asynchronous Parking Support	1	RO	Set to constant 1. This Host supports Asynchronous Schedule Parking.
30	Programmable Frame List Support	1	RO	This Host Controller supports programmable frame list length.
31	64-bit Addressing Support	0	RO	This Host does not support 64-bit addressing.

HCSP-PORTROUTE (Companion Port Route)

This is an optional EHCI register set. It is not provided by this Host Controller because companion host controllers are not implemented in the design.

Host Controller Operational Registers

This section defines the enhanced host controller operational registers. These registers are located after the capabilities registers. The operational register base are PLB Word (32-bit) aligned and is calculated by adding the value in the first capabilities register (CAPLENGTH) to the base address of the enhanced host controller register address space. **All registers are 32 bits in length. Software must read and write these registers using only PLB Word (32-bit) accesses to maintain the proper big-endian to little-endian overlay.**

Table 10: **Host Controller Operational Registers**

Operational Register Space Offset	Size (bytes)	Mnemonic	Register Name
00h	4	USBCMD	USB Command
04h	4	USBSTS	USB Status
08h	4	USBINTR	USB Interrupt Enable
0Ch	4	FRINDEX	USB Frame Index
10h	4	CTRLDSSEGMENT	4G Segment Selector
14h	4	PERIODICLISTBASE	Frame List Base Address
18h	4	ASYNCLISTADDR	Next Asynchronous List Address
1Ch-3Fh	36	Reserved	
40h	4	CONFIGFLAG	Configured Flag Register
44h	4	PORTSC	Port Status/Control Register
48h-A4h		Reserved	

Table 10: **Host Controller Operational Registers**

Operational Register Space Offset	Size (bytes)	Mnemonic	Register Name
A8h	4	Mode	Mode Register (1)
ACh-FFh		Reserved	

1. The Mode Register is required by the Linux programming model. It has no operational effect on the Host Controller function.

USBCMD (USB Command Register)

The USB Command Register provides the user application with operational control and status. The register layout is shown in Figure 8 and described in Table 11.

Big-endian/little-endian compatibility note:

The USBCMD register must be accessed by software as a 4-byte (PLB word) entity.

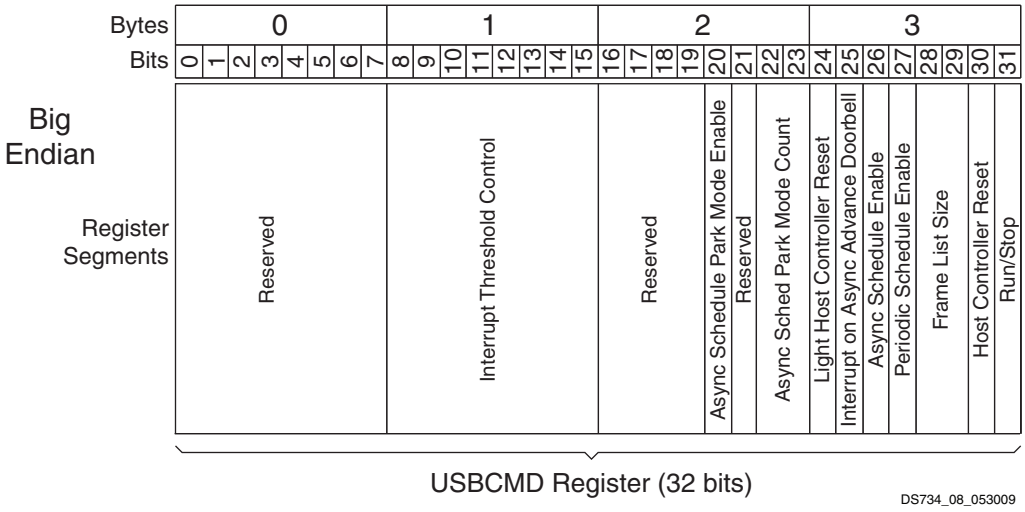

Figure 8: **USBCMD Register Layout**

Table 11: **USBCMD Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-7	Reserved	Zeros	RO	These bits are reserved and set to zero.
8-15	Interrupt Threshold Control	08h	R/W	Default 08h. This field is used by system software to select the maximum rate at which the host controller will issue interrupts. The only valid values are defined below. If software writes an invalid value to this register, the results are undefined. Value = Maximum Interrupt Interval 00h = Reserved 01h = 1 micro-frame 02h = 2 micro-frames 04h = 4 micro-frames 08h = 8 micro-frames (default, equates to 1 ms) 10h = 16 micro-frames (2 ms) 20h = 32 micro-frames (4 ms) 40h = 64 micro-frames (8 ms) Refer to EHCI Specification Section 4.15 for interrupts affected by this register. Any other value in this register yields undefined results. Software modifications to this bit while HCHalted bit is equal to zero results in undefined behavior.
16-19	Reserved	Zeros	RO	These bits are reserved and set to zero.
20	Async Schedule Park Mode Enable	1	R/W	Software uses this bit to enable or disable Async Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled.
21	Reserved	0	RO	This bit is reserved and set to zero.
22-23	Async Schedule Park Mode Count	3	R/W	This field contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. See EHCI Spec Section 4.10.3.2 for full operational details. Valid values are 1h to 3h. Software must not write a zero to this bit when *Park Mode Enable* is a one as this will result in undefined behavior.
24	Light Host Controller Reset	0	RO	The Light Host Controller Reset function is not supported. This bit is always 0.

Table 11: **USBCMD Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
25	Interrupt on Async Advance Doorbell	0	R/W	This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to *ring* the doorbell. When the host controller has evicted all appropriate cached schedule state, it sets the *Interrupt on Async Advance* status bit in the USBSTS register. If the *Interrupt on Async Advance Enable* bit in the USBINTR register is a one then the host controller will assert an interrupt at the next interrupt threshold. See Section 4.8.2 for operational details. The host controller sets this bit to a zero after it has set the *Interrupt on Async Advance* status bit in the USBSTS register to a one. Software should not write a one to this bit when the asynchronous schedule is disabled. Doing so will yield undefined results.
26	Asynchronous Schedule Enable	0	R/W	This bit controls whether the host controller skips processing the Asynchronous Schedule. Values mean: 0b = Do not process the Asynchronous Schedule 1b = Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
27	Periodic Schedule Enable	0	R/W	This bit controls whether the host controller skips processing the Periodic Schedule. Values mean: 0b = Do not process the Periodic Schedule 1b = Use the PERIODICLISTBASE register to access the Periodic Schedule.
28-29	Frame List Size	00b	R/W	This field specifies the size of the frame list. The size the frame list controls which bits in the Frame Index Register should be used for the Frame List Current index. Values mean: 00b = 1024 elements (4096 bytes) Default value 01b = 512 elements (2048 bytes) 10b = 256 elements (1024 bytes) – for resource-constrained environments 11b = **Reserved**

Table 11: **USBCMD Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
30	Host Controller Reset	0b	R/W	This control bit is used by software to reset the host controller. The effects of this on Root Hub registers are similar to a Chip Hardware Reset. When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial values. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. PCI Configuration registers are not affected by this reset. All operational registers, including port registers and port state machines are set to their initial values. Software must re initialize the host controller as described in EHCI Spec Section 4.1 in order to return the host controller to an operational state. This bit is set to zero by the Host Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Software should not set this bit to a one when the *HCHalted* bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior.
31	Run/Stop	0b	R/W	1=Run. 0=Stop. When set to a 1, the Host Controller proceeds with execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1. When this bit is set to 0, the Host Controller completes the current and any actively pipelined transactions on the USB and then halts. The Host Controller will halt within 16 micro-frames after software clears the Run bit. The HC Halted bit in the status register indicates when the Host Controller has finished its pending pipelined transactions and has entered the stopped state. Software must not write a one to this field unless the host controller is in the Halted state (i.e. *HCHalted* in the USBSTS register is a one). Doing so will yield undefined results.

USBSTS (USB Status Register)

This register indicates pending interrupts and various states of the Host Controller. The status resulting from a transaction on the serial bus is not indicated in this register. Software sets a bit to 0 in this register by writing a 1 to it. See EHCI Specification Section 4.15 for additional information concerning USB interrupt conditions. The register layout is shown in Figure 9 and described in Table 12.

Big-endian/little-endian compatibility note:

The USBSTS register must be accessed by software as a 4-byte (PLB word) entity.

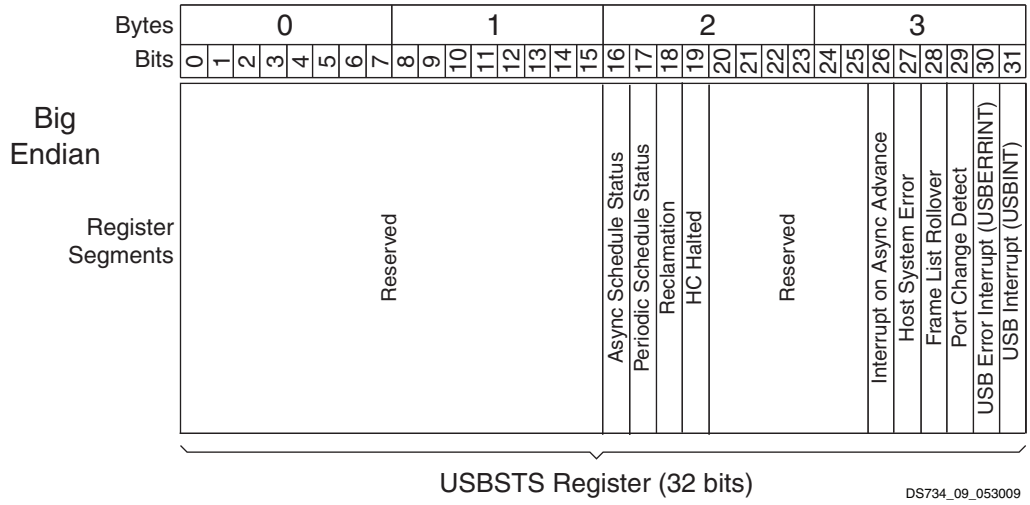

Figure 9: **USBSTS Register Layout**

Table 12: **USBSTS Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-15	Reserved	Zeros	RO	These bits are reserved and set to zero.
16	Async Schedule Status	0b	RO	The bit reports the current real status of the Asynchronous Schedule. If this bit is a zero then the status of the Asynchronous Schedule is disabled. If this bit is a one then the status of the Asynchronous Schedule is enabled. The Host Controller is not required to *immediately* disable or enable the Asynchronous Schedule when software transitions the *Asynchronous Schedule Enable* bit in the USBCMD register. When this bit and the *Asynchronous Schedule Enable* bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0).
17	Periodic Schedule Status	0b	RO	The bit reports the current real status of the Periodic Schedule. If this bit is a zero then the status of the Periodic Schedule is disabled. If this bit is a one then the status of the Periodic Schedule is enabled. The Host Controller is not required to *immediately* disable or enable the Periodic Schedule when software transitions the *Periodic Schedule Enable* bit in the USBCMD register. When this bit and the *Periodic Schedule Enable* bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0).
18	Reclamation	0b	RO	This is a read-only status bit, which is used to detect an empty asynchronous schedule. The operational model of empty schedule detection is described in EHCI Spec Section 4.8.3. The valid transitions for this bit are described in EHCI Spec Section 4.8.6.
19	HCHALTED	1b	RO	This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing as a result of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error).
20-25	Reserved	0	RO	These bits are reserved and set to zero.

Table 12: **USBSTS Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
26	Interrupt on Async Advance	0b	R/WC	System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the *Interrupt on Async Advance Doorbell* bit in the USBCMD register. This status bit indicates the assertion of that interrupt source.
27	Host System Error	0b	R/WC	The Host Controller sets this bit to 1 when a serious error occurs during a host system access involving the Host Controller module. When this error occurs, the Host Controller clears the Run/Stop bit in the Command register to prevent further execution of the scheduled TDs.
28	Frame List Rollover	0b	R/WC	The Host Controller sets this bit to a one when the *Frame List Index* (see EHCI Spec Section 2.3.4) rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the *Frame List Size* field of the USBCMD register) is 1024, the *Frame Index Register* rolls over every time FRINDEX[13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FRINDEX[12] toggles.
29	Port Change Detect	0b	R/WC	The Host Controller sets this bit to a one when any port for which the *Port Owner* bit is set to zero (see Section 2.3.9) has a change bit transition from a zero to a one or a *Force Port Resume* bit transition from a zero to a one as a result of a J-K transition detected on a suspended port. This bit will also be set as a result of the *Connect Status Change* being set to a one after system software has relinquished ownership of a connected port by writing a one to a port's *Port Owner* bit (see EHCI Spec Section 4.2.2). This bit is allowed to be maintained in the Auxiliary power well. Alternatively, it is also acceptable that on a D3 to D0 transition of the EHCI HC device, this bit is loaded with the OR of all of the PORTSC change bits (including: Force port resume, over-current change, enable/disable change and connect status change).
30	USB Error Interrupt	0b	R/WC	The Host Controller sets this bit to 1 when completion of a USB transaction results in an error condition (e.g., error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both this bit and USBINT bit are set. See EHCI Spec Section 4.15.1 for a list of the USB errors that will result in this bit being set to a one.
31	USB Interrupt	0b	R/WC	The Host Controller sets this bit to 1 on the completion of a USB transaction, which results in the retirement of a Transfer Descriptor that had its IOC bit set. The Host Controller also sets this bit to 1 when a short packet is detected (actual number of bytes received was less than the expected number of bytes).

USBINTR (USB Interrupt Enable Register)

This register enables and disables reporting of the corresponding interrupt to the user application or RTOS. When a bit is set and the corresponding interrupt is active, an interrupt is generated to the host. Interrupt sources that are disabled in this register still appear in the USBSTS to allow the software to poll for events. This register is shown in Figure 10 and described in Table 13.

Each interrupt enable bit description indicates if it is dependent on the interrupt threshold mechanism (see EHCI Spec Section 4.15).

Big-endian/little-endian compatibility note:

The USBINTR register must be accessed by software as a 4-byte (PLB word) entity.

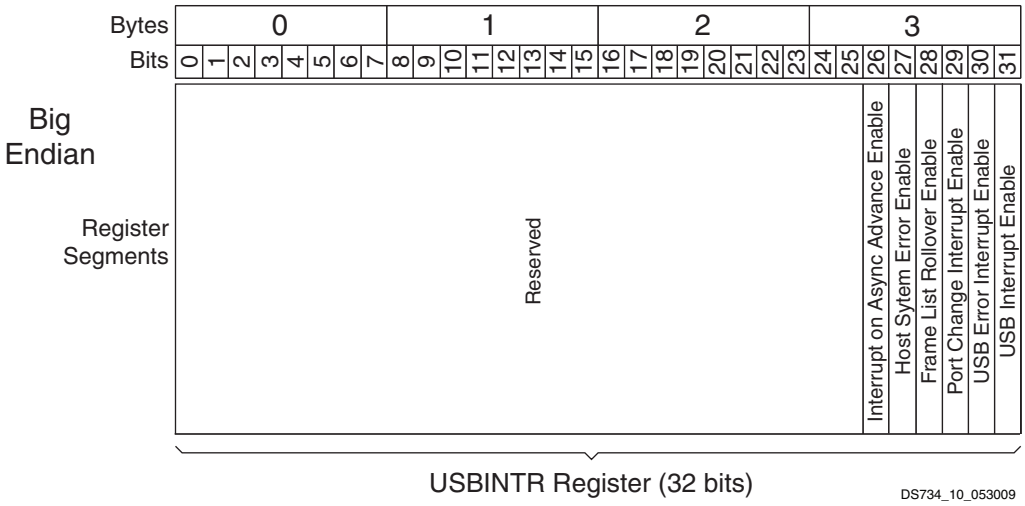

Figure 10: **USBINTR Register Layout**

Table 13: **USBINTR Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-25	Reserved	Zeros	RO	These bits are reserved and set to zero.
26	Interrupt on Async Advance Enable	0b	R/W	When this bit is a one, and the *Interrupt on Async Advance* bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *Interrupt on Async Advance* bit.
27	Host System Error Enable	0b	R/W	When this bit is a one, and the *Host System Error Status* bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the *Host System Error* bit.
28	Frame List Rollover Enable	0b	R/W	When this bit is a one, and the *Frame List Rollover* bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the *Frame List Rollover* bit.
29	Port Change Interrupt Enable	0b	R/W	When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the *Port Change Detect* bit.

Table 13: **USBINTR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
30	USB Error Interrupt Enable	0b	R/W	When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *USBERRINT* bit.
31	USB Interrupt Enable	0b	R/W	When this bit is a one, and the USBINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *USBINT* bit.

FRINDEX (Frame Index Register)

This register is used by the host controller to index into the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N:28] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the *Frame List Size* field in the USBCMD register as shown in Table 11, "USBCMD Register Details," on page 22. The layout of this register is shown in Figure 11 and described in Table 14.

This register must be written as 4 bytes (PLB Word). Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the Halted state as indicated by the *HCHalted* bit (Table 12, "USBSTS Register Details," on page 25). A write to this register while the Run/Stop bit is set to a one (see Table 11, "USBCMD Register Details," on page 22) produces undefined results. Writes to this register also affect the SOF value. See Section 4.5 of the EHCI Specification for details.

Big-endian/little-endian compatibility note:

The FRINDEX register must be accessed by software as a 4-byte (PLB word) entity.

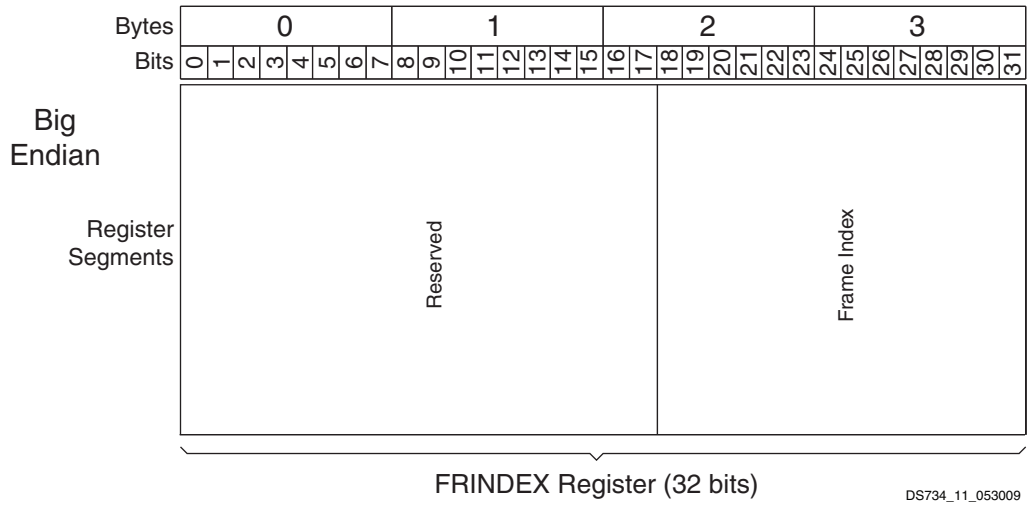

Figure 11: **FRINDEX Register Layout**

Table 14: **FRINDEX Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-17	Reserved	Zeros	RO	These bits are reserved and set to zero.
18-31	Frame Index	Zeros	R/W	The value in this register increments at the end of each time frame (e.g. micro-frame). Bits [N:28] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of *N* based on the value of the *Frame List Size* field in the USBCMD register.(1) (2) USBCMD[Frame List Size]/Number of Elements/N 00b/(1024)/18 01b/(512)/19 10b/(256)/20 11b/Reserved

1. The software frame number value for the bus SOF token is derived or alternatively managed from this register. See section 4.5 of the EHCI specification for an explanation of the SOF value management requirements of the host controller. The value of FRINDEX must be 125 µsec (1 micro-frame) ahead of the SOF token value. The SOF value may be implemented as an 11-bit shadow register. For this discussion, this shadow register is 11 bits and is named SOFV. SOFV updates every 8 micro-frames. (1 millisecond). An example implementation to achieve this behavior is to increment SOFV each time the FRINDEX[2:0] increments from a zero to a one.
2. Software must use the value of FRINDEX to derive the current micro-frame number, both for high-speed isochronous scheduling purposes and to provide the *get micro-frame number* function required for client drivers. Therefore, the value of FRINDEX and the value of SOFV must be kept consistent if chip is reset or software writes to FRINDEX. Writes to FRINDEX must also *write-through* FRINDEX[13:3] to SOFV[10:0]. In order to keep the update as simple as possible, software should never write a FRINDEX value where the three least significant bits are 111b or 000b. See section 4.5 of the EHCI Specification.

CTRLDSSEGMENT (Control Data Structure Segment Register)

This 32-bit register corresponds to the most significant address bits [63:32] for all EHCI data structures. Because the *64-bit Addressing Capability* field in HCCPARAMS register is a zero, this register is not used. Software cannot write to it and a read from this register will return zeros. The Register layout is shown in Figure 12 and described in Table 15.

Big-endian/little-endian compatibility note:

The CTRLDSSEGMENT register must be accessed by software as a 4-byte (PLB word) entity.

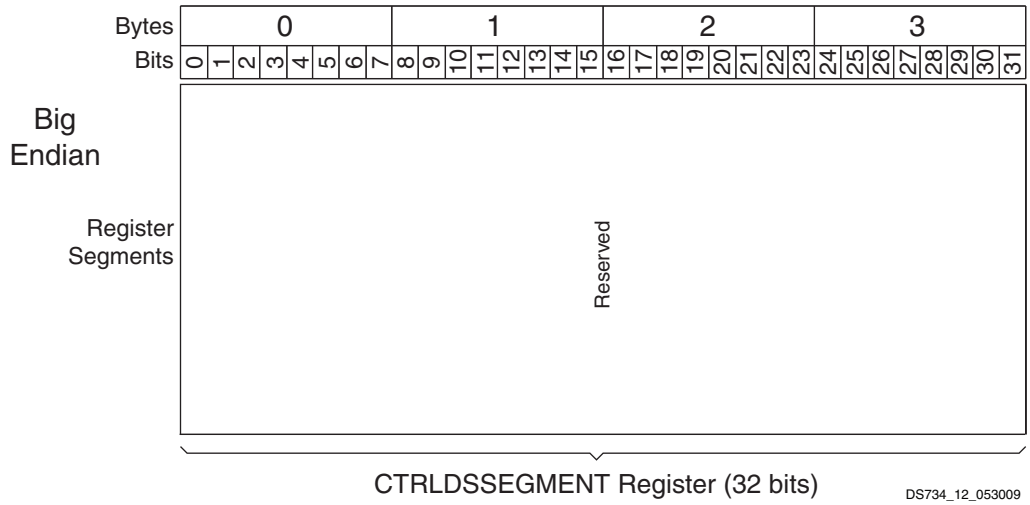

Figure 12: **CRTLDSSEGMENT Register Layout**

Table 15: **CTRLDSSEGMENT Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-31	Reserved	Zeros	RO	These bits are reserved and set to zero.

PERIODICLISTBASE (Periodic List Base Address Register)

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. System software loads this register prior to starting the schedule execution by the Host Controller (see EHCI Spec 4.1). The memory structure referenced by this physical memory pointer is assumed to be 4 kB (4000 bytes) aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. The register layout is shown in Figure 13 and described in Table 16.

Big-endian/little-endian compatibility note:

The PERIODICLISTBASE register must be accessed by software as a 4-byte (PLB word) entity.

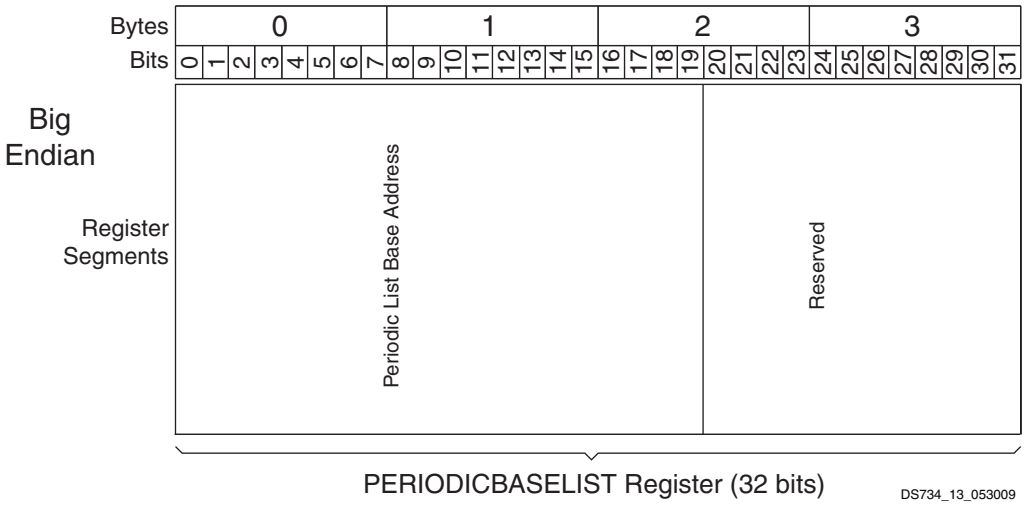

Figure 13: **PERIODICBASELIST Register Layout**

Table 16: **PERIODICBASELIST Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-19	Base Address	Zeros	R/W	These bits correspond to memory address signals [0:19],respectively of the starting address of the Periodic Schedule in System Memory.
20-31	Reserved	Zeros	R/W	These bits are reserved and set to zero. Software must write these bits as 0s.

ASYNCLISTADDR (Current Asynchronous List Address Register)

This 32-bit register contains the address of the next asynchronous queue head to be executed. Bits [4:0] of this register cannot be modified by system software and will always return a zero when read. The memory structure referenced by this physical memory pointer is assumed to be 32-byte (cache line) aligned. The register layout is shown in Figure 14 and described in Table 17.

Big-endian/little-endian compatibility note:

The ASYNCLISTADDR register must be accessed by software as a 4-byte (PLB word) entity.

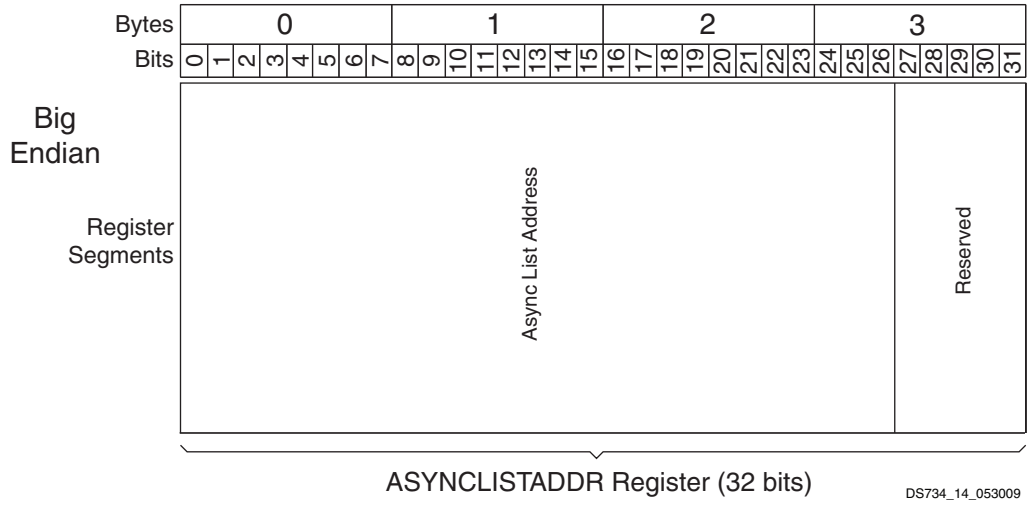

Figure 14: **ASYNCLISTADDR Register Layout**

Table 17: **ASYNCLISTADDR Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-26	Async Link Pointer	Zeros	R/W	These bits correspond to memory address signals [0:26],respectively. This field may only reference a Queue Head (QH) (see EHCI Spec Section 3.6).
27-31	Reserved	Zeros	RO	These bits are reserved and set to zero and their value has no effect on operation.

CONFIGFLAG (Configure Flag Register)

Because port routing is always via this Host Controller, the register will always be set to a constant value of 0x0001. The register layout is shown in Figure 15 and described in Table 18.

Big-endian/little-endian compatibility note:

The CONFIGFLAG register must be accessed by software as a 4-byte (PLB word) entity.

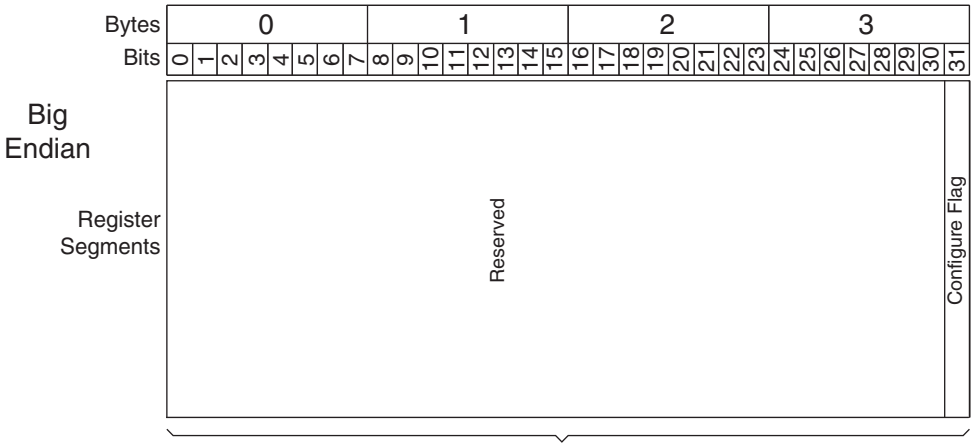

Figure 15: **CONFIGFLAG Register Layout**

Table 18: **CONFIGFLAG Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-30	Reserved	Zeros	RO	These bits are reserved and set to zero.
31	Configure Flag	1b	R/W	Host software would normally set this bit as the last action in its process of configuring the Host Controller (see EHCI Spec Section 4.1). However, this Host Controller implementation only allows the EHCI Controller to be routed to the single existing port. Therefor this bit is fixed at a constant '1' state and cannot be changed by writing to the register. 1b = Port routing control logic default-the USB port is always routed to this Host Controller. 0b = Not allowed

PORTSC (Port Status and Control Register)

This Host Controller implements one port register. The single port implementation is indicated in the HCSPARAMs register (See EHCI Spec Section 2.2.3). The initial conditions of the port are:

- No device connected,
- Port disabled

If the port has port power control, software cannot change the state of the port until after it applies power to the port by setting port power to a 1. Software must not attempt to change the state of the port until after power is stable on the port. The host is required to have power stable to the port within 20 milliseconds of the zero to one transition. The register layout is shown in Figure 16 and described in Table 19.

Note: When a device is attached, the port state transitions to the connected state and system software will process this as with any status change notification. Refer to EHCI Spec Section 4.3 for operational requirements for how change events interact with the port in suspend mode.

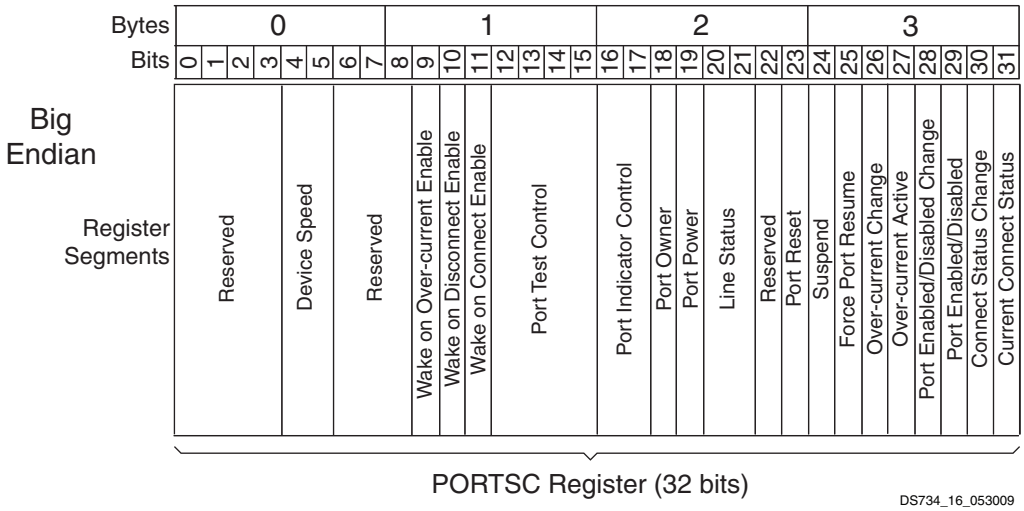

Figure 16: **PORTSC Register Layout**

Table 19: **PORTSC Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-3	Reserved	Zeros	RO	These bits are reserved and set to zero.
4-5	Device Speed	11b	RO	***Non-EHCI Compliant Field*** required by the Linux Programming model when a Transaction Translator is present in the Host controller (C_SUPPORT_USB_FS = 1) USB Device Speed 00 = Full Speed 01 = Low Speed 10 = High Speed 11 = No Device Attached This field is set by the Link Manager when it has determined the speed of the attached USB Device. If the Host Controller sets the PORTSC.PortEnable bit and the attached device speed reported in this field is not High Speed, then the Transaction Translator function in the Host Controller is enabled. This indicates to the SW Driver for the Host Controller that it must emulate a connection to a High Speed HUB (from the viewpoint of the RTOS). These bits are defaulted to "11" when Current Connect Status bit (bit 31) is set '0' indicating no device is connected.
6-8	Reserved	Zeros	RO	These bits are reserved and set to zero.

Table 19: **PORTSC Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
9	Wake on Over-current Enable	0b	R/W	The current implementation of this Host Controller ignores this bit and behaves as if the bit is set to a constant '1'. The Host will always wake on a detected overcurrent condition. This field is zero if *Port Power* is zero.
10	Wake on Disconnect Enable	0b	R/W	The current implementation of this Host Controller ignores this bit and behaves as if the bit is set to a constant '1'. The Host will always wake on a detected disconnect condition. This field is zero if *Port Power* is zero.
11	Wake on Connect Enable	0b	R/W	The current implementation of this Host Controller ignores this bit and behaves as if the bit is set to a constant '1'. The Host will always wake on a detected connect condition. This field is zero if *Port Power* is zero.
12-15	Port Test Control	Zeros	R/W	Port Test Control bits are currently unsupported by this Host Controller.
16-17	Port Indicator Control	00b	R/W	Writing to these bits has no effect if the *P_INDICATOR* bit in the HCSPARAMS register is a zero. If *P_INDICATOR* bit is a one, then the bit encodings are: Bit Value = Meaning 00b = Port indicators are off 01b = Amber 10b = Green 11b = Undefined Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if *Port Power* is zero.
18	Port Owner	0b	R/W	This bit is a R/W bit but is not used by HW. SW may set this bit to '1'. If this bit is set at a disconnect, HW will clear it when the disconnect is reported. See EHCI Spec Section 4.2 for operational details.

Table 19: **PORTSC Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
19	Port Power	0b	RO or R/W	The function of this bit depends on the value of the *Port Power Control* (*PPC*) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation PPC = 0b PP = 1b then Bit is RO. Host controller does not have port power control switches. Each port is hard-wired to power. PPC = 1b PP = 1b/0b Bit is R/W. Host controller has port power control switches. This bit represents the current setting of the switch (0 = off, 1 = on). When power is not available on a port (i.e. *PP* equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an over-current condition is detected on the powered port and *PPC* is a one, the PORTSC.*PP* bit will be transitioned by the Link Manager from a 1 to 0 (removing power from the port).
20-21	Line Status	00b	RO	These bits reflect the current logical levels of the D+ (bit 20) and D- (bit 21) signal lines. These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This field is valid only when the port enable bit is zero and the current connect status bit is set to a one. The encoding of the bits are: **00b = SE0** Not Low-speed device, perform EHCI reset **10b = J-state** Not Low-speed device, perform EHCI reset **01b = K-state** Low-speed device, release ownership of port **11b = Undefined** Not Low-speed device, perform EHCI reset. This value of this field is undefined if *Port Power* is zero.
22	Reserved	0b	RO	This bit is reserved for future use, and should return a value of zero when read.

Table 19: **PORTSC Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
23	Port Reset	0b	R/W	1 = Port is in Reset. 0 = Port is not in Reset. When software writes a one to this bit (from a zero), the bus reset sequence as defined in the USB Specification Revision 2.0 is started. Software writes a zero to this bit to terminate the bus reset sequence. Software must keep this bit at a one long enough to ensure the reset sequence, as specified in the USB Specification Revision 2.0, completes. Note: When software writes this bit to a one, it must also write a zero to the *Port Enable* bit. Note that when software writes a zero to this bit there may be a delay before the bit status changes to a zero. The bit status will not read as a zero until after the reset has completed. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (e.g. set the *Port Enable* bit to a one). **A host controller must terminate the reset and stabilize the state of the port within 2 milliseconds of the software transitioning this bit from a one to a zero.** For example: if the port detects that the attached device is high-speed during reset, then the host controller must have the port in the enabled state within 2ms of software writing this bit to a zero. The *HCHalted* bit in the USBSTS register should be a zero before software attempts to use this bit. The host controller may hold Port Reset asserted to a one when the *HCHalted* bit is a one. This field is zero if *Port Power* is zero.

Table 19: **PORTSC Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
24	Suspend	0b	R/W	1=Port in suspend state. 0=Port not in suspend state. Port Enabled Bit and Suspend bit of this register define the port states as follows: 0X = Disable 10 = Enable 11 = Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. A write of zero to this bit is ignored by the host controller. The host controller will unconditionally set this bit to a zero when: Software sets the ***Force Port Resume*** bit to a zero (from a one). Software sets the ***Port Reset*** bit to a one (from a zero). If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This field is zero if *Port Power* is zero.

Table 19: **PORTSC Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
25	Force Port Resume	0b	R/W	1 = Resume detected/driven on port. 0 = No resume (Kstate) detected/driven on port. This functionality defined for manipulating this bit depends on the value of the *Suspend* bit. For example, if the port is not suspended (*Suspend* and *Enabled* bits are a one) and software transitions this bit to a one, then the effects on the bus are undefined. Software sets this bit to a 1 to drive resume signaling. The Host Controller sets this bit to a 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to a one. If software sets this bit to a one, the host controller must not set the *Port Change Detect* bit. Note that when the EHCI controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. Software must appropriately time the Resume and set this bit to a zero when the appropriate amount of time has elapsed. Writing a zero (from one) causes the port to return to highs peed mode (forcing the bus below the port into a high-speed idle). This bit will remain a one until the port has switched to the high-speed idle. The host controller must complete this transition within 2 milliseconds of software setting this bit to a zero. This field is zero if *Port Power* is zero.
26	Over-current Change	0b	R/WC	1=This bit gets set to a one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position.
27	Over-current Active	0b	RO	1 = This port currently has an over-current condition. 0 = This port does not have an over-current condition. This bit will automatically transition from a one to a zero when the over current condition is removed.
28	Port Enable/Disable Change	0b	R/WC	1 = Port enabled/disabled status has changed. 0 = No change. For the root hub, this bit gets set to a one only when a port is disabled due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification for the definition of a Port Error). Software clears this bit by writing a 1 to it. This field is zero if *Port Power* is zero.

Table 19: **PORTSC Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
29	Port Enabled/Disabled	0b	R/W	1 = Enable. 0 = Disable. Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. The host controller will only set this bit to a one when the reset sequence determines that the attached device is a high-speed device. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. See EHCI Spec Section 4.2 for full details on port reset and enable. When the port is disabled (0b) downstream propagation of data is blocked on this port, except for reset. This field is zero if *Port Power* is zero.
30	Connect Status Change	0b	R/WC	1 = Change in Current Connect Status. 0 = No change. Indicates a change has occurred in the port's Current Connect Status. The host controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e., the bit will remain set). Software sets this bit to 0 by writing a 1 to it. This field is zero if *Port Power* is zero.
31	Current Connect Status	0b	RO	1=Device is present on port. 0=No device is present. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if *Port Power* is zero.

Mode (Mode Register)

This is a non-EHCI defined register. The Linux programming model requires a R/W register at offset 0XA8 into the Operational register space. This register is used when a Transaction Translator is configured into the Host Controller for Full Speed device support. The register has no operational effect on the Host Controller hardware. It is strictly for software use only. The register layout is shown in Figure 17 and detailed in Figure 20.

Big-endian/Little-endian compatibility note:

The Mode register must be accessed by software as a 4-byte (PLB word) entity.

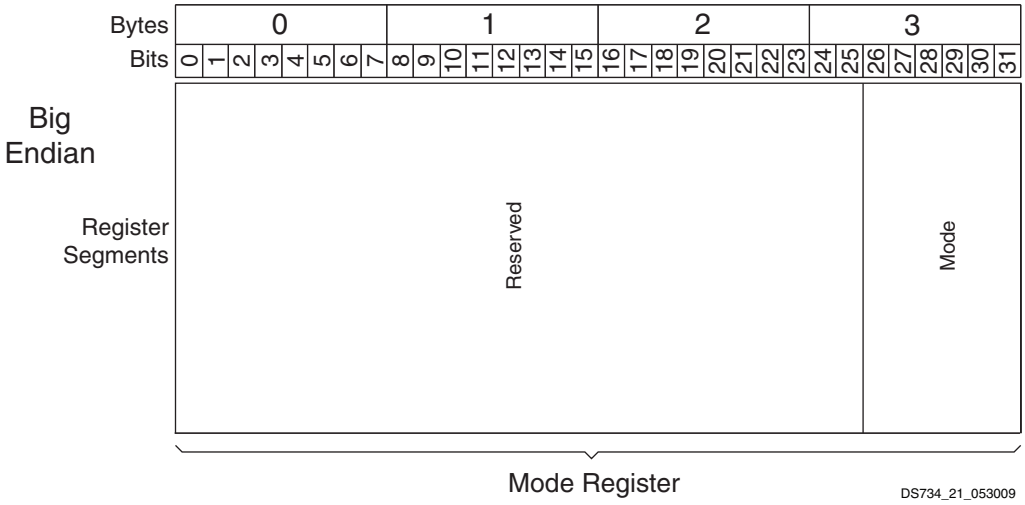

Figure 17: **Mode Register**

Table 20: **MODE Register Details**

Bits	Field Name	Default Value	Access Type	Description
0-25	Reserved	Zeros	RO	These bits are reserved and set to zero.
26-31	Mode	Zeros	R/W	These 6 bits are provided for the Linux programming model when a Transaction Translator is present in a Host Controller. These bits do not effect any functional operation within the XPS USB Host.

Application Information

Linux Big-Endian/Little Endian Considertions

To Linux, the XPS USB Host Controller has a big-endian register interface and uses big-endian memory descriptors. This requires that Linux be configured to handle this scenario.

There are two places in Linux that address endianess and must be set as follows:

1. Driver configuration, drivers/usb/host/Kconfig:
   select USB_EHCI_BIG_ENDIAN_DESC
   select USB_EHCI_BIG_ENDIAN_MMIO
2. driver implementation, drivers/usb/host/ehci-xilinx-of.c
   echi - >big_endian_mmio = 1;
   ehci ->big_endian_desc = 1;

PLBV46 Slave Interface

The USB Host design incorporates the PLBV46 Slave for the register access interface with PLBV46. The PLBV46 Slave interface has an internal Native Data Width of 32 bits but can attach to a PLBV46 Bus of 32, 64, and 128-bit data bus widths. The Slave interface only supports PLB Single Data Beat accesses of 1 to 4 bytes (32-bit Native Data Width). Bursting is not supported nor is it required for the USB Host register accesses via PLBV46.

PLBV46 Master Interface

Efficient data transfers with System Memory are controlled by the EHCI Controller via the Host's PLBV46 Master Interface. The Master design is integrated with and optimized for the USB Host application. The Master's internal interfaces are 32 bits, thus it has a fixed Native Data width of 32 bits. The Master is designed to support attachments to 32, 64, and 128-bit PLBV46 interface. Because the Master is a Native 32-bit design, it has no need to support Conversion Cycles and Burst Length Expansion. The Master Interface provides the required steering/mirroring functions for PLBV46.

USB Host Backend ULPI Interface

The USB PHY interface supports the ULPI V1.1 specification. There are no modifications to the usage except the bi-directional data bus cannot be implemented via internal FPGA fabric resources. Therefore, the data bus is implemented via the standard Xilinx methodology of an input bus, an output bus, and a direction control. These three elements are then combined by the implementation tools at the FPGA I/O boundary (external to the XPS USB Host design) to implement the actual bi-directional functionality required by the ULPI compliant PHY.

ULPI Clock Management

The Link Manager module of the XPS USB Host uses the input ULPI clock (60 MHz) from the USB 2.0 PHY as its main synchronization clock. A Global Clock Buffer (BUFG) is instantiated in the XPS USB Host top level to receive the raw ULPI Clock and buffer it onto a global clocking network within the target FPGA.

USB Host Clocking Architecture

The USB Host incorporates two clocking sources. The Embedded System side clocking is derived from the Slave PLB input clock (SPLB_Clk). **It is a system requirement of the XPS USB Host core that the PLBV46 Bus instance attached to the PLBV46 Master port must clocked with the same clock as the PLBV46 Slave port**. The input Master Port clock (MPLB_Clk) will be ignored internally by the Host.

The Link Manager primarily uses the ULPI clock from the external PHY chip (via a BUFG) as the clock. However, it also has internal interfaces that are synchronous to the PLB clock domain. Among these are the interfaces with the EHCI Controller and the EHCI Registers. Therefore, the Link Manager must accommodate clock domain crossing techniques (FIFOs and double registering). In addition, the support of PHY Low Power Mode requires the Link Manager to deal with the **loss of the ULPI clock** during the Suspend operation. The XPS USB Host clocking partition is graphically shown in Figure 18.

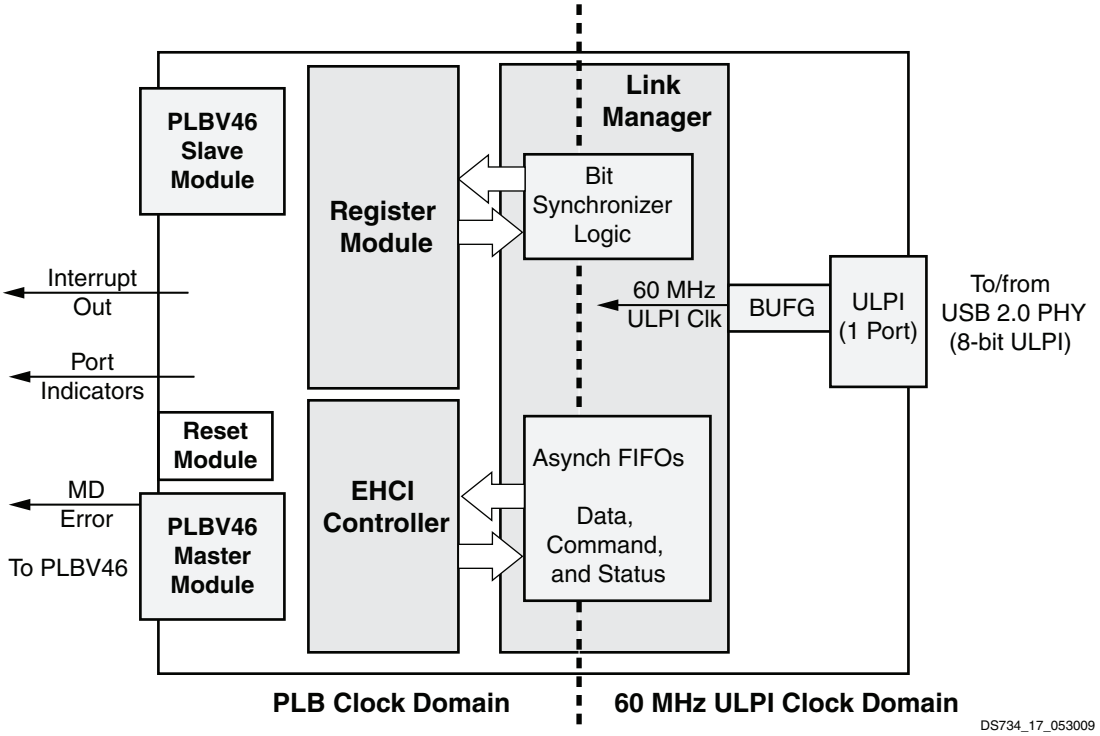

Figure 18: **XPS USB Host Clocking Partition**

Interface Timing

The USB Host Controller uses PLBV46 Slave and Master interfaces for the Embedded System I/O and the 8-bit ULPI Interface with an external (to FPGA) USB 2.0 PHY device. The PLBV46 interface timing is included in the document *IBM 128-bit Processor Local Bus, Architecture Specification, Version 4.6* with the Xilinx simplifications applied per Xilinx document *Xilinx SP026, PLBV46 Interface Simplifications, Version 1.0*. The ULPI interface timing of the USB Host Controller is described in the document *UTMI++ Low Pin Interface (ULPI) Specification, Revision 1.1*.

Design Implementation

Target Technology

The intended target technology for this design is Virtex-4, Virtex-5, Virtex-6, Spartan-3A, and Spartan-6 FPGAs.

Device Utilization and Performance Benchmarks

Core Performance

Because the XPS USB Host Controller core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When the XPS USB Host Controller core is combined with other designs in the system, the utilization of FPGA resources and timing of the XPS USB Host Controller design will vary from the results reported here.

The XPS USB Host Controller resource utilization for various parameter combinations measured with the Virtex-4 FPGA as the target device are detailed in Table 21.

Table 21: **Performance and Resource Utilization Benchmarks on Virtex-4 FPGA (xc4vfx60ff1152-10)**

Parameter Values (Other parameters at default values)			Device Resources				Performance
C_SUPPORT_USB_FS	C_MPLB_DWIDTH	C_SPLB_DWIDTH	Slices	Slice Registers	LUTs	Block RAMs	F_{MAX} (MHz)
0	64	64	4000	3394	4590	2	100
1	64	64	6600	4753	7652	5	100

The XPS USB Host Controller resource utilization for various parameter combinations measured with the Virtex-5 FPGA as the target device are detailed in Table 22..

Table 22: **Performance and Resource Utilization Benchmarks on Virtex-5 FPGA (xc5vfx70tff1136-1)**

Parameter Values (Other parameters at default values)			Device Resources				Performance
C_SUPPORT_USB_FS	C_MPLB_DWIDTH	C_SPLB_DWIDTH	Slices	Slice Registers	LUTs	Block RAMs	F_{MAX} (MHz)
0	128	128	2554	3387	3457	1	120
1	128	128	3767	4748	5714	3	120

The XPS USB Host Controller resource utilization for various parameter combinations measured with the Spartan-3A FPGA as the target device are detailed in Table 23.

Table 23: **Performance and Resource Utilization Benchmarks on the Spartan-3A FPGA (xc3sd3400afg676-4)**

Parameter Values (Other parameters at default values)			Device Resources				Performance
C_SUPPORT_USB_FS	C_MPLB_DWIDTH	C_SPLB_DWIDTH	Slices	Slice Registers	LUTs	Block RAMs	F_{MAX} (MHz)
0	32	32	4255	3392	4130	2	90
1	32	32	6268	4749	7320	5	90

System Performance

To measure the system performance (F_{MAX}) of the XPS USB Host Controller core, it was added as the Device Under Test (DUT) to a Virtex-4 FPGA system as shown in Figure 19, to a Virtex-5 FPGA system as shown in Figure 20, and to a Spartan-3A FPGA system as shown in Figure 21.

Because the XPS USB Host Controller core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other designs in the system, the utilization of FPGA resources and timing of the core design will vary from the results reported here.

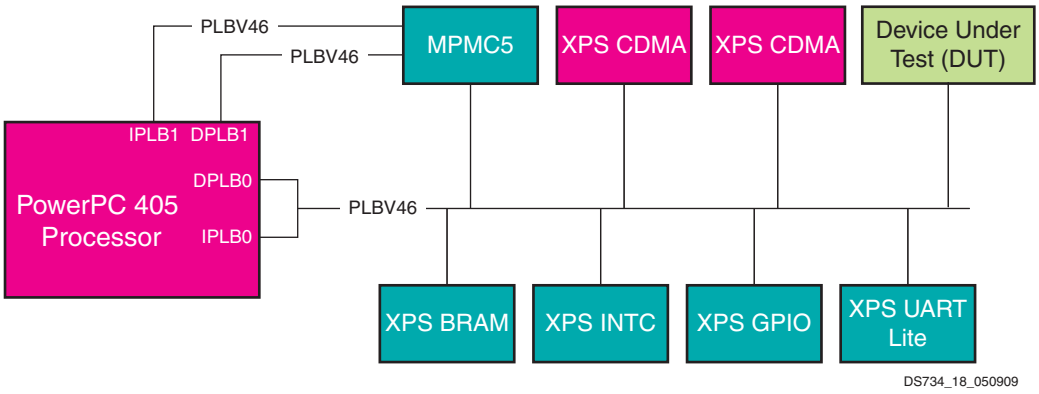

Figure 19: **Virtex-4 FX FPGA System with the XPS USB Host Controller Core as the DUT**

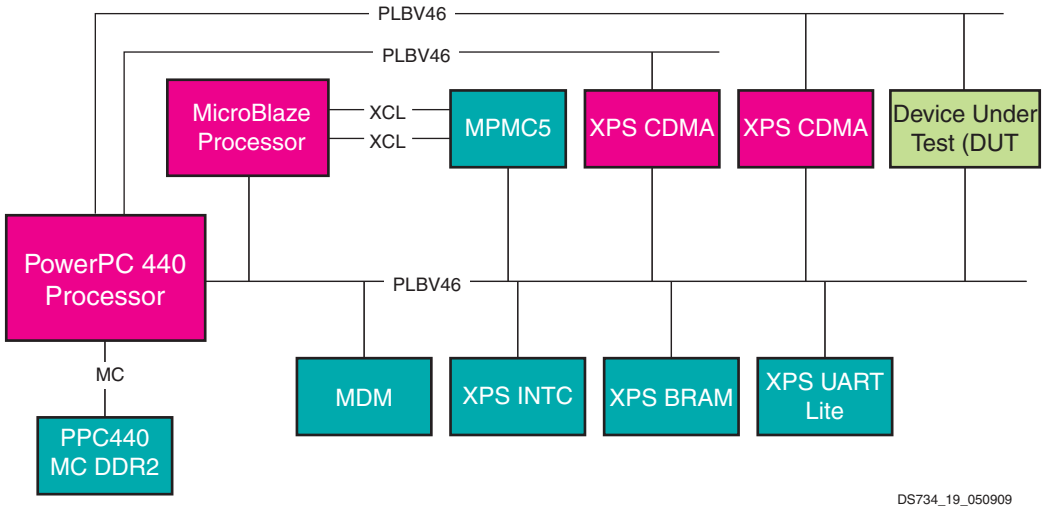

Figure 20: **Virtex-5 FX FPGA System with the XPS USB Host Controller Core as the DUT**

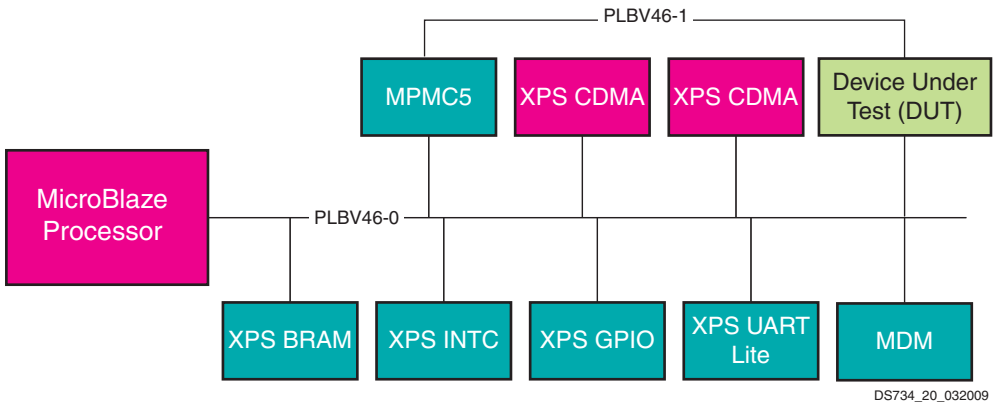

Figure 21: **Spartan-3A FPGA System with the XPS USB Host Controller Core as the DUT**

The target FPGA was then filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 24.

Table 24: **USB Host Controller System Performance**

Target FPGA	Target F_{MAX} (MHz)
xc3sd3400a-4	90
xc4vx60-10	100
xc5vfx70t-1	120

The target F_{MAX} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

XPS USB Host Known Issues or Limitations

- Low Speed devices are not supported
- Device Connect to Disconnect and Disconnect to Connect intervals of 4ms or less are not supported and can lead to port lock-up

Specification Exceptions

USB 2.0

- This host controller does not support low speed devices
- This host controller does not implement the USB 2.0 Test Modes

EHCI 1.0

- This Host Controller does not support PCI register space
- This Host Controller is always sensitive to Wake On Over-Current, Disconnect, and Connect Events regardless of the programming of PORTSC register bits (bits 9-11 Big Endian)
- Port Test Control bits in PORTSC register are not supported

PLBV46

The integrated PLBV46 Slave interface design does not support the following PLB V4.6 specification features for **Slave** devices:

- Aborts
- Address pipelining
- Parity
- Indeterminate Length Bursts
- Cacheline requests
- Fixed length Burst requests

The integrated PLBV46 Master design does not support the following PLB V4.6 specification features for **Master** devices:

- Aborts
- Address pipelining
- Parity
- Indeterminate Length Bursts
- Cacheline requests
- Fixed length Burst requests greater than 16 data beats
- Fixed Length Bursts of data width greater than 32-bits

Reference Documents

1. SP026 PLBV46 Interface Simplifications, Version 1.0
2. XAPP535 High Performance Multi-Port Memory Controller
3. DS643 Multi-Port Memory Controller (MPMC) (v5.02a)
4. UTMI++ Low Pin Interface (ULPI) Specification, Revision 1.1
5. Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 1.0
6. Universal Serial Bus Specification, Revision 2.0
7. IBM 128-bit Processor Local Bus, Architecture Specification, Version 4.6

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
6/24/09	1.0	Initial Xilinx release.
9/16/09	1.1	Added information relating to the addition of USB Full Speed Device support; made various clarifications and text edits; updated Figures 1, 4, 16, and created Figure 21; added the Speed Bits to the PORTSC Register definition; added the MODE Register definition.

Notice of Disclaimer