

USB 2.0 (USB20SR) Device IP Core

User Guide



System Level Solutions, Inc. (USA)
14100 Murphy Avenue
San Martin, CA 95046
(408) 852 - 0067

<http://www.slscorp.com>

IP Core Version:	1.3
Document Version:	1.3
Document Date:	January 2013

IP Usage Note

The Intellectual Property (IP) core is intended solely for our clients for physical integration into their own technical products after careful examination by experienced technical personnel for its suitability for the intended purpose.

The IP was not developed for or intended for use in any specific customer application. The firmware/software of the device may have to be adapted to the specific intended modalities of use or even replaced by other firmware/software in order to ensure flawless function in the respective areas of application.

Performance data may depend on the operating environment, the area of application, the configuration, and method of control, as well as on other conditions of use; these may deviate from the technical specifications, the Design Guide specifications, or other product documentation. The actual performance characteristics can be determined only by measurements subsequent to integration.

The reference designs were tested in a reference environment for compliance with the legal requirements applicable to the reference environment.

No representation is made regarding the compliance with legal, regulatory, or other requirements in other environments. No representation can be made and no warranty can be assumed regarding the suitability of the device for a specific purpose as defined by our customers.

SLS reserves the right to make changes to the hardware or firmware or software or to the specifications without prior notice or to replace the IP with a successor model to improve performance or design of the IP. Of course, any changes to the hardware or firmware or software of any IP for which we have entered into an agreement with our customers will be made only if, and only to the extent that, such changes can reasonably be expected to be acceptable to our customers.

Copyright©2013, System Level Solutions, Inc. (SLS) All rights reserved. SLS, an Embedded systems company, the stylized SLS logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of SLS in India and other countries. All other products or service names are the property of their respective holders. SLS products are protected under numerous U.S. and foreign patents and pending applications, mask working rights, and copyrights. SLS reserves the right to make changes to any products and services at any time without notice. SLS assumes no responsibility or liability arising out of the application or use of any information, products, or service described herein except as expressly agreed to in writing by SLS. SLS customers are advised to obtain the latest version of specifications before relying on any published information and before orders for products or services.

ug_ipusb20sr_1.3

Introduction

This guide helps users to know about the basics of USB 2.0 (USB20SR), the software based enumeration device IP Core.

Table below shows the revision history of this user guide.

Version	Date	Description
1.3	January 2013	- Added DCVERSION register - Changed offset of MAIN_CSR and FRM_NAT registers - Changed range (0x00 to 0xFF) of Phy register address in ULPI_REG_ACCESS
1.2	July 2011	Replace EPn_INT with EPn_IMS in Chapter 4
1.1	February 2010	Added 7 th and 29 th bit description in EPn_IMS register as per new IP core version
1.0	July 2009	First Publication of USB20SR, Software based enumeration device IP Core

How To Find Information

- The Adobe Acrobat Find feature allows you to search the contents of a PDF file. Use Ctrl + F to open the Find dialog box. Use Shift + Ctrl + N to open to the Go To Page dialog box.
- Bookmarks serve as an additional table of contents.
- Thumbnail icons, which provide miniature preview of each page, provide a link to the pages.
- Numerous links shown in Navy Blue color allow you to jump to related information.

How to Contact SLS

For the most up-to-date information about SLS products, go to the SLS worldwide website at <http://www.slscorp.com>. For additional information about SLS products, consult the source shown below.

Information Type	E-mail
Product literature services, SLS literature services, Non-technical customer services, Technical support.	support@slscorp.com

Typographic Conventions

The user guide uses the typographic conventions as shown below:

Visual Cue	Meaning
Bold Type with Initial Capital letters	All headings and Sub headings Titles in a document are displayed in bold type with initial capital letters; Example: Core Architecture, Operation.
Bold Type with Italic Letters	All Definitions, Figure and Table Headings are displayed in Italics. Examples: <i>Table 4-1. Register Details, Figure 2-1. USB 20SR IP Core Architecture</i>
Italic type	Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <i><USB20SR Installation Path></i> .
1., 2.	Numbered steps are used in a list of items, when the sequence of items is important. such as steps listed in procedure.
•	Bullets are used in a list of items when the sequence of items is not important.
	The hand points to special information that requires special attention
	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes.
	The feet direct you to more information on a particular topic.

<i>About this Guide</i>	<i>iii</i>
Introduction	iii
How To Find Information	iii
How to Contact SLS	iv
Typographic Conventions	v
 1. Introduction	 1
Features	2
Core Resources	2
Further Information.....	3
 2. Core Architecture	 4
ULPI PHY	5
ULPI Interface.....	5
Protocol Layer.....	5
EndPoint Registers	5
EP0 Controller.....	5
On Chip RAM.....	6
Micro Controller/Processor Interface.....	6
 3. Operation	 7
EndPoints	8
Buffer Pointers	8
Data Organization	8
Interrupts	9
Timing.....	9
Software Interaction.....	9
 4. Core Registers	 11
DCVERSION.....	14
FUNC_ADR.....	15

INT_MSK	15
INT_SRC	16
MAIN_CSR	21
FRM_NAT	23
TEST MODE	24
SETUP_PACK_1	25
SETUP_PACK_2	25
ULPI_PHY_CS	25
ULPI_REG_ACCESS	26
D_SPEED_SEL	27
D_CNCT	28
Endpoint Registers	28
EPn_CSR	28
EPn_IMS	31
EPn_BUF	33
5. Core IOs	36
6. Using USB20SR IP in SOPC Builder	37

1. Introduction

The USB 2.0 Device IP Core (USB20SR) is a RAM based, 32-bit Avalon interface and ULPI interface support. The core supports both High Speed (480 Mbps) and Full Speed (12 Mbps) functionality. The core supports three endpoints Control, IN and OUT. Support of up to 15 endpoints can be added inside the IP Core as per customer request. Endpoint can be used for IN or OUT operation at a time. Support of IN and OUT operation on a single endpoint can be added as per customer request to add support of 15 IN and 15 OUT endpoint. Each endpoint has an endpoint controller that supports interrupt, bulk, and isochronous transfers.

The core is an RTL design in Verilog that implements an USB device controller on an ASIC or FPGA. The core has been optimized for popular FPGA devices and its functionality has been verified on the real hardware. It is provided as Altera Quartus II Mega function (Altera SOPC Builder ready component) and integrates easily into any SOPC Builder generated system using Nios[®] II Avalon bus.

This user guide will provide you with some basic technical details of USB20SR device core acting as an Avalon slave in Altera's SOPC builder. SOPC builder is a software tool that allows for the creation of a Nios II system module or a more general multi master **System On A Programmable Chip** module. A complete Nios II system module contains a Nios II (soft core 32 bit RISC) processor and its associated system peripherals.



For development kits lacking USB interface, SLS has ULPI interface Snap On Board available, along with a USB 2.0 device IP core integrated into SOPC builder. The Snap On Board snaps on to the Altera Standard Santa Cruz header, therefore, now your Nios development kit is with USB 2.0 interface without any extra effort. So now using SOPC builder one can build a system with USB interface in just a few minutes.



To develop your Embedded System using USB 2.0 ULPI interface refer to [USB 2.0 On-The-Go \(ULPI\) Snap On Board Reference Manual](#). You can also

refer to [CoreCommander Reference Manual](#).

Features

Following are the USB20SR Device IP core features:

- USB 2.0 USB IF high-speed certified
- Supports both High Speed (480 Mbps) and Full Speed (12 Mbps)
- High speed or Full speed operation selection through Software



Low speed (1.5 Mbps) support also available on special request. Please contact support@slscorp.com for more details

ULPI Interface support

- Pre configured for 3 endpoints
 - CONTROL
 - IN
 - OUT
- Configurable for up to 15 IN/OUT endpoints which can support Bulk, Interrupt, Isochronous functionality on each endpoint on customer request at additional cost.
- Fully software controlled CONTROL, IN and OUT endpoints
- Avalon Bus compliant
- Optimized LE count

Core Resources

[Table 1-1](#) shows the LE usage of the Core.

<i>Table 1-1. Core LE Usage Summary</i>				
Supported Devices	LEs	Memory Bits	M4k/M9k Blocks	Performance (fmax) Mhz
Cyclone	2533	32768	8 m4k	83Mhz
Cyclone II	2527	32768	8 M4K	72Mhz
Cyclone III	2530	32768	8 M9K	90Mhz
Cyclone IV	2532	32768	8 M9K	92Mhz
Stratix	2534	32768	8 M9K	81Mhz
Stratix II	1652	32768	8 M9K	96Mhz

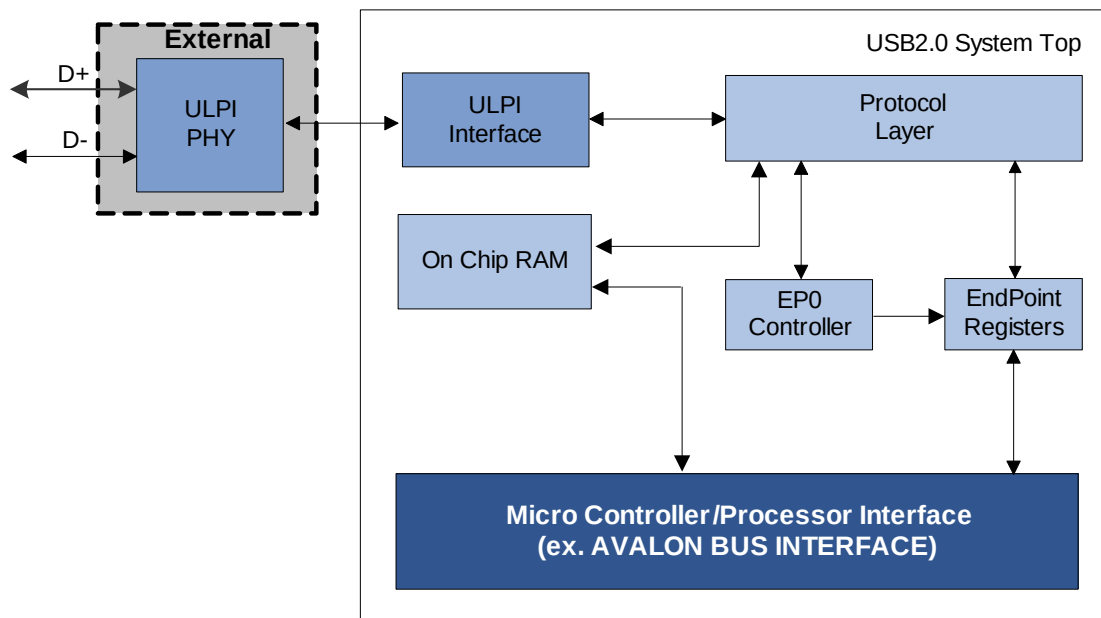
Further Information

For information about USB20SR IP Core installation directory structure and its content, licensing, component implementation and its support, refer **readme.html** located at *<USB20SR Installation Path>\usb20sr*. *<USB20SR Installation Path>* is the installation directory. The default installation directory is *c:\altera\<version #>\ip\sls*. Where version no. is Quartus setup version.'

2. Core Architecture

The USB 2.0 Device communicates through two differential lines (D+ & D-) that connect to a transceiver. The transceiver or physical interface, in turn, connects to the core via interface signals. Since the core supports all the transfer types - Control, Bulk, Isochronous and Interrupt, each transfer type retains a separate pipe for OUT and IN operation. [Figure 2-1.](#) illustrates overall architecture of the USB20SR IP core.

Figure 2-1. USB20SR Device IP Core Architecture



Each of the blocks is described in detail below:

The Micro Controller/Processor interface provides a bridge between Host Interfaces (ex. Nios II Processor) and internal data memory and control registers.

ULPI PHY

The ULPI PHY chip is the external chip that provides a link between USB2.0 IP and the physical USB data lines D+ & D-. The maximum clock from the PHY is 60MHz in 8 bit mode. All the blocks of USB IP Core related to USB activity run from the clock provided by the ULPI PHY for synchronization.

ULPI Interface

The ULPI block connects through the external PHY. It controls speed negotiation as well as handles all functions related to the line signals. It also handles the ULPI PHY chip configuration for device operation.

Protocol Layer

This block De assembles/Assembles the packet, handles all the standard USB 2.0 protocol handshakes and control correspondence. Proto layer also handles all error conditions of USB protocol.

EndPoint Registers

This block has control and status registers for each endpoints. One can configure any specific endpoint through these registers via Avalon interface. Detail description of each register as well as it's bit is given inside the [Table 4-1 Register Description](#).

EP0 Controller

This module will be implemented fully inside the hardware when the IP Core is enabled in debug mode. IP Core can be switched into debug mode in place of normal mode to verify the ULPI Phy chip interface functionality with the HOST PC.

The section below explains the steps to verify the ULPI PHY chip functionality with the host PC.

1. Open the **usb20sr_debug.v** file located at *<USB20SR Installation Path>\hardware\component\hdl*.
2. Disable the "'**define SR_ENABLE**'" line inside the file.
3. Recompile your design.
4. Download the .sof file on your board.
5. Connect the device with the Host PC through the USB 2.0 compliant cable.
6. The default SLS device should be enumerated on the Host PC without running a Nios II project.
7. If the device enumeration fails with the host PC, then there will be an error in generating the reference design or PHY interface on the board.

On Chip RAM

This memory is used to store transmitted as well as received data for End-Points. Size of this memory is 2Kbyte for IN and 2Kbyte for OUT endpoints in default configuration of the IP Core. Micro controller and device interfaces with this memory through two different memory ports with separate clock for each. Size of this memory should be increased through SOPC builder parameter settings, if number of endpoint increases to use separate memory buffers for each endpoint.

Micro Controller/ Processor Interface

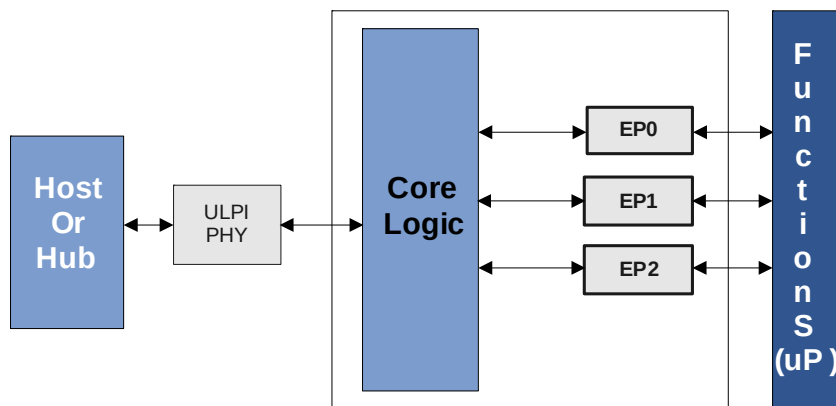
This block provides a consistent core interface between the internal functions of the core and the function-specific host or micro controller.

3. Operation

This core has been designed to compatible with avalon specific interface. The [Figure 3-1.](#) below shows how the core connects to the function micro controller and HOST.

IP core functionality is controlled by function controller as shown in [Figure 3-1.](#) through endpoint registers.

Figure 3-1. Operational Block Diagram



The USB core uses onchip memory (512*32) for IN operation as well as (512*32) for OUT operation separately. IN and OUT memory starts from the same location inside the IP Core (0x20000).

IN memory is write only memory and OUT memory is read only memory for processor. For device OUT memory is write only memory and IN memory is read only memory. No software intervention is needed between endpoint access. Double buffer mechanism is used for reducing the latency requirement on the software, and increasing USB throughput.

EndPoints

This USB core supports 3 endpoints (Control, IN, OUT). The function controller must set up the endpoints by writing to the endpoint register: EPn_CSR, EPn_IMS, EPn_BUFx.

Buffer Pointers

The buffer pointers point to the input/output data structure in memory. If all buffers are not allocated, the core will respond with NAK acknowledge to the USB host.

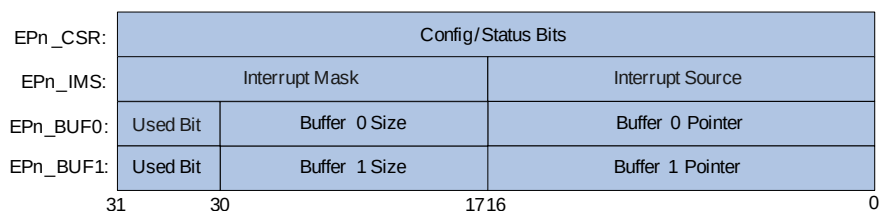
This core supports a double buffering feature which reduces the latency requirements on the functions micro controller and driver software. Data is being retrieved/filled from/to the buffers in a round robin fashion. When data is sent to/from an endpoint, first buffer0 is used. When the first buffer is empty/full, the function controller, may be notified via an interrupt. The function controller can refill/empty buffer0 now. The core will now use buffer 1 for the next operation. When the second buffer is full/empty, the function controller is interrupted, and the core will use buffer0 again, and so on. A buffer that has the used bit set cause the core for replying with NAK/NYET acknowledgments to the host.

The Buffer's Used bits indicate when a buffer has been used (this information is also provided in the Interrupt Source Register). The function controller must clear these bits after it has emptied/refilled the buffer.

Data Organization

Since the buffer memory is 32 bits wide and USB defines all transactions on byte boundaries it is important to understand the relationship of data in the buffer to actual USB byte sequence. This USB core supports Little Endian byte ordering.

Figure 3-2. Data Organization



The buffer pointer always point to byte 0. The USB core always fetches four bytes from the buffer memory. The actual final byte that is transmitted in the N^{th} transaction depends on the Buffer Size. The MaxPacketSize must always be a multiple of 4 bytes.

Interrupts

The USB core provides interrupt outputs. The output is fully programmable. The usage of the interrupt is up to the system into which the USB core is incorporated.

The interrupt mechanism in the USB core consists of a two level hierarchy:

1. **The main interrupt source register (INT_SRC)** indicates interrupts that are endpoint independent. These interrupts indicate overall events that have either global meaning for all endpoints or can not be associated with an endpoint because of an error condition.
2. **The endpoint interrupt source register** indicates events that are specific to an endpoint.

Timing

The interrupt output are asserted when the condition that is enabled in the interrupt mask occurs. They remain asserted until the main interrupt register is read.

Software Interaction

An interrupt handler should first read the main interrupt source register (INT_SRC) to determine the source of an interrupt. It must remember the value that was read until it is done processing of each interrupt source. If any of the bits 15 to 0 are set, the interrupt handler should also read the appropriate

endpoint interrupt register to determine endpoint specific events. Multiple interrupt sources may be indicated at any given time. Software should be prepared to handle every interrupt source it cares about.

A care must be taken not to lose interrupt sources, as the main interrupt source register is cleared after interrupt source register read operation.

4. Core Registers

This section describes all the registers inside the USB20SR Core. The **Register** field describes the name of the register. The **Offset** field describes the offset of register in USB IP. The **Access** field describes the type of access to the register that is read or write. **Description** field describes the type and function of register. [Table 4-1](#) shows the register details.

Table 4-1. Register Description

Sr. No.	Register	Offset	Width	Access	Description
1	DCVERSION	0x00	32	RO	IP Core Version/Product ID description register
2	FUNC_ADR	0x04	8	RW	USB function address register
3	INT_MSK	0x08	16	RW	Interrupt Mask for endpoint independent interrupt sources
4	INT_SRC	0x0c	32	ROC	Interrupt source register
5	MAIN_CSR	0x10	8	RW	Control/Status register
6	FRM_NAT	0x14	32	RO	Frame number and time
7	TEST_MODE	0x20	8	RW	Test mode register for enabling test mode inside the IP Core
8	SETUP_PACK_1	0x30	32	RO	First four bytes of received setup packet
9	SETUP_PACK_2	0x34	32	RO	Second four bytes of received setup packet
10	ULPI_PHY_CS	0x1f0	8	RW	ULPI PHY Chip enable or disable register
11	ULPI_REG_ACCESS	0x1f4	32	RW	ULPI PHY Chip register read/write register
12	D_SPEED_SEL	0x1f8	8	RW	Device IP Core speed select register
13	D_CNCT	0x1fc	8	RW	Device Connect/Disconnect register
14	EP0_CSR	0x40	32	RW	EndPoint 0 (Control Endpoint): CSR Register
15	EP0_IMS	0x44	32	ROC	EndPoint 0 (Control Endpoint): Interrupt Register
16	EP0_BUFFER0	0x48	32	RW	EndPoint 0 (Control Endpoint): Buffer0 Register

Table 4-1. Register Description

Sr. No.	Register	Offset	Width	Access	Description
17	EP0_BUFFER1	0x4c	32	RW	EndPoint 0 (Control Endpoint): Buffer1 Register
18	EP1_CSR	0x50	32	RW	EndPoint 1 CSR Register
19	EP1_IMS	0x54	32	RW and ROC	EndPoint 1 Interrupt Register
20	EP1_BUFFER0	0x58	32	RW	EndPoint 1 Buffer0 Register
21	EP1_BUFFER1	0x5c	32	RW	EndPoint 1 Buffer1 Register
22	EP2_CSR	0x60	32	RW	EndPoint 2 CSR Register
23	EP2_IMS	0x64	32	ROC	EndPoint 2 Interrupt Register
24	EP2_BUFFER0	0x68	32	RW	EndPoint 2 Buffer0 Register
25	EP2_BUFFER1	0x6c	32	RW	EndPoint 2 Buffer1 Register
26	EP3_CSR	0x70	32	RW	EndPoint 3 CSR Register
27	EP3_IMS	0x74	32	ROC	EndPoint 3 Interrupt Register
28	EP3_BUFFER0	0x78	32	RW	EndPoint 3 Buffer0 Register
29	EP3_BUFFER1	0x7c	32	RW	EndPoint 3 Buffer1 Register
30	EP4_CSR	0x80	32	RW	EndPoint 4 CSR Register
31	EP4_IMS	0x84	32	ROC	EndPoint 4 Interrupt Register
32	EP4_BUFFER0	0x88	32	RW	EndPoint 4 Buffer0 Register
33	EP4_BUFFER1	0x8c	32	RW	EndPoint 4 Buffer1 Register
34	EP5_CSR	0x90	32	RW	EndPoint 5 CSR Register
35	EP5_IMS	0x94	32	ROC	EndPoint 5 Interrupt Register
36	EP5_BUFFER0	0x98	32	RW	EndPoint 5 Buffer0 Register
37	EP5_BUFFER1	0x9c	32	RW	EndPoint 5 Buffer1 Register
38	EP6_CSR	0xA0	32	RW	EndPoint 6 CSR Register
39	EP6_IMS	0xA4	32	ROC	EndPoint 6 Interrupt Register
40	EP6_BUFFER0	0xA8	32	RW	EndPoint 6 Buffer0 Register
41	EP6_BUFFER1	0xAc	32	RW	EndPoint 6 Buffer1 Register
42	EP7_CSR	0xB0	32	RW	EndPoint 7 CSR Register

Table 4-1. Register Description

Sr. No.	Register	Offset	Width	Access	Description
43	EP7_IMS	0xB4	32	ROC	EndPoint 7 Interrupt Register
44	EP7_BUFFER0	0xB8	32	RW	EndPoint 7 Buffer0 Register
45	EP7_BUFFER1	0xBc	32	RW	EndPoint 7 Buffer1 Register
46	EP8_CSR	0xC0	32	RW	EndPoint 8 CSR Register
47	EP8_IMS	0xC4	32	ROC	EndPoint 8 Interrupt Register
48	EP8_BUFFER0	0xC8	32	RW	EndPoint 8 Buffer0 Register
49	EP8_BUFFER1	0xCc	32	RW	EndPoint 8 Buffer1 Register
50	EP9_CSR	0xD0	32	RW	EndPoint 9 CSR Register
51	EP9_IMS	0xD4	32	ROC	EndPoint 9 Interrupt Register
52	EP9_BUFFER0	0xD8	32	RW	EndPoint 9 Buffer0 Register
53	EP9_BUFFER1	0xDc	32	RW	EndPoint 9 Buffer1 Register
54	EP10_CSR	0xE0	32	RW	EndPoint 10 CSR Register
55	EP10_IMS	0xE4	32	ROC	EndPoint 10 Interrupt Register
56	EP10_BUFFER0	0xE8	32	RW	EndPoint 10 Buffer0 Register
57	EP10_BUFFER1	0xEc	32	RW	EndPoint 10 Buffer1 Register
58	EP11_CSR	0xF0	32	RW	EndPoint 11 CSR Register
59	EP11_IMS	0xF4	32	ROC	EndPoint 11 Interrupt Register
60	EP11_BUFFER0	0xF8	32	RW	EndPoint 11 Buffer0 Register
61	EP11_BUFFER1	0xFc	32	RW	EndPoint 11 Buffer1 Register
62	EP12_CSR	0x100	32	RW	EndPoint 12 CSR Register
63	EP12_IMS	0x104	32	ROC	EndPoint 12 Interrupt Register
64	EP12_BUFFER0	0x108	32	RW	EndPoint 12 Buffer0 Register
65	EP12_BUFFER1	0x10c	32	RW	EndPoint 12 Buffer1 Register
66	EP13_CSR	0x110	32	RW	EndPoint 13 CSR Register
67	EP13_IMS	0x114	32	ROC	EndPoint 13 Interrupt Register
68	EP13_BUFFER0	0x118	32	RW	EndPoint 13 Buffer0 Register
69	EP13_BUFFER1	0x11c	32	RW	EndPoint 13 Buffer1 Register

Table 4-1. Register Description

Sr. No.	Register	Offset	Width	Access	Description
70	EP14_CSR	0x120	32	RW	EndPoint 14 CSR Register
71	EP14_IMS	0x124	32	ROC	EndPoint 14 Interrupt Register
72	EP14_BUFFER0	0x128	32	RW	EndPoint 14 Buffer0 Register
73	EP14_BUFFER1	0x12c	32	RW	EndPoint 14 Buffer1 Register
74	EP15_CSR	0x130	32	RW	EndPoint 15 CSR Register
75	EP15_IMS	0x134	32	ROC	EndPoint 15 Interrupt Register
76	EP15_BUFFER0	0x138	32	RW	EndPoint 15 Buffer0 Register
77	EP15_BUFFER1	0x13c	32	RW	EndPoint 15 Buffer1 Register



1. Endpoint buffer memory (2K size) is starting from offset address 0X20000.
2. The above table includes all 15 endpoint register details of the IP Core with its offset. From these endpoint registers, only those are applicable whose support is enabled inside the hardware. For example if IP Core supports 6 endpoints then all endpoints from 0 to 5 are applicable while endpoints from 6 to 15 are not applicable. The IP Core supports 3 endpoints by default.

DCVERSION

This register represent IP Core Version/Product ID description of the core. .

[Table 4-2](#) shows the DCVERSION register description.

Table 4-2. DCVERSION Register Details

Bit	Access	Description
31:16	RO	These bits represent the hexadecimal value of Product ID. (i.e. 0A04 for USB20SR)
15:8	RO	These bits represent the hexadecimal value of IP Core Version. (i.e. 13 for IP Core Version 1.3)
7:0	RO	These bits represent the hexadecimal value of Customer Number and Customer Release Number. (i.e. 0 for Customer Number and 0 for Customer Release Number, if any)

Value after Reset: 00000000h

FUNC_ADR

This register is used to set the function address of the device received in set address request from host. Function address should be written inside this register before replying the status stage of the SET_address request. Device address will be set inside the IP Core after status stage reply of the SET_ADDRESS request. [Table 4-3](#) shows the list of FUNC_ADR register description.

Table 4-3. FUNC_ADR Register Details

Bit	Access	Description
7	RW	Reserved
6:0	RW	Function address value

Value after Reset: 00h

INT_MSK

The interrupt mask register defines functionality of interrupt output with regard to event not associated to endpoints. Processor can enable or disable the interrupts as per the application requirement. The information-only-interrupts can be disabled while other interrupts should be handled properly by processor for configuration of the device according to the device speed as well as data communication to and from the host PC. [Table 4-4](#) shows the list of Interrupt Mask register description.

Table 4-4. Interrupt Mask (INT_MSK) Register Details

Bit	Access	Description
15:10	RW	Reserved
9	RW	Speed Negotiation Done Mask 1: Enable interrupt generation due to Speed Negotiation Done 0: Disable interrupt generation due to Speed Negotiation Done
8	RW	USB Reset Mask 1: Enable interrupt generation due to USB reset 0: Disable interrupt generation due to USB reset
7	RW	UTMI RX Error Mask 1: Enable interrupt generation due to UTMI RX Error 0: Disable interrupt generation due to UTMI RX Error

6	RW	Device Detached Mask 1: Enable interrupt generation due to device is detached 0: Disable interrupt generation due to device is detached
5	RW	Device Attach Mask 1: Enable interrupt generation due to device is attached 0: Disable interrupt generation due to device is attached
4	RW	Resume Mode Mask 1: Enable interrupt generation due to device resumes 0: Disable interrupt generation due to device resumes
3	RW	Suspend Mode Mask 1: Enable interrupt generation due to device suspends 0: Disable interrupt generation due to device suspends
2	RW	No such endpoint Mask 1: Enable interrupt generation due to no such endpoint error 0: Disable interrupt generation due to no such endpoint error
1	RW	PID Error Mask 1: Enable interrupt generation due to PID Error 0: Disable interrupt generation due to PID Error
0	RW	CRC5 Error Mask 1: Enable interrupt generation due to CRC5 error 0: Disable interrupt generation due to CRC5 error

Value after Reset: **0000h**

INT_SRC

This register identifies the interrupt source. Whenever the function controller receives an interrupt, the interrupt handler must read this register to determine the source and cause of an interrupt. All endpoint independent device interrupt clears, once read operation is completed over the register. The software interrupt handler must make sure that it keeps whatever information is required to handle the interrupt. [Table 4-5](#) shows the interrupt source register description.

Table 4-5. Interrupt Source (INT_SRC) Register Details

Bit	Access	Description
31:30	RO	Reserved

29	ROC	Speed Negotiation Done After getting this interrupt, processor should configure the endpoints according to the device speed. Device sets this bit, while the speed negotiation process is completed during reset stage. Device performs the speed negotiation process to decide the device communication with host (high speed or full speed mode). 1: Speed negotiation process is completed. 0: Speed negotiation process is not completed.
28	ROC	USB Reset Software must reconfigure the endpoint registers after detecting this interrupt. Device sets this interrupt bit when host performs USB reset operation over the device. Device resets the IP Core internally and all registers. Host performs this operation when a device is attached with the host and also when an application driver resets the device whenever it finds any problem with the device communication. 1: Device is reset. 0: Device is not reset.
27	ROC	USB Rx Error This interrupt bit is for information purpose only. Device sets this interrupt whenever it finds any undesired D+ and D- value during data receive operation. Data lines status is verified continuously for attached speed mode of the device. Device takes all necessary steps to handle this error. 1: Rxerror is received 0: Rxerror is not received
26	ROC	Detached Device sets this interrupt bit when it is detached from the host and vbus is removed or the device is disconnected through disconnect register (D_CNCT). 1: Device is disconnected 0: Device is not disconnected
25	ROC	Attached Device sets this interrupt bit when it is attached to the host pc and Vbus goes high. Device should be enabled to connect with host through connect register (D_CNCT) for getting this interrupt. 1: Device is attached 0: Device is not attached
24	ROC	Resume Device sets this interrupt bit whenever it resumes from suspend state. Host controller resumes the device from suspend state if it requires to perform any operation over a suspended device or device responds with remote wake up event. 1: Device is resumed 0: Device is not resumed

23	ROC	Suspend Device sets this bit when there is no any activity on USB data lines for 3ms and goes into suspend state. Software can use this information for power saving of component used with USB. 1: Device is suspended 0: Device is not suspended
22	ROC	No Such Endpoint This interrupt bit is for information purpose only. This bit provides status of one possible error of USB communication. Device sets this bit whenever any endpoint number selected inside the received token is not supported by the device. Device takes all necessary action to handle this error itself and no any intervention is needed from processor. e.g. Token received with EP no = 4 and device supports only three EPs. 1: Token is received with endpoint number error 0: Token is not received with endpoint number error
21	ROC	PID Error This interrupt bit is for information purpose only. This bit provides status of one possible error of USB communication. Device sets this bit whenever any token is received with PID checksum error from host. Device takes all necessary action to handle this error itself and no any intervention is needed from processor. 1: Packet received with PID checksum error. 0: Packet with PID checksum error is not received.
20	ROC	Bad Token (CRC5 error) This interrupt bit is for information purpose only. This bit provides status of one possible error of USB communication. Device sets this bit whenever any token is received with CRC5 error from host. Device takes all necessary action to handle this error itself and no any intervention is needed from software. 1: Packet is received with token error. 0: Packet is not received with token error
19:16	RO	Reserved
15	RO	Endpoint 15 interrupt Device sets this bit when any interrupt bit of endpoint15 is set to high. This bit gets cleared when software reads endpoint15 interrupt register. Details of the endpoint interrupt event is given inside the endpoint interrupt register information. 1: Interrupt bit is set inside endpoint15. 0: Interrupt bit is not set inside endpoint15.

14	RO	Endpoint 14 interrupt Device sets this bit when any interrupt bit of endpoint14 is set to high. This bit gets cleared when software reads endpoint14 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint14. 0: Interrupt bit is not set inside endpoint14.
13	RO	Endpoint 13 interrupt Device sets this bit when any interrupt bit of endpoint13 is set to high. This bit gets cleared when software reads endpoint13 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint13. 0: Interrupt bit is not set inside endpoint13.
12	RO	Endpoint 12 interrupt Device sets this bit when any interrupt bit of endpoint12 is set to high. This bit gets cleared when software reads endpoint12 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint12. 0: Interrupt bit is not set inside endpoint12.
11	RO	Endpoint 11 interrupt Device sets this bit when any interrupt bit of endpoint11 is set to high. This bit gets cleared when software reads endpoint11 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint11. 0: Interrupt bit is not set inside endpoint11.
10	RO	Endpoint 10 interrupt Device sets this bit when any interrupt bit of endpoint10 is set to high. This bit gets cleared when software reads endpoint10 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint10. 0: Interrupt bit is not set inside endpoint10.
9	RO	Endpoint 9 interrupt Device sets this bit when any interrupt bit of endpoint9 is set to high. This bit gets cleared when software reads endpoint9 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint9. 0: Interrupt bit is not set inside endpoint9.

8	RO	Endpoint 8 interrupt Device sets this bit when any interrupt bit of endpoint8 is set to high. This bit gets cleared when software reads endpoint8 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint8. 0: Interrupt bit is not set inside endpoint8.
7	RO	Endpoint 7 interrupt Device sets this bit when any interrupt bit of endpoint7 is set to high. This bit gets cleared when software reads endpoint7 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint7. 0: Interrupt bit is not set inside endpoint7.
6	RO	Endpoint 6 interrupt Device sets this bit when any interrupt bit of endpoint6 is set to high. This bit gets cleared when software reads endpoint6 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint6. 0: Interrupt bit is not set inside endpoint6.
5	ROs	Endpoint 5 interrupt Device sets this bit when any interrupt bit of endpoint5 is set to high. This bit gets cleared when software reads endpoint5 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint5. 0: Interrupt bit is not set inside endpoint5.
4	RO	Endpoint 4 interrupt Device sets this bit when any interrupt bit of endpoint4 is set to high. This bit gets cleared when software reads endpoint4 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint4. 0: Interrupt bit is not set inside endpoint4.
3	RO	Endpoint 3 interrupt Device sets this bit when any interrupt bit of endpoint3 is set to high. This bit gets cleared when software reads the endpoint3 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint3. 0: Interrupt bit is not set inside endpoint3.

2	RO	Endpoint 2 interrupt Device sets this bit when any interrupt bit of endpoint2 is set to high. This bit gets cleared when software reads endpoint2 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint2. 0: Interrupt bit is not set inside endpoint2.
1	RO	Endpoint 1 interrupt Device sets this bit when any interrupt bit of endpoint1 is set to high. This bit gets cleared when software reads the endpoint1 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint1. 0: Interrupt bit is not set inside endpoint1.
0	RO	Endpoint 0 interrupt Device sets this bit when any interrupt bit of endpoint0 is set to high. This bit gets cleared when software reads the endpoint0 interrupt register. Details of the endpoint interrupt event is given inside endpoint interrupt register information. 1: Interrupt bit is set inside endpoint0. 0: Interrupt bit is not set inside endpoint0.

Value after reset: 00000000h



Register detail given above includes all 15 endpoint interrupt but only those are applicable which are available inside the hardware.



To get more information about USB error, read the proto layer section of USB2.0 specification.

MAIN_CSR

This is the main configuration and status register of the core. This register provides current status of the device. [Table 4-6](#) shows the Control/Status register description.

Table 4-6. Control/Status (MAIN_CSR) Register Details

Bit	Access	Description
7	R/W	Remote Wakeup This bit is used to generate a remote wakeup event from the device while device is in suspend state. The device should be enumerated with remote wake up functionality support to use this bit. After the remote wake up operation is completed successfully, the device clears this bit. Remote wake up should not be performed until host enables the device for remote wake up event through Set feature. Device clears this bit after performing a resume request. 1: Device is enabled for remote wake up 0: Device is disabled for remote wake up
6	RO	Reserved
5	RO	Reserved
4:3	RO	Linestate These bits provide the current state of D+ and D- line. Linestate[0] = D+ line Linestate[1] = D- line
2	RO	Device attached This bit is used to check the status of the attached device. Device sets this bit, when it is attached to the Host and Vbus is available from the Host PC Port. 1: Device is attached to the host. 0: Device is detached from the host.

1	RO	Device speed This bit is used to configure the endpoints according to device speed. Device sets this bit when it is connected into high speed mode with the Host PC after the successful speed negotiation. Device clears this bit when it is connected to the Host PC into full speed mode due to speed negotiation is failed. While the Host PC does not support EHCI controller or EHCI is disabled, speed negotiation is failed and device is connected into full speed mode. When the device is enabled for high speed mode using D_SPEED_SEL register and it is attached into full speed mode with the host then the message will be displayed that device can be connected into high speed mode. 1: Device is attached into high speed mode 0: Device is attached into full speed mode
0	RO	Suspend This status bit is used to check whether the device is in suspend state or not. Device sets this bit when it goes into suspend state and clears the bit when it resumes from suspend state. When the device is in suspend state there would not be any USB device related operation from the host. 1: Device is suspended 0: Device is active

Value after Reset: 00h

FRM_NAT

This register tracks frame number as received from the SOF token, and the frame time. [Table 4-7](#) shows the Frame Number and Time register description.

Table 4-7. Frame Number and Time (FRM_NAT) Register Details

Bit	Access	Description
31:28	RO	Number of frames with the same frame number (this field may be used to determine current microframe)
27	RO	Reserved
26:16	RO	Frame number as received from SOF token
15:12	RO	Reserved
11:0	RO	Time since last SOF in 0.5 uS resolution

Value after reset: 00000000h

TEST MODE

This register is used to configure the device into specific test mode for compliance test of the device. Different test modes are used for compliance test to check verify the device against its all electrical test cases. Once device is enabled into any test mode, it can not be used for any other operation. External reset should be used to exit the device from the test mode.

[Table 4-8](#) shows the Test Mode register description.

Table 4-8. Test Mode (TEST MODE) Register Details

Bit	Access	Description
7:5	RO	Reserved
4	RW	Enable_test_mode This bit is used to enable test mode on device selected from first four bits of this register for compliance test. At a time only one test mode should be selected in first four bits. 1: Test mode enable 0: Test mode disable
3	RW	Test_packet_mode This bit is used to select Test-packet mode for compliance test. 1: test mode is selected 0: test mode is not selected
2	RW	Test_se0_mode This bit is used to select Test-SE0 mode for compliance test. 1: test mode is selected 0: test mode is not selected
1	RW	Test_k_mode This bit is used to select Test-K mode for compliance test. 1: test mode is selected 0: test mode is not selected
0	RW	Test_J_mode This bit is used to select Test-J mode for compliance test. 1: test mode is selected 0: test mode is not selected

Value after Reset: 00h

SETUP_PACK_1

This register is used to read first four bytes received of setup packet.

[Table 4-9](#) shows the First Setup Packet register description.

Table 4-9. Setup Packet (SETUP_PACK_1) Register Details

Bit	Access	Description
31:0	RO	First four bytes of setup packet Setup packet is received with 8 bytes data information and from that, first four bytes are stored in this register. After getting a setup packet receive interrupt on control endpoint, processor should read this register for getting information of first four bytes of setup packet.

Value after reset: 00000000h

SETUP_PACK_2

This register is used to read second four bytes received of setup packet.

[Table 4-10](#) shows the Second Setup Packet register description.

Table 4-10. Setup Packet (SETUP_PACK_2) Register Details

Bit	Access	Description
31:0	RO	Second four bytes of setup packet Second four bytes of setup packet is stored in this register. Processor should read this register after reading first four bytes of setup packet to get the complete information of received setup packet.

Value after reset: 00000000h

ULPI_PHY_CS

This register is used to control the chip select pin of the ULPI PHY Chip through processor. Processor can enable or disable the ULPI PHY Chip through this register as per the application requirement. [Table 4-11](#) shows the ULPI PHY Chip Enable register description.

Table 4-11. ULPI PHY Chip Enable (ULPI_PHY_CS) Register Details

Bit	Access	Description
7:1	RW	Reserved
0	RW	This bit is used to enable or disable the ULPI PHY chip signal. Default bit value of this bit is 1. ULPI PHY chip select should be disabled to use its IO pins in sharing mode. 1: ULPI PHY chip select enable. 0: ULPI PHY chip select disable.

Value after Reset: 00h

ULPI_REG_ACCESS

ULPI PHY Chip has number of different registers inside it to control its functionality through Phy Chip interface. This register is used to provide read or write access of the ULPI chip register to the processor. Processor can read or write the specific register of ULPI PHY Chip through proper configuration of the register inside the USB IP Core. [Table 4-12](#) shows the ULPI PHY Chip Register Read/Write register description.

Table 4-12. ULPI PHY Chip Register Read/Write (ULPI_REG_ACCESS) Register Details

Bit	Access	Description
31:26	RO	Reserved
25	RW	Phy Reg read enable This bit is used to enable PHY chip register read operation inside the device IP Core. This bit should be set by the processor when it want to read any register of the PHY chip. Device clears this bit when it completes register read operation and load the read data. 1: Reg read operation enable. 0: Reg read operation disable or completed.
24	RW	Phy Reg write enable This bit is used to enable PHY chip register write operation. This bit should be set by the processor when it want to write any register of the PHY chip. Device clears this bit when it completes register write operation. 1: Reg write operation enable. 0: Reg write operation disable or completed

23:16	RO	Phy Reg Readdata These bits are used to store readdata received from selected register during read operation.
15:8	RW	Phy Reg writedata These bits of the register is used to specify write data which will be loaded inside the selected PHY register when write operation is enabled through write_en bit setting.
7:0	RW	Phy register address This address bits are used to select PHY chip register on which read or write operation needs to be performed.

Value after Reset: 00h

D_SPEED_SEL

This register is used to configure the USB device core for specific speed communication. With the use of this register, device core can be used as a USB20 high speed device or it can be used as a USB1.1 Full speed device.

[Table 4-13](#) shows the Device Speed Select register description.

Table 4-13. Device Speed Select (D_SPEED_SEL) Register Details

Bit	Access	Description
7:1	RW	Reserved
0	RW	This bit is used to select speed mode for USB device IP Core. Device can be set into high or full mode before it is connected to the host PC. In high speed mode, device can be attached to host in high speed if EHCI controller is available and speed negotiation is successful otherwise it is connected into full speed with full speed controller. In full speed mode device can be attached to the host into full speed only even though enhanced host controller is present. 1: Device is enabled for high speed connection 0: Device is enabled for full speed connection

Value after Reset: 01h

D_CNCT

This register is used to control device connect/disconnect operation through processor. This is 8 bit register. For connection of the device with the host, 0x01 should be written to the register and to disconnect the device from host 0x00 should be written to the register. [Table 4-14](#) shows the Device Connect/

Disconnect register description.

Table 4-14. Device Connect / Disconnect (D_CNCT) Register Details

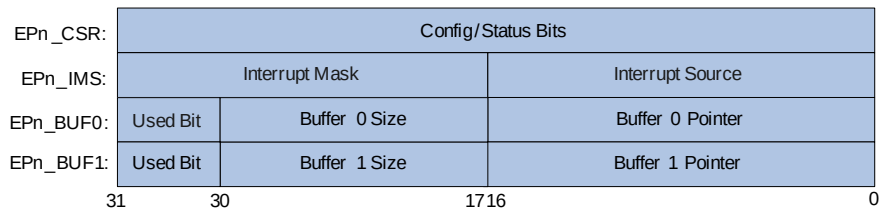
Bit	Access	Description
7:1	RW	Reserved for future use
0	RW	Connect/Disconnect 1 = Device is enabled for connection 0 = Device is disabled for connection

Value after Reset: 00h

Endpoint Registers

Each endpoint has 4 registers associated with it. These registers have exactly the same definition for each endpoint except for control endpoint (please refer a control endpoint related special notes where its flow is different compare to other endpoints.)

Figure 4-1. Endpoint Register



EPn_CSR

This is the control and status register of the endpoint. Through this register processor can configure the endpoint with few basic functionality support. It can also use this register to get back some information during communication to take few decisions based on it. Detail description for usage of each bit of this register is given in table below. [Table 4-15](#) shows the Endpoint CSR register description

Table 4-15. Endpoint CSR (EPn_CSR) Register Details

Bit	Access	Description
31:30	RW	Buffer_select Buffer_select[0] This register bit is used to select a buffer for next transaction over selected endpoint. Each endpoint of the IP Core has double buffer support for reducing data communication latency over a endpoint. Device toggles this bit after successful transaction over a selected endpoint. This bit is used with buffer done bit of the buffer registers to select the buffer for next transaction. To use single buffer communication over a device, buffer-done bit of buffer1 should always be disabled. IP Core buffer selection is as given below. 0: Buffer0 of the endpoint is selected. 1: Buffer1 of the endpoint is selected. Buffer_select[1] This bit usage is reserved. Support of this bit is given to add support for four buffer over a endpoint, which is configured for IN and OUT operation support as per the custom request. By default endpoint can perform a single direction communication either IN or out. <i>Note:</i> Control endpoint does not use this information for buffer selection and it is reserved as it support data communication on both direction. Control endpoint uses buffer0 for OUT operation and buffer1 for IN operation.
29:28	RW	DATAPID_sel These two bits are used by device to keep track of the next expected datapid for the transaction over a endpoint. Device handles these two bits on its own to keep track of the next datapid once it get initialized. Processor should reset these bits after getting device reset or setup packet request with endpoint clear feature or set interface or set configuration. Misuse of these bits can give loss of data packets. Refer proto layer specification for more detailed information of data pid usage for different types of USB application design. <i>Note:</i> Control endpoint does not use this information and hardware itself takes care for this data bit.
27:26	RW	EP_TYPE Endpoint Type These two control bits are used by device to select the type of the endpoint. 00: Control Endpoint 01:IN EndPoint 10: OUT Endpoint 11: Reserved

25:24	RW	TR_TYPE Transfer Type This two bits are used to configure the endpoint for different types of transfer support on it. USB controller handles transaction over the endpoint differently depending on this input. 00: Interrupt 01: Isochronous 10: Bulk 11: Reserved
23:22	RW	EP_DIS Temporarily Disable EP These two bits are used to control the endpoint during error condition. Halt feature of the endpoint is set to send stall handshake to the host controller. Endpoint can be disabled temporary through these bits for avoiding any process by controller over the endpoint. 00: Normal 01: Force the core to ignore transfer over this EP 10: Set EP HALT 11: Reserved
21:18	RO	EP_number
17	RW	LRG_OK These bits are used to show the number. Endpoint numbers are fixed from hardware itself and it can not be changed through software. 1 - Accept data packets of more than MAX_PL_SZ (RX only) 0 - Ignore data packets of more than MAX_PL_SZ (RX only)
16	RW	Small_ok This bit is used by device to check whether endpoint is configured for receiving packet with payload size less than max packet length. If this bit is set to 1, then device will accept the received packet with payload less than the maximum packet length size setting of the endpoint. If this bit is set to 0 then endpoint will receive only those packet from host which has data payload size as its maximum packet length size. Device will discard all small packets received from host on its own and no information will be provided to host.
15	RO	Reserved
14	RO	Reserved
13	RO	Reserved

12:11	RW	TF_FR Number of transaction per microframe (HS mode only) These bits are used for isochronous data communication only. These bits are not used for bulk and interrupt transfer over a device. These bits are used to give information of the number of transaction per microframe. Bit value setting depends on the endpoint description given inside the enumeration process.
10:0	RW	MAX_PL_SZ Maximum payload size (MaxPacketSize) in bytes These bits are used by device to get information of max payload size of the endpoint. Different endpoint can be configured with different maximum packet length within its max limit for select transfer type. Endpoint maximum value should be matched with the endpoint max packet length description given inside the endpoint descriptor during enumeration process. Mismatch between these two can give unexpected device behavior.

Value after Reset: All bits of the register will be zero after reset except endpoint number.

EPn_IMS

The interrupt register for each endpoint has mask bits for interrupt output and bits that indicate the interrupt source when an interrupt has been received. Interrupt source bits are set depending on its mask bit setting given inside the same register. Interrupt bit clears after read operation of the register.

[Table 4-16](#) shows the Endpoint Interrupt Mask/Source register description.

Table 4-16. Endpoint Interrupt Mask/Source (EPn_IMS) Register Details

Bit	Access	Description
31:30	RO	Reserved
29	RW	ISO out packet Error Mask 1: Enable interrupt generation due to ISO Out Packet Error 0: Disable interrupt generation due to ISO Out Packet Error
28	RW	PID Sequence Error Mask 1: Enable interrupt generation due to PID Sequence Error 0: Disable interrupt generation due to PID Sequence Error
27	RW	Buffer used Mask 1: Enable interrupt generation when buffer used bit of buffer registers is set. 0: Disable interrupt generation when buffer used bit of buffer registers is set.
26	RW	Unsupported PID Mask 1: Enable interrupt generation due to unsupported PIDError 0: Disable interrupt generation due to unsupported PIDError

25	RW	CRC 16 error Mask 1: Enable interrupt generation due to CRC16 error occurred in received data packet. 0: Disable interrupt generation due to CRC16 error occurred in received data packet
24	RW	Time out error Mask 1: Enable interrupt generation due to timeout error 0: Disable interrupt generation due to timeouterror
23:08	RO	Reserved
7	ROC	ISO out packet lose This interrupt is only supported for isochronous out endpoint and for other endpoints. It is reserved with value set to 0. Device sets this bit for isochronous out endpoint whenever any received packet lose on isochronous out endpoint. 1: Received packet is lose 0: Received packet is not lose
6	ROC	Setup packet receive This interrupt is only supported for control endpoint and for other endpoints it is reserved with value set to 0. Device sets this bit for control endpoint whenever any setup packet request is received on control endpoint. 1: Setup packet is received 0: Setup packet is not received
5	ROC	PID Sequence Error This interrupt bit is for information purpose only. Device sets this bit whenever it receives a data packet with PID sequence error on out endpoint. Device takes necessary action to handle this error condition. 1:Data packet is received with PID sequence error 0:Data packet is not received with PID sequence error
4	ROC	Buffer 1 Used Device sets this bit whenever the IN/OUT operation gets completed on an endpoint buffer1. For IN endpoint, device sets this interrupt bit when it transmits all data from buffer 1 and buffer 1 is empty. For OUT endpoint, device sets this bit when it receives short data packet or if there is no enough space to receive other max_packet_size packet inside the buffer 1 at the end of the current packet receive operation. It also sets the buffer used bit of the buffer1 register of the endpoint before this interrupt is set to high. 1: Operation is completed on buffer 1. 0: Operation is not completed on buffer 1 <i>Note:</i> Control endpoint uses single buffer communication for IN and OUT operation. Device sets buffer0 used bit when out operation is finished.

3	ROC	Buffer 0 Used Device sets this bit whenever IN/OUT operation gets completed on an endpoint buffer0. For IN endpoint, device sets this interrupt bit when it transmits all data from buffer 0 and buffer 0 is empty. (updated buffer size value of buffer register = 0). For OUT endpoint, device sets this bit when it receives short data packet or if there is no enough space to receive other max_packet_size packet inside the buffer0 at the end of the packet receive operation. It also sets the buffer used bit of the buffer0 register of the endpoint before this interrupt is set high. 1: Operation is completed on buffer 0 0: Operation is not completed on buffer 0 <i>Note:</i> Control endpoint uses single buffer communication for IN and OUT operation. Device sets buffer0 used bit when out operation is finished.
2	ROC	Unsupported PID error This interrupt bit is for information purpose only. Device sets this bit whenever it receives unsupported PID for the endpoint. It means if endpoint is configured as out endpoint and it receives Token byte for IN operation selection. Device takes necessary action for handling this error. 1: Unsupported PID is received inside token byte 0: Unsupported PID is not received inside token byte
1	ROC	Bad packet error This interrupt bit is for information purpose only. This interrupt receives on OUT endpoint only. Device sets this bit whenever it receives data packet with CRC16 error. 1: Data packet is received with CRC16 error. 0: Data packet is not received with CRC16 error.
0	ROC	Time out error This interrupt bit is for information purpose only. Device sets this bit on different condition for IN and out endpoint. Device sets this bit for out endpoint when data packet is not received after out token packet within out operation time out limitation as per the specification. For IN endpoint, device sets this bit when it does not receive ACK token from host after data packet transmission within IN operation time out limitation as per the specification. Device takes necessary action to handle this error. 1: Time out error is received 0: Time out error is not received

Value after Reset: 00000000h

EPn_BUF

The endpoint buffer register holds the buffer pointer for each endpoint. Each endpoint has two buffer registers, thus allowing double buffering as well as buffer start address inside the on chip buffer memory. Each buffer register has

exactly the same definition and functionality. Control endpoint supports IN and OUT operation both and it uses two buffer registers for different operation and support single buffer for each direction communication. Control endpoint uses Buffer0 for all OUT operation over it and buffer1 for all IN operation over it. [Table 4-17](#) shows the Endpoint Buffer register description.

Table 4-17. Endpoint Buffer (EPn_BUF) Register Details

Bit	Access	Description
31	RW	USED This bit is used to control the buffer usage by the processor and device IP Core. When the bit value of this bit is set to 1, processor can read or write to buffer0 memory space. While the bit value is set to 0, device IP Core can use the buffer0 memory space to store received data or to transmit data in response of IN token received for endpoint. Default value after reset is 1 for this bit. For IN endpoint, processor resets this bit with proper setting of buffer pointer as well as size, once it completes copying of data inside the on chip buffer memory. For OUT endpoint, processor clears this bit to receive data from host. Device uses the endpoint buffer if its used bit is cleared by the processor. Device sets the used bit when it completes the usage of buffer. Device does not receive or transmit data on endpoint if there is no any buffer available for given endpoint and sends NACK token in response of IN or OUT operation. 1: Buffer is not available for the device and processor can process on buffer space as well as can configure the buffer register for next operation. 0: Buffer is available for the device and processor can not process the buffer space as well as buffer register until the used bit is set by the device. <i>Note:</i> The used bit inside the buffer registers of control endpoint is cleared when setup packet receive operation starts inside the IP core.

30:17	RW	BUF_SIZE These bits of the register is used by the processor to inform the size of the buffer to device IP Core. The processor can change value of these bits before clearing the used bit of the register. Device IP Core uses this information, if used bit is cleared by processor to check how many number of received bytes can be stored inside the buffer or how many number of bytes can be transmitted from buffer in response of IN token. Device updates the value of these bits as it progress with its use. e.g. device buffer register is allocated with size 1024 for OUT endpoint and device receives a 512 byte successfully. Then after sending ACK handshake it reduces the buffer size with 512 bytes (received number of bytes) space. Similarly for IN operation.
16:0	RW	BUF_PTR These bits are used to inform the device about the starting location of the allocated buffer for endpoint. Processor decides the starting location of the endpoint buffer and informs about that to the device IP Core through these bits. Device IP Core can store or transmit data from starting location of the buffer up to its available size, if used bit of buffer register is cleared. Device IP Core updates buffer pointer with the number of bytes transmitted from buffer or received inside the buffer to keep track of starting location for the next data packet inside the buffer to transmit or receive if its used bit is cleared.

Value after Reset: **FFFF0000h(Control EP)**

Value after Reset: **FFFFFFFFh(EP1:EP15)**



1. All endpoints except control endpoint support double buffering or reducing latency on data transmit or receive operation for device. Also All endpoints except control endpoint supports only in one direction communication depending on its CSR register configuration. It means endpoint configured as an OUT endpoint can not support IN operation over it. This support can be added as per the customer request.
2. For control endpoint, buffer0 is used for OUT operation while buffer1 is used for IN endpoint.
3. During IN endpoint operation, the buffer act as write only register while in OUT endpoint operation buffer act as read only register.

5. Core IOs

Table 5-1 list all the IOs of the USB20SR IP core.

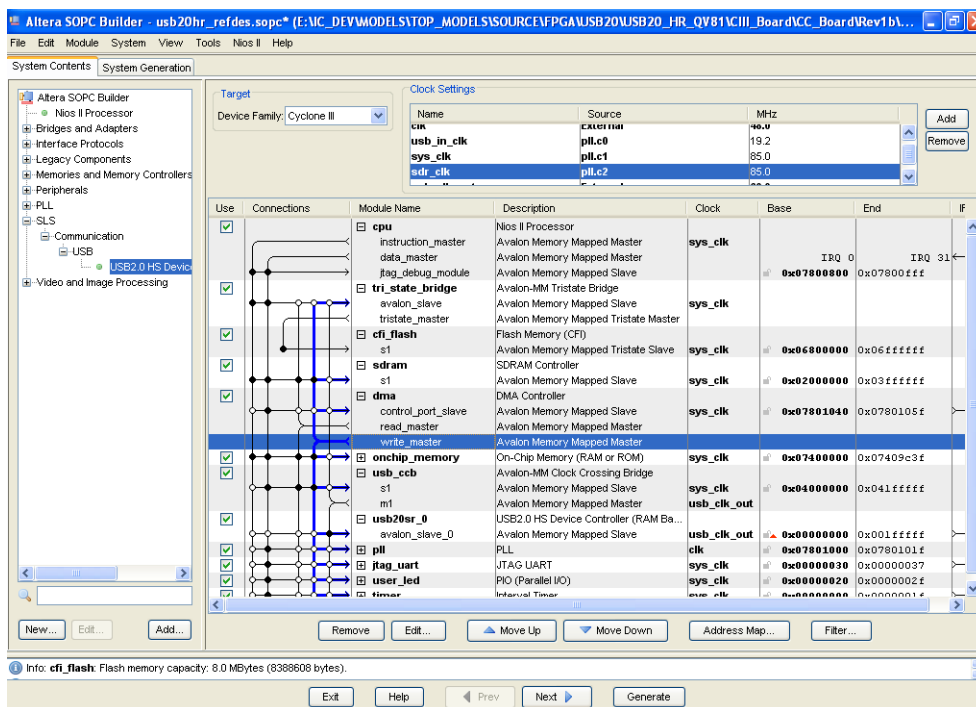
Table 5-1. Core IOs			
Name	Width	Direction	Description
Data	8	Bi-Di	Bi-Direction Data Line for ULPI interface
Dir	1	In	Direction line of ULPI interface. Use the Fast output register, Fast input register, Fast enable register setting from assignment editor of Quartus II software Tool to implement control and data registers of bidirectional lines inside the IO pin
Nxt	1	In	Next line of ULPI interface
Stp	1	Out	Stop line for ULPI interface
Extern_reset	1	In	External reset input of IP Core reset as well as ULPI Phy chip reset. Reset input is active high signal
Phy_cs_n	1	Out	Chipselect of ULPI Phy chip. Chipselect is active low signal
Phy_reset_n	1	Out	Reset pin of ULPI phy chip. Phy reset signal is active low signal
phy_clk	1	In	Input Phy clock (60 MHz)

6. Using USB20SR IP in SOPC Builder

To use USB20SR Device IP core in SOPC Builder follow the steps below:

1. Running **usb20sr_<licensetype>_v<version #>.exe** will automatically place the USB20SR component into SOPC Builder GUI. You can see that USB20SR component is now added in the final view of the SOPC builder system. See [Figure 6-1](#).

Figure 6-1. USB20SR Device IP Core in SOPC Builder



2. Since the core is a slave, there must be an Avalon master. Usually, this master is the Nios processor. Using the SOPC builder tool, include the Nios processor (must have the Nios development kit). Then add any other peripherals required for the final design and assign the base address and irq.



To generate complete reference design from scratch refer tutorial of design from scratch if you are new to SOPC builder tool.



For further reference, refer [Core Registers](#) Section.