USB Function IP Core

Author: Rudolf Usselmann
rudi@asics.ws

Rev. 0.2
January 8, 2001

Preliminary Draft

Revision History

Rev.	Date	Author	Description
0.1	6/1/01	Rudolf Usselmann	First Draft
0.2	8/1/01	Rudolf Usselmann	Second Draft

1. Introduction

The Universal Serial Bus (USB) has evolved to the standard interconnect between computers and peripherals. Everything from a mouse to a camera can be connected via USB. With the new USB 2.0 specification, data rates of over 480 Mb/s are possible.

The Universal Serial Bus is a point to point interface. Multiple peripherals are attached through a HUB to the host.

This core provides a function (peripheral device) interface. It can be used to interface almost any peripheral to a computer via USB. This core fully complies to the USB 2.0 specification and can operate at USB Full and High Speed rates (12 and 480 Mb/s).

This core requires an external PHY (transceiver) that complies to the UTMI specification.

The UTMI Specification, can be downloaded from:
http://developer.intel.com/technology/usb/download/USB_TMI_spec.pdf.

The following companies have announced PHY chips:

Lucent: USS2X1
http://www.lucent.com/micro/usb/usbdocs.html

NEC: uPD720120
http://www.necel.com/home.nsf/Main?ReadForm&Multimedia+Products

Philips: ISP1501
http://www.semiconductors.philips.com/pip/isp1501-01/

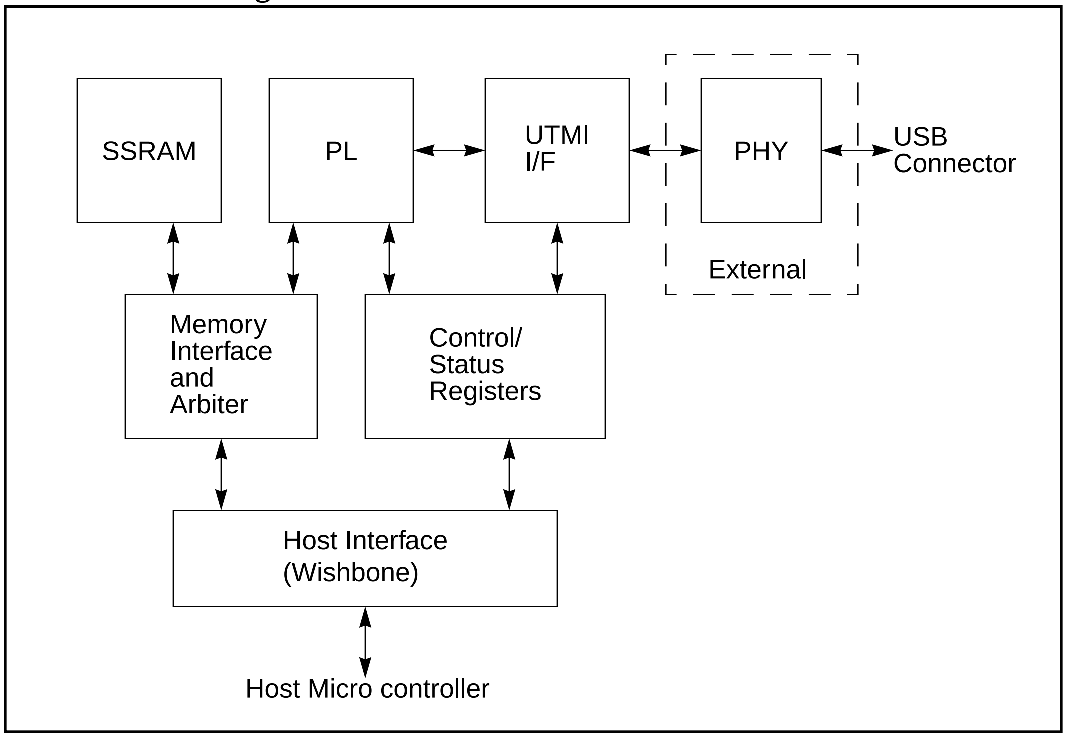
This are all that I have found. If you know of other please email me: rudi@asics.ws

2. Architecture

This section describes the internal architecture of the USB Function Controller.

Below figure illustrates the overall architecture of the core. The host micro controller interface provides a bridge between the internal data FIFO and control registers to the hosts micro controller (or CPU). The data FIFO and control registers interface to the Protocol Layer block (PL). The protocol layer interfaces to serial interface block (SEI). The SEI interfaces to the physical interface block (PHY). Each of the blocks is described in detail below.

Figure 1: Core Architecture Overview


SSRAM
PL
UTMI I/F
PHY
USB Connector
External
Memory Interface and Arbiter
Control/ Status Registers
Host Interface (Wishbone)
Host Micro controller

2.1. Clocks

The USB core has two clock domains. The UTMI interface block, runs of the clock provided by the PHY. The maximum clock output from the PHY is 60 MHz. The actual clock frequency depends on the operation mode (High Speed/Full Speed). The UTMI block includes synchronization logic to the rest of the USB core.
All other blocks run of the clock from the host interface. The goal is that the minimum frequency is at least 100Mhz.

2.2. Host Interface

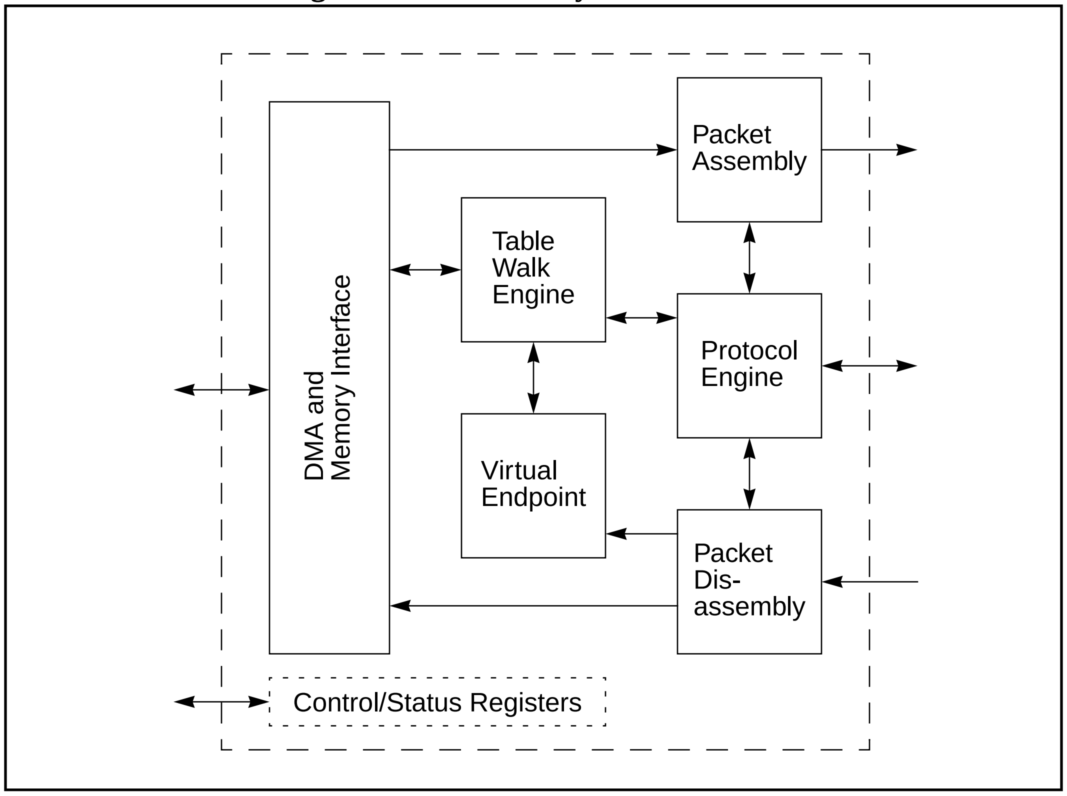
The host interface blocks provides a consistent core interface between the internal functions of the core and the function specific host or micro controller.

The host interface is WISHBONE SoC bus specification Rev. B compliant.

2.3. Protocol Layer (PL)

The protocol layer is responsible for all USB data IO and control communications.

Figure 2: Protocol Layer Block



2.3.1. DMA & Memory Interface

This block interfaces to the shared Memory. It provides random memory access and also DMA block transfers.

2.3.2. Table Walk Engine

Whenever a IN or OUT token is received, this block searches the endpoint list for the appropriate endpoint. If the endpoint is found the data packet is acknowledged or data is send out. If not no action is taken.

2.3.3. Virtual Endpoint

The virtual endpoint temporarily holds the configuration data from the endpoint found in the list or provides default endpoint configuration.

2.3.4. Protocol Engine

This block handles all the standard USB protocol handshakes and control correspondence.

2.3.5. Packet Assembly

This block assembles packets and places them in to the output FIFO.

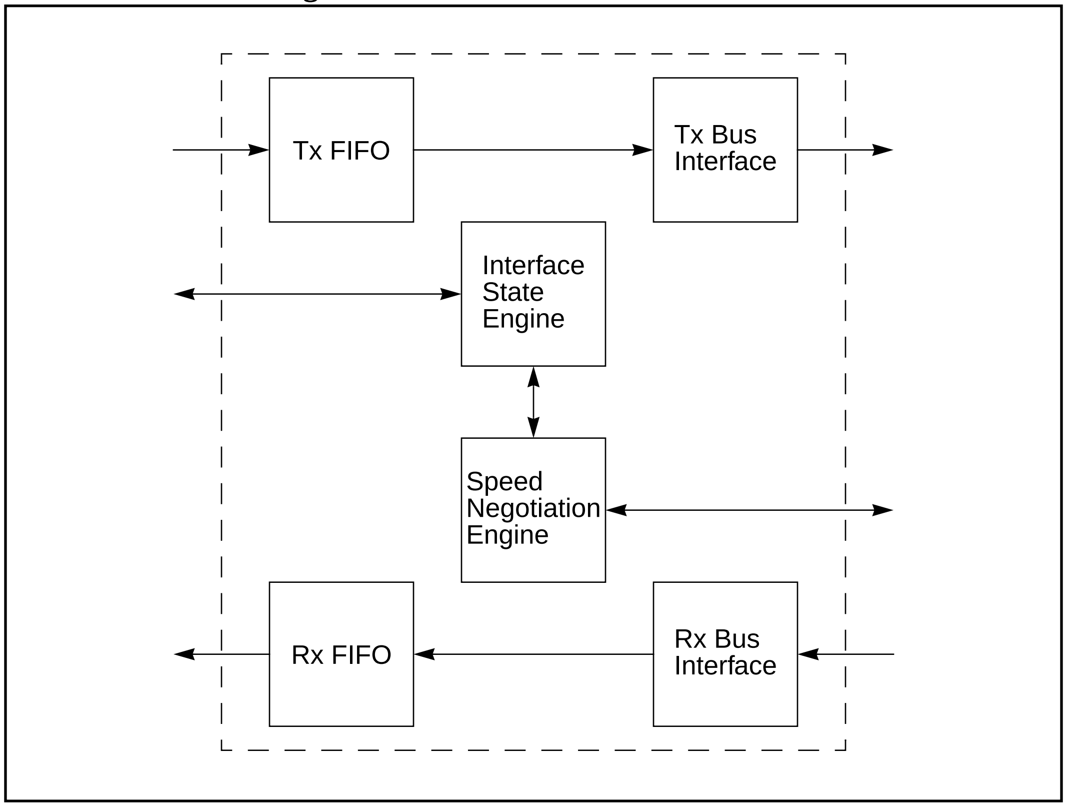
2.3.6. Packet Disassembly

This block decodes all incoming packets and forwards the decoded data to the appropriate blocks.

2.4. UTMI I/F

This is the interface block to the UTMI compliant PHY (transceiver).

Figure 3: UTMI Interface Block



2.4.1. Interface State Engine

This block handles the interface state. It controls suspend/resume modes and Full Speed/High Speed switching.

2.4.2. Speed Negotiation Engine

This block negotiates the speed of the USB interface and handles suspend and reset detection.

2.4.3. Rx & Tx FIFOs

The FIFOs hold the temporary receive and transmit data. They also used as a means to synchronize between the UTMI clock and the host interface clock.

2.4.4. Rx & Tx Bus Interface

This blocks ensure proper handshaking with the receive and transmit interfaces of the PHY.

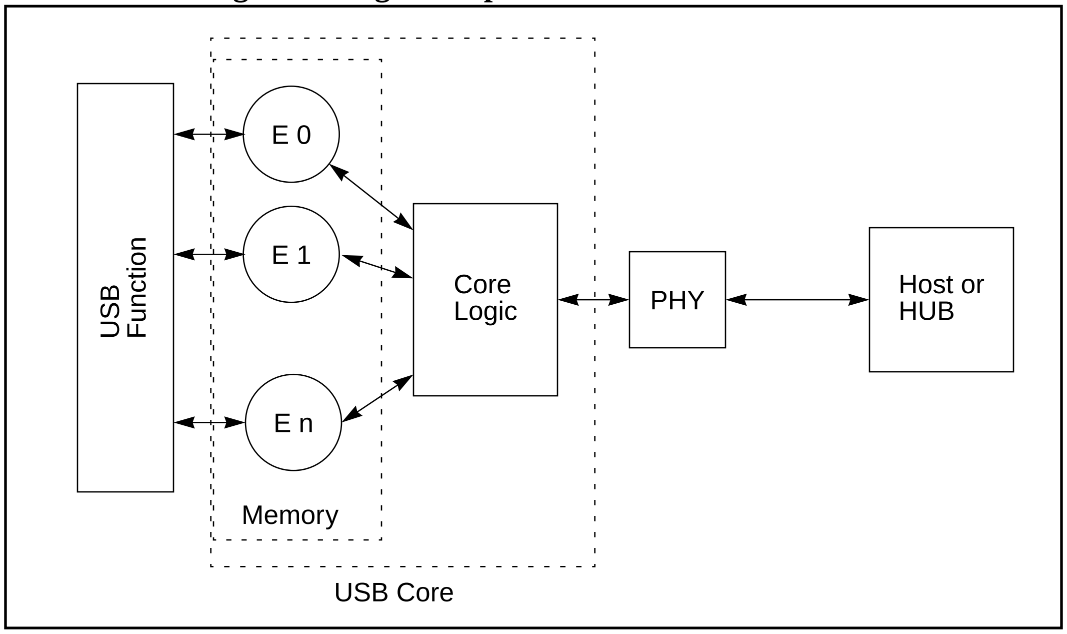
3. Operation

This section describes the operation of the USB function controller. It first discusses the logical interface to the host micro controller (function) and the logical USB interface.

The USB core uses a local configuration memory which is used to define endpoints and as a temporary data storage. The memory size is user definable, and can be used in three ways:

1) As a shared buffer between all end points. In this configuration, the USB controller will send an interrupt to the controller to fill/empty the buffer when needed for each endpoint.
2) Each endpoint has it's own dedicated input/output buffer. No software intervention is needed when different endpoints are accessed. Multiple buffer may be set up, allowing for automatic transmission of larger frames.
3) Any combination of 1) and 2)

Figure 4: Logical Representation of USB

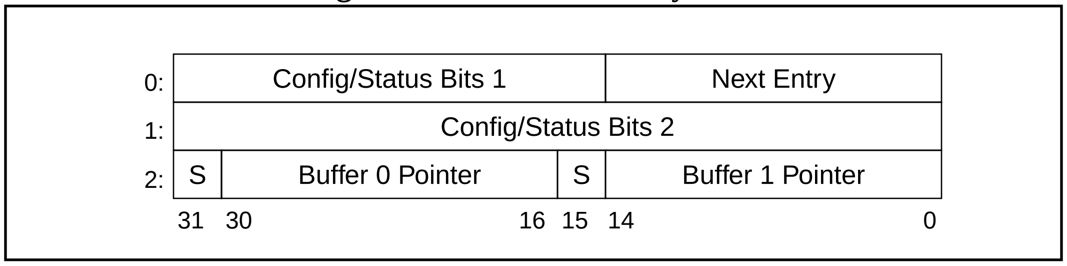


3.1. Endpoints

This USB core supports up to 15 input endpoints and 15 output endpoints (maximum number of endpoint per USB 2.0 specification). Each endpoint is defined by creating an entry in a linked list in the controllers memory. The TBD register points to the beginning of the linked list. The USB core will search the entire list for the desired endpoint each time it receives a token addressing a specific endpoint, if the desired endpoint is not found the default endpoint configuration is used.

Each entry in the linked list consists of the following fields:

Figure 5: Linked List Entry



3.1.1. Linked List Entry

The configuration and status bits specify the operation mode of the end point and the endpoint ID, as well as reporting any specific endpoint status back to the controller.

Table 1: Config Status Bits 1

Bit #	No of Bits	Description
31-28	4	Endpoint Number
27	1	1: Control endpoint; 0: Data Endpoint
26	1	1: Input Endpoint 0: Output endpoint (This bit has no meaning if bit 27 is 1.
25:24	2	Transfer Type 00: Reserved 01: Interrupt 10: Isochronous 11: Bulk
23		
22		
19		
18		
17:16	2	Interrupt Enable 00: Interrupts disabled 01: Assert Interrupt when a transfer has completed (successful or error) 10: Assert interrupt only on errors 11: Assert Interrupt only on fatal errors (HALT condition)
15	1	RESERVED

Table 1: Config Status Bits 1

Bit #	No of Bits	Description
14:0	15	Next Entry Pointer This pointer, points to the next entry in the endpoint linked list. If the value is all ones (7FFFh), then there are no more entries in the list.

Table 2: Config Status Bits 2

Bit #	No of Bits	Description
12:11	2	Number of transactions per micro frame
10:0	11	Maximum payload size

Table 3: Buffer Pointers

Bit #	No of Bits	Description
31		Shared Bit 0
30-16		Buffer 0 pointer
15		Shared bit 1
14:0		Buffer 1 pointer

3.1.2. Buffer Pointers

The buffer pointers point to the input/output data structure in memory. If the S bit is set, this indicates that the buffer is a shared buffer. In this case a interrupt is generated and the core waits for the controller to clear the TBD bit in the TBD register. Clearing the TBD bit, will cause the USB core to perform the transmit/receive operation. A value of all ones (7FFFh) indicates that the buffer has not been allocated. If both buffer are not allocated the core will respond with TBD acknowledgments to the USB host.

This USB core supports double buffering feature which reduces the latency requirements on the functions micro controller and driver software. Double buffering is enabled when both buffer pointers have been set. Data is being retrieved/ filled to/from the buffers in a ping-pong fashion. When data is sent to/from the endpoint, first buffer 0 is used. When the first transfer completes, the function controller is notified via an interrupt. The function controller can refill/empty buffer 0 now. The USB core will use buffer 1 for the next operation. When the second operation completes, the function controller is interrupted, and the USB core will use buffer 0 again, and so on.
<buffer usage for control endpoints> TBD

3.2. Default Endpoint

The default endpoint specifies the behavior of the USB core for unspecified endpoints.
<How to configure and use> TBD

3.3. USB core memory size

This USB core, includes a memory block that is uses for storing data and endpoint control information. The memory is 32 bits wide. Depending on the application, the user should chose the appropriate memory size based on the following:

- **Linked List**
  For each endpoint 12 bytes are required in the linked list. If only one endpoint is used, the default endpoint registers can be used for control and status of the endpoint.

- **Data buffer**
  USB packet sizes range from 64 to 1024 bytes, depending on transfer mode, and speed of the interface.
- **Control & Configuration Structure**
  Endpoint 0 is the default control and configuration structure. The minimum size for this structure is XX bytes.

Based on the above information, using shared buffers, the memory can be as small as 1024 bytes. The maximum supported memory size is 128 Kilobytes.

4. Core Configuration and Status Registers

This section describes all control and status register inside the USB function. The *Address* field indicates a relative address in hexadecimal. *Width* specifies the number of bits in the register, and *Access* specifies the valid access types to that register. Where RW stands for read and write access, RO for read only access. A ‘C’ appended to RW or RO, indicates that some or all of the bits are cleared after a read.

Table 4: Control/Status Registers

Name	Addr.	Width	Access	Description
CSR	0	32	RWC	Control/Status Register
FA	1	7	RO	Function Address
ELL	2	15	RW	Pointer to the beginning of the endpoint linked list in memory
DE_CS0	3	32	RW	Default endpoint Configuration Status Register 0
DE_CS1	4	32	R	Default endpoint Configuration Status Register 1
INT_MSKA	5	16	RW	Interrupt Mask for interrupt A (int_a)
INT_MSKB	6	16	RW	Interrupt Mask for interrupt B (int_b)
INT_SRC	7	16	RO	Interrupt Source register
	8		RW	
	9		RW	
	A		RW	

4.1. Control Status Register (CSR)

This is the main configuration and status register of the USB core.

Table 5: CSR Register

Bit #	Access	Description
5		
4		
3		
2		

Table 5: CSR Register

Bit #	Access	Description
1		
0		

Reset Value:

CSR: TBD h

4.2. Function Address Register (FA)

The function address is set by the host when the function is configured.

Reset Value:

FA: 00h

4.3. Endpoint Linked List Register (EEL)

The endpoint linked list register points to the beginning of the linked list in the local memory. When this register is all ones (7fff) it means that the linked list has not been set up.

Reset Value:

ELL: 7FFFh

4.4. Default Endpoint Control Status Registers (DE_CS)

This two registers may be used to specify a default endpoint behavior.

Table 6: DE_CS0 Register

Bit #	Access	Description
31	RW	
21	RW	
20	RW	
19	RW	

Table 6: DE_CS0 Register

Bit #	Access	Description
18	RW	
17	RW	
16	RW	
15-0	RO	RESERVED (Software should always write 0)

Table 7: DE_CS1 Register

Bit #	Access	Description
31	RW	Shared Bit 0
30:16	RW	Buffer 0 Pointer
15	RW	Shared Bit 1
14:0	RW	Buffer 1 Pointer

Reset Value:

DE_CS0: TBD h
DE_CS1: FFFFFFFF h

4.5. Interrupt Mask Registers (INT_MSK)

The interrupt mask registers defines the functionality of int_a and int_b outputs.

Table 8: Interrupt Mask Register

Bit #	Access	Description
7	RW	
6	RW	
5	RW	
4	RW	
3	RW	
2	RW	

Table 8: Interrupt Mask Register

Bit #	Access	Description
1	RW	
0	RW	

Reset Value:

INT_MSKA: 00h
INT_MSKB: 00h

4.6. Interrupt Source Register (INR_SRC)

TBD

Table 9: Interrupt mask Register

Bit #	Access	Description
	RW	
	RW	
	RW	
	RW	
7	RW	
6	RW	
5:4	RO	Buffer Number
3:0	RO	Endpoint Number

Note:

It is essential that the function controller reads the value of the INT_SRC register immediately when it receives and interrupt. Failure to do so, may resulting loosing an interrupt source.

5. IOs

This section lists all IOs of the USB core. Each clock domain is contained in a separate subsection

5.1. Host Interface IOs

The host interface is a WISHBONE Rev B compliant interface. This USB core works as a slave device only. When it need the intervention of the local micro controller, it will assert INTA_O or INTB_O.

Table 10: Host Interface (WISHBONE)

Name	Width	Dir ection	Description
CLK_I	1	I	Clock input
RST_I	1	I	Reset Input
CS_I	1	I	Core Select (From Address decoder)
ADDR_I	15	I	Address Input
DATA_I	32	I	Data Input
DATA_O	32	O	Data Output
SEL_I	4	I	Indicates which bytes are valid on data bus. Whenever this signal is not fh during a valid access, the ERR_O is asserted.
ACK_O	1	O	Acknowledgment Output. Indicates a normal Cycle termination.
ERR_O	1	O	Error acknowledgment output. Indicates an abnormal cycle termination.
RTY_O	1	O	Retry Output. Indicates that the interface is not ready, and the master should retry this operation. [Most Likely not used in this core.]
WE_I	1	I	Indicates a Write Cycle when asserted high.
STB_I	1	I	Indicates t6he beginning of a valid transfer cycle.
INTA_O	1	O	Interrupt Output A
INTB_O	1	O	Interrupt Output A

5.2. UTMI IOs

The UTMI interface is a USB 2.0 UTMI specification Version 1.03 compliant interface.

Table 11: UTMI Interface

Name	**Width**	**Dir ection**	**Description**
clock	1	I	Clock