USBFC
(USB Function Controller)

EIFUFAL501

User's Manual

Doc #: 88-02-E01
Revision: 2.0
Date: 03/24/98

SEIKO EPSON CORPORATION

USB Function Controller (USBFC)

1. Highlights

1.1 Features

- USB Specification Version 1.0 Compliant
- Bridges between a Processor-Independent local bus and a USB bus
- USB device bandwidth of up to 12Mb/sec
- USB Bulk, Isochronous, Interrupt, and Control transfers.
- Independent 64 byte transmit and receive FIFOs to maximize throughput.
- Supports local CPU or DMA data transfers.
- 3.3V operating voltage

1.2 Overview

The USB Function Controller (USBFC) allows bulk or isochronous data transfers between a generic local bus and a Universal Serial Bus (USB). The USBFC supports the connection between a host computer and an intelligent peripheral such as a digital camera or scanner.

The three main components of the USBFC are the USB Bus Interface, the dual 64 byte FIFOs, and a Local Bus Interface.

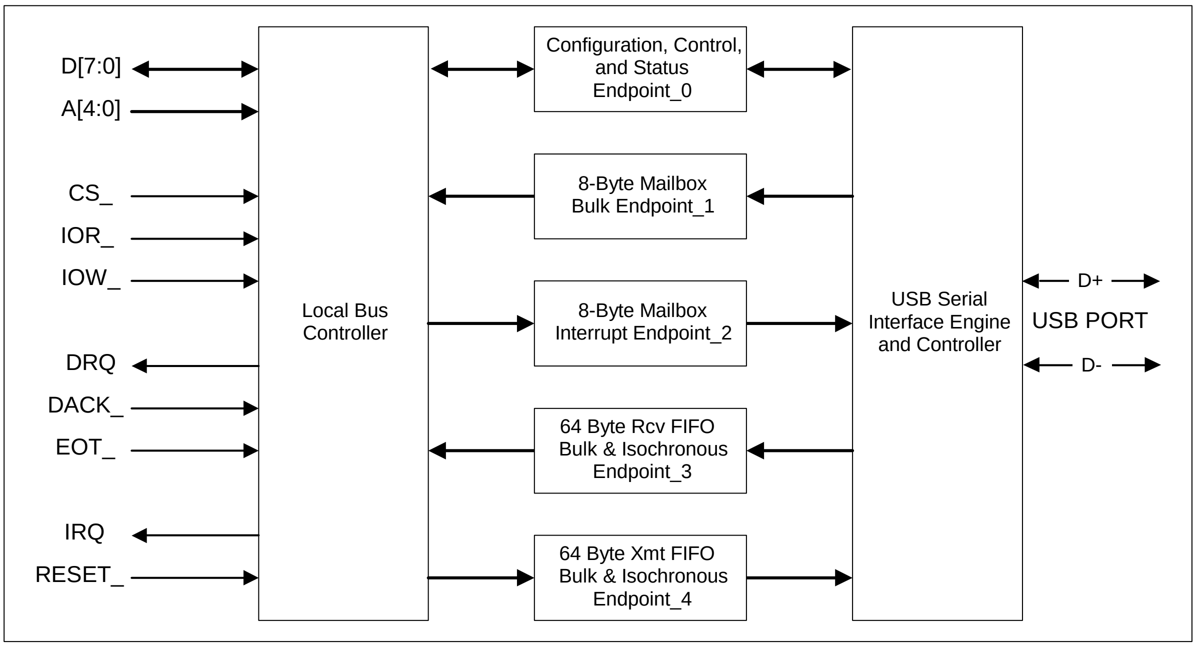
The USB Interface is responsible for the following functions:

- Host to device Communication
- Bulk or isochronous endpoints to access FIFOs
- Interrupt endpoint to access Local to USB Mailboxes
- Bulk endpoint to access USB to Local Mailboxes

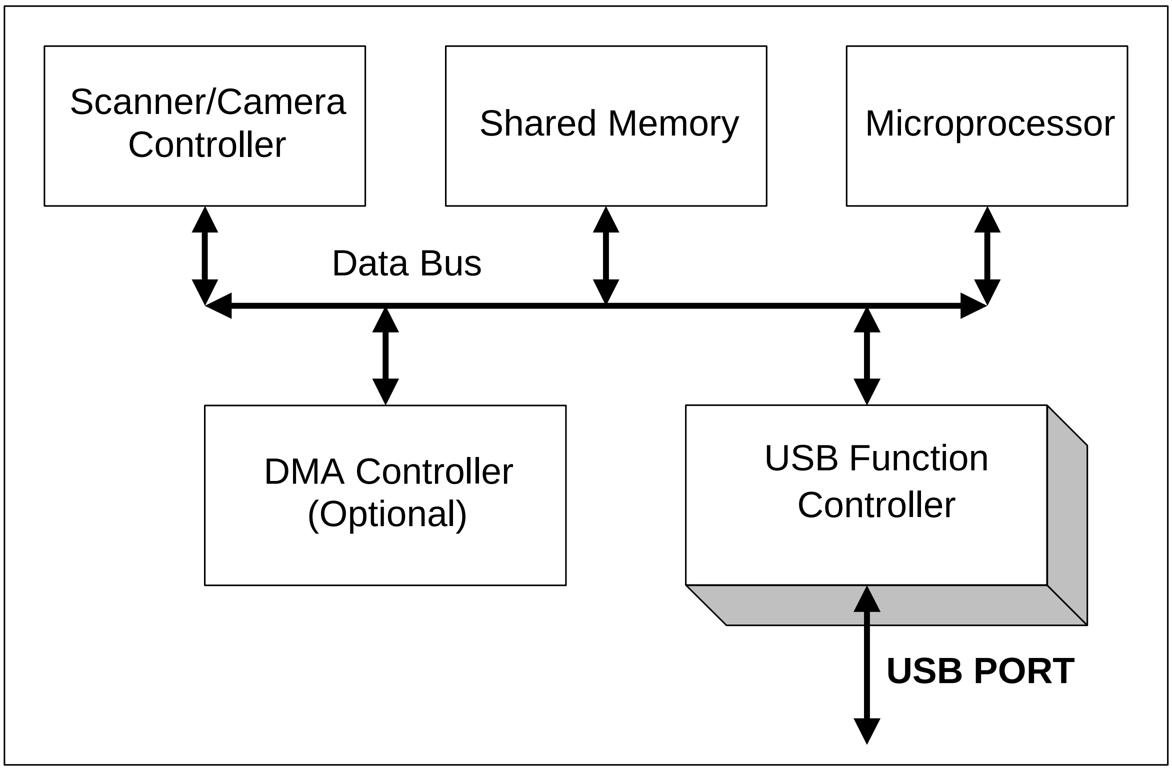
The Local Bus Interface is responsible for the following functions:

- FIFO Control
- Local CPU interface
- Local DMA controller interface
- Interrupts

1.3 USBFC Block Diagram

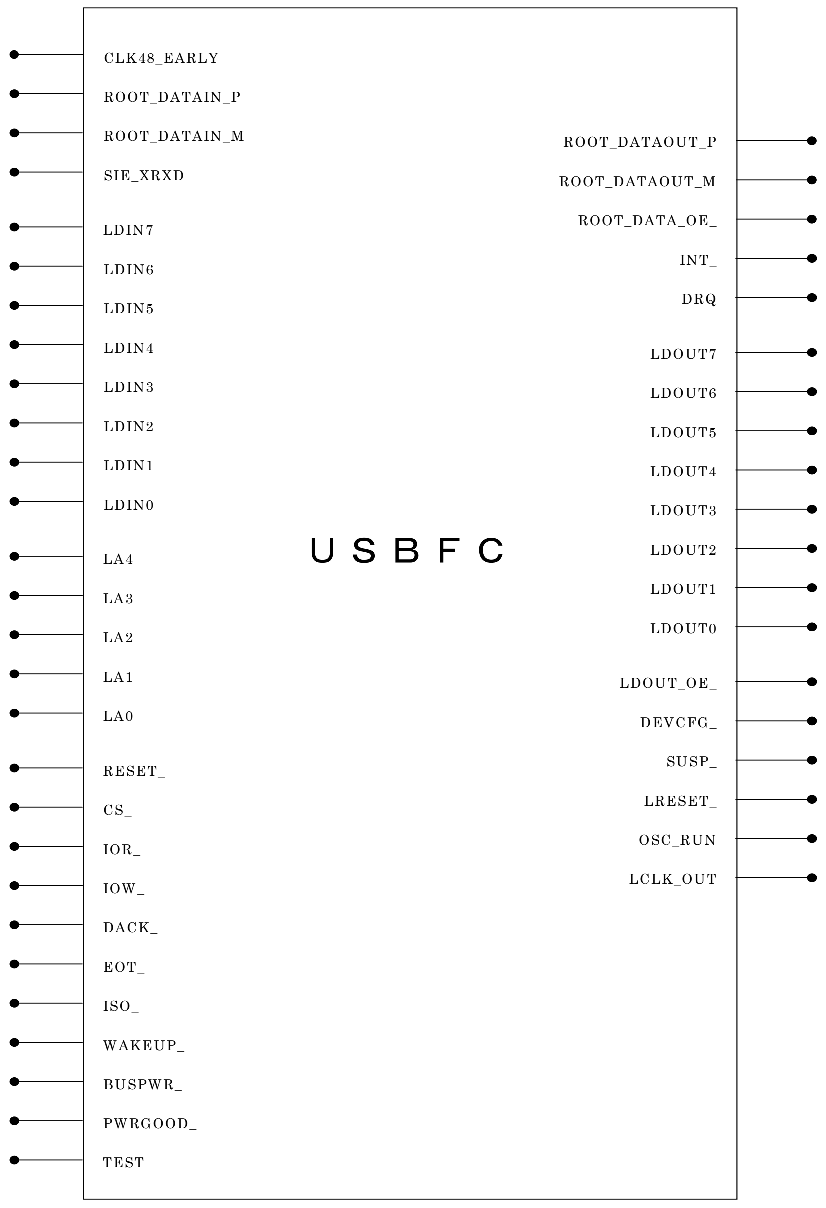

D[7:0]
A[4:0]
CS_
IOR_
IOW_
DRQ
DACK_
EOT_
IRQ
RESET_
Local Bus Controller
Configuration, Control, and Status Endpoint_0
8-Byte Mailbox Bulk Endpoint_1
8-Byte Mailbox Interrupt Endpoint_2
64 Byte Rcv FIFO Bulk & Isochronous Endpoint_3
64 Byte Xmt FIFO Bulk & Isochronous Endpoint_4
USB Serial Interface Engine and Controller
D+
USB PORT
D-

1.4 USBFC Typical System Block Diagram



2. Signal Description

2.1 Symbol Diagram for USBFC



2.2 Signal Description

NOTE: Input signal must be driven externally when the USBFC is in the suspended state.
The signal which has underscore (_) at the end of its name is active low.

Signal Name	Type	Description
CLK48_EARLY	Input	**48 MHz Oscillator input.**
RESET_	Input	**Reset.** External reset. Connect to local or power-on reset. To reset when oscillator is stopped (initial power-up or in suspend state), assert for at least one ms. When oscillator is running, assert for at least five 48-MHz clock periods.
ROOT_DATAIN_P, ROOT_DATAIN_M	Input	**USB Data Input.** Two differential input signals (DP & DM) of the USB port.
SIE_XRXD	Input	**USB Data Differential Receiver.** Amplified input signal extracted from the two incoming differential input signals of the USB data port. DP > DM : 1, DP < DM : 0
ROOT_DATAOUT_P, ROOT_DATAOUT_M	Output	**USB Data Output.** Two differential output signals (DP & DM) of the USB port.
ROOT_DATA_OE_	Output	**USB Port Output Enable.** This active low output is true when USBFC is driving the USB port data output lines (ROOT_DATAOUT_P & ROOT_DATAOUT_M).
LDIN[7:0]	Input	**Data Input.** LDIN7 is the most-significant bit.
LDOUT[7:0]	Output	**Data Output.** LDOUT7 is the most-significant bit.
LDOUT_OE_	Output	**Data Output Enable.** This active low output is true when the USBFC is driving the Data Output Bus (LDOUT).
LA[4:0]	Input	**Address Bus.** The local address bus is used by devices on the local bus to select registers within the USBFC.
CS_	Input	**Chip Select.** The chip select is used by devices on the local bus to enable access to registers within the USBFC. This signal is ignored if DACK_ is asserted.
IOR_	Input	**I/O Read.** The I/O read strobe is asserted along with CS_ and LA[4:0] when a device on the local bus reads from an internal register or the FIFO. It also allows the FIFO to be read during DMA transfers when DACK_ is asserted.
IOW_	Input	**I/O Write.** The I/O write strobe is asserted along with CS_ and LA[4:0] when a device on the local bus writes to an internal register or the FIFO. It also allows the FIFO to be written during DMA transfers when DACK_ is asserted.
DRQ	Output	**DMA Request.** This signal indicates to an external DMA controller that a byte should be transferred to/from the FIFO. During a transfer, DRQ remains asserted until the DACK_ input goes active.
DACK_	Input	**DMA Acknowledge.** This signal from the external DMA controller is used to transfer data to/from the FIFO in response to

		DRQ. IOR_ and IOW_ determine the direction of the DMA transfer.
EOT_	Input	**DMA End of Transfer.** This signal from the external DMA controller is used to terminate a DMA transfer. If it is asserted during a DMA cycle, the current byte will be transferred, but no additional bytes will be requested. EOT_ can be programmed to cause a USB interrupt.
INT_	Output	**Interrupt Request Output.** The interrupt request output is used to interrupt a processor on the local bus. There are several sources of this interrupt which are described in the Register Description Section.
DEVCFG_	Output	**Device Configured.** This active low output is true when the USBFC has been configured by the USB host.
SUSP_	Output	**Device Suspended.** This active low output is true when the USBFC has been suspended by the USB host.
OSC_RUN	Output	**Oscillator Run.** Clock input to CLK48_EARLY shoud be enabled depend on OSC_RUN.
LCLK_OUT	Output	**Local Clock.** This is a buffered output from the internal 48 MHz oscillator. This signal is not driven while the device is suspended.
LRESET_	Output	**Local Reset.** This active low output is asserted when either the RESET_ pin is asserted, or a USB port reset is detected.
ISO_	Input	**Isochronous Mode Select.** This active low input selects isochronous mode for USB transfers to and from the FIFOs.
WAKEUP_	Input	**Wakeup.** This active low input causes the USBFC to perform a USB remote wakeup.
BUSPWR_	Input	**Bus Powered.** This active low input indicates that the logic external to the USBFC is powered by the USB bus. If this input is high, then the external logic is self-powered.
PWRGOOD_	Input	**Power Good.** This active low input indicates that an external power supply used for self-powered mode is operational.
TEST	Input	**Test.** For normal operation, connect this pin to ground.

3. Functional Description

3.1 USB Interface

The USBFC is a USB function device, and as a result is always a slave to the USB host. All USB data transfers to and from the USBFC USB port are initiated by the USB host. There are five USB endpoints associated with the USBFC:

- Endpoint 0. This control endpoint is used to initialize the device, and provides access to USB configuration, control and status registers.
- Endpoint 1. This endpoint supports bulk transfers from the USB host to the USBFC receive mailboxes.
- Endpoint 2. This endpoint supports interrupt transfers from the USBFC transmit mailboxes to the USB host.
- Endpoint 3. This endpoint supports bulk or isochronous data transfers from the USB host to the USBFC Receive FIFO.
- Endpoint 4. This endpoint supports bulk or isochronous data transfers from the USBFC Transmit FIFO to the USB host.

3.2 Local Bus

Bulk or isochronous data passes between the local bus and the USB bus through a pair of 64 byte FIFOs. A CPU on the local bus can provide data to or accept data from the USB bus via the FIFOs in the USBFC. A pair of 8-byte mailbox registers provide a means for the local and host CPUs to exchange messages. The receive mailbox is implemented as a Bulk Data endpoint, and the transmit mailbox as an Interrupt endpoint.

3.2.1 CPU Controlled USB to Local Bus Transfers

For host to device transfers, the local and host CPUs first arrange to transfer a block of data from host memory to local shared memory. The USB host performs a bulk or isochronous data transfer over the USB bus to the receive FIFO in the USBFC.

If the FIFO fills up during a bulk transfer, the USBFC will return a USB NAK acknowledge to the host, signaling that the data could not be accepted. Packet data which is in the FIFO or has already been read by the CPU should be discarded.

If the local CPU has stalled this endpoint, the USBFC will not store any data into the FIFO, and will respond with a STALL acknowledge.

The local CPU can either start polling for valid FIFO data immediately after setting up the transfer with the USB host, or can wait for a packet complete interrupt. As the FIFO is filling up from the USB side, the local CPU can poll the FIFO status register to determine when a byte is available. Otherwise it can wait until the packet complete interrupt and read the entire packet at once.

Once an end of packet occurs, an interrupt can be generated to the local CPU. The local CPU can read a status port to detect whether the packet was acknowledged with an ACK, NAK, or STALL. If none of these acknowledge bits are set, then a timeout has occurred. For NAK or timeout conditions at the completion of bulk transfers, the USB host will send another OUT token, and the USBFC should receive the same packet again.

3.2.2 CPU Controlled Local Bus to USB Transfers

For device to host transfers, the local CPU first writes the data block from local memory into the transmit FIFO. While writing data into the USBFC, the local CPU must keep track of whether there is space available in the FIFO by monitoring the Transmit FIFO Count register.

After the block has been loaded into the transmit FIFO, the local and host CPUs arrange to transfer the block of data from the transmit FIFO to host memory. The USB host sends an IN token to the USBFC and starts a USB bulk or isochronous read from the transmit FIFO. The CPU writes and USB read operations could occur concurrently if the local CPU can provide data at a fast enough rate to keep up with the USB bus.

When the transmit FIFO becomes empty, the USBFC will terminate the packet with an EOP (end of packet), signaling that there is no more data available. Once an end of packet occurs, an interrupt can be generated to the local CPU. The local CPU can read a status port to detect whether the packet was acknowledged with an ACK from the host, or whether the USBFC responded to the IN token with a NAK or STALL. If none of these acknowledge bits are set, then a timeout has occurred. For NAK or timeout conditions at the completion of bulk transfers, the USB host will send another IN token, and the USBFC should re-transmit the same packet.

3.2.3 DMA Controlled USB to Local Bus Transfers

A Direct Memory Access (DMA) controller may be used on the local bus to transfer data to and from the USBFC. For host to device transfers, the local and host CPUs first arrange to transfer a block of data from host memory to local shared memory. The local CPU then programs the DMA controller for fly-by demand mode transfers. In this mode, transfers occur only when the USBFC requests them, and the data is read from the USBFC receive FIFO and written into local memory during the same bus transaction. The DMA address counter is programmed to point to the destination memory block in local shared memory, and the byte count to the number of bytes in the block to be transferred.

After the DMA controller has been programmed, the DMA request enable bit is set in the USBFC. The USB host performs USB bulk or isochronous data transfers over the USB bus to the receive FIFO in the USBFC. If the FIFO fills up during a bulk transfer, the USBFC will return a USB NAK acknowledge to the host, signaling that the data could not be accepted. Packet data which is in the FIFO or has already been transferred by the DMA should be discarded.

If the local CPU has stalled this endpoint, the USBFC will not store any data into the FIFO, and will respond with a STALL acknowledge.

As long as there is data available in the FIFO, the USBFC will request local DMA transfers by asserting DRQ. The DMA controller then requests the local bus from the local CPU. After the DMA controller has been granted the bus, it drives a valid memory address and asserts DACK_, IOR_, and MEMW_, thus transferring a byte from the USBFC receive FIFO to memory. This process continues until the DMA byte count reaches zero. A local bus interrupt may be programmed to occur when the DMA has finished.

Once an end of packet occurs, an interrupt can be generated to the local CPU. The local CPU can read a status port to detect whether the packet was acknowledged with an ACK, NAK, or STALL. If none of these acknowledge bits are set, then a timeout has occurred. For NAK or timeout conditions at the completion of bulk transfers, the USB host will send another OUT token, and the USBFC should receive the same packet again.

An early end-of-packet (EOP) can be detected by the local CPU if the DMA count is non-zero. The local and host CPUs should then decide how to proceed.

3.2.4 DMA Controlled Local Bus to USB Transfers

For device to host transfers, the local and host CPUs first arrange to transfer a block of data from local memory to host memory. The local CPU then programs the DMA controller for fly-by demand mode transfers. In this mode, transfers occur only when the USBFC requests them, and the data is read from local memory and written into the USBFC FIFO during the same bus transaction. The DMA address counter is programmed to point to the source memory block in local memory, and the byte count to the number of bytes in the block to be transferred. After the DMA controller has been programmed, the DMA request enable bit is set in the USBFC. As long as there is space available in the FIFO and the byte count is non-zero, the USBFC will request DMA transfers by asserting DRQ. The DMA controller then requests the local bus from the local CPU. After the DMA controller has been granted the bus, it drives a valid memory address and asserts DACK_, MEMR_, and IOW_, thus transferring a byte from memory to the USBFC Transmit FIFO.

After the DMA has been started, the local CPU can signal the USB host to start a bulk read using endpoint 2. Isochronous packets occur at pre-arranged intervals, so no signaling is required. The USB host sends an IN token to the USBFC and starts a USB bulk or isochronous data transfer from the transmit FIFO. The DMA transfers continue until the DMA byte count reaches zero. An interrupt can be generated to the local CPU when the DMA has finished.

When the transmit FIFO becomes empty, the USBFC will terminate the packet with an EOP (end of packet), signaling that there is no more data available. Once an end of packet occurs, an interrupt can be generated to the local CPU. The local CPU can read a status port to detect whether the packet was acknowledged with an ACK from the host, or whether the USBFC responded to the IN token with a NAK or STALL. If none of these acknowledge bits are set, then a timeout has occurred. For NAK or timeout conditions at the completion of bulk transfers, the USB host will send another IN token, and the USBFC should re-transmit the same packet.

3.2.5 Terminating DMA Transfers

The EOT_ signal is used to halt a DMA transfer, and is typically provided by an external DMA controller. It should be asserted while DACK_ and IOR_ or IOW_ are simultaneously active to indicate that DMA activity has stopped. Although an EOT_ signal indicates that DMA has terminated, the USB transfer is not complete until the last byte has been transferred from the FIFO to the USB bus. The EOT_ resets the USBFC DMA request enable bit.

If no EOT_ signal is provided by the DMA controller, the DMA transfer can be halted at any time by resetting the USBFC DMA request enable bit. If the USBFC DMA request enable bit is cleared during the middle of a DMA cycle, the current cycle will complete before DMA requests are terminated.

3.2.6 USB Endpoint 1 Receive Mailboxes

Endpoint 1 is used for bulk transfers from the USB host to a set of 8 receive mailbox registers which are read by the local CPU. The format of the data written to the mailbox is user defined. To transfer an 8 byte packet, the host first performs a USB 8-byte bulk transfer to the endpoint 1 receive mailbox registers. A receive mailbox valid bit (bit 1 of register **IRQSTAT1**) is then set which can cause a local bus interrupt. If the USB host tries to write to these registers when the valid bit is set, a NAK acknowledge will be returned. When the local CPU receives the interrupt, it reads the 8 bytes and clears the valid bit.. The USB host can then send another 8-byte packet to this endpoint. An index pointer is used to access the receive mailboxes. It must be initialized by the local CPU, and is automatically incremented after the local CPU reads the receive mailbox data register

3.2.7 USB Endpoint 2 Transmit Mailboxes

Endpoint 2 is used for interrupt transfers to the USB host from a set of 8 transmit mailbox registers which are written by the local CPU. To transfer an 8 byte packet, the local CPU writes data into the 8 registers and sets the transmit mailbox valid bit. The host performs a USB 8-byte interrupt transfer from the endpoint 2 transmit mailbox registers. After the USB interrupt transfer has completed, the transmit mailbox valid bit is cleared. The CPU should only write to the transmit mailbox registers when the valid bit is not set. This guarantees that a previous interrupt transfer has completed before the register values are changed. If the USB host tries to read endpoint 2 when the valid bit is not set, a NAK acknowledge is returned. An index pointer is used to access the transmit mailboxes. It must be initialized by the local CPU, and is automatically incremented after the local CPU reads or writes the transmit mailbox data register

3.3 Suspend Mode

When there is a three millisecond period of inactivity on the USB, the USB specification requires a device to enter into a low-power suspended state. The device may not draw more than 500 µA of current while in this state. To facilitate this, the USBFC provides a suspend - request interrupt and a suspend bit in the USB status register. Additionally, the USBFC allows the local CPU to send a "device remote wake-up" request -- a request from the local CPU to wake-up the USB.

3.3.1 The Suspend Sequence

The typical sequence of operation is as follows: during device configuration, the local CPU will configure the USBFC to interrupt on a suspend request (using the **IRQENB1** register). When the USB is idle for three milliseconds, the USBFC will generate a suspend - request interrupt.

The local CPU accepts this interrupt by clearing the corresponding bit in the **IRQSTAT1** register, and performs the tasks required to ensure that not more than 500 µA of current is drawn from the USB power bus. Then it writes a 1 to bit 7 of the USB status register (**USBSTAT**) to initiate the suspend.

In suspend mode, OSC_RUN output pin goes low, and therefore the clock input into CLK48_EARLY shoud be stopped. The SUSP_ output pin will be low while the device is in the suspended state. Note that input pins on the USBFC should not be allowed to float during suspend mode. The USBFC will leave suspend mode by detecting traffic on the USB bus or by a device remote wake-up from the local CPU.

3.3.2 Device-Remote Wake-Up

The local CPU signals a device remote wake-up by driving the WAKEUP_ input pin low. The USBFC will send a 10-ms wake-up signal to the USB host, and drive OSC_RUN high and restart the clock input into CLK48_EARLY. Two milliseconds after the WAKEUP_ pin is asserted, the SUSP_ line is driven high to indicate that the USBFC has completed its wake-up.

3.3.3 Host-Initiated Wake-Up

The host may wake-up the USBFC by any non-idle state on the USB. The USBFC will detect the host's wake-up request, and drive OSC_RUN high and restart the clock input into CLK48_EARLY. Two milliseconds later, the SUSP_ output signal is driven high to indicate that the USBFC has completed its wake-up.

3.4 USBFC Power Configuration

The USB specification defines both bus-powered and self-powered devices. A *bus-powered* device is a peripheral which derives all of its power from the upstream USB connector, while a *self-powered* device has an external power supply. The USBFC is well-suited for both types of applications.

The most significant consideration when deciding whether to build a bus-powered or a self-powered device is power consumption. The USB specification lays out the following requirements for maximum current draw:

- A peripheral not configured by the host (signified on the USBFC by the DEVCFG_ output pin) can draw only 100 mA from the USB power pins.
- A device may not draw more than 500 mA from the USB connector's power pins.
- In suspend mode, the peripheral may not draw more than 500 µA from the USB connector's power pins

If these power considerations can be met without the use of an external power supply, the peripheral can be bus-powered; otherwise a self-powered design should be implemented.

In case of self-powered and also power-sensitive applications, the USBFC can be forced to enter low-power suspend mode when disconnected from the USB. Setting bit 7 of the **USBSTAT** register when USB power has been removed forces the USBFC to enter low-power suspend mode. The USBFC will automatically wake-up when the peripheral is re-connected to the USB. *Do not force suspend mode unless the peripheral is disconnected from the USB.*

4. Local Registers

4.1 Register Description

The USBFC occupies a 32 byte local register space which can be accessed by a CPU on the local bus. The Endpoint 1 Receive Mailbox Registers are written by the USB host, and the Endpoint 2 Transmit Mailbox Registers are read by the USB host. After the USBFC is powered-up or reset, the registers are set to their default values.

Writes to unused registers are ignored, and reads from unused registers return a value of 0. For compatibility with future revisions, unused bits within a register should always be written with a zero.

NOTE: The USB device and configuration descriptors cannot be read by the USB host until the USBENB bit in the DMA control register is set. Until then, the device enumeration process cannot complete, so the device will not be recognized on the USB.

4.2 Register Summary

Address	Register Name	Register Description	USB Endpoint
0	DCTL	DMA Control	
1	IRQENB1	Interrupt Enable 1	
2	IRQSTAT1	Interrupt Status 1	
3	IRQENB2	Interrupt Enable 2	
4	IRQSTAT2	Interrupt Status 2	
5-7		Reserved	
8	EP1IDX	Endpoint 1 Index Register	
9	EP1DATA	Endpoint 1 Receive Mailbox Data Port (USB to Local)	1
A-B		Reserved	
C	EP2IDX	Endpoint 2 Transmit Mailbox Index Register	
D	EP2DATA	Endpoint 2 Transmit Mailbox Data Port (Local to USB)	2
E	EP2POLL	Endpoint 2 Interrupt Polling Interval	
F		Reserved	
10	EP3DATA	Endpoint 3 Receive FIFO Data	3
11	EP3COUNT	Endpoint 3 Receive FIFO Count	
12	EP3STAT	Endpoint 3 Receive FIFO Status	
13	EP3PKSZ	Endpoint 3 Maximum Packet Size	
14	EP4DATA	Endpoint 4 Transmit FIFO Data	4
15	EP4COUNT	Endpoint 4 Transmit FIFO Count	
16	EP4STAT	Endpoint 4 Transmit FIFO Status	
17	EP4PKSZ	Endpoint 4 Maximum Packet Size	
18	REVISION	USBFC Revision	
19	USBSTAT	USB Status	
1A	FRAMEMSB	Frame Counter MSB	
1B	FRAMELSB	Frame Counter LSB	
1C	EXTIDX	Extended Register Index	
1D	EXTDATA	Extended Register Data	
1E-1F		Reserved	

4.3 (Address 00h; DCTL) DMA Control Register

Bits	Description	Read	Write	Default Value
7	**Reserved.**	Yes	No	0
6	**Software EOT.** This bit determines the response to an IN request from Endpoint 4 when the transmit FIFO is empty. If either this bit or the DMA EOT input pin are asserted, the USBFC responds to an In request from Endpoint 4 with an ACK and a zero length packet if the FIFO is empty. If neither this bit nor the DMA EOT input are asserted, the USBFC responds to an In request from Endpoint 4 with an NACK if the FIFO is empty, indicating that it expects to transmit more data. This bit can be cleared by an I/O write with a data value of 0. It is automatically cleared when the USBFC responds to the host with a zero length packet when the FIFO is empty.	Yes	Yes	0
5	**USB Enable.** Any device or configuration descriptor reads from the host will be acknowledged with a NAK until this bit is set. This allows time for the local CPU to set up the interrupt polling register, maximum packet size registers, and other configuration registers (e.g. Product ID and Vendor ID) before the host reads the descriptors.	Yes	Yes	0
4	**Endpoint 4 Stall.** If this bit is set, host bulk reads from the transmit FIFO will result in a STALL acknowledge by the USBFC. No data will be returned to the USB host.	Yes	Yes	0
3	**Endpoint 3 Stall.** If this bit is set, host bulk writes to the receive FIFO will result in a STALL acknowledge by the USBFC. Receive data will be discarded.	Yes	Yes	0
2	**DMA Request.** This status bit reflects the state of the DRQ output pin, and allows a CPU on the local bus to monitor DMA transfers.	Yes	No	0
1	**DMA Request Enable.** Writing a 1 to this bit causes the USBFC to start requesting DMA cycles from a DMA controller on the local bus. If the EOT_ input is asserted, this bit is automatically reset. A CPU on the local bus may also explicitly reset this bit to terminate a DMA transfer. This bit can be read to determine whether a DMA transfer is still in progress.	Yes	Yes	0
0	**DMA Direction.** This bit determines the direction of data flow during a DMA transfer. 1 = Local bus to USB; 0 = USB to Local bus	Yes	Yes	0

4.4 (Address 01h; IRQENB1) Interrupt Enable Register 1

Bits	Description	Read	Write	Default Value
7	**Suspend Request Interrupt Enable.** When set, this bit enables a local interrupt to be set when the USB host is requesting the USBFC to enter suspend mode.	Yes	Yes	0
6	**SOF Interrupt Enable.** When set, this bit enables a local interrupt to be set when a start-of-frame packet is received by the USBFC.	Yes	Yes	0
5	**EOT Interrupt Enable.** When set, this bit enables the local interrupt to be asserted when EOT_ signal is received from the DMA controller.	Yes	Yes	0
4	**Endpoint 4 Interrupt Enable.** When set, this bit enables a local interrupt to be set when a USB Endpoint 4 Data Packet has been sent by the USBFC.	Yes	Yes	0
3	**Endpoint 3 Interrupt Enable.** When set, this bit enables a local interrupt to be set when a USB Endpoint 3 Data Packet has been received by the USBFC.	Yes	Yes	0
2	**Endpoint 2 Interrupt Enable.** When set, this bit enables a local interrupt to be set when the USB Endpoint 2 Receive Mailbox registers have been read by the USB host.	Yes	Yes	0
1	**Endpoint 1 Interrupt Enable.** When set, this bit enables a local interrupt to be set when the USB Endpoint 1 Transmit Mailbox registers have been written to by the USB host.	Yes	Yes	0
0	**Reserved.**	Yes	No	0

4.5 (Address 02h; IRQSTAT1) Interrupt Status Register 1

NOTE: These status bits are set independently of the corresponding interrupt enable bits. Writing a 0 to these bits has no effect.

Bits	Description	Read	Write	Default Value
7	**Suspend Request Interrupt Status.** This bit indicates when a suspend-request has been received by the USBFC. This status bit is cleared by writing a 1.	Yes	Yes/CLR	0
6	**SOF Interrupt Status.** This bit indicates when a start-of-frame packet has been received by the USBFC. This status bit is cleared by writing a 1.	Yes	Yes/CLR	0
5	**EOT Interrupt Status.** This bit indicates when the EOT_ input has been asserted simultaneously with DACK_ and either IOR_ or IOW_, indicating the completion of a DMA transfer. This status bit is cleared by writing a 1.	Yes	Yes/CLR	0
4	**Endpoint 4 Interrupt Status.** This bit indicates when a USB Endpoint 4 Data packet has been sent by the USBFC. This status bit is cleared by writing a 1.	Yes	Yes/CLR	0
3	**Endpoint 3 Interrupt Status (Receive FIFO Valid).** This bit indicates when a USB Endpoint 3 Data packet has been received by the USBFC. This status bit is cleared by writing a 1.	Yes	Yes/CLR	0
2	**Endpoint 2 Interrupt Status.** This bit indicates when the USB Endpoint 2 Mailbox registers have been read by the USB host. This status bit is cleared by writing a 1.	Yes	Yes/CLR	0
1	**Endpoint 1 Interrupt Status (Receive Mailbox Valid).** This bit indicates when the USB Endpoint 1 Mailbox registers have been written to by the USB host. This status bit is cleared by writing a 1.	Yes	Yes/CLR	0
0	**Upper Interrupt Active.** At least one interrupt status bit is set in the IRQSTAT2 register.	Yes	No	0

4.6 (Address 03h; IRQENB2) Interrupt Enable Register 2

Bits	Description	Read	Write	Default Value
7:2	**Reserved.**	Yes	No	0
1	**Transmit FIFO Almost Empty Interrupt Enable.** When set, this bit enables a local interrupt to be generated when the Transmit FIFO Almost Empty status bit is set.	Yes	Yes	0
0	**Receive FIFO Almost Full Interrupt Enable.** When set, this bit enables a local interrupt to be generated when the Receive FIFO Almost Full status bit is set.	Yes	Yes	0

4.7 (Address 04h; IRQSTAT2) Interrupt Status Register 2

NOTE: These status bits are set independently of the corresponding interrupt enable bits. Writing a 0 to these bits has no effect.

Bits	Description	Read	Write	Default Value
7:2	**Reserved.**	Yes	No	0
1	**Transmit FIFO Almost Empty Status.** This bit is set when the number of bytes in the Transmit FIFO is equal to the Transmit FIFO Almost Empty Threshold, and another byte is sent to the USB bus from the FIFO. This status bit is cleared by writing a 1.	Yes	Yes/Clr	0
0	**Receive FIFO Almost Full Status.** This bit is set when the number of bytes in the Receive FIFO is equal to the Receive FIFO Almost Full Threshold, and another byte is received from the USB bus into the FIFO. This status bit is cleared by writing a 1.	Yes	Yes/Clr	0

4.8 (Address 08h; EP1IDX) Endpoint 1 Index Register

Bits	Description	Read	Write	Default Value
7:3	**Reserved.**	Yes	No	0
2:0	**Endpoint 1 Index Register.** This register determines which Endpoint 1 Receive Mailbox is accessed when the Endpoint 1 Receive Mailbox Data port is read. This register is automatically incremented after the Endpoint 1 Receive Mailbox Data port is read. This index register wraps around to zero when it reaches the maximum count.	Yes	Yes	0

4.9 (Address 09h; EP1DATA) Endpoint 1 Receive Mailbox Data

Bits	Description	Read	Write	Default Value
7:0	**Endpoint 1 Receive Mailbox Data.** This port is used to read data from one of the receive mailbox registers. Data is returned from the register selected by the Endpoint 1 Index Register. The eight receive mailbox registers are written by a USB bulk transfer to endpoint 1, and can be used to pass messages from the USB host to the local CPU. The format and content of the messages are user defined. If enabled, USB writes to this register can generate a local interrupt.	Yes	USB	0

4.10 (Address 0Ch; EP2IDX) Endpoint 2 Index Register

Bits	Description	Read	Write	Default Value
7:3	**Reserved.**	Yes	No	0
2:0	**Endpoint 2 Index Register.** This register determines which Endpoint 2 Transmit Mailbox is accessed when the Endpoint 2 Transmit Mailbox Data Port is read or written. This register is automatically incremented after the Endpoint 2 Transmit Mailbox Data port is read or written. This index register wraps around to zero when it reaches the maximum count.	Yes	Yes	0

4.11 (Address 0Dh; EP2DATA) Endpoint 2 Transmit Mailbox Data

Bits	Description	Read	Write	Default Value
7:0	**Endpoint 2 Transmit Mailbox Data.** This port is used to read or write one of the transmit mailbox registers. The register being accessed is selected by the Endpoint 2 Index Register. The eight Transmit Mailbox Registers are written by the local CPU and are read by a USB interrupt transfer from endpoint 2. They can be used to pass messages from the local CPU to the USB host. The format and content of the messages are user defined. If enabled, USB reads from this register can generate a local interrupt.	Yes	Yes	0

4.12 (Address 0Eh; EP2POLL) Endpoint 2 Interrupt Polling Interval Register

Bits	Description	Read	Write	Default Value
7:0	**Interrupt Polling Interval Register.** This register specifies the Endpoint 2 interrupt polling interval in milliseconds. It can read by the host through the endpoint 2 descriptor.	Yes	Yes	0xFF

4.13 (Address 10h; EP3DATA) Endpoint 3 Receive FIFO Data Register

Bits	Description	Read	Write	Default Value
7:0	**Endpoint 3 Receive FIFO Data Register.** This register is used by the local CPU to read data from the USB receive FIFO. The FIFO is written by the USB host using bulk or isochronous transfers to endpoint 3.	Yes	No	0

4.14 (Address 11h; EP3COUNT) Endpoint 3 Receive FIFO Count Register

Bits	Description	Read	Write	Default Value
7:0	**Receive FIFO Count.** This register returns the number of receive FIFO entries containing valid entries. Values range from 0 (empty) to 64 (full).	Yes	No	0

4.15 (Address 12h; EP3STAT) Endpoint 3 Receive FIFO Status Register

Bits	Description	Read	Write	Default Value
7:5	**Reserved.**	Yes	No	0
4	**Receive FIFO Flush.** Writing to this bit causes the receive FIFO to be flushed. Reading this bit always returns a 0.	Yes	Yes/Clr	0
3	**Receive FIFO Overflow.** If set, this bit indicates that an attempt was made by the USB host to write to the receive FIFO when the receive FIFO was full. Writing a 1 clears this bit.	Yes	Yes/Clr	0
2	**Receive FIFO Underflow.** If set, this bit indicates that an attempt was made to read the receive FIFO when the receive FIFO was empty. Writing a 1 clears this bit.	Yes	Yes/Clr	0
1	**Receive FIFO Full.** If set, this bit indicates that the receive FIFO is full.	Yes	No	0
0	**Receive FIFO Empty.** If set, this bit indicates that the receive FIFO is empty.	Yes	No	1

4.16 (Address 13h; EP3PKSZ) Endpoint 3 Maximum Packet Size Register

Bits	Description	Read	Write	Default Value
7:0	**Endpoint 3 Max Packet Size Register.** This register specifies the maximum packet size for endpoint 3 in units of 8 bytes (default = 64 bytes). It can be read by the host through the endpoint 3 descriptor.	Yes	Yes	0x08

4.17 (Address 14h; EP4DATA) Endpoint 4 Transmit FIFO Data Register

Bits	Description	Read	Write	Default Value
7:0	**Transmit FIFO Data Register.** This register is used by the local CPU to write data to the transmit FIFO. The FIFO is read by the USB host using bulk or isochronous transfers from endpoint 4.	No	Yes	0

4.18 (Address 15h; EP4COUNT) Endpoint 4 Transmit FIFO Count Register

Bits	Description	Read	Write	Default Value
7:0	**Transmit FIFO Count.** This register returns the number of transmit FIFO entries containing valid entries. Values range from 0 (empty) to 64 (full).	Yes	No	0

4.19 (Address 16h; EP4STAT) Endpoint 4 Transmit FIFO Status Register

Bits	Description	Read	Write	Default Value
7:6	**Reserved.**	Yes	No	0
5	**Transmit FIFO Valid.** If set, this bit allows the data in the FIFO to be read by the next read from the host. This bit is automatically cleared by a host read. This bit is only used if bit 0 in register **FIFOCTL** is set.	Yes	Yes	0
4	**Transmit FIFO Flush.** Writing to this bit causes the transmit FIFO to be flushed. Reading this bit always returns a 0.	Yes	Yes/Clr	0
3	**Transmit FIFO Overflow.** If set, this bit indicates that an attempt was made by the local CPU to write to the transmit FIFO when the transmit FIFO was full. Writing a 1 clears this bit.	Yes	Yes/Clr	0
2	**Transmit FIFO Underflow.** If set, this bit indicates that an attempt was made by the USB host to read the transmit FIFO when the transmit FIFO was empty. Writing a 1 clears this bit.	Yes	Yes/Clr	0
1	**Transmit FIFO Full.** If set, this bit indicates that the transmit FIFO is full.	Yes	No	0
0	**Transmit FIFO Empty.** If set, this bit indicates that the transmit FIFO is empty.	Yes	No	1

4.20 (Address 17h; EP4PKSZ) Endpoint 4 Maximum Packet Size Register

Bits	Description	Read	Write	Default Value
7:0	**Endpoint 4 Max Packet Size Register.** This register specifies the maximum packet size for endpoint 4 in units of 8 bytes (default = 64 bytes). It can be read by the host through the endpoint 4 descriptor.	Yes	Yes	0x08

4.21 (Address 18h; REVISION) Revision Register

Bits	Description	Read	Write	Default Value
7:0	**USBFC Revision.** This register returns current silicon revision number of the USBFC.	Yes	No	Current Revision

4.22 (Address 19h; USBSTAT) USB Status Register

Bits	Description	Read	Write	Default Value
7	**Suspend Control.** If set, this bit indicates that there is a pending suspend request from the USB host. Writing a 1 clears this bit and causes the USBFC to enter suspended mode.	Yes	Yes/Clr	0
6	**USB Endpoint 4 STALL.** The last USB IN token could not be serviced because the endpoint was stalled (DCTL register bit 4 set), and was acknowledged with a STALL. Writing a 1 clears this bit.	Yes	Yes/Clr	0
5	**USB Endpoint 4 NAK.** The last USB packet transmitted (IN packet) encountered a FIFO underrun condition, and was acknowledged with a NAK. Writing a 1 clears this bit.	Yes	Yes/Clr	0
4	**USB Endpoint 4 ACK.** The last USB packet transmitted (IN packet) was successfully acknowledged with an ACK from the USB host. Writing a 1 clears this bit.	Yes	Yes/Clr	0
3	**USB Endpoint 3 STALL.** The last USB packet received (OUT packet) could not be accepted because the endpoint was stalled (DCTL register bit 3 set), and was acknowledged with a STALL. Writing a 1 clears this bit.	Yes	Yes/Clr	0
2	**USB Endpoint 3 NAK.** The last USB packet received (OUT packet) could not be accepted, and was acknowledged with a NAK. The receive FIFO data will be corrupted and the local CPU should flush the FIFO. Writing a 1 clears this bit.	Yes	Yes/Clr	0
1	**USB Endpoint 3 ACK.** The last USB packet received (OUT packet) was successfully acknowledged with an ACK. Writing a 1 clears this bit.	Yes	Yes/Clr	0
0	**Endpoint 2 Valid.** When this bit is set, the 8-byte endpoint 2 mailbox registers have been written by the local CPU, but not yet read by the USB host. The local CPU should not write into these registers while this bit is set.	Yes	Yes	0

4.23 (Address 1Ah; FRAMEMSB) Frame Counter MSB Register

Bits	Description	Read	Write	Default Value
7:3	**Reserved.**	Yes	No	0
2:0	**Frame Counter MSB.** This register contains the most-significant bits of the frame counter from the most recent start-of-frame packet.	Yes	No	0

4.24 (Address 1Bh; FRAMELSB) Frame Counter LSB Register

Bits	Description	Read	Write	Default Value
7:0	**Frame Counter LSB.** This register contains the least-significant bits of the frame counter from the most recent start-of-frame packet.	Yes	No	0

4.25 (Address 1Ch; EXTIDX) Extended Register Index

Bits	Description	Read	Write	Default Value
7:0	**Extended Register Index.** This register selects which extended data register is accessed when the EXTDATA port is read or written.	Yes	Yes	0

4.26 (Address 1Dh; EXTDATA) Extended Register Data

Bits	Description	Read	Write	Default Value
7:0	**Extended Data.** This port provides access to one of the extended data registers. The index of the current register is held in the EXTIDX register.	See Below	See Below	See Below

4.26.1 (Address 1Dh, Index 00h; VIDMSB) Vendor ID MSB

Bits	Description	Read	Write	Default Value
7:0	**Vendor ID MSB.** This register determines the most significant byte of the Vendor ID during a ‘Get Device Descriptor’ request.	Yes	Yes	0x04

4.26.2 (Address 1Dh, Index 01h; VIDLSB) Vendor ID LSB

Bits	Description	Read	Write	Default Value
7:0	**Vendor ID LSB.** This register determines the least significant byte of the Vendor ID during a ‘Get Device Descriptor’ request.	Yes	Yes	0xB8

4.26.3 (Address 1Dh, Index 02h; PIDMSB) Product ID MSB

Bits	Description	Read	Write	Default Value
7:0	**Product ID MSB.** This register determines the most significant byte of the Product ID during a ‘Get Device Descriptor’ request.	Yes	Yes	0x88

4.26.4 (Address 1Dh, Index 03h; PIDLSB) Product ID LSB

Bits	Description	Read	Write	Default Value
7:0	**Product ID LSB.** This register determines the least significant byte of the Product ID during a ‘Get Device Descriptor’ request.	Yes	Yes	0x21

4.26.5 (Address 1Dh, Index 04h; RELMSB) Release Number MSB

Bits	Description	Read	Write	Default Value
7:0	**Release Number MSB.** This register determines the most significant byte of the device release number during a ‘Get Device Descriptor’ request.	Yes	Yes	0x01

4.26.6 (Address 1Dh, Index 05h; RELLSB) Release Number LSB

Bits	Description	Read	Write	Default Value
7:0	**Release Number LSB.** This register determines the least significant byte of the vendor-assigned revision code during a ‘Get Device Descriptor’ request.	Yes	Yes	0x00

4.26.7 (Address 1Dh, Index 06h; RCVAFTH) Receive FIFO Almost Full Threshold

Bits	Description	Read	Write	Default Value
7:6	**Reserved.**	Yes	No	0x00
5:0	**Receive FIFO Almost Full Threshold.** This register determines the threshold at which the receive FIFO almost full status bit is set.	Yes	Yes	0x3C

4.26.8 (Address 1Dh, Index 07h; XMTAETH) Transmit FIFO Almost Empty Threshold

Bits	Description	Read	Write	Default Value
7:6	**Reserved.**	Yes	No	0x00
5:0	**Transmit FIFO Almost Empty Threshold.** This register determines the threshold at which the transmit FIFO almost empty status bit is set.	Yes	Yes	0x04

4.26.9 (Address 1Dh, Index 08h; USBCTL) USB Control

Bits	Description	Read	Write	Default Value
7:1	**Reserved.**	Yes	No	0x00
0	**USB String Enable.** When set, this bit allows the default Vendor and Product ID String Descriptors to be returned to the host. When this bit is cleared, the string index values in the Device Descriptor are set to zero, and string descriptor reads are acknowledged with a stall.	Yes	Yes	0x1

4.26.10 (Address 1Dh, Index 09h; MAXPWR) Maximum Power Consumption

Bits	Description	Read	Write	Default Value
7:0	**Maximum Current.** The amount of current drawn by the peripheral from the USB port in increments of 2 mA. The USBFC reports this value to the host controller in the configuration descriptor. The default is 500 mA (0xFA * 2 mA).	Yes	Yes	0xFA

4.26.11 (Address 1Dh, Index 0Ah; PKTCTL) Packet Control

Bits	Description	Read	Write	Default Value
7	**EP4 Data Toggle Bit.** Contains the value of the Data Toggle bit to be sent in response to the next IN token to endpoint 4 from the USB host.	Yes	Yes	0x0 / 0x1 (Toggle)
6	**EP3 Data Toggle Bit.** Contains the value of the Data Toggle bit expected in the next DATA packet to endpoint 3 from the USB host.	Yes	Yes	0x0 / 0x1 (Toggle)
5	**EP2 Data Toggle Bit.** Contains the value of the Data Toggle bit to be sent in response to the next IN token to endpoint 2 from the USB host.	Yes	Yes	0x0 / 0x1 (Toggle)
4	**EP1 Data Toggle Bit.** Contains the value of the Data Toggle bit expected in the next DATA packet to endpoint 1 from the USB host.	Yes	Yes	0x0 / 0x1 (Toggle)
3	**EP4 Data Toggle Mode.** When set, this bit resets the Data Toggle bit to zero at the end of a USB transfer from EP4. When cleared, the Data Toggle bit strictly toggles.	Yes	Yes	0x0
2	**EP3 Data Toggle Mode.** When set, this bit resets the Data Toggle bit to zero at the end of a USB transfer to EP3. When cleared, the Data Toggle bit strictly toggles.	Yes	Yes	0x0
1	**Reserved.**	Yes	No	0x0
0	**EP1 Data Toggle Mode.** When set, this bit resets the Data Toggle bit to zero at the end of a USB transfer to EP1. When cleared, the Data Toggle bit strictly toggles.	Yes	Yes	0x0

4.26.12 (Address 1Dh, Index 0Bh; LOCALCTL) Local-Side Control

Bits	Description	Read	Write	Default Value
7:1	**Reserved.**	Yes	No	0x00
0	**Local Clock Output.** This bit controls the output of the LCLK pin. 0 = 48 MHz clock derived from CLKIN signal 1 = 12 MHz clock derived from USB bitstream (for testing purposes)	Yes	Yes	0x0

4.26.13 (Address 1Dh, Index 0Ch; FIFOCTL) FIFO Control

Bits	Description	Read	Write	Default Value
7:1	**Reserved.**	Yes	No	0x00
0	**Transmit FIFO Valid Mode.** When set, this bit causes a NAK response to a host read request from the transmit FIFO (EP4) unless the FIFO Valid bit (in register EP4STAT) is set. When this bit is cleared, any data waiting in the transmit FIFO will be sent in response to a host read request, and the FIFO Valid bit is ignored.	Yes	Yes	0x0

5. Standard Device Requests

5.1 Control 'IN' Transactions

5.1.1 Get Device Status

Offset	Number of Bytes	Description	Default Value
0	2	bits 15:2 = Reserved bit 1 = Device Remote Wakeup enabled bit 0 = Device is operating in Self-Powered mode. (depends on PWRGOOD_ input pin)	0x0001

5.1.2 Get Interface Status

Offset	Number of Bytes	Description	Default Value
0	2	bits 15:0 = Reserved	0x0000

5.1.3 Get Endpoint 0,1,2,3,4 Status

Offset	Number of Bytes	Description	Default Value
0	2	bits 15:1 = Reserved bit 0 = Endpoint is stalled	0x0000

5.1.4 Get Device Descriptor (18 Bytes)

Offset	Number of Bytes	Description	Default Value
0	1	Length	0x12
1	1	Type (device)	0x01
2	2	USB Specification Release Number	0x0100
4	1	Class Code	0x00
5	1	Sub Class Code	0x00
6	1	Protocol	0x00
7	1	Maximum Endpoint 0 Packet Size	0x08
8	2	Vendor ID	Determined by Local Register VIDMSB, VIDLSB
10	2	Product ID	Determined by Local Register PIDMSB, PIDLSB
12	2	Device Release Number	Determined by Local Register PIDMSB, PIDLSB
14	1	Index of string descriptor describing manufacturer	0x01
15	1	Index of string descriptor describing product	0x02
16	1	Index of string descriptor describing serial number	0x00
17	1	Number of configurations	0x01

5.1.5 Get Configuration Descriptor (46 bytes)

Note that all interface and endpoint descriptors are returned when this request is issued

Offset	Number of Bytes	Description	Default Value
Configuration Descriptor			
0	1	Length	0x09
1	1	Type (configuration)	0x02
2	2	Total length returned for this configuration	0x002E
4	1	Number of Interfaces	0x01
5	1	Number of this configuration	0x01
6	1	Index of string descriptor describing this configuration	0x00
7	1	Attributes bit 7 = Bus Powered (depends on BUSPWR_ input pin) bit 6 = Self-Powered (depends on BUSPWR_ input pin) bit 5 = Remote-Wakeup bits 4:0 = Reserved	0x60 (if self powered) 0xA0 (if bus powered)
8	1	Maximum USB power required (in 2 mA units) (depends on BUSPWR_ input pin)	0x32 (if self powered) 0xFA (if bus powered)
Interface 0 Descriptor			
0	1	Size of this descriptor in bytes	0x09
1	1	Type (interface)	0x04
2	1	Number of this interface	0x00
3	1	Alternate Interface	0x00
4	1	Number of endpoints used by this interface (excluding endpoint 0)	0x04
5	1	Class Code	0x00
6	1	Sub Class Code	0x00
7	1	Device Protocol	0x00
8	1	Index of string descriptor describing this interface	0x00
Endpoint 1 Descriptor			
0	1	Size of this descriptor	0x07
1	1	Descriptor Type (endpoint)	0x05
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	0x01
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	0x02
4	2	Maximum packet size of this endpoint	0x0008
6	1	Interval for polling endpoint (not used)	0x00

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Default Value
Endpoint 2 Descriptor			
0	1	Size of this descriptor	0x07
1	1	Descriptor Type (endpoint)	0x05
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	0x82
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	0x03
4	2	Maximum packet size of this endpoint	0x0008
6	1	Interval for polling endpoint	Determined by Local Register EP2POLL
Endpoint 3 Descriptor			
0	1	Size of this descriptor	0x07
1	1	Descriptor Type (endpoint)	0x05
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	0x03
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	0x02 for bulk 0x01 for isochronous
4	2	Maximum packet size of this endpoint for bulk mode Bus Time for isochronous mode	Determined by Local Register EP3PKSZ
6	1	Interval for polling endpoint	0x00 for bulk 0x01 for isochronous
Endpoint 4 Descriptor			
0	1	Size of this descriptor	0x07
1	1	Descriptor Type (endpoint)	0x05
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	0x84
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	0x02 for bulk 0x01 for isochronous
4	2	Maximum packet size of this endpoint for bulk mode Bus Time for isochronous mode	Determined by Local Register EP4PKSZ
6	1	Interval for polling endpoint	0x00 for bulk 0x01 for isochronous

5.1.6 Get String Descriptor 0

Offset	Number of Bytes	Description	Default Value
0	2	Language ID (English = 09, U.S. = 04)	0x0403 0x0409

5.1.7 Get String Descriptor 1

Offset	Number of Bytes	Description	Default Value
0	24	Manufacturer Descriptor	0x1803 “SEIKO EPSON”

5.1.8 Get String Descriptor 2

Offset	Number of Bytes	Description	Default Value
0	66	Product Descriptor	0x4203 “USB Interface Controller TEST2.0”

5.1.9 Get Configuration

Offset	Number of Bytes	Description	Default Value
0	1	Returns current device configuration	0x00

5.1.10 Get Interface

Offset	Number of Bytes	Description	Default Value
0	1	Returns current alternate setting for the specified interface	0x00

5.2 Control ‘OUT’ Transactions

5.2.1 Set Address

Offset	Number of Bytes	Description	Default Value
--	0	Sets USB address of device Value = device address, Index = 0	--

5.2.2 Set Configuration

Offset	Number of Bytes	Description	Default Value
--	0	Sets the device configuration Value = Configuration value (0 or 1 supported),	--

5.2.3 Set Interface

Offset	Number of Bytes	Description	Default Value
--	0	Selects alternate setting for specified interface Value = Alternate setting, Index = specified interface	--

5.2.4 Device Clear Feature

Offset	Number of Bytes	Description	Default Value

--	0	Clear the selected device feature
		Value = feature selector
		FS = 1 --> Device Remote Wakeup (disable)
			--

5.2.5 Device Set Feature

Offset	Number of Bytes	Description	Default Value

--	0	Set the selected device feature
		Value = feature selector
		FS = 1 --> Device Remote Wakeup (enable)
			--

5.2.6 Endpoint 0,1,2,3,4 Clear Feature

Offset	Number of Bytes	Description	Default Value

--	0	Clear the selected endpoint feature
		Value = feature selector, Index = endpoint number
		FS = 0 --> Endpoint stall (clears stall bit)
			--

5.2.7 Endpoint 0,1,2,3,4 Set Feature

Offset	Number of Bytes	Description	Default Value

--	0	Set the selected endpoint feature
		Value = feature selector, Index = endpoint number
		FS = 0 --> Endpoint stall (sets stall bit)
			--

5.3 Endpoint 1 'OUT' Transactions (Receive Mailboxes)

Offset	Number of Bytes	Description	Default Value
--	8	Host writes 8 bytes to the receive mailboxes using bulk OUT transactions	--

5.4 Endpoint 2 'IN' Transactions (Transmit Mailboxes)

Offset	Number of Bytes	Description	Default Value
--	8	Host reads 8 bytes from the transmit mailboxes using interrupt IN transactions	--

5.5 Endpoint 3 'OUT' Transactions (Receive FIFO)

Offset	Number of Bytes	Description	Default Value
--	up to 64	Host writes data into the receive FIFO using bulk or isochronous OUT transactions	--

5.6 Endpoint 4 'IN' Transactions (Transmit FIFO)

Offset	Number of Bytes	Description	Default Value
--	up to 64	Host reads data from the transmit FIFO using bulk or isochronous IN transactions	--

6. Vendor Device Requests

6.1 Device Clear Feature

Offset	Number of Bytes	Description	Default Value

--	0	Clear the selected device feature
		Value = feature selector
		FS = 0x80 --> Timing test mode (clears test bit)
			--

6.2 Device Set Feature

Offset	Number of Bytes	Description	Default Value

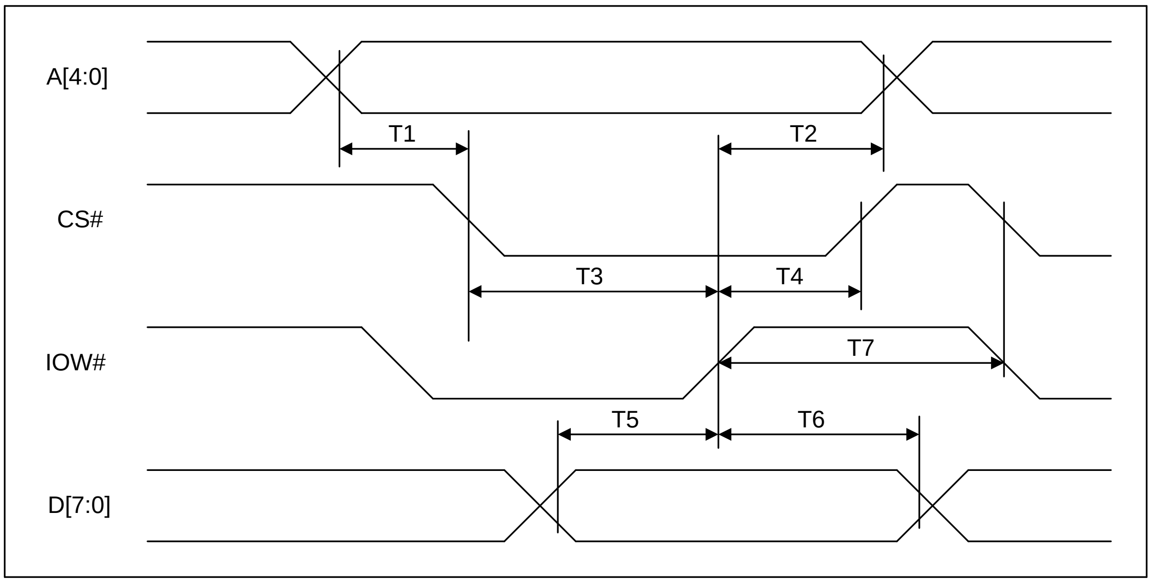
--	0	Set the selected device feature
		Value = feature selector
		FS = 0x80 --> Timing test mode (sets test bit)
			--

7. Timing

7.1 Local Bus Write to Register

NAME	DESCRIPTION	MIN	MAX	UNIT
T1	Address setup to write enable*	10		ns
T2	Address hold from end of write enable*	0		ns
T3	Write enable width*	25		ns
T4	Chip select hold from end of IOW_	0		ns
T5	Data setup to end of write enable*	5		ns
T6	Data hold time from end of IOW_	5		ns
T7	I/O Recovery Time	60		ns

- *Write enable is the occurrence of both IOW_ and CS_.*

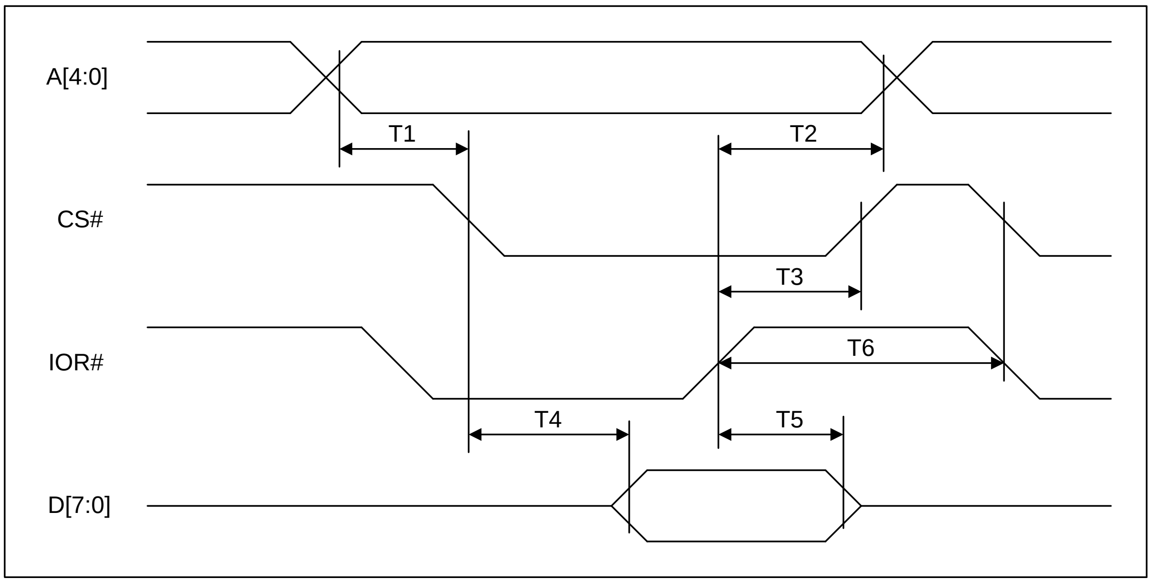


7.2 Local Bus Read from Register

NAME	DESCRIPTION	MIN	MAX	UNIT
T1	Address setup to read enable*	10		ns
T2	Address hold from end of read enable*	0		ns
T3	Chip select hold from end of IOR_	0		ns
T4	Data access time from read enable*		8	ns
T5	Data tri-state time from end of IOR_	9		ns
T6	I/O Recovery Time	60		ns

- *Read enable is the occurrence of both IOR_ and CS_*

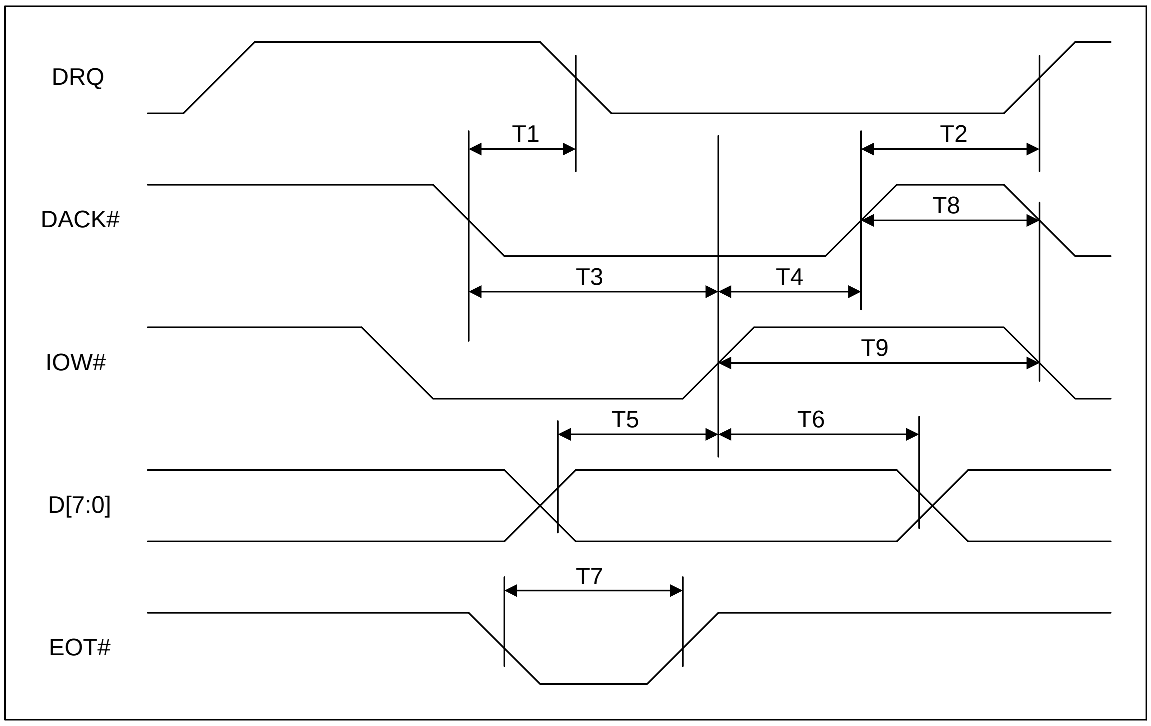
.



7.3 DMA Write to FIFO

NAME	DESCRIPTION	MIN	MAX	UNIT
T1	DRQ false from DACK_ true		58	ns
T2	DACK_ false to DRQ true	29		ns
T3	Write enable width*	50		ns
T4	DACK_ hold from end of IOW_	0		ns
T5	Data setup to end of write enable*	50		ns
T6	Data hold time from end of IOW_	25		ns
T7	Width of EOT_ pulse (see note)	50		ns
T8	DACK_ recovery time	30		ns
T9	DMA recovery time	60		ns

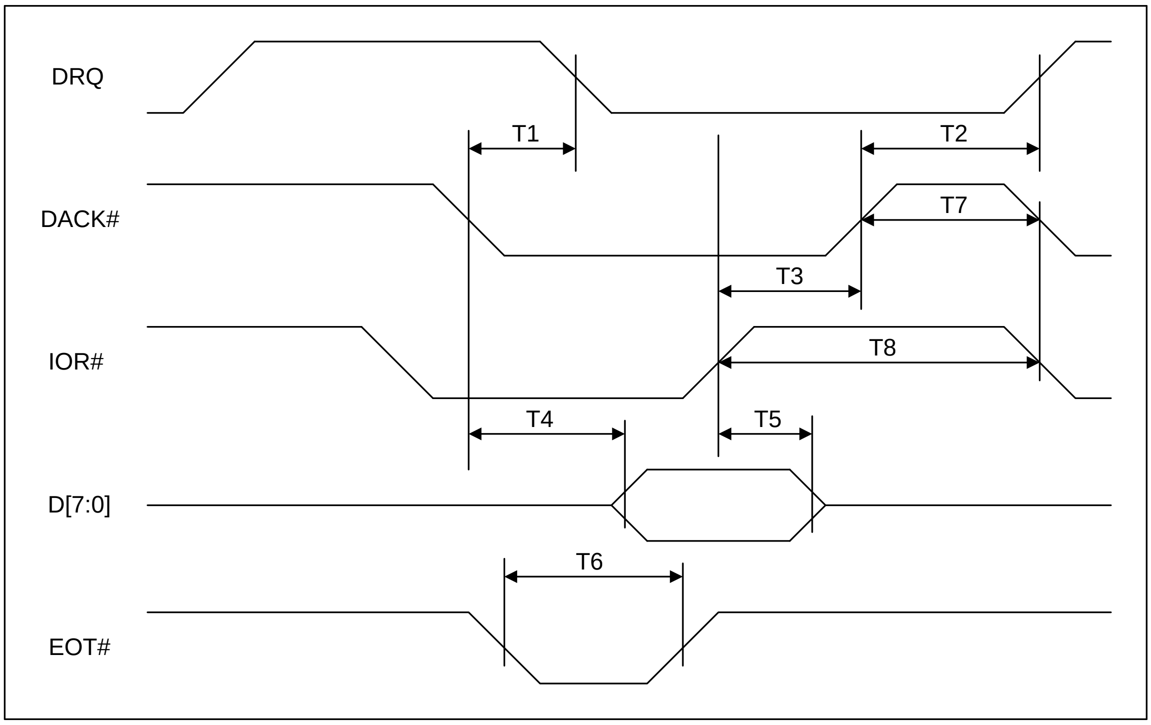
** Write enable is the occurrence of both IOW_ and DACK_.*
Note: EOT_, IOW_, and DACK_ must be concurrently true for at least T7 for proper recognition for the EOT_ pulse.



7.4 DMA Read from FIFO

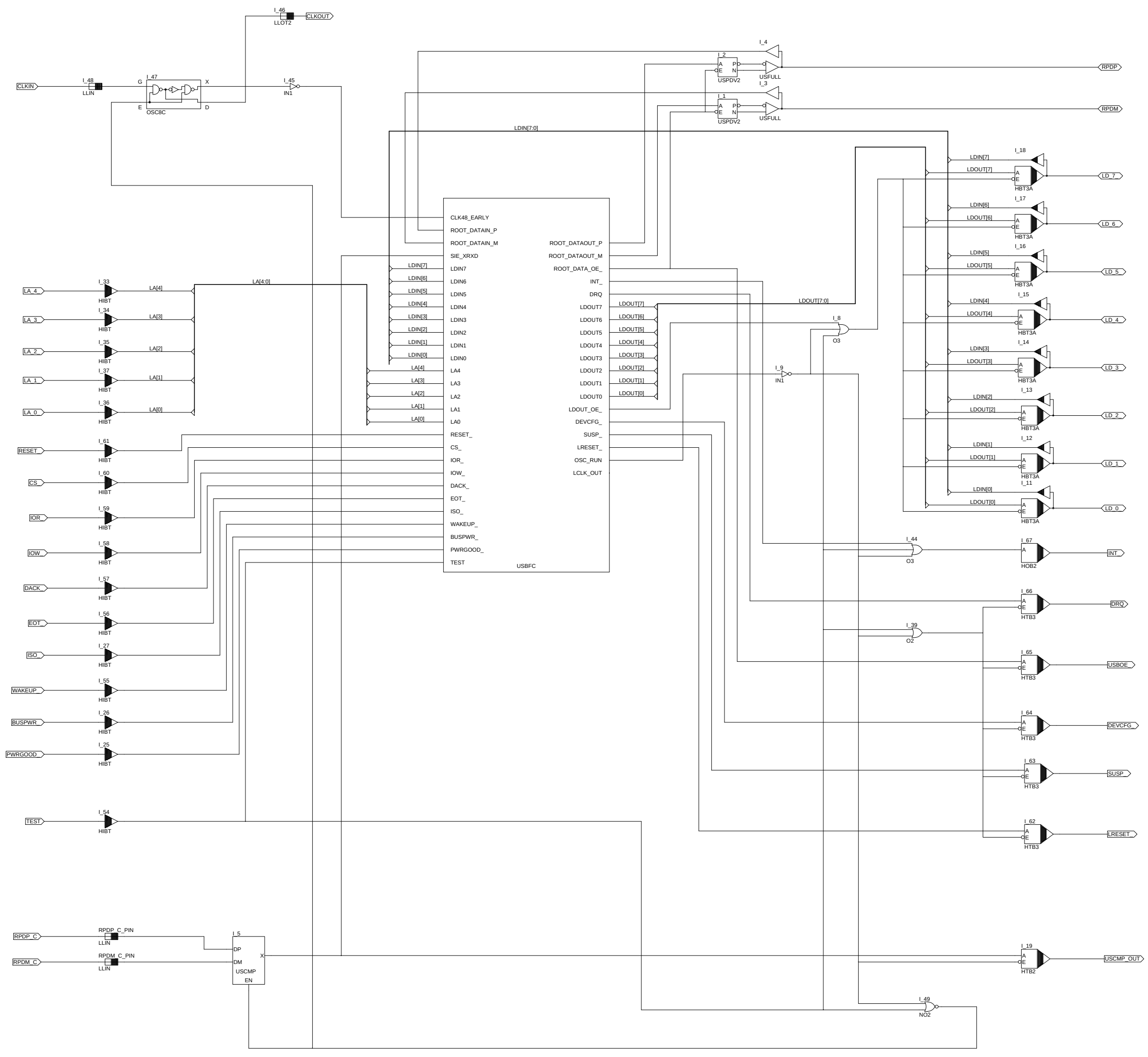
NAME	DESCRIPTION	MIN	MAX	UNIT
T1	DRQ false from DACK_ true		58	ns
T2	DACK_ false to DRQ true	29		ns
T3	DACK_ hold time from end of IOR_	0		ns
T4	Data access time from read enable*		8	ns
T5	Data hold time from end of IOR_	9		ns
T6	Width of EOT_ pulse (see note)	50		ns
T7	DACK_ recovery time	30		ns
T8	DMA recovery time	60		ns

** Read enable is the occurrence of both IOR_ and DACK_.*
Note: EOT_, IOR_, and DACK_ must be concurrently true for at least T6 for proper recognition of the EOT_ pulse.



8. Test Circuit

Please design your circuit so that USBFC can be seen from outside chip as the diagram below in the USBFC test mode.



Oscillator cell (OSC8C) is included just for a design example.　OSC_RUN, however, should be able to control the clock input into CLK48_EARLY.

CLKOUT pin is not included in the test pattern.　LCLK_OUT pin of USSBFC is not tested.

This sample circuit can be an example of a minimam addition to USBFC to make a simple usb chip when a CPU is external to the chip.