USB Module

NOTE: This chapter is an excerpt from the *MSP430x5xx and MSP430x6xx Family User's Guide*.

This chapter describes the USB module that is available in some devices.

Topic **Page**

1.1 USB Introduction

The features of the USB module include:

- Fully compliant with the USB 2.0 full-speed specification
  - Full-speed device (12 Mbps) with integrated USB transceiver (PHY)
  - Up to eight input and eight output endpoints
  - Supports control, interrupt, and bulk transfers
  - Supports USB suspend, resume, and remote wakeup
- A power supply system independent from the PMM system
  - Integrated 3.3-V LDO regulator with sufficient output to power entire MSP430 and system circuitry from 5-V VBUS
  - Integrated 1.8-V LDO regulator for PHY and PLL
  - Easily used in either bus-powered or self-powered operation
  - Current-limiting capability on 3.3-V LDO output
  - Autonomous power-up of device on arrival of USB power possible (low or no battery condition)
- Internal 48-MHz USB clock
  - Integrated programmable PLL
  - Highly-flexible input clock frequencies for use with lowest-cost crystals
- 1904 bytes of dedicated USB buffer space for endpoints, with fully configurable size to a granularity of eight bytes
- Timestamp generator with 62.5-ns resolution
- When USB is disabled
  - Buffer space is mapped into general RAM, providing additional 2KB to the system
  - USB interface pins become high-current general purpose I/O pins

NOTE: Use of the word *device*

The word *device* is used throughout the chapter. This word can mean one of two things, depending on the context. In a USB context, it means what the USB specification refers to as a device, function, or peripheral; that is, a piece of equipment that can be attached to a USB host or hub. In a semiconductor context, it refers to an integrated circuit such as the MSP430.

To avoid confusion, the term *USB device* in this document refers to the USB-context meaning of the word. The word *device* by itself refers to silicon devices such as the MSP430.

Figure 1-1 shows a block diagram of the USB module.

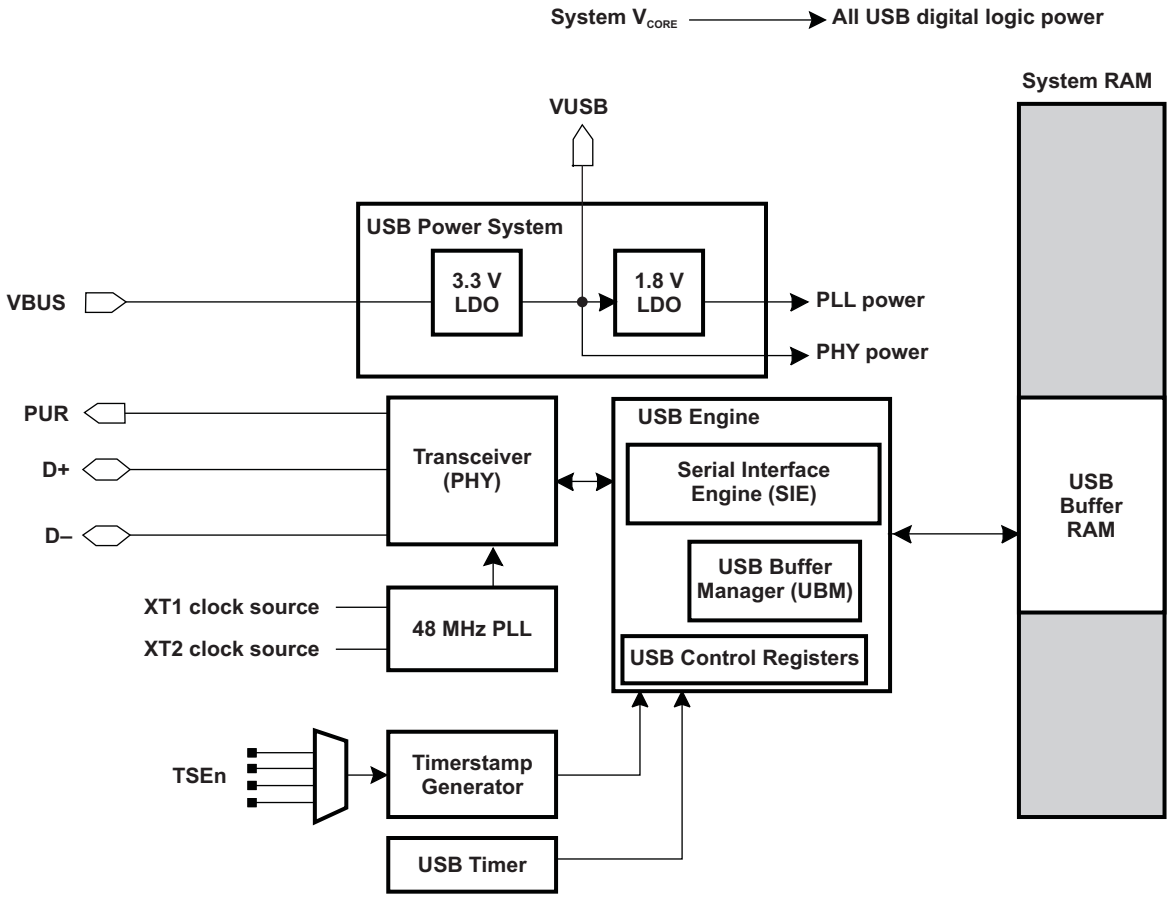

Figure 1-1. USB Block Diagram

1.2 USB Operation

The USB module is a comprehensive full-speed USB device compliant with the USB 2.0 specification.

The USB engine coordinates all USB-related traffic. It consists of the USB SIE (serial interface engine) and USB Buffer Manager (UBM). All traffic received on the USB receive path is de-serialized and placed into receive buffers in the USB buffer RAM. Data in the buffer RAM marked 'ready to be sent' are serialized into packets and sent to the USB host.

The USB engine requires an accurate 48-MHz clock to sample the incoming data stream. This is generated by a PLL that is fed from one of the system oscillators (XT1 or XT2). A crystal of 4 MHz or greater is required. In addition to crystal operation, the crystal bypass mode can also be used to supply the clock required by the PLL. The PLL is very flexible and can adapt to a wide range of crystal and input frequencies, allowing for cost-effective clock designs.

NOTE: The reference clock to the PLL depends on the device configuration. On devices that contain the optional XT2, the reference clock to the PLL is XT2CLK, regardless of whether or not XT1 is available. If the device has only XT1, then the reference is XT1CLK. See the device-specific data sheet for clock sources available.

NOTE: The USB module only supports active operation during power modes AM through LPM1.

The USB buffer memory is where data is exchanged between the USB interface and the application software. It is also where the usage of endpoints 1 to 7 are defined. This buffer memory is implemented such that it can be easily accessed like RAM by the CPU or DMA while USB module is not in suspend condition.

1.2.1 USB Transceiver (PHY)

The physical layer interface (USB transceiver) is a differential line driver directly powered from VUSB (3.3 V). The line driver is connected to the DP and DM pins, which form the signaling mechanism of the USB interface.

When the PUSEL bit is set, DP and DM are configured to function as USB drivers controlled by the USB core logic. When the bit is cleared, these two pins become "Port U", which is a pair of high-current general purpose I/O pins. In this case, the pins are controlled by the Port U control registers. Port U is powered from the VUSB rail, separate from the main device DVCC. If these pins are to be used, whether for USB or general purpose use, it is necessary that VUSB be properly powered from either the internal regulators or an external source.

1.2.1.1 D+ Pullup Via PUR Pin

When a full-speed USB device is attached to a USB host, it must pull up the D+ line (DP pin) for the host to recognize its presence. The MSP430 USB module implements this with a software-controlled pin that activates a pullup resistor. The bit that controls this function is PUR_EN. If software control is not desired, the pullup can be connected directly to VUSB.

1.2.1.2 Shorts on Damaged Cables and Clamping

USB devices must tolerate connection to a cable that is damaged, such that it has developed shorts on either ground or VBUS. The device should not become damaged by this event, either electrically or physically. To this end, the MSP430 USB power system features a current limitation mechanism that limits the available transceiver current in the event of a short to ground. The transceiver interface itself therefore does not need a current limiting function.

Note that if VUSB is to be powered from a source other than the integrated regulator, the absence of current-limiting in the transceiver means that the external power source must itself be tolerant of this same shorting event, through its own means of current limiting.

1.2.1.3 Port U Control

When PUSEL is cleared, the Port U pins (PU.0 and PU.1) function as general-purpose, high-current I/O pins. These pins can only be configured together as either both inputs or both outputs. Port U is supplied by the VUSB rail. If the 3.3-V LDO is not being used in the system (disabled), the VUSB pin can be supplied externally.

PUOPE controls the enable of both outputs residing on the Port U pins. Setting PUIPE =1 causes both input buffers to be enabled. When Port U outputs are enabled (PUOPE = 1), the PUIN0 and PUIN1 pins mirror what is present on the outputs assuming PUIPE = 1. To use the Port U pins as inputs, the outputs should be disabled by setting PUOPE = 0, and enabling the input buffers by setting PUIPE = 1. Once configured as inputs (PUIPE = 1), the PUIN0 and PUIN1 bits can be read to determine the respective input values.

When PUOPE is set, both Port U pins function as outputs, controlled by PUOUT0 and PUOUT1. When driven high, they use the VUSB rail, and they are capable of a drive current higher than other I/O pins on the device. See the device-specific datasheet for parameters.

By default, PUOPE and PUIPE are cleared. PU.0 and PU.1 are high-impedance (input buffers are disabled and outputs are disabled).

1.2.2 USB Power System

The USB power system incorporates dual LDO regulators (3.3 V and 1.8 V) that allow the entire MSP430 device to be powered from 5-V VBUS when it is made available from the USB host. Alternatively, the power system can supply power only to the USB module, or it can be unused altogether, as in a fully self-powered device. The block diagram is shown in Figure 1-2.

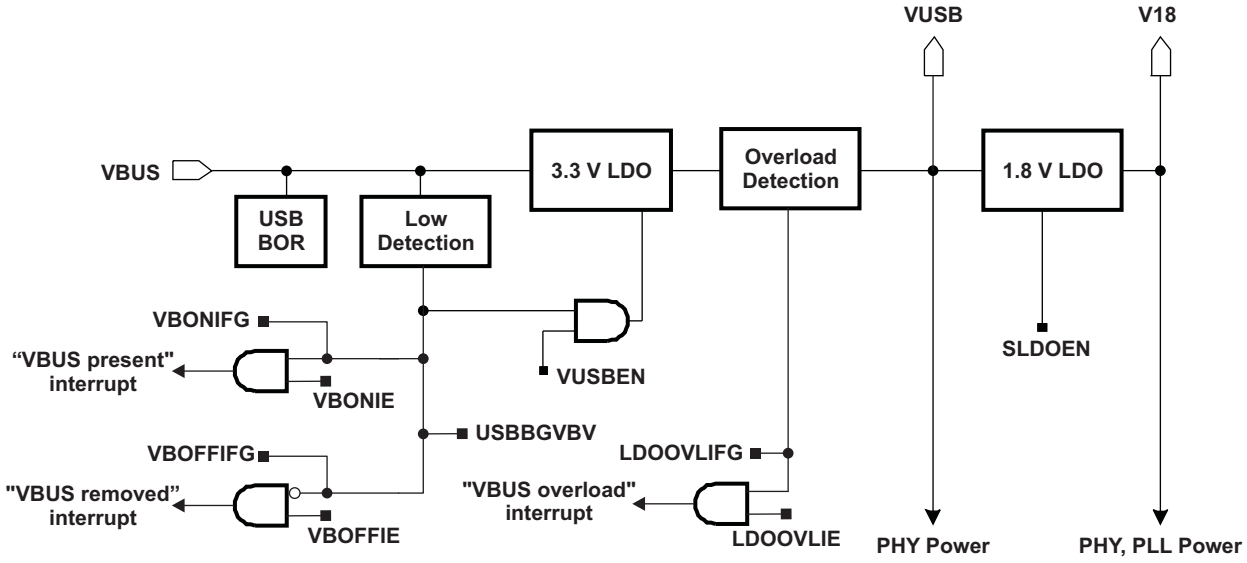

Figure 1-2. USB Power System

The 3.3-V LDO receives 5 V from VBUS and provides power to the transceiver, as well as the VUSB pin. Using this setup prevents the relatively high load of the transceiver and PLL from loading a local system power supply, if used. Thus it is very useful in battery-powered devices.

The 1.8-V LDO receives power from the VUSB pin – which is to be sourced either from the internal 3.3-V LDO or externally – and provides power to the USB PLL and transceiver. The 1.8-V LDO in the USB module is not related to the LDO that resides in the MSP430 Power Management Module (PMM).

The inputs and outputs of the LDOs are shown in Figure 1-2. VBUS, VUSB, and V18 need to be connected to external capacitors. The V18 pin is not intended to source other components in the system, rather it exists solely for the attachment of a load capacitor.

1.2.2.1 Enabling and Disabling

The 3.3-V LDO is enabled or disabled by setting or clearing VUSBEN, respectively. Even if enabled, if the voltage on VBUS is detected to be low or nonexistent, the LDO is suspended. No additional current is consumed while the LDO is suspended. When VBUS rises above the USB power brownout level, the LDO reference and low voltage detection become enabled. When VBUS rises further above the launch voltage V_{LAUNCH}, the LDO module becomes enabled (see Figure 1-3). See device-specific data sheet for value of V_{LAUNCH}.

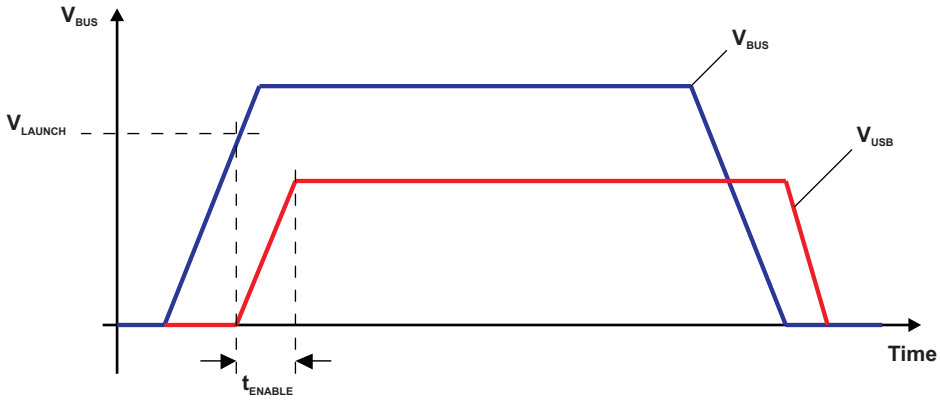

Figure 1-3. USB Power Up and Down Profile

The 1.8-V LDO can be enabled or disabled by setting SLDOEN accordingly. By default, the 1.8-V LDO is controlled automatically according to whether power is available on VBUS. This auto-enable feature is controlled by SLDOAON. In this case, that the SLDOEN bit does not reflect the state of the 1.8-V LDO. If the user wishes to know the state while using the auto-enable feature, the USBBGVBV bit in USBPWRCTL can be read. In addition, to disable the 1.8-V LDO, SLDOAON must be cleared along with SLDOEN. If providing VUSB from an external source, rather than through the integrated 3.3-V LDO, keep in mind that if 5 V is not present on VBUS, the 1.8-V LDO is not automatically enabled. In this situation, either VBUS much be attached to USB bus power, or the SLDOAON bit must be cleared and SLDOEN set.

It is required that power from the USB cable's VBUS be directed through a Schottky diode prior to entering the VBUS terminal. This prevents current from draining into the cable's VBUS from the LDO input, allowing the MSP430 to tolerate a suspended or unpowered USB cable that remains electrically connected.

The VBONIFG flag can be used to indicate that the voltage on VBUS has risen above the launch voltage. In addition to the VBONIFG being set, an interrupt is also generated when VBONIE = 1. Similarly, the VBOFFIFG flag can be used to indicate that the voltage on VBUS has fallen below the launch voltage. In addition to the VBOFFIFG being set, an interrupt is also generated when VBOFFIE = 1. The USBBGVBV bit can also be polled to indicate the level of VBUS; that is, above or below the launch voltage.

1.2.2.2 Powering the Rest of the MSP430 From USB Bus Power Via VUSB

The output of the 3.3-V LDO can be used to power the entire MSP430 device, sourcing the DVCC rail. If this is desired, the VUSB and DVCC should be connected externally. Power from the 3.3-V LDO is sourced into DVCC (see Figure 1-4).

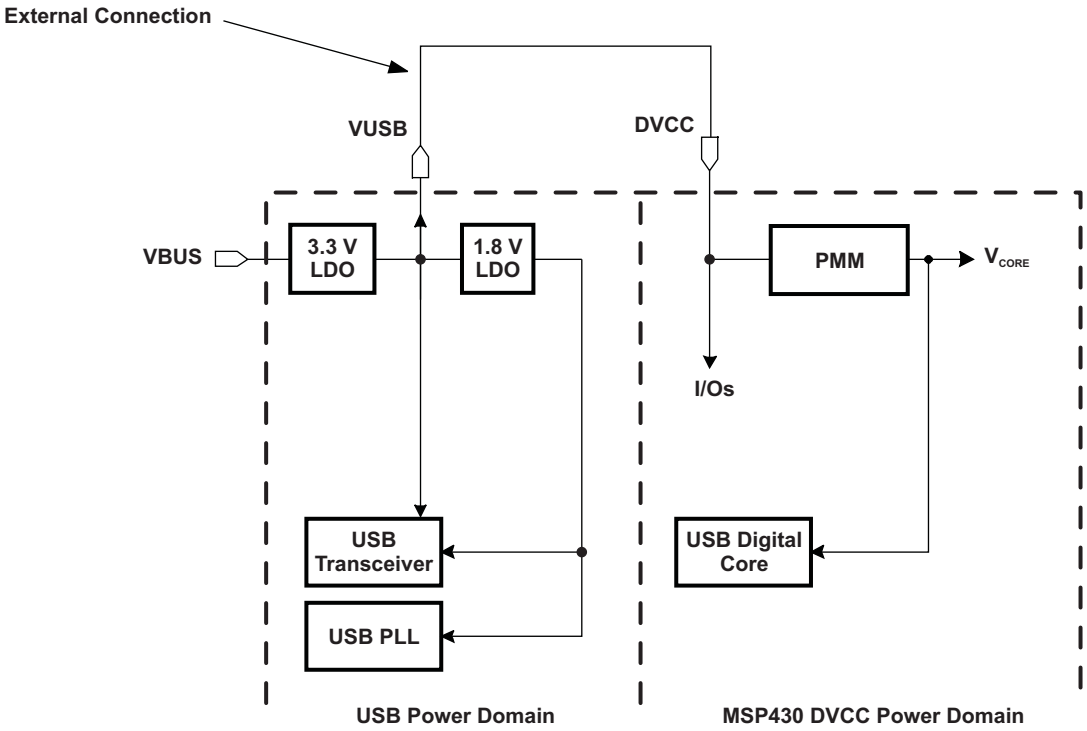

Figure 1-4. Powering Entire MSP430 From VBUS

With this connection made, the MSP430 allows for autonomous power up of the device when VBUS rises above V_{LAUNCH}. If no voltage is present on V_{CORE} – meaning the device is unpowered (or, in LPMx.5 mode) – then both the 3.3-V and 1.8-V LDOs automatically turn on when VBUS rises above V_{LAUNCH}.

Note that if DVCC is being driven from VUSB in this manner, and if power is available from VUSB, attempting to place the device into LPMx.5 results in the device immediately re-powering. This is because it re-creates the conditions of the autonomous feature described above (no V_{CORE} but power available on VBUS). The resulting drop of V_{CORE} would cause the system to immediately power up again.

When DVCC is being powered from VUSB, it is up to the user to ensure that the total current being drawn from VBUS stays below I_{DET}.

1.2.2.3 Powering Other Components in the System from VUSB

There is sufficient current capacity available from the 3.3-V LDO to power not only the entire MSP430 but also other components in the system, via the VUSB pin.

If the device is to always be connected to USB, then perhaps no other power system is needed. If it only occasionally connects to USB and is battery-powered otherwise, then sourcing system power via the 3.3-V LDO takes power burden away from the battery. Alternatively, if the battery is rechargeable, the recharging can be driven from VUSB.

1.2.2.4 Self-Powered Devices

Some applications may be self-powered, in that the VUSB power is supplied externally. In these cases, the 3.3-V LDO would be disabled (VUSBEN = 0). For proper USB operation, the voltage on VBUS can still be detected, even while the 3.3-V LDO disabled, by setting USBDETEN = 1. When VBUS rises above the USB power brownout level, low voltage detection becomes enabled. When VBUS rises further above the launch voltage V_{LAUNCH}, the voltage on VBUS is detected.

1.2.2.5 Current Limitation and Overload Protection

The 3.3-V LDO features current limitation to protect the transceiver during shorted-cable conditions. A short or overload condition – that is, when the output of the LDO becomes current-limited to I_{DET}. This is reported to software via the VUOVLIFG flag. See device-specific data sheet for value of I_{DET}.

If this event occurs, it means USB operation may become unreliable, due to insufficient power supply. As a result, software may wish to cease USB operation. If the OVLAOFF bit is set, USB operation is automatically terminated by clearing VUSBEN.

During overload conditions, VUSB and V18 drop below their nominal output voltage. In power scenarios where DVCC is exclusively supplied from VUSB, repetitive system restarts may be triggered as long the short or overload condition exists. For this reason, firmware should avoid re-enabling USB after detection of an overload on the previous power session, until the cause of failure can be identified. Ultimately, it is the user's responsibility to ensure that the current drawn from VBUS does not exceed I_{DET}.

The VUOVLIFG flag can be used to indicate an overcurrent condition on the 3.3-V LDO. When an overcurrent condition is detected, VUOVLIFG = 1. In addition to the VUOVLIFG being set, an interrupt is also generated when VUOVLIE = 1.

The USB power system brownout circuit is supplied from VBUS or DVCC, whichever carries the higher voltage.

1.2.3 USB Phase-Locked Loop (PLL)

The PLL provides the low-jitter high-accuracy clock needed for USB operation (see Figure 1-5).

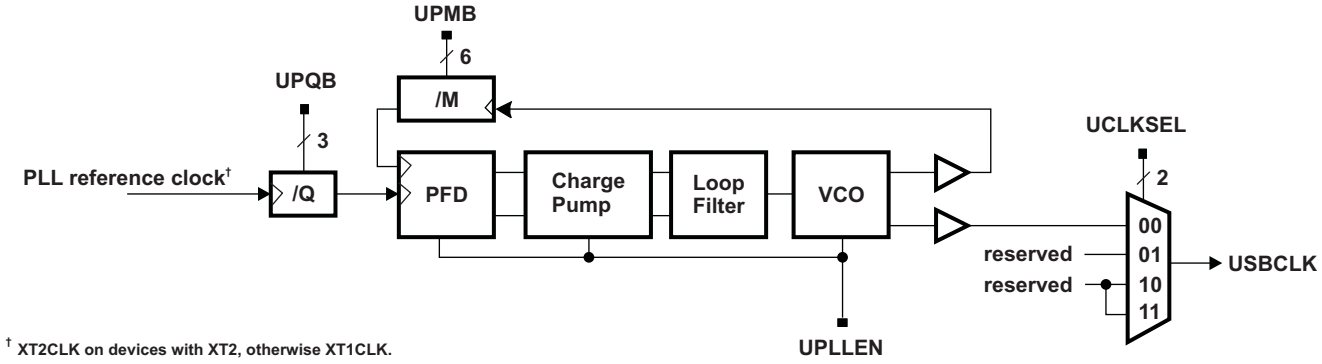

Figure 1-5. USB-PLL Analog Block Diagram

The reference clock to the PLL depends on the device configuration. On devices that contain the optional XT2, the reference clock to the PLL is XT2CLK, regardless if XT1 is available. If the device has only XT1, then the reference is XT1CLK. A four-bit prescale counter controlled by the UPQB bits allows division of the reference to generate the PLL update clock. The UPMB bits control the divider in the feedback path and define the multiplication rate of the PLL (see Equation 1).

$$f_{OUT} = CLK_{SEL} \times \frac{DIVM}{DIVQ} \quad \text{with} \quad \frac{CLK_{SEL}}{DIVQ} = f_{UPD} \geq 1.5\ \text{MHz} \tag{1}$$

Where

CLK_{SEL} is the PLL reference clock frequency

DIVQ is derived from Table 1-1

DIVM represents the value of UPMB field

Table 1-2 lists some common clock input frequencies for CLK_{SEL}, along with the appropriate register settings for generating the nominal 48-MHz clock required by the USB serial engine. For crystal operation, a 4 MHz or higher crystal is required. For crystal bypass mode of operation, 1.5 MHz is the lowest external clock input possible for CLK_{SEL}.

If USB operation is used in a bus-powered configuration, disabling the PLL is necessary to pass the USB requirement of not consuming more than 500 µA. The UPLLEN bit enables or disables the PLL. The PFDEN bit must be set to enable the phase and frequency discriminator. Out-of-lock, loss-of-signal, and out-of-range are indicated and flagged in the interrupt flags OOLIFG, LOSIFG, OORIFG, respectively.

NOTE: UCLKSEL bits should always be cleared, which is the default operation. All other combinations are reserved for future usages.

Table 1-1. USB-PLL Pre-Scale Divider

UPQB	DIVQ
000	1
001	2
010	3
011	4
100	6
101	8
110	13
111	16

Table 1-2. Register Settings to Generate 48 MHz Using Common Clock Input Frequencies

CLK_{SEL} (MHz)	UPQB	UPMB	DIVQ	DIVM	CLKLOOP (MHz)	UPLLCLK (MHz)	ACCURACY (ppm)
1.5	000	011111	1	32	1.5	48	0
1.6	000	011101	1	30	1.6	48	0
1.7778	000	011010	1	27	1.7778	48	0
1.8432	000	011001	1	26	1.8432	47.92	-1570
1.8461	000	011001	1	26	1.8461	48	0
1.92	000	011000	1	25	1.92	48	0
2	000	010111	1	24	2	48	0
2.4	000	010011	1	20	2.4	48	0
2.6667	000	010001	1	18	2.6667	48	0
3	000	001111	1	16	3	48	0
3.2	001	011110	2	30	1.6	48	0
3.5556	001	011010	2	27	1.7778	48	0
3.84	001	011001	2	25	1.92	48	0
4[(1)]	001	010111	2	24	2	48	0
4.1739	001	010110	2	23	2.086	48	0
4.3636	001	010101	2	22	2.1818	48	0
4.5	010	011111	3	32	1.5	48	0
4.8	001	010011	2	20	2.4	48	0
5.33 ≠ (16/3)	001	010001	2	18	2.6667	48	0
5.76	010	011000	3	25	1.92	48	0
6	010	010111	3	24	2	48	0
6.4	011	011101	4	30	1.6	48	0
7.2	010	010011	3	20	2.4	48	0
7.68	011	011000	4	25	1.92	48	0
8[(1)]	010	010001	3	18	2.6667	48	0
9	010	001111	3	16	3	48	0

(1) This frequency can be automatically detected by the factory-supplied BSL, for use in production programming of the MSP430 via USB. See *MSP430 Programming With the Bootloader (BSL)* for details.

Table 1-2. Register Settings to Generate 48 MHz Using Common Clock Input Frequencies (continued)

CLK_{SEL} (MHz)	UPQB	UPMB	DIVQ	DIVM	CLKLOOP (MHz)	UPLLCLK (MHz)	ACCURACY (ppm)
9.6	011	010011	4	20	2.4	48	0
10.66 ≠ (32/3)	011	010001	4	18	2.6667	48	0
12[(1)]	011	001111	4	16	3	48	0
12.8	101	011101	8	30	1.6	48	0
14.4	100	010011	6	20	2.4	48	0
16	100	010001	6	18	2.6667	48	0
16.9344	100	010000	6	17	2.8224	47.98	-400
16.94118	100	010000	6	17	2.8235	48	0
18	100	001111	6	16	3	48	0
19.2	101	010011	8	20	2.4	48	0
24[(1)]	101	001111	8	16	3	48	0
25.6	111	011101	16	30	1.6	48	0
26.0	110	010111	13	24	2	48	0
32	111	010111	16	24	2.6667	48	0

1.2.3.1 Modifying the Divider Values

Updating the values of UPQB (DIVQ) and UPMB (DIVM) to select the desired PLL frequency must occur simultaneously to avoid spurious frequency artifacts. The values of UPQB and UPMB can be calculated and written to their buffer registers; the final update of UPQB and UPMB occurs when the upper byte of UPLLDIVB (UPQB) is written.

1.2.3.2 PLL Error Indicators

The PLL can detect three kinds of errors. Out-of-lock (OOL) is indicated if a frequency correction is performed in the same direction (that is, up or down) for four consecutive update periods. Loss-of-signal (LOS) is indicated if a frequency correction is performed in the same direction (that is, up or down) for 16 consecutive update periods. Out-of-range (OOR) is indicated if PLL was unable to lock for more than 32 update periods.

OOL, LOS, and OOR trigger their respective interrupt flags (USBOOLIFG, USBLOSIFG, USBOORIFG) if errors occur, and interrupts are generated if enabled by their enable bits (USBOOLIE, USBLOSIE, USBOORIE).

1.2.3.3 PLL Startup Sequence

To achieve the fastest startup of the PLL, the following sequence is recommended.

1. Enable VUSB and V18.
2. Wait 2 ms for external capacitors to charge, so that proper VUSB is in place. (During this time, the USB registers and buffers can be initialized.)
3. Activate the PLL, using the required divider values.
4. Wait 2 ms and check PLL. If it stays locked, it is ready to be used.

1.2.4 USB Controller Engine

The USB controller engine transfers data packets arriving from the USB host into the USB buffers, and also transmits valid data from the buffers to the USB host. The controller engine has dedicated, fixed buffer space for input endpoint 0 and output endpoint 0, which are the default USB endpoints for control transfers.

The 14 remaining endpoints (seven input and seven output) may have one or more USB buffers assigned to them. All the buffers are located in the USB buffer memory. This memory is implemented as "multiport" memory, in that it can be accessed both by the USB buffer manager and also by the CPU and DMA.

Each endpoint has a dedicated set of descriptor registers that describe the use of that endpoint (see Figure 1-6). Configuration of each endpoint is performed by setting its descriptor registers. These data structures are located in the USB buffer memory and contain address pointers to the next memory buffer for receive or transmit.

Assigning one or two data buffers to an endpoint, of up to 64 bytes, requires no further software involvement after configuration. If more than two buffers per endpoint are desired, however, software must change the address pointers on the fly during a receive or transmit process.

Synchronization of empty and full buffers is done using validation flags. All events are indicated by flags and fire a vector interrupt when enabled. Transfer event indication can be enabled separately.

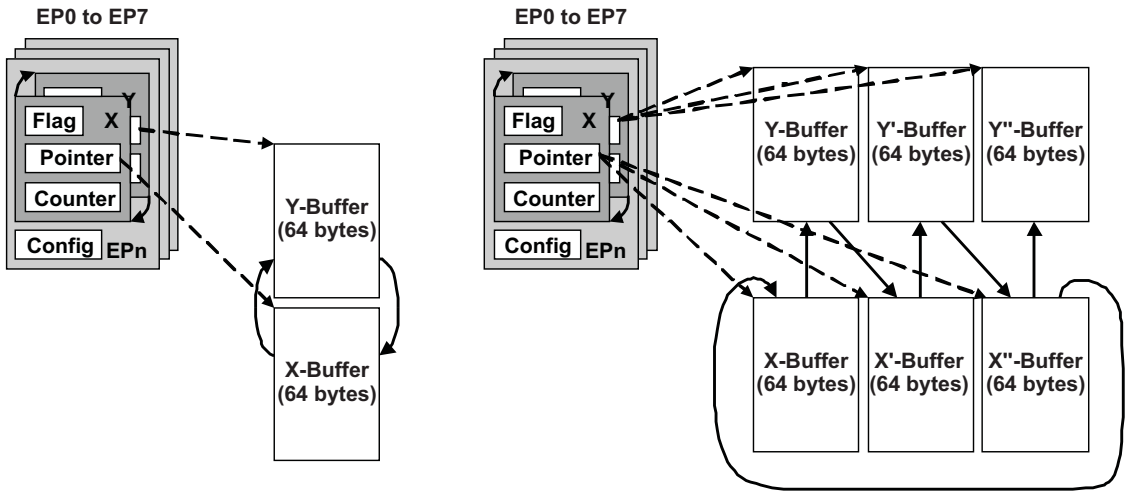

Figure 1-6. Data Buffers and Descriptors

1.2.4.1 USB Serial Interface Engine (SIE)

The SIE logic manages the USB packet protocol requirements for the packets being received and transmitted on the bus. For packets being received, the SIE decodes the packet identifier field (packet ID) to determine the type of packet being received and to ensure the packet ID is valid. For token and data packets being received, the SIE calculates the packet cycle redundancy check (CRC) and compares the value to the CRC contained in the packet to verify that the packet was not corrupted during transmission.

For token and data packets being transmitted, the SIE generates the CRC that is transmitted with the packet. For packets being transmitted, the SIE also generates the synchronization field (SYNC), which is an eight-bit field at the beginning of each packet. In addition, the SIE generates the correct packet ID for all packets being transmitted.

Another major function of the SIE is the overall serial-to-parallel conversion of the data packets being received or transmitted.

1.2.4.2 USB Buffer Manager (UBM)

The USB buffer manager provides the control logic that interfaces the SIE to the USB endpoint buffers.

One of the major functions of the UBM is to decode the USB device address to determine if the USB host is addressing this particular USB device. In addition, the endpoint address field and direction signal are decoded to determine which particular USB endpoint is being addressed. Based on the direction of the USB transaction and the endpoint number, the UBM either writes or reads the data packet to or from the appropriate USB endpoint data buffer.

The TOGGLE bit for each output endpoint configuration register is used by the UBM to track successful output data transactions. If a valid data packet is received and the data packet ID matches the expected packet ID, the TOGGLE bit is toggled. Similarly, the TOGGLE bit for each input endpoint configuration is used by the UBM to track successful input data transactions. If a valid data packet is transmitted, the TOGGLE bit is toggled. If the TOGGLE bit is cleared, a DATA0 packet ID is transmitted in the data packet to the host. If the TOGGLE bit is set, a DATA1 packet ID is transmitted in the data packet to the host. See Section 1.3 regarding details of USB transfers.

1.2.4.3 USB Buffer Memory

The USB buffer memory contains the data buffers for all endpoints and for SETUP packets. In that the buffers for endpoints 1 to 7 are flexible, there are USB buffer configuration registers that define them, and these too are in the USB buffer memory. (Endpoint 0 is defined with a set of registers in the USB control register space.) Storing these in open memory allows for efficient, flexible use, which is advantageous because use of these endpoints is very application-specific.

This memory is implemented as "multiport" memory, in that it can be accessed both by the USB buffer manager and also by the CPU and DMA. The SIE allows CPU or DMA access, but reserves priority. As a result, CPU or DMA access is delayed using wait states if a conflict arises with an SIE access.

When the USB module is disabled (USBEN = 0), the buffer memory behaves like regular RAM. When changing the state of the USBEN bit (enabling or disabling the USB module), the USB buffer memory should not be accessed within four clocks before and eight clocks after changing this bit, as doing so reconfigures the access method to the USB memory.

Accessing of the USB buffer memory by CPU or DMA is only possible if the USB PLL is active. When a host requests suspend condition the application software (for example, USB stack) of client has to switch off the PLL within 10 ms. Note that the MSP430 USB suspend interrupt occurs around 5 ms after the host request.

Each endpoint is defined by a block of six configuration "registers" (based in RAM, they are not true registers in the strict sense of the word). These registers specify the endpoint type, buffer address, buffer size and data packet byte count. They define an endpoint buffer space that is 1904 bytes in size. An additional 24 bytes are allotted to three remaining blocks – the EP0_IN buffer, the EP0_OUT buffer, and the SETUP packet buffer (see Table 1-3).

Table 1-3. USB Buffer Memory Map

Memory	Short Form	Access Type	Address Offset
Start of buffer space	STABUFF	Read/Write	0000h
1904 bytes of configurable buffer space	⋮	Read/Write	⋮
End of buffer space	TOPBUFF	Read/Write	076Fh
Output endpoint_0 buffer	USBOEP0BUF	Read/Write	0770h
		Read/Write	⋮
		Read/Write	0777h
Input endpoint_0 buffer	USBIEP0BUF	Read/Write	0778h
		Read/Write	⋮
		Read/Write	077Fh
Setup Packet Block	USBSUBLK	Read/Write	0780h
		Read/Write	⋮
		Read/Write	0787h

Software can configure each buffer according to the total number of endpoints needed. Single or double buffering of each endpoint is possible.

Unlike the descriptor registers for endpoints 1 to 7, which are defined as memory entries in USB RAM, endpoint 0 is described by a set of four registers (two for output and two for input) in the USB control register set. Endpoint 0 has no base-address register, since these addresses are hardwired. The bit positions have been preserved to provide consistency with endpoint_n (n = 1 to 7).

1.2.4.4 USB Fine Timestamp

The USB module is capable of saving a timestamp associated with particular USB events (see Figure 1-7). This can be useful in compensating for delays in software response. The timestamp values are based on the USB module's internal timer, driven by USBCLK.

Up to four events can be selected to generate the timestamp, selected with the TSESEL bits. When they occur, the value of the USB timer is transferred to the timestamp register USBTSREG, and thus the exact moment of the event is recorded. The trigger options include one of three DMA channels, or a software-driven event. The USB timer cannot be directly accessed by reading.

Furthermore, the value of the USB timer can be used to generate periodic interrupts. Since the USBCLK can have a frequency different from the other system clocks, this gives another option for periodic system interrupts. The UTSEL bits select the divider from the USB clock. UTIE must be set for an interrupt vector to get triggered.

The timestamp register is set to zero on a frame-number-receive event and pseudo-start-of-frame.

TSGEN enables or disables the time stamp generator.

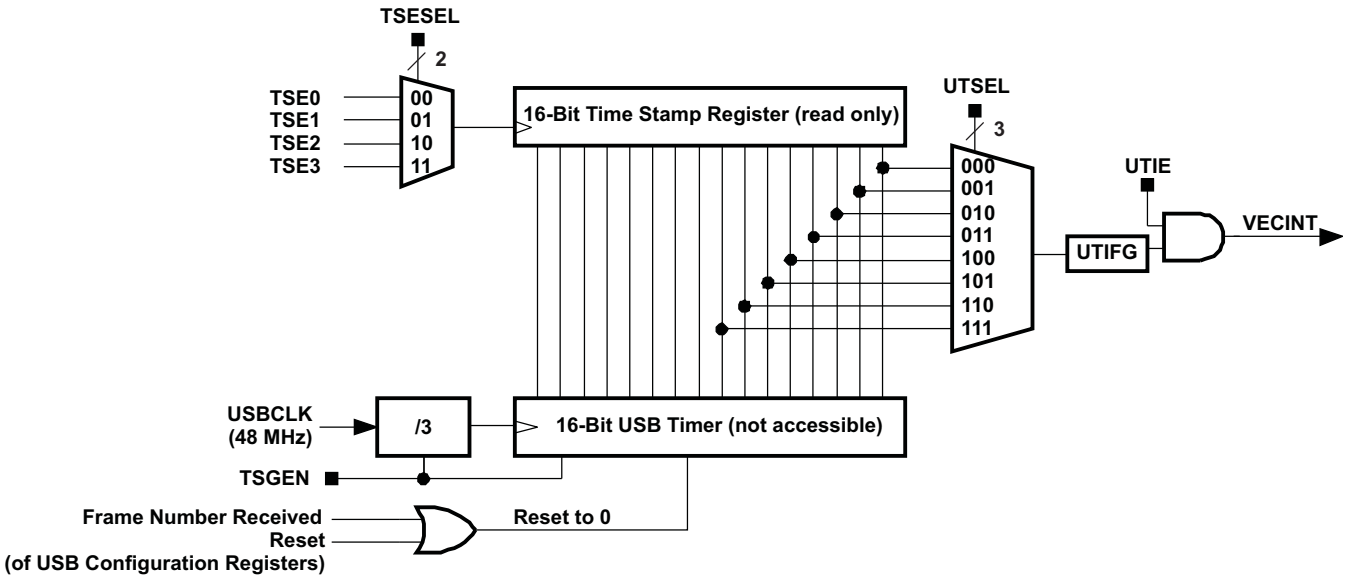

Figure 1-7. USB Timer and Time Stamp Generation

1.2.4.5 Suspend and Resume Logic

The USB suspend and resume logic detects suspend and resume conditions on the USB bus. These events are flagged in SUSRIFG and RESRIFG, respectively, and they fire dedicated interrupts, if the interrupts are enabled (SUSRIE and RESRIE).

The remote wakeup mechanism, in which a USB device can cause the USB host to awaken and resume the device, is triggered by setting the RWUP bit of the USBCTL register.

See Section 1.2.6 for more information.

1.2.4.6 Reset Logic

A PUC resets the USB module logic. When FRSTE = 1, the logic is also reset when a USB reset event occurs on the bus, triggered from the USB host. (A USB reset also sets the RSTRIFG flag.) USB buffer memory is not reset by a USB reset.

1.2.5 USB Vector Interrupts

The USB module uses a single interrupt vector generator register to handle multiple USB interrupts. All USB-related interrupt sources trigger the USBVECINT (also called USBIV) vector, which then contains a 6-bit vector value that identifies the interrupt source. Each of the interrupt sources results in a different offset value read. The interrupt vector returns zero when no interrupt is pending.

Reading the interrupt vector register clears the corresponding interrupt flag and updates its value. The interrupt with highest priority returns the value 0002h; the interrupt with lowest priority returns the value 003Eh when reading the interrupt vector register. Writing to this register clears all interrupt flags.

For each input and output endpoints resides an USB transaction interrupt indication enable. Software may set this bit to define if interrupts are to be flagged in general. To generate an interrupt the corresponding interrupt enable and flag must be set.

Table 1-4. USB Interrupt Vector Generation

USBVECINT Value	Interrupt Source	Interrupt Flag Bit	Interrupt Enable Bit	Indication Enable Bit
0000h	no interrupt	–	–	–
0002h	USB-PWR drop ind.	USBPWRCTL.VUOVLIFG	USBPWRCTL.VUOVLIE	–
0004h	USB-PLL lock error	USBPLLIR.USBPLLOOLIFG	USBPLLIR.USBPLLOOLIE	–
0006h	USB-PLL signal error	USBPLLIR.USBPLLOSIFG	USBPLLIR.USBPLLLOSIE	–
0008h	USB-PLL range error	USBPLLIR.USBPLLOORIFG	USBPLLIR.USBPLLOORIE	–
000Ah	USB-PWR VBUS-on	USBPWRCTL.VBONIFG	USBPWRCTL.VBONIE	–
000Ch	USB-PWR VBUS-off	USBPWRCTL.VBOFFIFG	USBPWRCTL.VBOFFIE	–
000Eh	reserved	–	–	–
0010h	USB timestamp event	USBMAINTL.UTIFG	USBMAINTL.UTIE	–
0012h	Input Endpoint-0	USBIEPIFG.EP0	USBIEPIE.EP0	USBIEPCNFG_0.USBIIE
0014h	Output Endpoint-0	USBOEPIFG.EP0	USBOEPIE.EP0	USBOEPCNFG_0.USBIIE
0016h	RSTR interrupt	USBIFG.RSTRIFG	USBIE.RSTRIE	–
0018h	SUSR interrupt	USBIFG.SUSRIFG	USBIE.SUSRIE	–
001Ah	RESR interrupt	USBIFG.RESRIFG	USBIE.RESRIE	–
001Ch	reserved	–	–	–
001Eh	reserved	–	–	–
0020h	Setup packet received	USBIFG.SETUPIFG	USBIE.SETUPIE	–
0022h	Setup packet overwrite	USBIFG.STPOWIFG	USBIE.STPOWIE	–
0024h	Input Endpoint-1	USBIEPIFG.EP1	USBIEPIE.EP1	USBIEPCNF_1.USBIIE
0026h	Input Endpoint-2	USBIEPIFG.EP2	USBIEPIE.EP2	USBIEPCNF_2.USBIIE
0028h	Input Endpoint-3	USBIEPIFG.EP3	USBIEPIE.EP3	USBIEPCNF_3.USBIIE
002Ah	Input Endpoint-4	USBIEPIFG.EP4	USBIEPIE.EP4	USBIEPCNF_4.USBIIE
002Ch	Input Endpoint-5	USBIEPIFG.EP5	USBIEPIE.EP5	USBIEPCNF_5.USBIIE
002Eh	Input Endpoint-6	USBIEPIFG.EP6	USBIEPIE.EP6	USBIEPCNF_6.USBIIE
0030h	Input Endpoint-7	USBIEPIFG.EP7	USBIEPIE.EP7	USBIEPCNF_7.USBIIE
0032h	Output Endpoint-1	USBOEPIFG.EP1	USBOEPIE.EP1	USBOEPCNF_1.USBIIE
0034h	Output Endpoint-2	USBOEPIFG.EP2	USBOEPIE.EP2	USBOEPCNF_2.USBIIE
0036h	Output Endpoint-3	USBOEPIFG.EP3	USBOEPIE.EP3	USBOEPCNF_3.USBIIE
0038h	Output Endpoint-4	USBOEPIFG.EP4	USBOEPIE.EP4	USBOEPCNF_4.USBIIE
003Ah	Output Endpoint-5	USBOEPIFG.EP5	USBOEPIE.EP5	USBOEPCNF_5.USBIIE
003Ch	Output Endpoint-6	USBOEPIFG.EP6	USBOEPIE.EP6	USBOEPCNF_6.USBIIE
003Eh	Output Endpoint-7	USBOEPIFG.EP7	USBOEPIE.EP7	USBOEPCNF_7.USBIIE

1.2.6 Power Consumption

USB functionality consumes more power than is typically drawn in the MSP430. Since most MSP430 applications are power sensitive, the MSP430 USB module has been designed to protect the battery by ensuring that significant power load only occurs when attached to the bus, allowing power to be drawn from VBUS.

The two components of the USB module that draw the most current are the transceiver and the PLL. The transceiver can consume large amounts of power while transmitting, but in its quiescent state – that is, when not transmitting data – the transceiver actually consumes very little power. This is the amount specified as I_{IDLE}. This amount is so little that the transceiver can be kept active during suspend mode without presenting a problem for bus-powered applications. Fortunately the transceiver always has access to VBUS power when drawing the level of current required for transmitting.

The PLL consumes a larger amount of current. However, it need only be active while connected to the host, and the host can supply the power. When the PLL is disabled (for example, during USB suspend), USBCLK automatically is sourced from the VLO.

1.2.7 Suspend and Resume

All USB devices must support the ability to be suspended into a no-activity state, and later resumed. When suspended, a device is not allowed to consume more than 500uA from the USB's VBUS power rail, if the device is drawing any power from that source. A suspended device must also monitor for a resume event on the bus.

The host initiates a suspend condition by creating a constant idle state on the bus for more than 3.0 ms. It is the responsibility of the software to ensure the device enters its low power suspend state within 10 ms of the suspend condition. The USB specification requires that a suspended bus-powered USB device not draw in excess of 500 µA from the bus.

1.2.7.1 Entering Suspend

When the host suspends the USB device, a suspend interrupt is generated (SUSRIFG). From this point, the software has 10 ms to ensure that no more than 500uA is being drawn from the host via VBUS.

For most applications, the integrated 3.3-V LDO is being used. In this case, the following actions should be taken:

- Disable the PLL by clearing UPLLEN (UPLLEN = 0)
- Limit all current sourced from VBUS that causes the total current sourced from VBUS equal to 500 µA minus the suspend current, $I_{SUSPEND}$ (see the device-specific data sheet).

Disabling the PLL eliminates the largest on-chip draw of power from VBUS. During suspend, the USBCLK is automatically sourced by the VLO (VLOCLK), allowing the USB module to detect resume when it occurs. It is a good idea to also then ensure that the RESRIE bit is also set, so that an interrupt is generated when the host resumes the device. If desired, the high frequency crystal can also be disabled to save additional system power, however it does not contribute to the power from VBUS since it draws power from the DVCC supply.

1.2.7.2 Entering Resume Mode

When the USB device is in a suspended condition, any non-idle signaling, including reset signaling, on the host side is detected by the suspend and resume logic and device operation is resumed. RESRIFG is set, causing an USB interrupt. The interrupt service routine can be used to resume USB operation.

1.3 USB Transfers

The USB module supports control, bulk, and interrupt data transfer types. In accordance with the USB specification, endpoint 0 is reserved for the control endpoint and is bidirectional. In addition to the control endpoint, the USB module is capable of supporting up to 7 input endpoints and 7 output endpoints. These additional endpoints can be configured either as bulk or interrupt endpoints. The software handles all control, bulk, and interrupt endpoint transactions.

1.3.1 Control Transfers

Control transfers are used for configuration, command, and status communication between the host and the USB device. Control transfers to the USB device use input endpoint 0 and output endpoint 0. The three types of control transfers are control write, control write with no data stage, and control read. Note that the control endpoint must be initialized before connecting the USB device to the USB.

1.3.1.1 Control Write Transfer

The host uses a control write transfer to write data to the USB device. A control write transfer consists of a setup stage transaction, at least one output data stage transaction, and an input status stage transaction.

The stage transactions for a control write transfer are:

- Setup stage transaction:
  1. Input endpoint 0 and output endpoint 0 are initialized by programming the appropriate USB endpoint configuration blocks. This entails enabling the endpoint interrupt (USBIIE = 1) and enabling the endpoint (UBME = 1). The NAK bit for both input endpoint 0 and output endpoint 0 must be cleared.
  2. The host sends a setup token packet followed by the setup data packet addressed to output endpoint 0. If the data is received without an error, then the UBM writes the data to the setup data packet buffer, sets the setup stage transaction bit (SETUPIFG = 1) in the USB Interrupt Flag register (USBIFG), returns an ACK handshake to the host, and asserts the setup stage transaction interrupt. Note that as long as SETUPIFG = 1, the UBM returns a NAK handshake for any data stage or status stage transactions regardless of the endpoint 0 NAK or STALL bit values.
  3. The software services the interrupt, reads the setup data packet from the buffer, and then decodes the command. If the command is not supported or invalid, the software should set the STALL bit in the output endpoint 0 configuration register (USBOEPCNFG_0) and the input endpoint 0 configuration register (USBIEPCNFG_0) . This causes the device to return a STALL handshake for any data or status stage transaction. For control write transfers, the packet ID used by the host for the first data packet output is a DATA1 packet ID and the TOGGLE bit must match.

NOTE: When using USBIV, SETUPIFG is cleared upon reading USBIV. In addition, the NAK on input endpoint 0 and output endpoint 0 are also cleared. In this case, the host may send or receive the next setup packet even if MSP430 did not perform the first setup packet. To prevent this, first read the SETUPIFG directly, perform the required setup, and then use the USBIV for further processing.

NOTE: The priority of input endpoint 0 is higher than the setup flag inside USBIV (SETUPIFG). Therefore, if both the USBIEPIFG.EP0 and SETUPIFG are pending, reading USBIV gives the higher priority interrupt (EP0) as opposed to SETUPIFG. Therefore, read SETUPIFG directly, process the pending setup packet, then proceed to read the USBIV.

- Data stage transaction:
  1. The host sends an OUT token packet followed by a data packet addressed to output endpoint 0. If the data is received without an error, the UBM writes the data to the output endpoint buffer (USBOEP0BUF), updates the data count value, toggles the TOGGLE bit, sets the NAK bit, returns an ACK handshake to the host , and asserts the output endpoint interrupt 0 (OEPIFG0).
  2. The software services the interrupt and reads the data packet from the output endpoint buffer. To read the data packet, the software first needs to obtain the data count value inside the USBOEPBCNT_0 register. After reading the data packet, the software should clear the NAK bit to allow the reception of the next data packet from the host.
  3. If the NAK bit is set when the data packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the data packet is received, the UBM simply returns a STALL handshake to the host. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host.
- Status stage transaction:
  1. For input endpoint 0, the software updates the data count value to zero, sets the TOGGLE bit, then clears the NAK bit to enable the data packet to be sent to the host. Note that for a status stage transaction, a null data packet with a DATA1 packet ID is sent to the host.
  2. The host sends an IN token packet addressed to input endpoint 0. After receiving the IN token, the UBM transmits a null data packet to the host. If the data packet is received without errors by the host, then an ACK handshake is returned. The UBM then toggles the TOGGLE bit and sets the NAK bit.

3. If the NAK bit is set when the IN token packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the IN token packet is received, the UBM simply returns a STALL handshake to the host. If no handshake packet is received from the host, then the UBM prepares to retransmit the same data packet again.

1.3.1.2 Control Write Transfer with No Data Stage Transfer

The host uses a control write transfer to write data to the USB device. A control write with no data stage transfer consists of a setup stage transaction and an input status stage transaction. For this type of transfer, the data to be written to the USB device is contained in the two byte value field of the setup stage transaction data packet.

The stage transactions for a control write transfer with no data stage transfer are:

- Setup stage transaction:
  1. Input endpoint 0 and output endpoint 0 are initialized by programming the appropriate USB endpoint configuration blocks. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt (USBIIE = 1), initializing the TOGGLE bit, enabling the endpoint (UBME = 1). The NAK bit for both input endpoint 0 and output endpoint 0 must be cleared.
  2. The host sends a setup token packet followed by the setup data packet addressed to output endpoint 0. If the data is received without an error then the UBM writes the data to the setup data packet buffer, sets the setup stage transaction (SETUP) bit in the USB status register, returns an ACK handshake to the host, and asserts the setup stage transaction interrupt. Note that as long as the setup transaction (SETUP) bit is set, the UBM returns a NAK handshake for any data stage or status stage transaction regardless of the endpoint 0 NAK or STALL bit values.
  3. The software services the interrupt and reads the setup data packet from the buffer then decodes the command. If the command is not supported or invalid, the software should set the STALL bits in the output endpoint 0 and the input endpoint 0 configuration registers before clearing the setup stage transaction (SETUP) bit. This causes the device to return a STALL handshake for data or status stage transactions. After reading the data packet and decoding the command, the software should clear the interrupt, which automatically clears the setup stage transaction status bit.

NOTE: When using USBIV, the SETUPIFG is cleared upon reading USBIV. In addition, the NAK on input endpoint 0 and output endpoint 0 is also cleared. In this case, the host may send or receive the next setup packet even if MSP430 did not perform the first setup packet. To prevent this, first read the SETUPIFG directly, perform the required setup, and then use the USBIV for further processing.

NOTE: The priority of input endpoint 0 is higher than setup flag inside USBIV. Therefore, if both the USBIEPIFG.EP0 and SETUPIFG are pending, reading the USBIV gives the higher priority interrupt (EP0) as opposed to the SETUPIFG. Therefore, read SETUPIFG directly, process the pending setup packet, then proceed to read the USBIV.

- Status stage transaction:
  1. For input endpoint 0, the software updates the data count value to zero, sets the TOGGLE bit, then clears the NAK bit to enable the data packet to be sent to the host. Note that for a status stage transaction a null data packet with a DATA1 packet ID is sent to the host.
  2. The host sends an IN token packet addressed to input endpoint 0. After receiving the IN token, the UBM transmits a null data packet to the host. If the data packet is received without errors by the host, then an ACK handshake is returned. The UBM then toggles the TOGGLE bit, sets the NAK bit, and asserts the endpoint interrupt.
  3. If the NAK bit is set when the IN token packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the IN token packet is received, the UBM simply returns a STALL handshake to the host. If no handshake packet is received from the host, then the UBM prepares to retransmit the same data packet again.

1.3.1.3 Control Read Transfer

The host uses a control read transfer to read data from the USB device. A control read transfer consists of a setup stage transaction, at least one input data stage transaction and an output status stage transaction.

The stage transactions for a control read transfer are:

- Setup stage transaction:
  1. Input endpoint 0 and output endpoint 0 are initialized by programming the appropriate USB endpoint configuration blocks. This entails enabling the endpoint interrupt (USBIIE = 1) and enabling the endpoint (UBME = 1). The NAK bit for both input endpoint 0 and output endpoint 0 must be cleared.
  2. The host sends a setup token packet followed by the setup data packet addressed to output endpoint 0. If the data is received without an error, then the UBM writes the data to the setup buffer, sets the setup stage transaction (SETUP) bit in the USB status register, returns an ACK handshake to the host, and asserts the setup stage transaction interrupt. Note that as long as the setup transaction (SETUP) bit is set, the UBM returns a NAK handshake for any data stage or status stage transactions regardless of the endpoint 0 NAK or STALL bit values.
  3. The software services the interrupt and reads the setup data packet from the buffer then decodes the command. If the command is not supported or invalid, the software should set the STALL bits in the output endpoint 0 and the input endpoint 0 configuration registers before clearing the setup stage transaction (SETUP) bit. This causes the device to return a STALL handshake for a data stage or status stage transactions. After reading the data packet and decoding the command, the software should clear the interrupt, which automatically clears the setup stage transaction status bit. The software should also set the TOGGLE bit in the input endpoint 0 configuration register. For control read transfers, the packet ID used by the host for the first input data packet is a DATA1 packet ID.

NOTE: When using USBIV, the SETUPIFG is cleared upon reading USBIV. In addition, it also the clears NAK on input endpoint 0 and output endpoint 0. In this case, the host may send or receive the next setup packet even if MSP430 did not perform the first setup packet. To prevent this, first read the SETUPIFG directly, perform the required setup, and then use the USBIV for further processing.

NOTE: The priority of input endpoint 0 is higher than the setup flag inside USBIV. Therefore, if both the USBIEPIFG.EP0 and SETUPIFG are pending, reading the USBIV gives the higher priority interrupt (EP0) as opposed to the SETUPIFG. Therefore, read SETUPIFG directly, process the pending setup packet, then proceed to read the USBIV.

- Data stage transaction:
  1. The data packet to be sent to the host is written to the input endpoint 0 buffer by the software. The software also updates the data count value then clears the input endpoint 0 NAK bit to enable the data packet to be sent to the host.
  2. The host sends an IN token packet addressed to input endpoint 0. After receiving the IN token, the UBM transmits the data packet to the host. If the data packet is received without errors by the host, then an ACK handshake is returned. The UBM sets the NAK bit and asserts the endpoint interrupt.
  3. The software services the interrupt and prepares to send the next data packet to the host.
  4. If the NAK bit is set when the IN token packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the IN token packet is received, the UBM simply returns a STALL handshake to the host. If no handshake packet is received from the host, then the UBM prepares to retransmit the same data packet again.
  5. The software continues to send data packets until all data has been sent to the host.
- Status stage transaction:
  1. For output endpoint 0, the software sets the TOGGLE bit, then clears the NAK bit to enable the data packet to be sent to the host. Note that for a status stage transaction a null data packet with a DATA1 packet ID is sent to the host.
  2. The host sends an OUT token packet addressed to output endpoint 0. If the data packet is received

without an error then the UBM updates the data count value, toggles the TOGGLE bit, sets the NAK bit, returns an ACK handshake to the host, and asserts the endpoint interrupt.

3. The software services the interrupt. If the status stage transaction completed successfully, then the software should clear the interrupt and clear the NAK bit.
4. If the NAK bit is set when the input data packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the in data packet is received, the UBM simply returns a STALL handshake to the host. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host.

1.3.2 Interrupt Transfers

The USB module supports interrupt data transfers both to and from the host. Devices that need to send or receive a small amount of data with a specified service period are best served by the interrupt transfer type. Input endpoints 1 through 7 and output endpoints 1 through 7 can be configured as interrupt endpoints.

1.3.2.1 Interrupt OUT Transfer

The steps for an interrupt OUT transfer are:

1. The software initializes one of the output endpoints as an output interrupt endpoint by programming the appropriate endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint, and clearing the NAK bit.
2. The host sends an OUT token packet followed by a data packet addressed to the output endpoint. If the data is received without an error then the UBM writes the data to the endpoint buffer, updates the data count value, toggles the toggle bit, sets the NAK bit, returns an ACK handshake to the host, and asserts the endpoint interrupt.
3. The software services the interrupt and reads the data packet from the buffer. To read the data packet, the software first needs to obtain the data count value. After reading the data packet, the software should clear the interrupt and clear the NAK bit to allow the reception of the next data packet from the host.
4. If the NAK bit is set when the data packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the data packet is received, the UBM simply returns a STALL handshake to the host. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host device.

In double buffer mode, the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM writes the data packet to the X buffer. If the toggle bit is a 1, the UBM writes the data packet to the Y buffer. When a data packet is received, the software could determine which buffer contains the data packet by reading the toggle bit. However, when using double buffer mode, the possibility exists for data packets to be received and written to both the X and Y buffer before the software responds to the endpoint interrupt. In this case, simply using the toggle bit to determine which buffer contains the data packet would not work. Hence, in double buffer mode, the software should read the X buffer NAK bit, the Y buffer NAK bit, and the toggle bits to determine the status of the buffers.

1.3.2.2 Interrupt IN Transfer

The steps for an interrupt IN transfer are:

1. The software initializes one of the input endpoints as an input interrupt endpoint by programming the appropriate endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint, and setting the NAK bit.
2. The data packet to be sent to the host is written to the buffer by the software. The software also updates the data count value then clears the NAK bit to enable the data packet to be sent to the host.
3. The host sends an IN token packet addressed to the input endpoint. After receiving the IN token, the UBM transmits the data packet to the host. If the data packet is received without errors by the host, then an ACK handshake is returned. The UBM then toggles the toggle bit, sets the NAK bit, and asserts the endpoint interrupt.

4. The software services the interrupt and prepares to send the next data packet to the host.
5. If the NAK bit is set when the in token packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the IN token packet is received, the UBM simply returns a STALL handshake to the host. If no handshake packet is received from the host, then the UBM prepares to retransmit the same data packet again.

In double buffer mode, the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM reads the data packet from the X buffer. If the toggle bit is a 1, the UBM reads the data packet from the Y buffer.

1.3.3 Bulk Transfers

The USB module supports bulk data transfers both to and from the host. Devices that need to send or receive a large amount of data without a suitable bandwidth are best served by the bulk transfer type. In endpoints 1 through 7 and out endpoints 1 through 7 can all be configured as bulk endpoints.

1.3.3.1 Bulk OUT Transfer

The steps for a bulk OUT transfer are:

1. The software initializes one of the output endpoints as an output bulk endpoint by programming the appropriate endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint, and clearing the NAK bit.
2. The host sends an out token packet followed by a data packet addressed to the output endpoint. If the data is received without an error then the UBM writes the data to the endpoint buffer, updates the data count value, toggles the toggle bit, sets the NAK bit, returns an ACK handshake to the host, and asserts the endpoint interrupt.
3. The software services the interrupt and reads the data packet from the buffer. To read the data packet, the software first needs to obtain the data count value. After reading the data packet, the software should clear the interrupt and clear the NAK bit to allow the reception of the next data packet from the host.
4. If the NAK bit is set when the data packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the data packet is received, the UBM simply returns a STALL handshake to the host. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host.

In double buffer mode, the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM writes the data packet to the X buffer. If the toggle bit is a 1, the UBM writes the data packet to the Y buffer. When a data packet is received, the software could determine which buffer contains the data packet by reading the toggle bit. However, when using double buffer mode, the possibility exists for data packets to be received and written to both the X and Y buffer before the software responds to the endpoint interrupt. In this case, simply using the toggle bit to determine which buffer contains the data packet would not work. Hence, in double buffer mode, the software should read the X buffer NAK bit, the Y buffer NAK bit, and the toggle bits to determine the status of the buffers.

1.3.3.2 Bulk IN Transfer

The steps for a bulk IN transfer are:

1. The software initializes one of the input endpoints as an input bulk endpoint by programming the appropriate endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint, and setting the NAK bit.
2. The data packet to be sent to the host is written to the buffer by the software. The software also updates the data count value then clears the NAK bit to enable the data packet to be sent to the host.
3. The host sends an IN token packet addressed to the input endpoint. After receiving the IN token, the UBM transmits the data packet to the host. If the data packet is received without errors by the host, then an ACK handshake is returned. The UBM then toggles the toggle bit, sets the NAK bit, and asserts the endpoint interrupt.

4. The software services the interrupt and prepares to send the next data packet to the host.
5. If the NAK bit is set when the in token packet is received, the UBM simply returns a NAK handshake to the host. If the STALL bit is set when the In token packet is received, the UBM simply returns a STALL handshake to the host. If no handshake packet is received from the host, then the UBM prepares to retransmit the same data packet again.

In double buffer mode , the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM reads the data packet from the X buffer. If the toggle bit is a 1, the UBM reads the data packet from the Y buffer.

1.4 USB Registers

The USB register space is subdivided into configuration registers, control registers, and USB buffer memory.

The configuration and control registers are physical registers located in peripheral memory, while the buffer memory is implemented in RAM. See the device-specific data sheet for base addresses of these register groupings.

The USB control registers can be written only while the USB module is enabled.

When the USB module is disabled, it no longer uses the RAM buffer memory. This memory then behaves as a 2KB RAM block and can be used by the CPU or DMA without any limitation.

1.4.1 USB Configuration Registers

The configuration registers control the hardware functions needed to make a USB connection, including the PHY, PLL, and LDOs.

Access to the configuration registers is allowed or disallowed using the USBKEYPID register. Writing the proper value (9628h) unlocks the configuration registers and enables access. Writing any other value disables access while leaving the values of the registers intact. Locking should be done intentionally after the configuration is finished. Read access is available without the need to write to the USBKEYPID register.

The configuration registers are listed in Table 1-5. All addresses are expressed as offsets; the base address can be found in the device-specific data sheet.

All registers are byte and word accessible.

Table 1-5. USB Configuration Registers

Offset	Acronym	Register Name	Type	Reset	Section
00h	USBKEYPID	USB controller key and ID register	Read/Write	0000h	Section 1.4.1.1
02h	USBCNF	USB controller configuration register	Read/Write	0000h	Section 1.4.1.2
04h	USBPHYCTL	USB-PHY control register	Read/Write	0000h	Section 1.4.1.3
08h	USBPWRCTL	USB-PWR control register	Read/Write	1850h	Section 1.4.1.4
10h	USBPLLCTL	USB-PLL control register	Read/Write	0000h	Section 1.4.1.5
12h	USBPLLDIVB	USB-PLL divider buffer register	Read/Write	0000h	Section 1.4.1.6
14h	USBPLLIR	USB-PLL interrupt register	Read/Write	0000h	Section 1.4.1.7

1.4.1.1 USBKEYPID Register

USB Key Register

Figure 1-8. USBKEYPID Register

15	14	13	12	11	10	9	8
USBKEY							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
USBKEY							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 1-6. USBKEYPID Register Description

Bit	Field	Type	Reset	Description
15-0	USBKEY	RW	00h	Key register. Must be written with a value of 9628h to be recognized as a valid key. This "unlocks" the configuration registers. If written with any other value, the registers become "locked". Reads back as A528h if the registers are unlocked.

1.4.1.2 USBCNF Register

USB Module Configuration Register

This register can be modified only when USBKEYPID is unlocked.

Figure 1-9. USBCNF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved			FNTEN	BLKRDY	PUR_IN	PUR_EN	USB_EN
r0	r0	r0	rw-0	rw-0	r	rw-0	rw-0

Can be modified only when USBKEYPID is unlocked.

Table 1-7. USBCNF Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always reads as 0.
4	FNTEN	RW	0h	Frame number receive trigger enable for DMA transfers 0b = Frame number receive trigger is blocked. 1b = Frame number receive trigger is gated through to DMA.
3	BLKRDY	RW	0h	Block transfer ready signaling for DMA transfers 0b = DMA triggering is disabled. 1b = DMA is triggered whenever the USB bus interface can accept new write transfers.
2	PUR_IN	R	0h	PUR input value. This bit reflects the input value present on PUR. This bit may be used as an indication to start a USB based boot loading program (USB-BSL). The PUR input logic is powered by VUSB. PUR_IN returns zero when VUSB is zero.
1	PUR_EN	RW	0h	PUR pin enable 0b = PUR pin is in high-impedance state 1b = PUR pin is driven high
0	USB_EN	RW	0h	USB module enable 0b = USB module is disabled 1b = USB module is enabled

1.4.1.3 USBPHYCTL Register

USB-PHY Control Register

This register can be modified only when USBKEYPID is unlocked.

Figure 1-10. USBPHYCTL Register

15	14	13	12	11	10	9	8
Reserved						Reserved	PUIPE
r0	r0	r0	r0	r0	r0	rw-0	rw-0
7	6	5	4	3	2	1	0
PUSEL	Reserved	PUOPE	Reserved	PUIN1	PUIN0	PUOUT1	PUOUT0
rw-0	r	rw-0	rw-0	r	r	rw-0	rw-0

Can be modified only when USBKEYPID is unlocked.

Table 1-8. USBPHYCTL Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved. Always reads as 0.
9	Reserved	RW	0h	Reserved. Always write as 0.
8	PUIPE	RW	0h	PU input enable. This bit is valid only when PUSEL = 0. 0b = PU.0 and PU.1 inputs are disabled 1b = PU.0 and PU.1 inputs are enabled
7	PUSEL	RW	0h	USB port function select. This bit selects the function of the PU.0/DP and PU.1/DM pins. 0b = PU.0 and PU.1 function selected (general purpose I/O) 1b = DP and DM function selected (USB terminals)
6	Reserved	R	0h	Reserved. Always reads as 0.
5	PUOPE	RW	0h	PU output enable. This bit is valid only when PUSEL = 0. 0b = PU.0 and PU.1 outputs are disabled 1b = PU.0 and PU.1 outputs are enabled.
4	Reserved	RW	0h	Reserved. Always write as 0.
3	PUIN1	R	0h	PU.1 input data. This bit reflects the logic value on the PU.1 terminal when PUIPE = 1.
2	PUIN0	R	0h	PU.0 input data. This bit reflects the logic value on the PU.0 terminal when PUIPE = 1.
1	PUOUT1	RW	0h	PU.1 output data. This bits defines the value of the PU.1 pin when PUOPE = 1.
0	PUOUT0	RW	0h	PU.0 output data. This bits defines the value of the PU.0 pin when PUOPE = 1.

1.4.1.4 USBPWRCTL Register

USB-Power Control Register

This register can be modified only when USBKEYPID is unlocked.

Figure 1-11. USBPWRCTL Register

15	14	13	12	11	10	9	8
Reserved			SLDOEN	VUSBEN	VBOFFIE	VBONIE	VUOVLIE
r0	r0	r0	rw-1	rw-1	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved	SLDOAON	OVLAOFF	USBDETEN	USBBGVBV	VBOFFIFG	VBONIFG	VUOVLIFG
r0	rw-1	rw-0	rw-1	r	rw-0	rw-0	rw-0

Can be modified only when USBKEYPID is unlocked.

Table 1-9. USBPWRCTL Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12	SLDOEN	RW	1h	1.8-V (secondary) LDO enable. When set, the LDO is enabled. Can be cleared only if SLDOAON = 0. 0b = 1.8-V LDO is disabled 1b = 1.8-V LDO is enabled
11	VUSBEN	RW	1h	3.3-V LDO enable. When set, the LDO is enabled. 0b = 3.3-V LDO is disabled 1b = 3.3-V LDO is enabled
10	VBOFFIE	RW	0h	VBUS "going OFF" interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
9	VBONIE	RW	0h	VBUS "coming ON" interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
8	VUOVLIE	RW	0h	VUSB overload indication interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
7	Reserved	R	0h	Reserved. Always reads as 0.
6	SLDOAON	RW	1h	1.8-V LDO auto-on enable 0b = LDO must be turned on manually using SLDOEN 1b = A "VBUS coming on" enables the secondary LDO, but SLDOEN is not set automatically.
5	OVLAOFF	RW	0h	LDO overload auto-off enable 0b = During an overload on the 3.3-V LDO, the LDO automatically enters current-limiting mode and stays there until the condition stops. 1b = An overload indication clears the VUSBEN bit.
4	USBDETEN	RW	1h	Enable bit for VBUS on and off events. 0b = USB module does not detect USB-PWR VBUS on and off events 1b = USB module does detect USB-PWR VBUS on and off events
3	USBBGVBV	R	0h	VBUS valid 0b = VBUS is not valid yet 1b = VBUS is valid and within bounds

Table 1-9. USBPWRCTL Register Description (continued)

Bit	Field	Type	Reset	Description
2	VBOFFIFG	RW	0h	VBUS "going OFF" interrupt flag. This bit indicates that VBUS fell below the launch voltage. It is automatically cleared when the corresponding vector of the USB interrupt vector register is read, or if a value is written to the interrupt vector register. 0b = VBUS did not fall below the launch voltage. 1b = VBUS fell below the launch voltage.
1	VBONIFG	RW	0h	VBUS "coming ON" interrupt flag. This bit indicates that VBUS rose above the launch voltage. This bit is automatically cleared when the corresponding vector of the USB interrupt vector register is read, or if a value is written to the interrupt vector register. 0b = VUSB did not rise above the launch voltage. 1b = VUSB rose above the launch voltage.
0	VUOVLIFG	RW	0h	VUSB overload interrupt flag. This bit indicates that the 3.3-V LDO entered an overload condition. 0b = No overload condition detected. 1b = Overload condition detected.

1.4.1.5 USBPLLCTL Register

USB-PLL Control Register

This register can be modified only when USBKEYPID is unlocked.

Figure 1-12. USBPLLCTL Register

15	14	13	12	11	10	9	8
Reserved						UPFDEN	UPLLEN
r0	r0	r0	r-0	r0	r0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCLKSEL		Reserved					
rw-0	rw-0	r0	r0	r0	r0	r0	r0

Can be modified only when USBKEYPID is unlocked.

Table 1-10. USBPLLCTL Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved. Always reads as 0.
9	UPFDEN	RW	0h	Phase frequency discriminator (PFD) enable 0b = PFD is disabled 1b = PFD is enabled
8	UPLLEN	RW	0h	PLL enable 0b = PLL is disabled 1b = PLL is enabled
7-6	UCLKSEL	RW	0h	USB module clock select. Must always be written with 00. 00b = PLLCLK (default) 01b = Reserved 10b = Reserved 11b = Reserved
5-0	Reserved	R	0h	Reserved. Always reads as 0.

1.4.1.6 USBPLLDIVB Register

USB-PLL Clock Divider Buffer Register

This register can be modified only when USBKEYPID is unlocked.

Figure 1-13. USBPLLDIVB Register

15	14	13	12	11	10	9	8
Reserved					UPQB		
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved		UPMB					
r0	r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBKEYPID is unlocked.

Table 1-11. USBPLLDIVB Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10-8	UPQB	RW	0h	PLL pre-scale divider buffer register. These bits select the pre-scale division value. The value of this register is transferred to UPQB as soon it is written. 000b = $f_{UPD} = f_{REF}$ 001b = $f_{UPD} = f_{REF} / 2$ 010b = $f_{UPD} = f_{REF} / 3$ 011b = $f_{UPD} = f_{REF} / 4$ 100b = $f_{UPD} = f_{REF} / 6$ 101b = $f_{UPD} = f_{REF} / 8$ 110b = $f_{UPD} = f_{REF} / 13$ 111b = $f_{UPD} = f_{REF} / 16$
7-6	Reserved	R	0h	Reserved. Always reads as 0.
5-0	UPMB	RW	0h	USB PLL feedback divider buffer register. These bits select the value of the feedback divider. The value of this register is transferred to UPMB automatically when UPQB is written. 000000b = Feedback division rate: 1 000001b = Feedback division rate: 2 ⋮ 111111b = Feedback division rate: 64

1.4.1.7 USBPLLIR Register

USB-PLL Interrupt Register

This register can be modified only when USBKEYPID is unlocked.

Figure 1-14. USBPLLIR Register

15	14	13	12	11	10	9	8
Reserved					USBOORIE	USBLOSIE	USBOOLIE
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved					USBOORIFG	USBLOSIFG	USBOOLIFG
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0

Can be modified only when USBKEYPID is unlocked.

Table 1-12. USBPLLIR Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10	USBOORIE	RW	0h	PLL out-of-range interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
9	USBLOSIE	RW	0h	PLL loss-of-signal interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
8	USBOOLIE	RW	0h	PLL out-of-lock interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
7-3	Reserved	R	0h	Reserved. Always reads as 0.
2	USBOORIFG	RW	0h	PLL out-of-range interrupt flag 0b = No interrupt pending 1b = Interrupt pending
1	USBLOSIFG	RW	0h	PLL loss-of-signal interrupt flag 0b = No interrupt pending 1b = Interrupt pending
0	USBOOLIFG	RW	0h	PLL out-of-lock interrupt flag 0b = No interrupt pending 1b = Interrupt pending

1.4.2 USB Control Registers

The control registers affect core USB operations that are fundamental for any USB connection. This includes control endpoint 0, interrupts, bus address and frame, and timestamps. Control of endpoints other than zero are found in the operation registers. Unlike the operation registers, the control registers are actual physical registers, whereas the operation registers exist in RAM, which can be re-allocated to general-purpose use.

The control registers are listed in Table 1-13. All addresses are expressed as offsets; the base address can be found in the device-specific data sheet.

All registers are byte and word accessible.

Table 1-13. USB Control Registers

Offset	Acronym	Register Name	Type	Reset	Section
00h	USBIEPCNF_0	Input endpoint_0: Configuration	Read/Write	00h	Section 1.4.2.1
01h	USBIEPBCNT_0	Input endpoint_0: Byte Count	Read/Write	80h	Section 1.4.2.2
02h	USBOEPCNFG_0	Output endpoint_0: Configuration	Read/Write	00h	Section 1.4.2.3
03h	USBOEPBCNT_0	Output endpoint_0: Byte count	Read/Write	00h	Section 1.4.2.4
0Eh	USBIEPIE	Input endpoint interrupt enables	Read/Write	00h	Section 1.4.2.5
0Fh	USBOEPIE	Output endpoint interrupt enables	Read/Write	00h	Section 1.4.2.6
10h	USBIEPIFG	Input endpoint interrupt flags	Read/Write	00h	Section 1.4.2.7
11h	USBOEPIFG	Output endpoint interrupt flags	Read/Write	00h	Section 1.4.2.8
12h	USBVECINT or USBIV	Vector interrupt register	Read/Write	0000h	Section 1.4.2.9
16h	USBMAINT	Timestamp maintenance register	Read/Write	0000h	Section 1.4.2.10
18h	USBTSREG	Timestamp register	Read/Write	0000h	Section 1.4.2.11
1Ah	USBFN	USB frame number	Read only	0000h	Section 1.4.2.12
1Ch	USBCTL	USB control register	Read/Write	00h	Section 1.4.2.13
1Dh	USBIE	USB interrupt enable register	Read/Write	00h	Section 1.4.2.14
1Eh	USBIFG	USB interrupt flag register	Read/Write	00h	Section 1.4.2.15
1Fh	USBFUNADR	Function address register	Read/Write	00h	Section 1.4.2.16

1.4.2.1 USBIEPCNF_0 Register

USB Input Endpoint-0 Configuration Register

Figure 1-15. USBIEPCNF_0 Register

7	6	5	4	3	2	1	0
UBME	Reserved	TOGGLE	Reserved	STALL	USBIIE	Reserved	
rw-0	r0	r-0	r0	rw-0	rw-0	r0	r0

Can be modified only when USBEN = 1.

Table 1-14. USBIEPCNF_0 Register Description

Bit	Field	Type	Reset	Description
7	UBME	RW	0h	UBM in endpoint-0 enable 0b = UBM cannot use this endpoint 1b = UBM can use this endpoint
6	Reserved	R	0h	Reserved. Always reads as 0.
5	TOGGLE	R	0h	Toggle bit. Reads as 0, because the configuration endpoint does not need to toggle.
4	Reserved	R	0h	Reserved. Always reads as 0.
3	STALL	RW	0h	USB stall condition. When set, hardware automatically returns a stall handshake to the USB host for any transaction transmitted from endpoint-0. The stall bit is cleared automatically by the next setup transaction. 0b = Indicates no stall 1b = Indicates stall
2	USBIIE	RW	0h	USB transaction interrupt indication enable. Software may set this bit to define if interrupts are to be flagged in general. To generate an interrupt the corresponding interrupt flag must be set (IEPIE). 0b = Corresponding interrupt flag is not set 1b = Corresponding interrupt flag is set
1-0	Reserved	R	0h	Reserved. Always reads as 0.

1.4.2.2 USBIEPBCNT_0 Register

USB Input Endpoint-0 Byte Count Register

Figure 1-16. USBIEPBCNT_0 Register

7	6	5	4	3	2	1	0
NAK	Reserved			CNT			
rw-0	r0	r0	r0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-15. USBIEPBCNT_0 Register Description

Bit	Field	Type	Reset	Description
7	NAK	RW	0h	No acknowledge status bit. This bit is set by the UBM at the end of a successful USB IN transaction from endpoint-0, to indicate that the EP-0 IN buffer is empty. When this bit is set, all subsequent transactions from endpoint-0 result in a NAK handshake response to the USB host. To re-enable this endpoint to transmit another data packet to the host, this bit must be cleared by software. 0b = Buffer contains a valid data packet for host device 1b = Buffer is empty (Host-In request receives a NAK)
6-4	Reserved	R	0h	Reserved. Always reads as 0.
3-0	CNT	RW	0h	Byte count. The In_EP-0 buffer data count value should be set by software when a new data packet is written to the buffer. This four-bit value contains the number of bytes in the data packet. 0000b to 1000b are valid numbers for 0 to 8 bytes to be sent. 1001b to 1111b are reserved values (if used, defaults to 8).

1.4.2.3 USBOEPCNFG_0 Register

USB Output Endpoint-0 Configuration Register

Figure 1-17. USBOEPCNFG_0 Register

7	6	5	4	3	2	1	0
UBME	Reserved	TOGGLE	Reserved	STALL	USBIIE	Reserved	
rw-0	r0	r-0	r0	rw-0	rw-0	r0	r0

Can be modified only when USBEN = 1

Table 1-16. USBOEPCNFG_0 Register Description

Bit	Field	Type	Reset	Description
7	UBME	RW	0h	UBM out Endpoint-0 enable 0b = UBM cannot use this endpoint 1b = UBM can use this endpoint
6	Reserved	R	0h	Reserved. Always reads as 0.
5	TOGGLE	RW	0h	Toggle bit. Reads as 0, because the configuration endpoint does not need to toggle.
4	Reserved	R	0h	Reserved. Always reads as 0.
3	STALL	RW	0h	USB stall condition. When set, hardware automatically returns a stall handshake to the USB host for any transaction transmitted into endpoint-0. The stall bit is cleared automatically by the next setup transaction. 0b = Indicates no stall 1b = Indicates stall
2	USBIIE	RW	0h	USB transaction interrupt indication enable. Software may set this bit to define if interrupts are to be flagged in general. To generate an interrupt the corresponding interrupt flag must be set (OEPIE). 0b = Corresponding interrupt flag will not be set 1b = Corresponding interrupt flag will be set
1-0	Reserved	R	0h	Reserved. Always reads as 0.

1.4.2.4 USBOEPBCNT_0 Register

USB Output Endpoint-0 Byte Count Register

Figure 1-18. USBOEPBCNT_0 Register

7	6	5	4	3	2	1	0
NAK	Reserved			CNT			
rw-0	r0	r0	r0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-17. USBOEPBCNT_0 Register Description

Bit	Field	Type	Reset	Description
7	NAK	RW	0h	No acknowledge status bit. This bit is set by the UBM at the end of a successful USB out transaction into endpoint-0, to indicate that the EP-0 buffer contains a valid data packet and that the buffer data count value is valid. When this bit is set, all subsequent transactions to endpoint-0 result in a NAK handshake response to the USB host. To re-enable this endpoint to receive another data packet from the host, this bit must be cleared by software. 0b = No valid data in the buffer. The buffer is ready to receive a host OUT transaction 1b = The buffer contains a valid packet from the host that has not been picked up. (Any subsequent Host-Out requests receive a NAK.)
6-4	Reserved	R	0h	Reserved. Always reads as 0.
3-0	CNT	RW	0h	Byte count. This data count value is set by the UBM when a new data packet is received by the buffer for the out endpoint-0. The four-bit value contains the number of bytes received in the data buffer. 0000b to 1000b are valid numbers for 0 to 8 received bytes 1001b to 1111b are reserved values

1.4.2.5 USBIEPIE Register

USB Input Endpoint Interrupt Enable Register

Figure 1-19. USBIEPIE Register

7	6	5	4	3	2	1	0
IEPIE7	IEPIE6	IEPIE5	IEPIE4	IEPIE3	IEPIE2	IEPIE1	IEPIE0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-18. USBIEPIE Register Description

Bit	Field	Type	Reset	Description
7	IEPIE7	RW	0h	Input endpoint interrupt enable 7. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
6	IEPIE6	RW	0h	Input endpoint interrupt enable 6. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
5	IEPIE5	RW	0h	Input endpoint interrupt enable 5. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
4	IEPIE4	RW	0h	Input endpoint interrupt enable 4. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
3	IEPIE3	RW	0h	Input endpoint interrupt enable 3. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
2	IEPIE2	RW	0h	Input endpoint interrupt enable 2. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
1	IEPIE1	RW	0h	Input endpoint interrupt enable 1. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt

Table 1-18. USBIEPIE Register Description (continued)

Bit	Field	Type	Reset	Description
0	IEPIE0	RW	0h	Input endpoint interrupt enable 0. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt

1.4.2.6 USBOEPIE Register

USB Output Endpoint Interrupt Enable Register

Figure 1-20. USBOEPIE Register

7	6	5	4	3	2	1	0
OEPIE7	OEPIE6	OEPIE5	OEPIE4	OEPIE3	OEPIE2	OEPIE1	OEPIE0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-19. USBOEPIE Register Description

Bit	Field	Type	Reset	Description
7	OEPIE7	RW	0h	Output endpoint interrupt enable 7. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
6	OEPIE6	RW	0h	Output endpoint interrupt enable 6. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
5	OEPIE5	RW	0h	Output endpoint interrupt enable 5. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
4	OEPIE4	RW	0h	Output endpoint interrupt enable 4. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
3	OEPIE3	RW	0h	Output endpoint interrupt enable 3. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
2	OEPIE2	RW	0h	Output endpoint interrupt enable 2. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt
1	OEPIE1	RW	0h	Output endpoint interrupt enable 1. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt

Table 1-19. USBOEPIE Register Description (continued)

Bit	Field	Type	Reset	Description
0	OEPIE0	RW	0h	Output endpoint interrupt enable 0. This bit enables or disables whether an event can trigger an interrupt. It does not influence whether the event is flagged; the flag is enabled or disabled with the interrupt indication enable bit in the Endpoint descriptors. 0b = Event does not generate an interrupt 1b = Event does generate an interrupt

1.4.2.7 USBIEPIFG Register

USB Input Endpoint Interrupt Flag Register

Figure 1-21. USBIEPIFG Register

7	6	5	4	3	2	1	0
IEPIFG7	IEPIFG6	IEPIFG5	IEPIFG4	IEPIFG3	IEPIFG2	IEPIFG1	IEPIFG0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-20. USBIEPIFG Register Description

Bit	Field	Type	Reset	Description
7	IEPIFG7	RW	0h	Input endpoint interrupt flag 7. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.
6	IEPIFG6	RW	0h	Input endpoint interrupt flag 6. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.
5	IEPIFG5	RW	0h	Input endpoint interrupt flag 5. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.
4	IEPIFG4	RW	0h	Input endpoint interrupt flag 4. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.
3	IEPIFG3	RW	0h	Input endpoint interrupt flag 3. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.
2	IEPIFG2	RW	0h	Input endpoint interrupt flag 2. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.
1	IEPIFG1	RW	0h	Input endpoint interrupt flag 1. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.
0	IEPIFG0	RW	0h	Input endpoint interrupt flag 0. This bit is set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing zero to that bit location.

1.4.2.8 USBOEPIFG Register

USB Output Endpoint Interrupt Flag Register

Figure 1-22. USBOEPIFG Register

7	6	5	4	3	2	1	0
OEPIFG7	OEPIFG6	OEPIFG5	OEPIFG4	OEPIFG3	OEPIFG2	OEPIFG1	OEPIFG0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-21. USBOEPIFG Register Description

Bit	Field	Type	Reset	Description
7	OEPIFG7	RW	0h	Output endpoint interrupt flag 7. The output endpoint interrupt flag is set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.
6	OEPIFG6	RW	0h	Output endpoint interrupt flag 6. The output endpoint interrupt flag is set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.
5	OEPIFG5	RW	0h	Output endpoint interrupt flag 5. The output endpoint interrupt flag is set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.
4	OEPIFG4	RW	0h	Output endpoint interrupt flag 4. The output endpoint interrupt flag is set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.
3	OEPIFG3	RW	0h	Output endpoint interrupt flag 3. The output endpoint interrupt flag set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.
2	OEPIFG2	RW	0h	Output endpoint interrupt flag 2. The output endpoint interrupt flag set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.
1	OEPIFG1	RW	0h	Output endpoint interrupt flag 1. The output endpoint interrupt flag set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.

Table 1-21. USBOEPIFG Register Description (continued)

Bit	Field	Type	Reset	Description
0	OEPIFG0	RW	0h	Output endpoint interrupt flag 0. The output endpoint interrupt flag set to 1 by the UBM when a successful completion of a transaction occurs to that out endpoint. When the bit is set, a USB interrupt is generated. The interrupt flag is cleared when the MCU reads the value from the USBVECINT (USBIV) register corresponding with this interrupt, or when it writes any value to the interrupt vector register. An interrupt flag can also be cleared by writing a zero to this bit location.

1.4.2.9 USBVECINT Register

USB Interrupt Vector Register

This register is also referred to as USBIV.

Figure 1-23. USBVECINT Register

15	14	13	12	11	10	9	8
USBIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
USBIV							
r0	r0	r-0	r-0	r-0	r-0	r-0	r0

Table 1-22. USBVECINT Register Description

Bit	Field	Type	Reset	Description
15-0	USBIV	R	0h	USB interrupt vector value. This register is to be accessed as a whole word only. When an interrupt is pending, reading this register results in a value that can be added to the program counter to handle the corresponding event. Writing to this register clears all pending USB interrupt flags independent of the status of USBEN. 00h = No interrupt pending 02h = See Section 1.2.5.; Interrupt Priority: Highest 3Eh = Interrupt Priority: Lowest

1.4.2.10 USBMAINT Register

Timestamp Maintenance Register

Figure 1-24. USBMAINT Register

15	14	13	12	11	10	9	8
UTSEL			Reserved	TSE3	TSESEL		TSGEN
rw-0	rw-0	rw-0	r0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved						UTIE	UTIFG
r0	r0	r0	r0	r0	r0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-23. USBMAINT Register Description

Bit	Field	Type	Reset	Description
15-13	UTSEL	RW	0h	USB timer selection 000b = USB Timer Period: 4096 µs; Approximate Frequency: 250 Hz (244 Hz) 001b = USB Timer Period: 2048 µs; Approximate Frequency: 500 Hz (488 Hz) 010b = USB Timer Period: 1024 µs; Approximate Frequency: 1 kHz (977 Hz) 011b = USB Timer Period: 512 µs; Approximate Frequency: 2 kHz (1953 Hz) 100b = USB Timer Period: 256 µs; Approximate Frequency: 4 kHz (3906 Hz) 101b = USB Timer Period: 128 µs; Approximate Frequency: 8 kHz (7812 Hz) 110b = USB Timer Period: 64 µs; Approximate Frequency: 16 kHz (15625 Hz) 111b = USB Timer Period: 32 µs; Approximate Frequency: 31 kHz (31250 Hz)
12	Reserved	R	0h	Reserved. Always reads as 0.
11	TSE3	RW	0h	Timestamp Event 3 bit. This bit allows the triggering of a software-driven timestamp event (when TSESEL = 11b). 0b = No TSE3 event signaled 1b = TSE3 event signaled
10-9	TSESEL	RW	0h	Timestamp Event Selection. TSE[2:0] are connected to the event multiplexer of the three DMA channels of the DMA controller if not otherwise noted in data sheet 00b = TSE0 (DMA0) signal is qualified timestamp event 01b = TSE1 (DMA1) signal is qualified timestamp event 10b = TSE2 (DMA2) signal is qualified timestamp event 11b = Software-driven timestamp event
8	TSGEN	RW	0h	Timestamp generator enable 0b = Timestamp mechanism disabled 1b = Timestamp mechanism enabled
7-2	Reserved	R	0h	Reserved. Always reads as 0.
1	UTIE	RW	0h	USB timer interrupt enable bit 0b = USB timer interrupt disabled 1b = USB timer interrupt enabled
0	UTIFG	RW	0h	USB timer interrupt flag 0b = No interrupt pending 1b = Interrupt pending

1.4.2.11 USBTSREG Register

USB Timestamp Register

Figure 1-25. USBTSREG Register

15	14	13	12	11	10	9	8
TVAL							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
TVAL							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Can be modified only when USBEN = 1

Table 1-24. USBTSREG Register Description

Bit	Field	Type	Reset	Description
15-0	TVAL	R	0h	Timestamp high register. The timestamp value is updated by hardware from the USB timer. A qualified timestamp trigger signal causes the current timer value to be latched into this register.

1.4.2.12 USBFN Register

USB Frame Number Register

Figure 1-26. USBFN Register

15	14	13	12	11	10	9	8
Reserved					USBFN		
r0	r0	r0	r0	r0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
USBFN							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 1-25. USBFN Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved. Always reads as 0.
10-0	USBFN	R	0h	USB Frame Number register. The frame number bit values are updated by hardware; each USB frame with the frame number field value received in the USB start-of-frame packet. The frame number can be used as a timestamp. If the local (MSP430's) frame timer is not locked to the USB host's frame timer, then the frame number is automatically incremented from the previous value when a pseudo start-of-frame occurs.

1.4.2.13 USBCTL Register

USB Control Register

Figure 1-27. USBCTL Register

7	6	5	4	3	2	1	0
Reserved	FEN	RWUP	FRSTE	Reserved			DIR
r0	rw-0	rw-0	rw-0	r0	r0	r0	rw-0

Can be modified only when USBEN = 1

Table 1-26. USBCTL Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	FEN	RW	0h	Function Enable Bit. This bit needs to be set to enable the USB device to respond to USB transactions. If this bit is not set, the UBM ignores all USB transactions. It is cleared by a USB reset. (This bit is primarily intended for debugging.) 0b = Function is disabled 1b = Function is enabled
5	RWUP	RW	0h	Device Remote Wakeup request. The remote wake-up bit is set by software to request the suspend/resume logic to generate resume signaling upstream on the USB. This bit is used to exit a USB low-power suspend state when a remote wake-up event occurs. The bit is self-clearing. 0b = Writing 0 has no effect 1b = A Remote-Wakeup pulse is generated
4	FRSTE	RW	0h	Function Reset Connection Enable. This bit selects whether a bus reset on the USB causes an internal reset of the USB module. 0b = Bus reset does not cause a reset of the module 1b = Bus reset does cause a reset of the module
3-1	Reserved	R	0h	Reserved. Always reads as 0.
0	DIR	RW	0h	Data response to setup packet interrupt status bit. Software must decode the request and set/clear this bit to reflect the data transfer direction. 0b = USB data-OUT transaction (from host to device) 1b = USB data-IN transaction (from device to host)

1.4.2.14 USBIE Register

USB Interrupt Enable Register

Figure 1-28. USBIE Register

7	6	5	4	3	2	1	0
RSTRIE	SUSRIE	RESRIE	Reserved		SETUPIE	Reserved	STPOWIE
rw-0	rw-0	rw-0	r0	r0	rw-0	r0	rw-0

Can be modified only when USBEN = 1

Table 1-27. USBIE Register Description

Bit	Field	Type	Reset	Description
7	RSTRIE	RW	0h	USB reset interrupt enable. Causes an interrupt to be generated if the RSTRIFG bit is set. 0b = Function Reset interrupt disabled 1b = Function Reset interrupt enabled
6	SUSRIE	RW	0h	Suspend interrupt enable. Causes an interrupt to be generated if the SUSRIFG bit is set. 0b = Suspend interrupt disabled 1b = Suspend interrupt enabled
5	RESRIE	RW	0h	Resume interrupt enable. Causes an interrupt to be generated if the RESRIFG bit is set. 0b = Resume interrupt disabled 1b = Resume interrupt enabled
4-3	Reserved	R	0h	Reserved. Always reads as 0.
2	SETUPIE	RW	0h	Setup interrupt enable. Causes an interrupt to be generated if the SETUPIFG bit is set. 0b = Setup interrupt disabled 1b = Setup interrupt enabled
1	Reserved	R	0h	Reserved. Always reads as 0.
0	STPOWIE	RW	0h	Setup Overwrite interrupt enable. Causes an interrupt to be generated if the STPOWIFG bit is set. 0b = Setup Overwrite interrupt disabled 1b = Setup Overwrite interrupt enabled

1.4.2.15 USBIFG Register

USB Interrupt Flag Register

Figure 1-29. USBIFG Register

7	6	5	4	3	2	1	0
RSTRIFG	SUSRIFG	RESRIFG	Reserved		SETUPIFG	Reserved	STPOWIFG
rw-0	rw-0	rw-0	r0	r0	rw-0	r0	rw-0

Can be modified only when USBEN = 1

Table 1-28. USBIFG Register Description

Bit	Field	Type	Reset	Description
7	RSTRIFG	RW	0h	USB reset request bit. This bit is set to one by hardware in response to the host initiating a USB port reset. A USB reset causes a reset of the USB module logic, but this bit is not affected.
6	SUSRIFG	RW	0h	Suspend request bit. This bit is set by hardware in response to the host/hub causing a global or selective suspend condition.
5	RESRIFG	RW	0h	Resume request bit. This bit is set by hardware in response to the host/hub causing a resume event.
4-3	Reserved	R	0h	Reserved. Always reads as 0.
2	SETUPIFG	RW	0h	Setup transaction received bit. This bit is set by hardware when a SETUP transaction is received. As long as this bit is set, transactions on IN and OUT on endpoint-0 receive a NAK, regardless of their corresponding NAK bit value.
1	Reserved	R	0h	Reserved. Always reads as 0.
0	STPOWIFG	RW	0h	Setup overwrite bit. This bit is set by hardware when a setup packet is received while there is already a packet in the setup buffer.

1.4.2.16 USBFUNADR Register

USB Function Address Register

Figure 1-30. USBFUNADR Register

7	6	5	4	3	2	1	0
Reserved	FA6	FA5	FA4	FA3	FA2	FA1	FA0
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when USBEN = 1

Table 1-29. USBFUNADR Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6-0	FA[6:0]	RW	0h	Function address (USB address 0 to 127). These bits define the current device address assigned to this USB device. Software must write a value from 0 to 127 when a Set-Address command is received from the host.

1.4.3 USB Buffer Registers and Memory

The data buffers for all endpoints, as well as the registers that define endpoints 1 to 7, are stored in the USB RAM buffer memory. Doing so allows for efficient and flexible use of this memory. The memory area is known as the USB buffer memory, and the registers that define its use are the buffer descriptor registers.

The buffer memory blocks are listed in Table 1-30. The registers are listed in Table 1-31. All addresses are expressed as offsets; the base address can be found in the device-specific data sheet.

All memory is byte and word accessible.

Table 1-30. USB Buffer Memory

Offset	Acronym	Memory	Type
0000h	USBSTABUFF	Start of buffer space	Read/Write
⋮	⋮	1904 bytes of configurable buffer space	⋮
076Fh	USBTOPBUFF	End of buffer space	Read/Write
0770h	USBOEP0BUF	Output endpoint_0 buffer	Read/Write
⋮			⋮
0777h			Read/Write
0778h	USBIEP0BUF	Input endpoint_0 buffer	Read/Write
⋮			⋮
077Fh			Read/Write
0780h	USBSUBLK	Setup Packet Block	Read/Write
⋮			⋮
0787h			Read/Write

Table 1-31. USB Buffer Descriptor Registers

Offset	Acronym	Register Name	Type	Section
0788h	USBOEPCNF_1	Output Endpoint_1 Configuration Register	Read/Write	Section 1.4.3.1
0789h	USBOEPBBAX_1	Output Endpoint_1 X Buffer Base Address Register	Read/Write	Section 1.4.3.2
078Ah	USBOEPBCTX_1	Output Endpoint_1 X Byte Count Register	Read/Write	Section 1.4.3.3
078Dh	USBOEPBBAY_1	Output Endpoint_1 Y Buffer Base Address Register	Read/Write	Section 1.4.3.4
078Eh	USBOEPBCTY_1	Output Endpoint_1 Y Byte Count Register	Read/Write	Section 1.4.3.5
078Fh	USBOEPSIZXY_1	Output Endpoint_1 X and Y Buffer Size Register	Read/Write	Section 1.4.3.6
0790h	USBOEPCNF_2	Output Endpoint_2 Configuration Register	Read/Write	Section 1.4.3.1
0791h	USBOEPBBAX_2	Output Endpoint_2 X Buffer Base Address Register	Read/Write	Section 1.4.3.2
0792h	USBOEPBCTX_2	Output Endpoint_2 X Byte Count Register	Read/Write	Section 1.4.3.3
0795h	USBOEPBBAY_2	Output Endpoint_2 Y Buffer Base Address Register	Read/Write	Section 1.4.3.4
0796h	USBOEPBCTY_2	Output Endpoint_2 Y Byte Count Register	Read/Write	Section 1.4.3.5
0797h	USBOEPSIZXY_2	Output Endpoint_2 X and Y Buffer Size Register	Read/Write	Section 1.4.3.6
0798h	USBOEPCNF_3	Output Endpoint_3 Configuration Register	Read/Write	Section 1.4.3.1
0799h	USBOEPBBAX_3	Output Endpoint_3 X Buffer Base Address Register	Read/Write	Section 1.4.3.2
079Ah	USBOEPBCTX_3	Output Endpoint_3 X Byte Count Register	Read/Write	Section 1.4.3.3
079Dh	USBOEPBBAY_3	Output Endpoint_3 Y Buffer Base Address Register	Read/Write	Section 1.4.3.4
079Eh	USBOEPBCTY_3	Output Endpoint_3 Y Byte Count Register	Read/Write	Section 1.4.3.5
079Fh	USBOEPSIZXY_3	Output Endpoint_3 X and Y Buffer Size Register	Read/Write	Section 1.4.3.6

Table 1-31. USB Buffer Descriptor Registers (continued)

Offset	Acronym	Register Name	Type	Section
07A0h	USBOEPCNF_4	Output Endpoint_4 Configuration Register	Read/Write	Section 1.4.3.1
07A1h	USBOEPBBAX_4	Output Endpoint_4 X Buffer Base Address Register	Read/Write	Section 1.4.3.2
07A2h	USBOEPBCTX_4	Output Endpoint_4 X Byte Count Register	Read/Write	Section 1.4.3.3
07A5h	USBOEPBBAY_4	Output Endpoint_4 Y Buffer Base Address Register	Read/Write	Section 1.4.3.4
07A6h	USBOEPBCTY_4	Output Endpoint_4 Y Byte Count Register	Read/Write	Section 1.4.3.5
07A7h	USBOEPSIZXY_4	Output Endpoint_4 X and Y Buffer Size Register	Read/Write	Section 1.4.3.6
07A8h	USBOEPCNF_5	Output Endpoint_5 Configuration Register	Read/Write	Section 1.4.3.1
07A9h	USBOEPBBAX_5	Output Endpoint_5 X Buffer Base Address Register	Read/Write	Section 1.4.3.2
07AAh	USBOEPBCTX_5	Output Endpoint_5 X Byte Count Register	Read/Write	Section 1.4.3.3
07ADh	USBOEPBBAY_5	Output Endpoint_5 Y Buffer Base Address Register	Read/Write	Section 1.4.3.4
07AEh	USBOEPBCTY_5	Output Endpoint_5 Y Byte Count Register	Read/Write	Section 1.4.3.5
07AFh	USBOEPSIZXY_5	Output Endpoint_5 X and Y Buffer Size Register	Read/Write	Section 1.4.3.6
07B0h	USBOEPCNF_6	Output Endpoint_6 Configuration Register	Read/Write	Section 1.4.3.1
07B1h	USBOEPBBAX_6	Output Endpoint_6 X Buffer Base Address Register	Read/Write	Section 1.4.3.2
07B2h	USBOEPBCTX_6	Output Endpoint_6 X Byte Count Register	Read/Write	Section 1.4.3.3
07B5h	USBOEPBBAY_6	Output Endpoint_6 Y Buffer Base Address Register	Read/Write	Section 1.4.3.4
07B6h	USBOEPBCTY_6	Output Endpoint_6 Y Byte Count Register	Read/Write	Section 1.4.3.5
07B7h	USBOEPSIZXY_6	Output Endpoint_6 X and Y Buffer Size Register	Read/Write	Section 1.4.3.6
07B8h	USBOEPCNF_7	Output Endpoint_7 Configuration Register	Read/Write	Section 1.4.3.1
07B9h	USBOEPBBAX_7	Output Endpoint_7 X Buffer Base Address Register	Read/Write	Section 1.4.3.2
07BAh	USBOEPBCTX_7	Output Endpoint_7 X Byte Count Register	Read/Write	Section 1.4.3.3
07BDh	USBOEPBBAY_7	Output Endpoint_7 Y Buffer Base Address Register	Read/Write	Section 1.4.3.4
07BEh	USBOEPBCTY_7	Output Endpoint_7 Y Byte Count Register	Read/Write	Section 1.4.3.5
07BFh	USBOEPSIZXY_7	Output Endpoint_7 X and Y Buffer Size Register	Read/Write	Section 1.4.3.6
07C8h	USBIEPCNF_1	Input Endpoint_1 Configuration Register	Read/Write	Section 1.4.3.7
07C9h	USBIEPBBAX_1	Input Endpoint_1 X Buffer Base Address Register	Read/Write	Section 1.4.3.8
07CAh	USBIEPBCTX_1	Input Endpoint_1 X Byte Count Register	Read/Write	Section 1.4.3.9
07CDh	USBIEPBBAY_1	Input Endpoint_1 Y Buffer Base Address Register	Read/Write	Section 1.4.3.10
07CEh	USBIEPBCTY_1	Input Endpoint_1 Y Byte Count Register	Read/Write	Section 1.4.3.11
07CFh	USBIEPSIZXY_1	Input Endpoint_1 X and Y Buffer Size Register	Read/Write	Section 1.4.3.12
07D0h	USBIEPCNF_2	Input Endpoint_2 Configuration Register	Read/Write	Section 1.4.3.7
07D1h	USBIEPBBAX_2	Input Endpoint_2 X Buffer Base Address Register	Read/Write	Section 1.4.3.8
07D2h	USBIEPBCTX_2	Input Endpoint_2 X Byte Count Register	Read/Write	Section 1.4.3.9
07D5h	USBIEPBBAY_2	Input Endpoint_2 Y Buffer Base Address Register	Read/Write	Section 1.4.3.10
07D6h	USBIEPBCTY_2	Input Endpoint_2 Y Byte Count Register	Read/Write	Section 1.4.3.11
07D7h	USBIEPSIZXY_2	Input Endpoint_2 X and Y Buffer Size Register	Read/Write	Section 1.4.3.12
07D8h	USBIEPCNF_3	Input Endpoint_3 Configuration Register	Read/Write	Section 1.4.3.7
07D9h	USBIEPBBAX_3	Input Endpoint_3 X Buffer Base Address Register	Read/Write	Section 1.4.3.8
07DAh	USBIEPBCTX_3	Input Endpoint_3 X Byte Count Register	Read/Write	Section 1.4.3.9
07DDh	USBIEPBBAY_3	Input Endpoint_3 Y Buffer Base Address Register	Read/Write	Section 1.4.3.10
07DEh	USBIEPBCTY_3	Input Endpoint_3 Y Byte Count Register	Read/Write	Section 1.4.3.11
07DFh	USBIEPSIZXY_3	Input Endpoint_3 X and Y Buffer Size Register	Read/Write	Section 1.4.3.12

Table 1-31. USB Buffer Descriptor Registers (continued)

Offset	Acronym	Register Name	Type	Section
07E0h	USBIEPCNF_4	Input Endpoint_4 Configuration Register	Read/Write	Section 1.4.3.7
07E1h	USBIEPBBAX_4	Input Endpoint_4 X Buffer Base Address Register	Read/Write	Section 1.4.3.8
07E2h	USBIEPBCTX_4	Input Endpoint_4 X Byte Count Register	Read/Write	Section 1.4.3.9
07E5h	USBIEPBBAY_4	Input Endpoint_4 Y Buffer Base Address Register	Read/Write	Section 1.4.3.10
07E6h	USBIEPBCTY_4	Input Endpoint_4 Y Byte Count Register	Read/Write	Section 1.4.3.11
07E7h	USBIEPSIZXY_4	Input Endpoint_4 X and Y Buffer Size Register	Read/Write	Section 1.4.3.12
07E8h	USBIEPCNF_5	Input Endpoint_5 Configuration Register	Read/Write	Section 1.4.3.7
07E9h	USBIEPBBAX_5	Input Endpoint_5 X Buffer Base Address Register	Read/Write	Section 1.4.3.8
07EAh	USBIEPBCTX_5	Input Endpoint_5 X Byte Count Register	Read/Write	Section 1.4.3.9
07EDh	USBIEPBBAY_5	Input Endpoint_5 Y Buffer Base Address Register	Read/Write	Section 1.4.3.10
07EEh	USBIEPBCTY_5	Input Endpoint_5 Y Byte Count Register	Read/Write	Section 1.4.3.11
07EFh	USBIEPSIZXY_5	Input Endpoint_5 X and Y Buffer Size Register	Read/Write	Section 1.4.3.12
07F0h	USBIEPCNF_6	Input Endpoint_6 Configuration Register	Read/Write	Section 1.4.3.7
07F1h	USBIEPBBAX_6	Input Endpoint_6 X Buffer Base Address Register	Read/Write	Section 1.4.3.8
07F2h	USBIEPBCTX_6	Input Endpoint_6 X Byte Count Register	Read/Write	Section 1.4.3.9
07F5h	USBIEPBBAY_6	Input Endpoint_6 Y Buffer Base Address Register	Read/Write	Section 1.4.3.10
07F6h	USBIEPBCTY_6	Input Endpoint_6 Y Byte Count Register	Read/Write	Section 1.4.3.11
07F7h	USBIEPSIZXY_6	Input Endpoint_6 X and Y Buffer Size Register	Read/Write	Section 1.4.3.12
07F8h	USBIEPCNF_7	Input Endpoint_7 Configuration Register	Read/Write	Section 1.4.3.7
07F9h	USBIEPBBAX_7	Input Endpoint_7 X Buffer Base Address Register	Read/Write	Section 1.4.3.8
07FAh	USBIEPBCTX_7	Input Endpoint_7 X Byte Count Register	Read/Write	Section 1.4.3.9
07FDh	USBIEPBBAY_7	Input Endpoint_7 Y Buffer Base Address Register	Read/Write	Section 1.4.3.10
07FEh	USBIEPBCTY_7	Input Endpoint_7 Y Byte Count Register	Read/Write	Section 1.4.3.11
07FFh	USBIEPSIZXY_7	Input Endpoint_7 X and Y Buffer Size Register	Read/Write	Section 1.4.3.12

1.4.3.1 USBOEPCNF_n Register

Output Endpoint-n Configuration Register

This register can be modified only when USBEN = 1.

Figure 1-31. USBOEPCNF_n Register

7	6	5	4	3	2	1	0
UBME	Reserved	TOGGLE	DBUF	STALL	USBIIE	Reserved	
rw	r0	rw	rw	rw	rw	r0	r0

Can be modified only when USBEN = 1

Table 1-32. USBOEPCNF_n Register Description

Bit	Field	Type	Reset	Description
7	UBME	RW	0h	UBM out endpoint-n enable. This bit is to be set or cleared by software. 0b = UBM cannot use this endpoint 1b = UBM can use this endpoint
6	Reserved	R	0h	Reserved. Always reads as 0.
5	TOGGLE	RW	0h	Toggle bit. The toggle bit is controlled by the UBM and is toggled at the end of a successful out data stage transaction, if a valid data packet is received and the data packet's packet ID matches the expected packet ID.
4	DBUF	RW	0h	Double buffer enable. This bit can be set to enable the use of both the X and Y data packet buffers for USB transactions, for a particular out endpoint. Clearing it results in the use of single buffer mode. In this mode, only the X buffer is used. 0b = Primary buffer only (X-buffer only) 1b = Toggle bit selects buffer
3	STALL	RW	0h	USB stall condition. This bit can be set to cause endpoint transactions to be stalled. When set, the hardware automatically returns a stall handshake to the host for any transaction received on endpoint-0. The stall bit is cleared automatically by the next setup transaction. 0b = Indicates no stall 1b = Indicates stall
2	USBIIE	RW	0h	USB transaction interrupt indication enable. Can be set or cleared to define if interrupts are to be flagged in general. To generate an interrupt, the corresponding interrupt flag must be set (OEPIE). 0b = Corresponding interrupt flag will not be set 1b = Corresponding interrupt flag will be set
1-0	Reserved	R	0h	Reserved. Always reads as 0.

1.4.3.2 USBOEPBBAX_n Register

Output Endpoint-n X Buffer Base Address Register

This register can be modified only when USBEN = 1.

Figure 1-32. USBOEPBBAX_n Register

7	6	5	4	3	2	1	0
ADR							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-33. USBOEPBBAX_n Register Description

Bit	Field	Type	Reset	Description
7-0	ADR	RW	0h	X-buffer base address. These are the upper eight bits of the X-buffer's base address. The three LSBs are assumed to be zero, for a total of 11 bits. This value needs to be set by software. The UBM uses this value as the start address of a given transaction. It does not change this value at the end of a transaction.

1.4.3.3 USBOEPBCTX_n Register

Output Endpoint-n X Byte Count Register

This register can be modified only when USBEN = 1.

Figure 1-33. USBOEPBCTX_n Register

7	6	5	4	3	2	1	0
NAK	CNT						
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-34. USBOEPBCTX_n Register Description

Bit	Field	Type	Reset	Description
7	NAK	RW	0h	No-acknowledge status bit. The NAK status bit is set by the UBM at the end of a successful USB out transaction to that endpoint, to indicate that the USB endpoint-"n" buffer contains a valid data packet, and that the buffer data count value is valid. When this bit is set, all subsequent transactions to that endpoint result in a NAK handshake response to the USB host. To re-enable this endpoint to receive another data packet from the host, this bit must be cleared. 0b = No valid data in buffer. The buffer is ready to receive OUT packets from the host. 1b = The buffer contains a valid packet from the host, and it has not been picked up (subsequent host-out requests receive a NAK)
6-0	CNT	RW	0h	X-buffer data count. The Out_EP-n data count value is set by the UBM when a new data packet is written to the X-buffer for that out endpoint. It is set to the number of bytes received in the data buffer.

1.4.3.4 USBOEPBBAY_n Register

Output Endpoint-n Y Buffer Base Address Register

This register can be modified only when USBEN = 1.

Figure 1-34. USBOEPBBAY_n Register

7	6	5	4	3	2	1	0
ADR							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-35. USBOEPBBAY_n Register Description

Bit	Field	Type	Reset	Description
7-0	ADR	RW	0h	Y-buffer base address. These are the upper eight bits of the Y-buffer's base address. The three LSBs are assumed to be zero, for a total of 11 bits. This value needs to be set by software. The UBM uses this value as the start address of a given transaction. It does not change this value at the end of a transaction.

1.4.3.5 USBOEPBCTY_n Register

Output Endpoint-n X Byte Count Register

This register can be modified only when USBEN = 1.

Figure 1-35. USBOEPBCTY_n Register

7	6	5	4	3	2	1	0
NAK	CNT						
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-36. USBOEPBCTY_n Register Description

Bit	Field	Type	Reset	Description
7	NAK	RW	0h	No-acknowledge status bit. The NAK status bit is set by the UBM at the end of a successful USB out transaction to that endpoint, to indicate that the USB endpoint-"n" buffer contains a valid data packet, and that the buffer data count value is valid. When this bit is set, all subsequent transactions to that endpoint result in a NAK handshake response to the USB host. To re-enable this endpoint to receive another data packet from the host, this bit must be cleared. 0b = No valid data in buffer. The buffer is ready to receive OUT packets from the host. 1b = The buffer contains a valid packet from the host, and it has not been picked up (subsequent host-out requests receive a NAK)
6-0	CNT	RW	0h	Y-buffer data count. The Out_EP-n data count value is set by the UBM when a new data packet is written to the X-buffer for that out endpoint. It is set to the number of bytes received in the data buffer.

1.4.3.6 USBOEPSIZXY_n Register

Output Endpoint-n X and Y Buffer Size Register

This register can be modified only when USBEN = 1.

Figure 1-36. USBOEPSIZXY_n Register

7	6	5	4	3	2	1	0
Reserved	SIZx						
r0	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-37. USBOEPSIZXY_n Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6-0	SIZx	RW	0h	Buffer size count. This value needs to be set by software to configure the size of the X and Y data packet buffers. Both buffers are set to the same size, based on this value. 000:0000b to 100:0000b are valid numbers for 0 to 64 bytes. Any value greater than or equal to 100:0001b results in unpredictable results.

1.4.3.7 USBIEPCNF_n Register

Input Endpoint-n Configuration Register

This register can be modified only when USBEN = 1.

Figure 1-37. USBIEPCNF_n Register

7	6	5	4	3	2	1	0
UBME	Reserved	TOGGLE	DBUF	STALL	USBIIE	Reserved	
rw	r0	rw	rw	rw	rw	r0	r0

Can be modified only when USBEN = 1

Table 1-38. USBIEPCNF_n Register Description

Bit	Field	Type	Reset	Description
7	UBME	RW	0h	UBM in endpoint-n enable. This value needs to be set or cleared by software. 0b = UBM cannot use this endpoint 1b = UBM can use this endpoint
6	Reserved	R	0h	Reserved. Always reads as 0.
5	TOGGLE	RW	0h	Toggle bit. The toggle bit is controlled by the UBM and is toggled at the end of a successful in data stage transaction, if a valid data packet is transmitted. If this bit is cleared, a DATA0 packet ID is transmitted in the data packet to the host. If this bit is set, a DATA1 packet ID is transmitted in the data packet.
4	DBUF	RW	0h	Double buffer enable. This bit can be set to enable the use of both the X and Y data packet buffers for USB transactions, for a particular out endpoint. Clearing it results in the use of single buffer mode. In this mode, only the X buffer is used. 0b = Primary buffer only (X-buffer only) 1b = Toggle bit selects buffer
3	STALL	RW	0h	USB stall condition. This bit can be set to cause endpoint transactions to be stalled. When set, the hardware automatically returns a stall handshake to the host for any transaction received on endpoint-0. The stall bit is cleared automatically by the next setup transaction. 0b = Indicates no stall 1b = Indicates stall
2	USBIIE	RW	0h	USB transaction interrupt indication enable. Can be set or cleared to define if interrupts are to be flagged in general. To generate an interrupt the corresponding interrupt flag must be set (OEPIE). 0b = Corresponding interrupt flag will not be set 1b = Corresponding interrupt flag will be set
1-0	Reserved	R	0h	Reserved. Always reads as 0.

1.4.3.8 USBIEPBBAX_n Register

Input Endpoint-n X Buffer Base Address Register

This register can be modified only when USBEN = 1.

Figure 1-38. USBIEPBBAX_n Register

7	6	5	4	3	2	1	0
ADR							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-39. USBIEPBBAX_n Register Description

Bit	Field	Type	Reset	Description
7-0	ADR	RW	0h	X-buffer base address. These are the upper eight bits of the X-buffer's base address. The three LSBs are assumed to be zero, for a total of 11 bits. This value needs to be set by software. The UBM uses this value as the start address of a given transaction. It does not change this value at the end of a transaction.

1.4.3.9 USBIEPBCTX_n Register

Input Endpoint-n X Byte Count Register

This register can be modified only when USBEN = 1.

Figure 1-39. USBIEPBCTX_n Register

7	6	5	4	3	2	1	0
NAK	CNT						
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-40. USBIEPBCTX_n Register Description

Bit	Field	Type	Reset	Description
7	NAK	RW	0h	No-acknowledge status bit. The NAK status bit is set by the UBM at the end of a successful USB in transaction from that endpoint, to indicate that the EP-n in buffer is empty. For interrupt or bulk endpoints, when this bit is set, all subsequent transactions from that endpoint result in a NAK handshake response to the USB host. To re-enable this endpoint to transmit another data packet to the host, this bit must be cleared. 0b = Buffer contains a valid data packet for the host 1b = Buffer is empty (any host-In requests receive a NAK)
6-0	CNT	RW	0h	X-buffer data count. The In_EP-n X-buffer data count value must be set by software when a new data packet is written to the buffer. It should be the number of bytes in the data packet for interrupt, or bulk endpoint transfers. 000:0000b to 100:0000b are valid numbers for 0 to 64 bytes. Any value greater than or equal to 100:0001b results in unpredictable results.

1.4.3.10 USBIEPBBAY_n Register

Input Endpoint-n Y Buffer Base Address Register

This register can be modified only when USBEN = 1.

Figure 1-40. USBIEPBBAY_n Register

7	6	5	4	3	2	1	0
ADR							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-41. USBIEPBBAY_n Register Description

Bit	Field	Type	Reset	Description
7-0	ADR	RW	0h	Y-buffer base address. These are the upper eight bits of the Y-buffer's base address. The three LSBs are assumed to be zero, for a total of 11 bits. This value needs to be set by software. The UBM uses this value as the start address of a given transaction. It does not change this value at the end of a transaction.

1.4.3.11 USBIEPBCTY_n Register

Input Endpoint-n Y Byte Count Register

This register can be modified only when USBEN = 1.

Figure 1-41. USBIEPBCTY_n Register

7	6	5	4	3	2	1	0
NAK	CNT						
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-42. USBIEPBCTY_n Register Description

Bit	Field	Type	Reset	Description
7	NAK	RW	0h	No-acknowledge status bit. The NAK status bit is set by the UBM at the end of a successful USB in transaction from that endpoint, to indicate that the EP-n in buffer is empty. For interrupt or bulk endpoints, when this bit is set, all subsequent transactions from that endpoint result in a NAK handshake response to the host. To re-enable this endpoint to transmit another data packet to the host, this bit must be cleared. This bit is set by USB SW-init. 0b = Buffer contains a valid data packet for host device 1b = Buffer is empty (any host-in requests receive a NAK)
6-0	CNT	RW	0h	Y-Buffer data count. The In EP-n Y-buffer data count value needs to be set by software when a new data packet is written to the buffer. It should be the number of bytes in the data packet for interrupt, or bulk endpoint transfers. 000:0000b to 100:0000b are valid numbers for 0 to 64 bytes. Any value greater than or equal to 100:0001b results in unpredictable results.

1.4.3.12 USBIEPSIZXY_n Register

Input Endpoint-n X and Y Buffer Size Register

This register can be modified only when USBEN = 1.

Figure 1-42. USBIEPSIZXY_n Register

7	6	5	4	3	2	1	0
Reserved	SIZ						
r0	rw	rw	rw	rw	rw	rw	rw

Can be modified only when USBEN = 1

Table 1-43. USBIEPSIZXY_n Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6-0		RW	0h	Buffer size count. This value needs to be set by software to configure the size of the X and Y data packet buffers. Both buffers are set to the same size, based on this value. 000:0000b to 100:0000b are valid numbers for 0 to 64 bytes. Any value greater than or equal to 100:0001b results in unpredictable results.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265