# MSP430F5xx Family USB Module

# User's Guide

Literature Number: SLAU284
May 2009

# Contents

## List of Figures

## List of Tables

Chapter 1
SLAU284–May 2009

# USB Module

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 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 MSP430 upon arrival of USB power possible (low/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 2 KB 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.

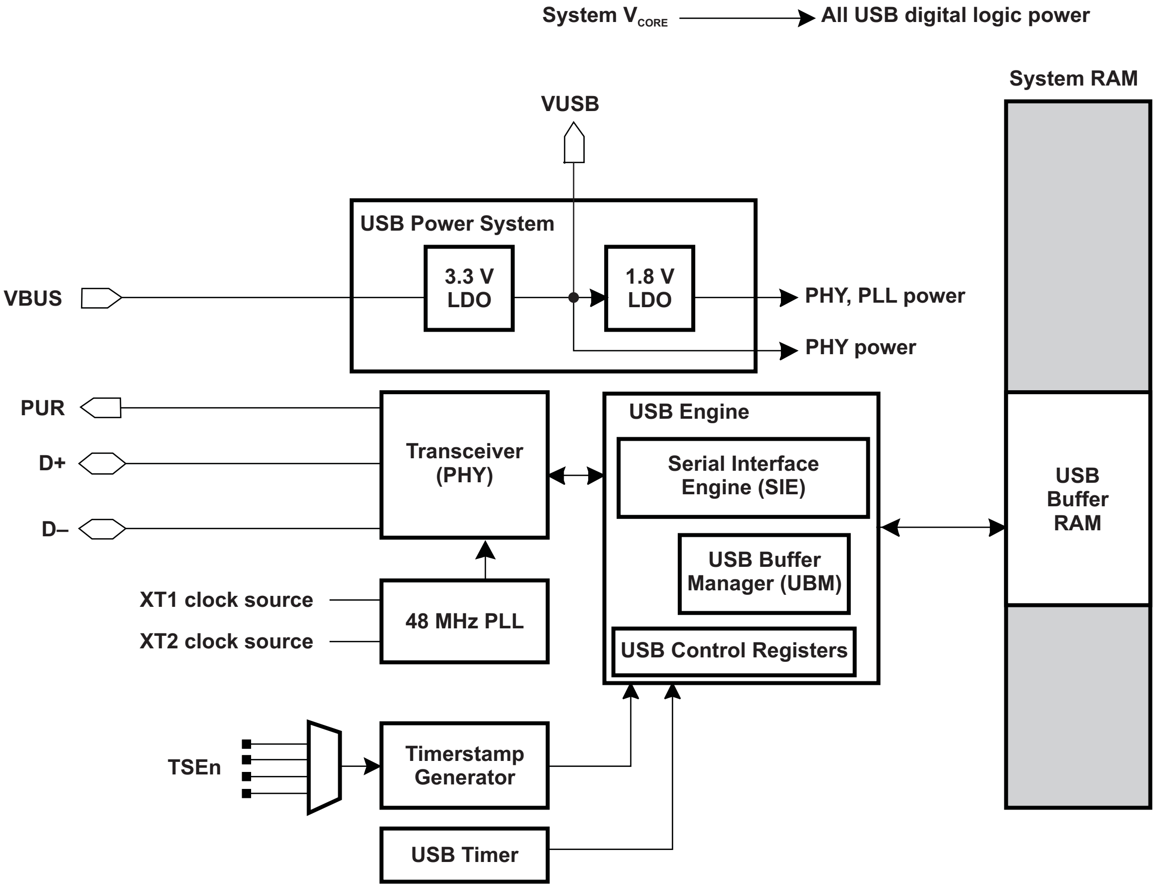

Figure 1-1. USB Block Diagram

## 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/XT2). A crystal greater than 1.5 MHz is required. However, the PLL is very flexible and can adapt to a wide range of frequencies, allowing design to the most cost-effective crystal frequency.

**Note:** Some devices only support XT1 in low frequency (LF) mode of operation. The PLL can only support inputs from the high frequency source i.e. XT1 in high frequency mode (HF) or XT2. For these devices, only XT2 can be used as the input into the PLL for USB operation. XT1 (HF mode) and XT2 bypass modes are also supported. Please refer to the device specific datasheet for clock sources available.

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 and/or DMA.

### *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/DM pins, which form the signaling mechanism of the USB interface.

When the PUSEL bit is set, DP/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 UPCR register. 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 – either from the internal regulators or an external source.

#### 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) in order 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.

#### 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.

### Port U Control

When PUSEL is cleared, the Port U pins (PU0/PU1, corresponding with DP/DM, respectively) function as general-purpose, high-current I/O pins. PUDIR controls the enable of both outputs residing on the Port U pins. The Port U pins are either both driving out, or both acting as inputs. When configured as inputs, the PUIN0/1 pins can be read to determine the input values. When Port U outputs are enabled, the PUIN0/1 will mirror what is present on the outputs.

When PUDIR is set, both Port U pins function as outputs, controlled by PUOUT0/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, PUDIR is cleared. PU0/PU1 therefore become high-impedance when the USB module is 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.

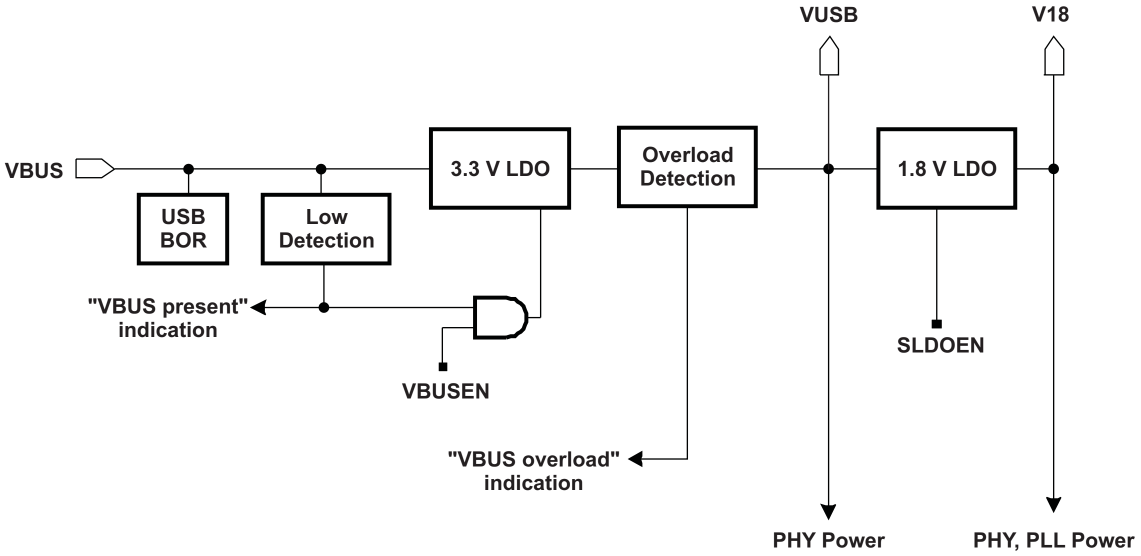

**Figure 1-2. USB Power System**

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.

### Enabling/Disabling

The 3.3-V LDO is enabled/disabled by setting/clearing VUSBEN. Even if enabled, if the voltage on VBUS is detected to be low or nonexistent, 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).

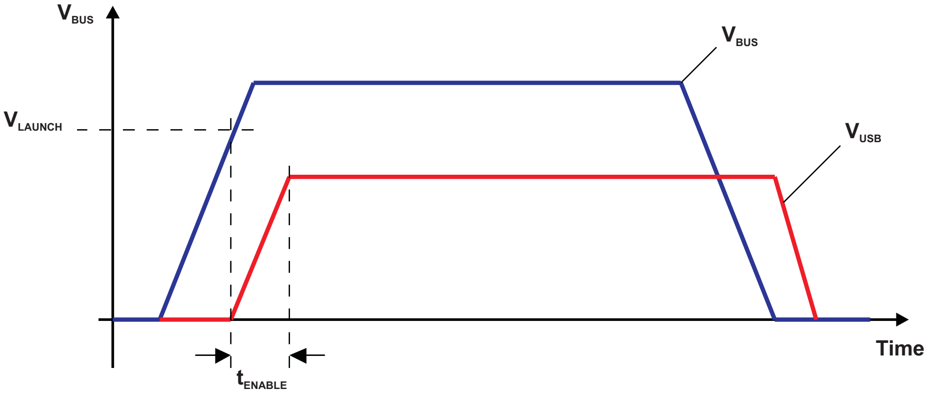

**Figure 1-3. USB Power Up/Down Profile**

The 1.8-V LDO can be enabled/disabled by setting SLDOEN accordingly. By default, SLDOEN is controlled automatically according to whether power is available on VBUS. This auto-enable feature is controlled by SLDOAON. 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/unpowered USB cable that remains electrically connected.

### 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).

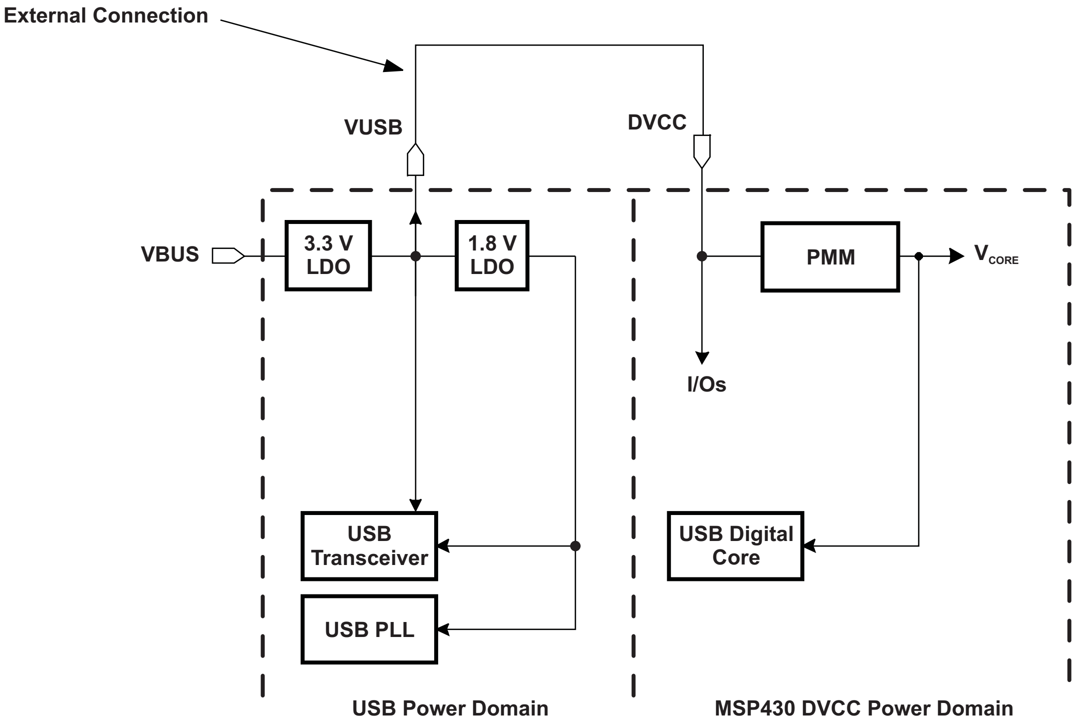

**Figure 1-4. Powering Entire MSP430 From VBUS**

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 LPM5 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 LPM5 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 $I_{DET}$.

#### 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.

#### Current Limitation / Overload Protection

The 3.3-V LDO features current limitation to protect the transceiver during shorted-cable conditions. A short/overload condition – that is, when the output of the LDO becomes current-limited to $I_{DET}$ – is reported to software via the VUOVLIFG flag.

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/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.

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

Ultimately, it is the user's responsibility to ensure that the current drawn from VBUS does not exceed $I_{DET}$.

### *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).

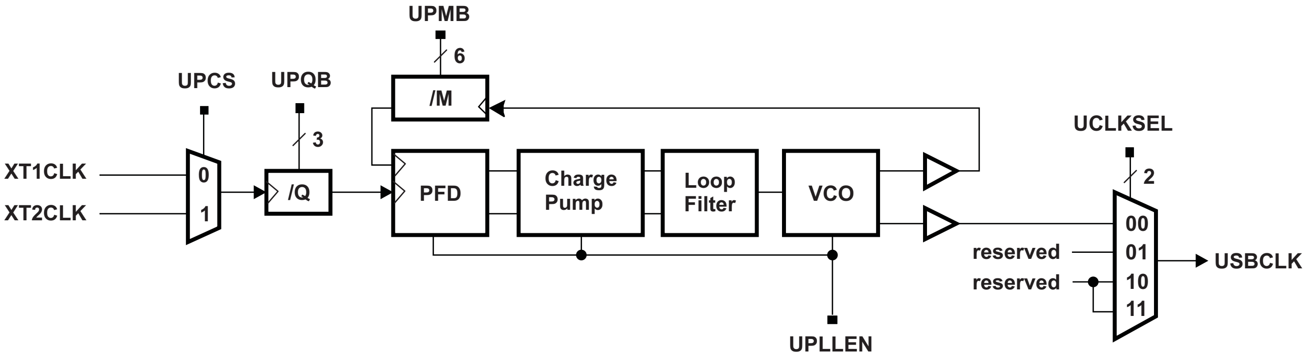

**Figure 1-5. USB-PLL Analog Block Diagram**

The selection of a reference clock is made via the UPCS bit. This allows selection of one of the two system crystal clock sources as a reference clock. 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 1-1).

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

Where

$CLK_{SEL}$ is the selected reference frequency (XT1CLK or XT2CLK)

DIVQ is derived from Table 1-1

DIVM represents the value of UPMB field

If USB operation is used in a bus-powered configuration, disabling the PLL is necessary in order to pass the USB requirement of not consuming more than 500 µA. The UPLLEN bit enables/disables the PLL. The PFDEN bit must be set in order to enable the phase/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 should always 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 | 12 |
| 111 | 16 |

**Table 1-2. Register Settings to Generate 48 MHz Using Common Crystals**

| CLKSEL (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 |
| 3.579545 | 001 | 011010 | 2 | 27 | 1.79 | 48.32 | 6666 |
| 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 |

[1] This frequency is supported by the factory-supplied BSL.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4.1943 | 001 | 010110 | 2 | 23 | 2.097 | 48.23 | 4884 |
| 4.332 | 001 | 010101 | 2 | 22 | 2.166 | 47.652 | -7250 |
| 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 | 010 | 010001 | 3 | 18 | 2.6667 | 48 | 0 |
| 9 | 010 | 001111 | 3 | 16 | 3 | 48 | 0 |
| 9.6 | 011 | 010011 | 4 | 20 | 2.4 | 48 | 0 |
| 10.66 ≈ (32/3) | 011 | 010001 | 4 | 18 | 2.6667 | 48 | 0 |
| 12 | 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 | 101 | 001111 | 8 | 16 | 3 | 48 | 0 |
| 25.6 | 111 | 011101 | 16 | 30 | 1.6 | 48 | 0 |
| 32 | 111 | 010111 | 16 | 24 | 2.6667 | 48 | 0 |

### 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.

### 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 (i.e., up/down) for four consecutive update periods. Loss-of-signal (LOS) is indicated if a frequency correction is performed in the same direction (i.e., up/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).

### 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/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 three buffers per endpoint are desired, however, software must change the address pointers on the fly during a receive/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.

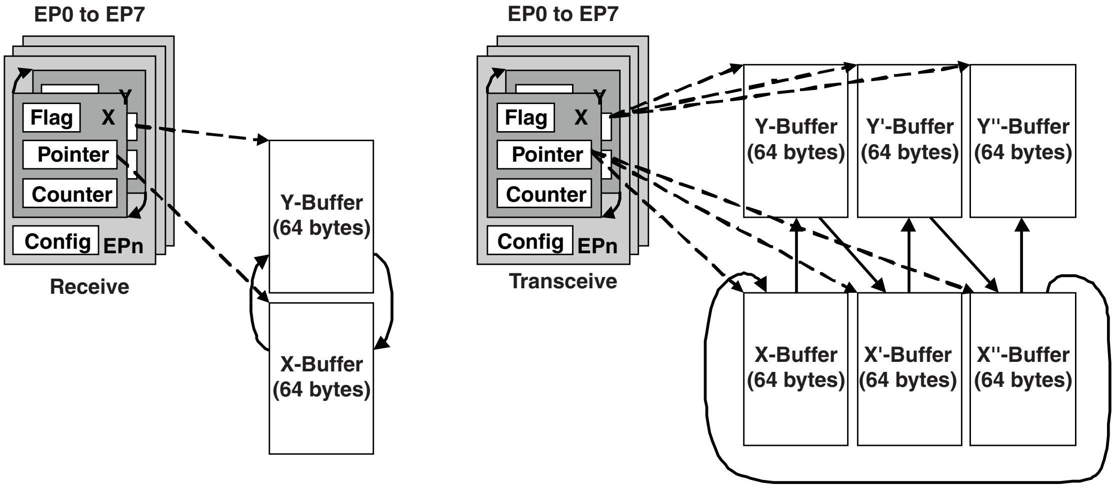

Figure 1-6. Data Buffers and Descriptors

### 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 (PID) to determine the type of packet being received and to ensure the PID 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 PID for all packets being transmitted.

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

### 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 function 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/from the appropriate USB endpoint data buffer.

### 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/DMA access, but reserves priority. As a result, CPU/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.

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 | USBIEPCNT_0 | 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).

### 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/disables the time stamp generator.

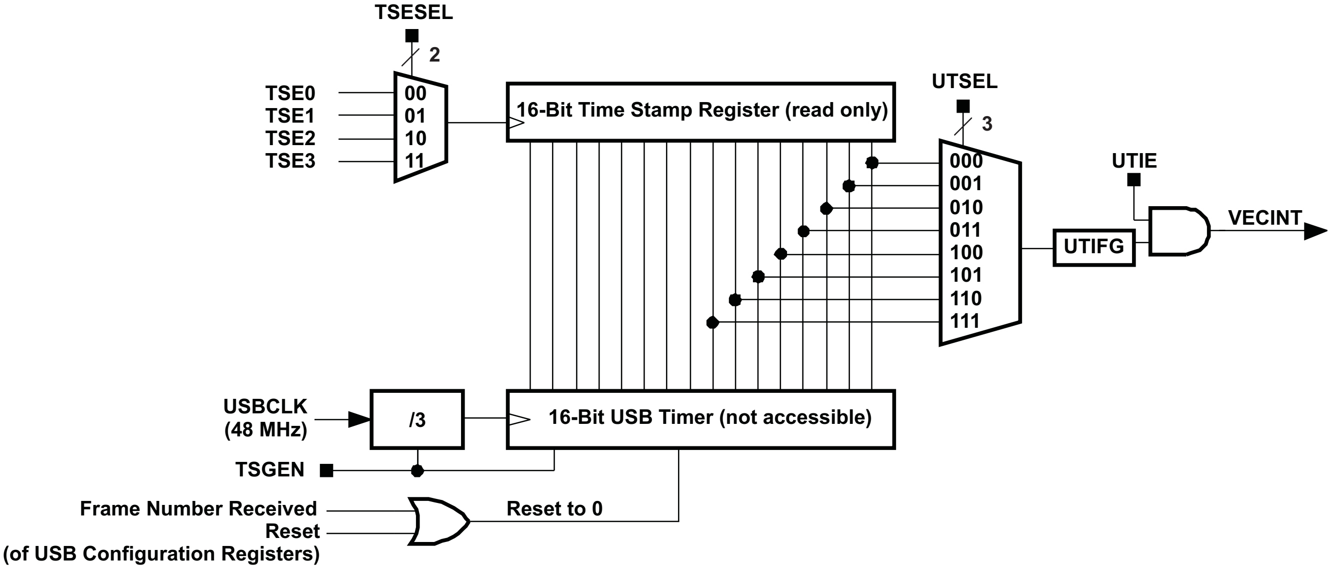

**Figure 1-7. USB Timer and Time Stamp Generation**

### Suspend/Resume Logic

The USB suspend/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.

The USB specification requires that a suspended bus-powered USB device not draw in excess of 500 µA from the bus. To meet this goal in a bus-powered device, it is generally necessary to disable the PLL. Disabling the PLL eliminates the extra power consumption. During suspend, the USBCLK is automatically sourced by the VLO (VLOCLK), allowing the USB module to detect resume when it occurs. See Section 1.2.6 for more information.

### Reset Logic

A PUC resets the USB module logic. 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 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 | – | – | – |
| 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 Entering Suspend Mode

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. To accomplish this, the following actions usually must take place:

- Disable the PLL by clearing UPLLEN
- Disable the high-frequency crystal oscillator that sources the PLL

It is a good idea to also then ensure that the RESRIE bit is also set, so that an interrupt will be generated when the host resumes the device.

## 1.3 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 datasheet for base addresses of these register groupings.

The USB control registers may only be written 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 2 KB RAM block, and can be used by the CPU or DMA without any limitation.

### *1.3.1 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. This register serves as a password. 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.

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 datasheet.

All registers are byte and word accessible.

**Table 1-5. USB Configuration Registers**

| Register | Short Form | Register Type | Address Offset | Initial State |
|---|---|---|---|---|
| USB controller key and ID register | USBKEYPID | Read/Write | 00h | 0000h |
| USB controller configuration register | USBCNF | Read/Write | 02h | 0000h |
| USB-PHY control register | USBPHYCTL | Read/Write | 04h | 0000h |
| USB-PWR control register | USBPWRCTL | Read/Write | 08h | 1850h |
| USB-PLL control register | USBPLLCTL | Read/Write | 10h | 0000h |
| USB-PLL divider buffer register | USBPLLDIVB | Read/Write | 12h | 0000h |
| USB-PLL interrupt register | USBPLLIR | Read/Write | 14h | 0000h |

**USBKEYPID, USB Key Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| **USBKEYPID**<br>Read as A5h, Must be written as 96h | | | | | | | |
| rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **USBKEYPID** | | | | | | | |
| rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 |

**USBKEYPID** Bits 15-0 Key register. Must be written with a value of 9628h in order 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.

### USBCNF, USB Module Configuration 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

**Reserved** Bits 15-5 Reserved. Read back as 0.

**FNTEN** Bit 4 Frame number receive trigger enable for DMA transfers

0 Frame number receive trigger is blocked.

1 Frame number receive trigger is gated through to DMA.

**BLKRDY** Bit 3 Block transfer ready signaling for DMA transfers

0 DMA triggering is disabled.

1 DMA is triggered whenever the USB bus interface can accept new write transfers.

**PUR_IN** Bit 2 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 VBUS. PUR_IN returns zero when VBUS is zero

**PUR_EN** Bit 1 PUR pin enable

0 PUR pin is in high-impedance state

1 PUR pin is driven high

**USB_EN** Bit 0 USB module enable

0 USB module is disabled

1 USB module is enabled

### USBPHYCTL, USB-PHY Control Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | **Reserved** | **Reserved** |
| r0 | r0 | r0 | r0 | r0 | r0 | rw-0 | rw-0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **PUSEL** | **Reserved** | **PUDIR** | **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

**Reserved** Bits 15-10 Reserved. Reads back as 0.

**Reserved** Bits 9-8 Reserved. **Must always be written with 0.**

**PUSEL** Bit 7 USB port function select. This bit selects the function of the PU0/DP and PU1/DM pins.

0 PU0 and PU1 function selected (general purpose I/O)

1 DP and DM function selected (USB terminals)

**Reserved** Bit 6 Reserved.

**PUDIR** Bit 5 USB port direction. This bit controls the direction of both PU0 and PU1. It is only valid when PUSEL = 0.

0 PU0 and PU1 output drivers are disabled.

1 PU0 and PU1 output drivers are enabled.

**Reserved** Bit 4 Reserved. **Must always be written with 0.**

**PUIN1** Bit 3 PU1 input data, This bit reflects the logic value on the PU1 terminal.

**PUIN0** Bit 2 PU0 input data, This bit reflects the logic value on the PU0 terminal.

**PUOUT1** Bit 1 PU1 output data. This bits defines the value of the PU1 pin when configured as port function and PUDIR = 1.

**PUOUT0** Bit 0 PU0 output data. This bits defines the value of the PU0 pin when configured as port function and PUDIR = 1.

### USBPWRCTL, USB-Power Control 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

**Reserved** Bits 15-13 Reserved. Reads back as 0.

**SLDOEN** Bit 12 1.8 V (secondary) LDO enable. When set, the LDO is enabled.

**VUSBEN** Bit 11 3.3-V LDO enable. When set, the LDO is enabled.

**VBOFFIE** Bit 10 VBUS "going OFF" interrupt enable

0 Interrupt disabled

1 Interrupt enabled

**VBONIE** Bit 9 VBUS "coming ON" interrupt enable

0 Interrupt disabled

1 Interrupt enabled

**VUOVLIE** Bit 8 VUSB overload indication interrupt enable

0 Interrupt disabled

1 Interrupt enabled

**Reserved** Bit 7 Reserved. Reads back as 0.

**SLDOAON** Bit 6 1.8-V LDO auto-on enable

0 LDO needs to be turned on manually via SLDOEN

1 A "VBUS coming on" transition sets SLDOEN

**OVLAOFF** Bit 5 LDO overload auto-off enable

0 During an overload on the 3.3-V LDO, the LDO automatically enters current-limiting mode and stays there until the condition stops.

1 An overload indication clears the VUSBEN bit.

**USBDETEN** Bit 4 Enable bit for VBUS-on/off events.

0 USB module will not detect USB-PWR VBUS-on/off events

1 USB module will detect USB-PWR VBUS-on/off events

**USBBGVBV** Bit 3 VBUS valid

0 VBUS is not valid yet

1 VBUS is valid and within bounds

**VBOFFIFG** Bit 2 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.

0 No interrupt pending

1 Interrupt pending

**VBONIFG** Bit 1 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.

0 No interrupt pending

1 Interrupt pending

**VUOVLIFG** Bit 0 VUSB overload interrupt flag. This bit indicates that the 3.3-V LDO entered an overload situation.

0 No interrupt pending

1 Interrupt pending

### USBPLLCTL, USB-PLL Control Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | **UPCS** | **Reserved** | | **UPFDEN** | **UPLLEN** |
| r0 | r0 | r0 | rw-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

**Reserved** Bits 15-13 Reserved. Reads back as 0.

**UPCS** Bit 12 PLL clock select

0 XT1CLK is selected as the reference clock

1 XT2CLK is selected as the reference clock

**Reserved** Bits 11-10 Reserved. Reads back as 0.

**UPFDEN** Bit 9 Phase frequency discriminator (PFD) enable

0 PFD is disabled

1 PFD is enabled

**UPLLEN** Bit 8 PLL enable

0 PLL is disabled

1 PLL is enabled

**UCLKSEL** Bits 7-6 USB module clock select. **Must always be written with 00.**

| UCLKSEL value | Selected Clock for USB Module |
|---|---|
| 00 | PLLCLK (default) |
| 01 | Reserved |
| 10 | Reserved |
| 11 | Reserved |

**Reserved** Bits 5-0 Reserved. Reads back as 0.

## USBPLLDIVB, USB-PLL Clock Divider Buffer 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

**Reserved** Bits 15-11 Reserved. Reads back as 0.

**UPQB** Bits 10-8 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.

| UPQB value | Pre-Scaling Ratio |
|---|---|
| 000 | $f_{UPD} = f_{REF}$ |
| 001 | $f_{UPD} = f_{REF} / 2$ |
| 010 | $f_{UPD} = f_{REF} / 3$ |
| 011 | $f_{UPD} = f_{REF} / 4$ |
| 100 | $f_{UPD} = f_{REF} / 6$ |
| 101 | $f_{UPD} = f_{REF} / 8$ |
| 110 | $f_{UPD} = f_{REF} / 12$ |
| 111 | $f_{UPD} = f_{REF} / 16$ |

**Reserved** Bits 7-6 Reserved. Reads back as 0.

**UPMB** Bits 5-0 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.

| UPMB value | Multiplying Factor |
|---|---|
| 000000 | Feedback division rate: 1 |
| 000001 | Feedback division rate: 2 |
| ⋮ | ⋮ |
| 111111 | Feedback division rate: 64 |

## USBPLLIR, USB-PLL Interrupt 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

| | | | |
|---|---|---|---|
| **Reserved** | Bits 15-11 | Reserved. Reads back as 0. | |
| **USBOORIE** | Bit 10 | PLL out-of-range interrupt enable | |
| | | 0 | Interrupt disabled |
| | | 1 | Interrupt enabled |
| **USBLOSIE** | Bit 9 | PLL loss-of-signal interrupt enable | |
| | | 0 | Interrupt disabled |
| | | 1 | Interrupt enabled |
| **USBOOLIE** | Bit 8 | PLL out-of-lock interrupt enable | |
| | | 0 | Interrupt disabled |
| | | 1 | Interrupt enabled |
| **Reserved** | Bits 7-3 | Reserved. Reads back as 0. | |
| **USBOORIFG** | Bit 2 | PLL out-of-range interrupt flag | |
| | | 0 | No interrupt pending |
| | | 1 | Interrupt pending |
| **USBLOSIFG** | Bit 1 | PLL loss-of-signal interrupt flag | |
| | | 0 | No interrupt pending |
| | | 1 | Interrupt pending |
| **USBOOLIFG** | Bit 0 | PLL out-of-lock interrupt flag | |
| | | 0 | No interrupt pending |
| | | 1 | Interrupt pending |

### 1.3.2 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 1-6. All addresses are expressed as offsets; the base address can be found in the device-specific datasheet.

All registers are byte and word accessible.

**Table 1-6. USB Control Registers**

| Register | | Short Form | Register Type | Address Offset | Initial State |
|---|---|---|---|---|---|
| Endpoint 0 configuration | Input endpoint_0: Configuration | USBIEPCNF_0 | Read/Write | 00h | 00h |
| | Input endpoint_0: Byte Count | USBIEPCNT_0 | Read/Write | 01h | 80h |
| | Output endpoint_0: Configuration | USBOEPCNF_0 | Read/Write | 02h | 00h |
| | Output endpoint_0: Byte count | USBOEPCNT_0 | Read/Write | 03h | 00h |
| Interrupts | Input endpoint interrupt enables | USBIEPIE | Read/Write | 0Eh | 00h |
| | Output endpoint interrupt enables | USBOEPIE | Read/Write | 0Fh | 00h |
| | Input endpoint interrupt flags | USBIEPIFG | Read/Write | 10h | 00h |
| | Output endpoint interrupt flags | USBOEPIFG | Read/Write | 11h | 00h |
| | Vector interrupt register | USBVECINT | Read/Write | 12h | 0000h |
| Timestamps | Timestamp maintenance register | USBMAINT | Read/Write | 16h | 0000h |
| | Timestamp register | USBTSREG | Read/Write | 18h | 0000h |
| Basic USB control | USB frame number | USBFN | Read only | 1Ah | 0000h |
| | USB control register | USBCTL | Read/Write | 1Ch | 00h |
| | USB interrupt enable register | USBIE | Read/Write | 1Dh | 00h |
| | USB interrupt flag register | USBIFG | Read/Write | 1Eh | 00h |
| | Function address register | USBFUNADR | Read/Write | 1Fh | 00h |

## USBIEPCNF_0 USB Input Endpoint-0 Configuration 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

**UBME** Bit 7 UBM in endpoint-0 enable

0 UBM cannot use this endpoint

1 UBM can use this endpoint

**Reserved** Bit 6 Reserved. Reads back as 0.

**TOGGLE** Bit 5 Toggle bit. Reads back 0, since the configuration endpoint does not need to toggle.

**Reserved** Bit 4 Reserved

**STALL** Bit 3 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.

0 Indicates no stall

1 Indicates stall

**USBIIE** Bit 2 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).

0 Corresponding interrupt flag is not set

1 Corresponding interrupt flag is set

**Reserved** Bits 1-0 Reserved. Reads back as 0.

## USBIEPBCNT_0 USB Input Endpoint-0 Byte Count 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

**NAK** Bit 7 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.

0 Buffer contains a valid data packet for host device

1 Buffer is empty (Host-In request receives a NAK)

**Reserved** Bits 6-4 Reserved. Reads back as 0.

**CNT** Bits 3-0 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)

### USBOEPCNFG_0 USB Output Endpoint-0 Configuration 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

**UBME** Bit 7 UBM out Endpoint-0 enable

0 UBM cannot use this endpoint

1 UBM can use this endpoint

**Reserved** Bit 6 Reserved. Reads back as 0.

**TOGGLE** Bit 5 Toggle bit. Reads back 0, since the configuration endpoint does not need to toggle.

**Reserved** Bit 4 Reserved. Reads back as 0.

**STALL** Bit 3 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.

0 Indicates no stall

1 Indicates stall

**USBIIE** Bit 2 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).

0 Corresponding interrupt flag will not be set

1 Corresponding interrupt flag will be set

**Reserved** Bits 1-0 Reserved. Reads back as 0.

### USBOEPBCNT_0 USB Output Endpoint-0 Byte Count 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

**NAK** Bit 7 No acknowledge status bit. This bit is set by the UBM at the end of a successful USB out transaction into endpoint-0, in order 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 will 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.

0 No valid data in the buffer. The buffer is ready to receive a host OUT transaction

1 The buffer contains a valid packet from the host that has not been picked up. (Any subsequent Host-Out requests receive a NAK.)

**Reserved** Bits 6-4 Reserved. Reads back as 0.

**CNT** Bits 3-0 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

### USBIEPIE, USB Input Endpoint Interrupt Enable 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

**IEPIEn** Bits 7-0 Input endpoint interrupt enable. These bits enable/disable whether an event can trigger an interrupt; they do not influence whether the event gets flagged. This is enabled/disabled with the interrupt indication enable bit in the Endpoint descriptors.

0 Event does not generate an interrupt

1 Event does generate an interrupt

### USBOEPIE, USB Output Endpoint Interrupt Enable 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

**IEPIEn** Bits 7-0 Output endpoint interrupt enable. These bits enable/disable whether an event can trigger an interrupt; they do not influence whether the event gets flagged. This is enabled/disabled with the interrupt indication enable bit in the Endpoint descriptors.

0 Event does not generate an interrupt

1 Event does generate an interrupt

### USBIEPIFG, USB Input Endpoint Interrupt Flag 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

**OEPIFGn** Bits 7-0 Input Endpoint Interrupt Flag. These bits are set by the UBM when a successful completion of a transaction occurs for this endpoint. When set, a USB interrupt will be generated. The interrupt flag will be cleared when the MCU reads the value from the USBVECINT 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.

### USBOEPIFG, USB Output Endpoint Interrupt Flag 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

**OEPIFGn** Bits 7-0 Output Endpoint Interrupt Flag. The output endpoint interrupt flag bits for a particular USB output endpoint are set to a "1" by the UBM when a successful completion of a transaction occurs to that out endpoint. When a bit is set, a USB interrupt will be generated. The interrupt flag will be cleared when the MCU reads the value from the USBVECINT 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 that bit location.

## USBVECINT, USB Interrupt Vector Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| r0 | r0 | r0 | r0 | r0 | r0 | r0 | r0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **USBIV** | | | | | **0** |
| r0 | r0 | r-0 | r-0 | r-0 | r-0 | r-0 | r0 |

**USBIV** Bits 15-0 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 will clear all pending USB interrupt flags independent of the status of USBEN.

| USBIV Contents | Interrupt Source | Interrupt Flag | Interrupt Priority |
|---|---|---|---|
| 00h | No interrupt pending | — | — |
| 02h | See Section 1.2.5 | | Highest |
| 3Eh | | | Lowest |

### USBMAINT, Timestamp Maintenance 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

**UTSEL** Bits 15-13 USB timer selection

| UTSEL | USB Timer Period | Approximate Frequency |
|---|---|---|
| 000 | 4096 µs | ~250 Hz (244 Hz) |
| 001 | 2048 µs | ~ 500 Hz (488 Hz) |
| 010 | 1024 µs | ~ 1 kHz (977 Hz) |
| 011 | 512 µs | ~ 2 kHz (1953 Hz) |
| 100 | 256 µs | ~ 4 kHz (3906 Hz) |
| 101 | 128 µs | ~ 8 kHz (7812 Hz) |
| 110 | 64 µs | ~ 16 kHz (15625 Hz) |
| 111 | 32 µs | ~ 31 kHz (31250 Hz) |

**Reserved** Bit 12 Reserved. Read back as 0

**TSE3** Bit 11 Timestamp Event #3 bit. This bit allows the triggering of a software-driven timestamp event (when TSESEL=11).

0 no TSE3 event signaled

1 TSE3 event signaled

**TSESEL** Bits 10-9 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 datasheet

| TSESEL | Source of Timestamp Event |
|---|---|
| 00 | TSE0 (DMA0) signal is qualified timestamp event |
| 01 | TSE1 (DMA1) signal is qualified timestamp event |
| 10 | TSE2 (DMA2) signal is qualified timestamp event |
| 11 | Software-driven timestamp event |

**TSGEN** Bit 8 Timestamp Generator Enable

0 Timestamp mechanism disabled

1 Timestamp mechanism enabled

**Reserved** Bits 7-2 Reserved. Read back as 0

**UTIE** Bit 1 USB timer interrupt enable bit

0 USB timer interrupt disabled

1 USB timer interrupt enabled

**UTIFG** Bit 0 USB timer interrupt flag

0 No interrupt pending

1 Interrupt pending

## USBTSREG, USB Timestamp 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

**TVAL** Bits 15-0 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.

## USBFN, USB Frame Number 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 |

**Reserved** Bits 15-11 Reserved. Read back as 0

**USBFN** Bits 10-0 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.

## USBCTL, USB Control 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

**Reserved** Bit 7 Reserved. Read back as 0.

**FEN** Bit 6 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 will ignore all USB transactions. It is cleared by a USB reset. (This bit is primarily intended for debugging.)

0 Function is disabled

1 Function is enabled

**RWUP** Bit 5 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.

0 Writing 0 has no effect

1 A Remote-Wakeup pulse will be generated

**FRSTE** Bit 4 Function Reset Connection Enable. This bit selects whether a bus reset on the USB causes an internal reset of the USB module.

0 Bus reset does not cause a reset of the module

1 Bus reset does cause a reset of the module

**Reserved** Bits 3-1 Reserved. Read back as 0.

**DIR** Bit 0 Data response to setup packet interrupt status bit. Software must decode the request and set/clear this bit to reflect the data transfer direction.

0 USB data-OUT transaction (from host to device)

1 USB data-IN transaction (from device to host)

## USBIE, USB Interrupt Enable 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

**RSTRIE** Bit 7 USB reset interrupt enable. Causes an interrupt to be generated if the RSTRIFG bit is set.

0 Function Reset interrupt disabled

1 Function Reset interrupt enabled

**SUSRIE** Bit 6 Suspend interrupt enable. Causes an interrupt to be generated if the SUSRIFG bit is set.

0 Suspend interrupt disabled

1 Suspend interrupt enabled

**RESRIE** Bit 5 Resume interrupt enable. Causes an interrupt to be generated if the RESRIFG bit is set.

0 Resume interrupt disabled

1 Resume interrupt enabled

**Reserved** Bits 4-3 Reserved. Read back as 0.

**SETUPIE** Bit 2 Setup interrupt enable. Causes an interrupt to be generated if the SETUPIFG bit is set.

0 Setup interrupt disabled

1 Setup interrupt enabled

**Reserved** Bit 1 Reserved. Read back as 0.

**STPOWIE** Bit 0 Setup Overwrite interrupt enable. Causes an interrupt to be generated if the STPOWIFG bit is set.

0 Setup Overwrite interrupt disabled

1 Setup Overwrite interrupt enabled

### USBIFG, USB Interrupt Flag 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

| | | |
|---|---|---|
| **RSTRIFG** | Bit 7 | 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 will not be affected. |
| **SUSRIFG** | Bit 6 | Suspend request bit. This bit is set by hardware in response to the host/hub causing a global or selective suspend condition. |
| **RESRIFG** | Bit 5 | Resume request bit. This bit is set by hardware in response to the host/hub causing a resume event. |
| **Reserved** | Bits 4-3 | Reserved. Read back as 0. |
| **SETUPIFG** | Bit 2 | 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. |
| **Reserved** | Bit 1 | Reserved. Read back as 0. |
| **STPOWIFG** | Bit 0 | 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. |

### USBFUNADR, USB Function Address 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

| | | |
|---|---|---|
| **Reserved** | Bit 7 | Reserved. Read back as 0. |
| **FA[6:0]** | Bits 6-0 | 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. |

### 1.3.3 USB Buffer Registers and Memory

The data buffers for all endpoints, as well as the registers that define endpoints 1-7, are stored in the USB RAM buffer memory. Doing so allows for efficient, 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 1-7. The registers are listed in Table 1-8. All addresses are expressed as offsets; the base address can be found in the device-specific datasheet.

All memory is byte and word accessible.

**Table 1-7. USB Buffer Memory**

| Memory | Short Form | Access Type | Address Offset |
|---|---|---|---|
| Start of buffer space | USBSTABUFF | Read/Write | 0000h |
| 1904 bytes of configurable buffer space | USBIEPCNT_0 | Read/Write | ⋮ |
| End of buffer space | USBTOPBUFF | 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 |

**Table 1-8. USB Buffer Descriptor Registers**

| Register | | Short Form | Access Type | Address Offset |
|---|---|---|---|---|
| Output Endpoint_1 | Configuration Register | USBOEPCNF_1 | Read/Write | 0788h |
| | X-buffer base address Register | USBOEPBBAX_1 | Read/Write | 0789h |
| | X-byte count Register | USBOEPBCTX_1 | Read/Write | 078Ah |
| | Y-buffer base address Register | USBOEPBBAY_1 | Read/Write | 078Dh |
| | Y-byte count Register | USBOEPBCTY_1 | Read/Write | 078Eh |
| | X/Y-buffer size Register | USBOEPSIZXY_1 | Read/Write | 078Fh |
| Output Endpoint_2 | Configuration Register | USBOEPCNF_2 | Read/Write | 0790h |
| | X-buffer base address Register | USBOEPBBAX_2 | Read/Write | 0791h |
| | X-byte count Register | USBOEPBCTX_2 | Read/Write | 0792h |
| | Y-buffer base address Register | USBOEPBBAY_2 | Read/Write | 0795h |
| | Y-byte count Register | USBOEPBCTY_2 | Read/Write | 0796h |
| | X/Y-buffer size Register | USBOEPSIZXY_2 | Read/Write | 0797h |
| Output Endpoint_3 | Configuration Register | USBOEPCNF_3 | Read/Write | 0798h |
| | X-buffer base address Register | USBOEPBBAX_3 | Read/Write | 0799h |
| | X-byte count Register | USBOEPBCTX_3 | Read/Write | 079Ah |
| | Y-buffer base address Register | USBOEPBBAY_3 | Read/Write | 079Dh |
| | Y-byte count Register | USBOEPBCTY_3 | Read/Write | 079Eh |
| | X/Y-buffer size Register | USBOEPSIZXY_3 | Read/Write | 079Fh |

### Table 1-8. USB Buffer Descriptor Registers (continued)

| Register | | Short Form | Access Type | Address Offset |
|---|---|---|---|---|
| Output Endpoint_4 | Configuration Register | USBOEPCNF_4 | Read/Write | 07A0h |
| | X-buffer base address Register | USBOEPBBAX_4 | Read/Write | 07A1h |
| | X-byte count Register | USBOEPBCTX_4 | Read/Write | 07A2h |
| | Y-buffer base address Register | USBOEPBBAY_4 | Read/Write | 07A5h |
| | Y-byte count Register | USBOEPBCTY_4 | Read/Write | 07A6h |
| | X/Y-buffer size Register | USBOEPSIZXY_4 | Read/Write | 07A7h |
| Output Endpoint_5 | Configuration Register | USBOEPCNF_5 | Read/Write | 07A8h |
| | X-buffer base address Register | USBOEPBBAX_5 | Read/Write | 07A9h |
| | X-byte count Register | USBOEPBCTX_5 | Read/Write | 07AAh |
| | Y-buffer base address Register | USBOEPBBAY_5 | Read/Write | 07ADh |
| | Y-byte count Register | USBOEPBCTY_5 | Read/Write | 07AEh |
| | X/Y-buffer size Register | USBOEPSIZXY_5 | Read/Write | 07AFh |
| Output Endpoint_6 | Configuration Register | USBOEPCNF_6 | Read/Write | 07B0h |
| | X-buffer base address Register | USBOEPBBAX_6 | Read/Write | 07B1h |
| | X-byte count Register | USBOEPBCTX_6 | Read/Write | 07B2h |
| | Y-buffer base address Register | USBOEPBBAY_6 | Read/Write | 07B5h |
| | Y-byte count Register | USBOEPBCTY_6 | Read/Write | 07B6h |
| | X/Y-buffer size Register | USBOEPSIZXY_6 | Read/Write | 07B7h |
| Output Endpoint_7 | Configuration Register | USBOEPCNF_7 | Read/Write | 07B8h |
| | X-buffer base address Register | USBOEPBBAX_7 | Read/Write | 07B9h |
| | X-byte count Register | USBOEPBCTX_7 | Read/Write | 07BAh |
| | Y-buffer base address Register | USBOEPBBAY_7 | Read/Write | 07BDh |
| | Y-byte count Register | USBOEPBCTY_7 | Read/Write | 07BEh |
| | X/Y-buffer size Register | USBOEPSIZXY_7 | Read/Write | 07BFh |
| Input Endpoint_1 | Configuration Register | USBIEPCNF_1 | Read/Write | 07C8h |
| | X-buffer base address Register | USBIEPBBAX_1 | Read/Write | 07C9h |
| | X-byte count Register | USBIEPBCTX_1 | Read/Write | 07CAh |
| | Y-buffer base address Register | USBIEPBBAY_1 | Read/Write | 07CDh |
| | Y-byte count Register | USBIEPBCTY_1 | Read/Write | 07CEh |
| | X/Y-buffer size Register | USBIEPSIZXY_1 | Read/Write | 07CFh |
| Input Endpoint_2 | Configuration Register | USBIEPCNF_2 | Read/Write | 07D0h |
| | X-buffer base address Register | USBIEPBBAX_2 | Read/Write | 07D1h |
| | X-byte count Register | USBIEPBCTX_2 | Read/Write | 07D2h |
| | Y-buffer base address Register | USBIEPBBAY_2 | Read/Write | 07D5h |
| | Y-byte count Register | USBIEPBCTY_2 | Read/Write | 07D6h |
| | X/Y-buffer size Register | USBIEPSIZXY_2 | Read/Write | 07D7h |
| Input Endpoint_3 | Configuration Register | USBIEPCNF_3 | Read/Write | 07D8h |
| | X-buffer base address Register | USBIEPBBAX_3 | Read/Write | 07D9h |
| | X-byte count Register | USBIEPBCTX_3 | Read/Write | 07DAh |
| | Y-buffer base address Register | USBIEPBBAY_3 | Read/Write | 07DDh |
| | Y-byte count Register | USBIEPBCTY_3 | Read/Write | 07DEh |
| | X/Y-buffer size Register | USBIEPSIZXY_3 | Read/Write | 07DFh |

Table 1-8. USB Buffer Descriptor Registers (continued)

| Register | | Short Form | Access Type | Address Offset |
|---|---|---|---|---|
| Input Endpoint_4 | Configuration Register | USBIEPCNF_4 | Read/Write | 07E0h |
| | X-buffer base address Register | USBIEPBBAX_4 | Read/Write | 07E1h |
| | X-byte count Register | USBIEPBCTX_4 | Read/Write | 07E2h |
| | Y-buffer base address Register | USBIEPBBAY_4 | Read/Write | 07E5h |
| | Y-byte count Register | USBIEPBCTY_4 | Read/Write | 07E6h |
| | X/Y-buffer size Register | USBIEPSIZXY_4 | Read/Write | 07E7h |
| Input Endpoint_5 | Configuration Register | USBIEPCNF_5 | Read/Write | 07E8h |
| | X-buffer base address Register | USBIEPBBAX_5 | Read/Write | 07E9h |
| | X-byte count Register | USBIEPBCTX_5 | Read/Write | 07EAh |
| | Y-buffer base address Register | USBIEPBBAY_5 | Read/Write | 07EDh |
| | Y-byte count Register | USBIEPBCTY_5 | Read/Write | 07EEh |
| | X/Y-buffer size Register | USBIEPSIZXY_5 | Read/Write | 07EFh |
| Input Endpoint_6 | Configuration Register | USBIEPCNF_6 | Read/Write | 07F0h |
| | X-buffer base address Register | USBIEPBBAX_6 | Read/Write | 07F1h |
| | X-byte count Register | USBIEPBCTX_6 | Read/Write | 07F2h |
| | Y-buffer base address Register | USBIEPBBAY_6 | Read/Write | 07F5h |
| | Y-byte count Register | USBIEPBCTY_6 | Read/Write | 07F6h |
| | X/Y-buffer size Register | USBIEPSIZXY_6 | Read/Write | 07F7h |
| Input Endpoint_7 | Configuration Register | USBIEPCNF_7 | Read/Write | 07F8h |
| | X-buffer base address Register | USBIEPBBAX_7 | Read/Write | 07F9h |
| | X-byte count Register | USBIEPBCTX_7 | Read/Write | 07FAh |
| | Y-buffer base address Register | USBIEPBBAY_7 | Read/Write | 07FDh |
| | Y-byte count Register | USBIEPBCTY_7 | Read/Write | 07FEh |
| | X/Y-buffer size Register | USBIEPSIZXY_7 | Read/Write | 07FFh |

## USBOEPCNF_n, Output Endpoint-n Configuration 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

**UBME** Bit 7 UBM out endpoint-n enable. This bit is to be set/cleared by software.

0 UBM cannot use this endpoint

1 UBM can use this endpoint

**Reserved** Bit 6 Reserved. Read back as 0.

**TOGGLE** Bit 5 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 PID matches the expected PID.

**DBUF** Bit 4 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.

0 Primary buffer only (X-buffer only)

1 Toggle bit selects buffer

**STALL** Bit 3 USB stall condition. This bit can be set to cause endpoint transactions to be stalled. When set, the hardware will automatically return a stall handshake to the host for any transaction received on endpoint-0. The stall bit is cleared automatically by the next setup transaction.

0 Indicates no stall

1 Indicates stall

**USBIIE** Bit 2 USB transaction interrupt indication enable. Can be set/cleared to define if interrupts are to be flagged in general. To generate an interrupt, the corresponding interrupt flag must be set (OEPIE).

0 Corresponding interrupt flag will not be set

1 Corresponding interrupt flag will be set

**Reserved** Bits 1-0 Reserved. Read back as 0.

## USBOEPBBAX_n, Output Endpoint-n X-buffer Base Address 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

**ADR** Bits 7-0 X-buffer base address. These are the upper seven 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.

### USBOEPBCTX_n, Output Endpoint-n X-byte Count 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

| | | |
|---|---|---|
| **NAK** | Bit 7 | 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, in order 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 will 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.<br>0 No valid data in buffer. The buffer is ready to receive OUT packets from the host.<br>1 The buffer contains a valid packet from the host, and it has not been picked up (subsequent host-out requests receive a NAK) |
| **CNT** | Bits 6-0 | 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.<br>000:0000b to 100:0000b are valid numbers for 0 to 64 bytes.<br>Any value ≥ 100:0001b results in unpredictable results. |

### USBOEPBBAY_n, Output Endpoint-n Y-buffer Base Address 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

| | | |
|---|---|---|
| **ADR** | Bits 7-0 | Y-buffer base address. These are the upper seven 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. |

### USBOEPBCTY_n, Output Endpoint-n X-byte Count 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

| | | |
|---|---|---|
| **NAK** | Bit 7 | 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, in order 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 will 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.<br>0 No valid data in buffer. The buffer is ready to receive OUT packets from the host.<br>1 The buffer contains a valid packet from the host, and it has not been picked up (subsequent host-out requests receive a NAK) |
| **CNT** | Bits 6-0 | 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.<br>000:0000b to 100:0000b are valid numbers for 0 to 64 bytes.<br>Any value ≥ 100:0001b results in unpredictable results. |

### USBOEPSIZXY_n, Output Endpoint-n X/Y-buffer Size 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

| | | |
|---|---|---|
| **Reserved** | Bit 7 | Reserved. Read back as 0. |
| **SIZx** | Bits 6-0 | 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.<br>000:0000b to 100:0000b are valid numbers for 0 to 64 bytes.<br>Any value ≥ 100:0001b results in unpredictable results. |

### USBIEPCNF_n, Input Endpoint-n Configuration 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

| | | |
|---|---|---|
| **UBME** | Bit 7 | UBM in endpoint-n enable. This value needs to be set/cleared by software.<br>0 UBM cannot use this endpoint<br>1 UBM can use this endpoint |
| **Reserved** | Bit 6 | Reserved. Read back as 0. |
| **TOGGLE** | Bit 5 | 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 PID is transmitted in the data packet to the host. If this bit is set, a DATA1 PID is transmitted in the data packet. |
| **DBUF** | Bit 4 | 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.<br>0 Primary buffer only (X-buffer only)<br>1 Toggle bit selects buffer |
| **STALL** | Bit 3 | USB stall condition. This bit can be set to cause endpoint transactions to be stalled. When set, the hardware will automatically return a stall handshake to the host for any transaction received on endpoint-0. The stall bit is cleared automatically by the next setup transaction.<br>0 Indicates no stall<br>1 Indicates stall |
| **USBIIE** | Bit 2 | USB transaction interrupt indication enable. Can be set/cleared to define if interrupts are to be flagged in general. To generate an interrupt the corresponding interrupt flag must be set (OEPIE).<br>0 Corresponding interrupt flag will not be set<br>1 Corresponding interrupt flag will be set |
| **Reserved** | Bits 1-0 | Reserved. Read back as 0. |

### USBIEPBBAX_n, Input Endpoint-n X-buffer Base Address 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

| | | |
|---|---|---|
| **ADR** | Bits 7-0 | X-buffer base address. These are the upper seven 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. |

## USBIEPBCTX_n, Input Endpoint-n X-byte Count 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

| | | |
|---|---|---|
| **NAK** | Bit 7 | 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, in order 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.<br>0 Buffer contains a valid data packet for the host<br>1 Buffer is empty (any host-In requests receive a NAK) |
| **CNT** | Bits 6-0 | 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.<br>000:0000b to 100:0000b are valid numbers for 0 to 64 bytes.<br>Any value ≥ 100:0001b results in unpredictable results. |

## USBIEPBBAY_n, Input Endpoint-n Y-buffer Base Address 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

| | | |
|---|---|---|
| **ADR** | Bits 7-0 | Y-buffer base address. These are the upper seven 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. |

## USBIEPBCTY_n, Input Endpoint-n Y-byte Count 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

| | | |
|---|---|---|
| **NAK** | Bit 7 | 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, in order 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.<br>0 Buffer contains a valid data packet for host device<br>1 Buffer is empty (any host-in requests receive a NAK) |
| **CNT** | Bits 6-0 | 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.<br>000:0000b to 100:0000b are valid numbers for 0 to 64 bytes.<br>Any value ≥ 100:0001b results in unpredictable results. |

### USBIEPSIZXY_n, Input Endpoint-n X/Y-buffer Size 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

| | | |
|---|---|---|
| **Reserved** | Bit 7 | Reserved. Read back as 0. |
| **SIZ** | Bits 6-0 | 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.<br>000:0000b to 100:0000b are valid numbers for 0 to 64 bytes.<br>Any value ≥ 100:0001b results in unpredictable results. |

# IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DLP® Products | www.dlp.com | Broadband | www.ti.com/broadband |
| DSP | dsp.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Clocks and Timers | www.ti.com/clocks | Medical | www.ti.com/medical |
| Interface | interface.ti.com | Military | www.ti.com/military |
| Logic | logic.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Power Mgmt | power.ti.com | Security | www.ti.com/security |
| Microcontrollers | microcontroller.ti.com | Telephony | www.ti.com/telephony |
| RFID | www.ti-rfid.com | Video & Imaging | www.ti.com/video |
| RF/IF and ZigBee® Solutions | www.ti.com/lprf | Wireless | www.ti.com/wireless |

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