20. USB – Universal Serial Bus Interface

20.1 Features

- USB 2.0 full speed (12Mbps) and low speed (1.5Mbps) device compliant interface
- Integrated on-chip USB transceiver, no external components needed
- 16 endpoint addresses with full endpoint flexibility for up to 31 endpoints
  - One input endpoint per endpoint address
  - One output endpoint per endpoint address
- Endpoint address transfer type selectable to
  - Control transfers
  - Interrupt transfers
  - Bulk transfers
  - Isochronous transfers
- Configurable data payload size per endpoint, up to 1023 bytes
- Endpoint configuration and data buffers located in internal SRAM
  - Configurable location for endpoint configuration data
  - Configurable location for each endpoint's data buffer
- Built-in direct memory access (DMA) to internal SRAM for:
  - Endpoint configurations
  - Reading and writing endpoint data
- Ping-pong operation for higher throughput and double buffered operation
  - Input and output endpoint data buffers used in a single direction
  - CPU/DMA controller can update data buffer during transfer
- Multi-packet transfer for reduced interrupt load and software intervention
  - Data payload exceeding maximum packet size is transferred in one continuous transfer
  - No interrupts or software interaction on packet transaction level
- Transaction complete FIFO for workflow management when using multiple endpoints
  - Tracks all completed transactions in a first-come, first-served work queue
- Clock selection independent of system clock source and selection
- Minimum 1.5MHz CPU clock required for low speed USB operation
- Minimum 12MHz CPU clock required for full speed operation
- Connection to event system
- On chip debug possibilities during USB transactions

20.2 Overview

The USB module is a USB 2.0 full speed (12Mbps) and low speed (1.5Mbps) device compliant interface.

The USB supports 16 endpoint addresses. All endpoint addresses have one input and one output endpoint, for a total of 31 configurable endpoints and one control endpoint. Each endpoint address is fully configurable and can be configured for any of the four transfer types: control, interrupt, bulk, or isochronous. The data payload size is also selectable, and it supports data payloads up to 1023 bytes.

No dedicated memory is allocated for or included in the USB module. Internal SRAM is used to keep the configuration for each endpoint address and the data buffer for each endpoint. The memory locations used for endpoint configurations and data buffers are fully configurable. The amount of memory allocated is fully dynamic, according to the number of endpoints in use and the configuration of these. The USB module has built-in direct memory access (DMA), and will read/write data from/to the SRAM when a USB transaction takes place.

To maximize throughput, an endpoint address can be configured for ping-pong operation. When done, the input and output endpoints are both used in the same direction. The CPU or DMA controller can then read/write one data buffer while the USB module writes/reads the others, and vice versa. This gives double buffered communication.

Multipacket transfer enables a data payload exceeding the maximum packet size of an endpoint to be transferred as multiple packets without software intervention. This reduces the CPU intervention and the interrupts needed for USB transfers.

For low-power operation, the USB module can put the microcontroller into any sleep mode when the USB bus is idle and a suspend condition is given. Upon bus resumes, the USB module can wake up the microcontroller from any sleep mode.

Figure 20-1. USB OUT transfer: data packet from host to USB device.

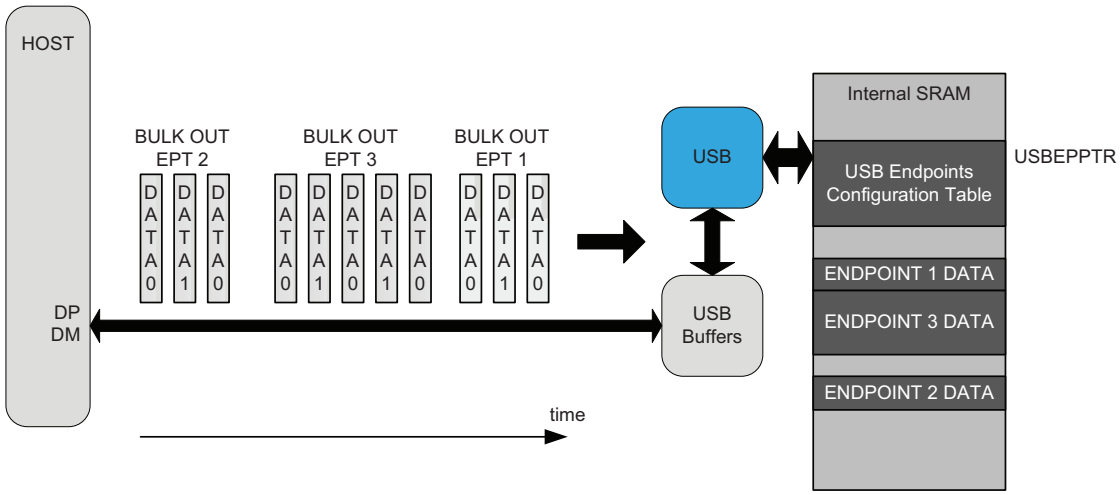
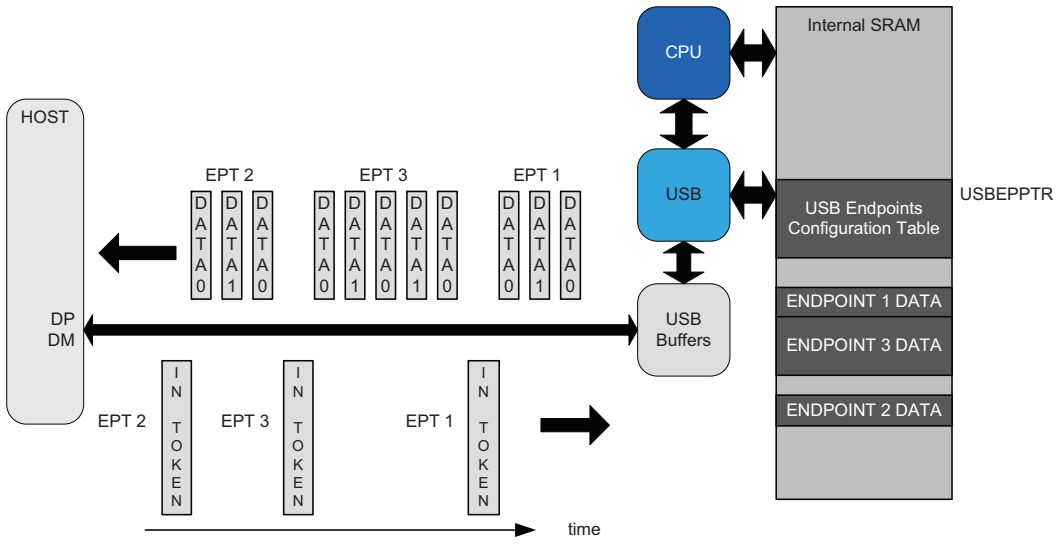


Figure 20-2. USB IN transfer: data packet from USB device to host after request from host.



20.3 Operation

This section gives an overview of the USB module operation during normal transactions. For general details on USB and the USB protocol, please refer to http://www.usb.org and the USB specification documents.

20.3.1 Start of Frame

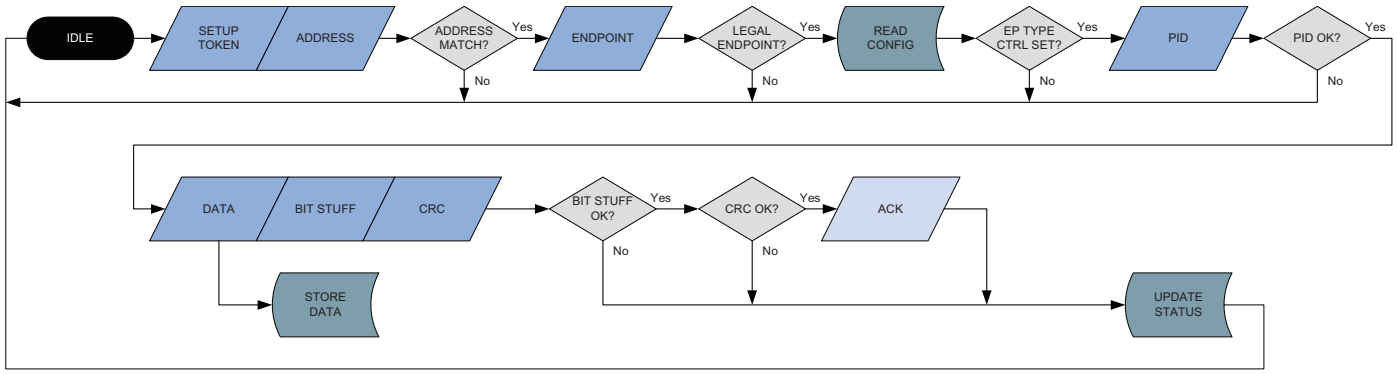
When a start of frame (SOF) token is detected and storing of the frame numbers is enabled, the frame number from the token is stored in the frame number register (FRAMENUM) and the start of frame interrupt flag (SOFIF) in the interrupt

flag B clear/set register (INTFLAGSBCLR/SET) is set. If there was a CRC or bit-stuff error, the frame error (FRAMEERR) flag in FRAMENUM is set.

20.3.2 SETUP

When a SETUP token is detected, the USB module fetches the endpoint control register (CTRL) from the addressed output endpoint in the endpoint configuration table. If the endpoint type is not set to control, the USB module returns to idle and waits for the next token packet.

Figure 20-3. SETUP transaction.



The USB module then fetches the endpoint data pointer register (DATAPTR) and waits for a DATA0 packet. If a PID error or any other PID than DATA0 is detected, the USB module returns to idle and waits for the next token packet.

The incoming data are written to the data buffer pointed to by DATAPTR. If a bit-stuff error is detected in the incoming data, the USB module returns to idle and waits for the next token packet. If the number of received data bytes exceeds the endpoint's maximum data payload size, as specified by the data size (SIZE) in the endpoint CTRL register, the remaining received data bytes are discarded. The packet will still be checked for bit-stuff and CRC errors. Software must never report a maximum data payload size to the host that is greater than specified in SIZE. If there was a bit-stuff or CRC error in the packet, the USB module returns to idle and waits for the next token packet.

If data was successfully received, an ACK handshake is returned to the host, and the number of received data bytes, excluding the CRC, is written to the endpoint byte counter (CNT). If the number of received data bytes is the maximum data payload specified by SIZE, no CRC data are written in the data buffer. If the number of received data bytes is the maximum data payload specified by SIZE minus one, only the first CRC data byte is written in the data buffer. If the number of received data bytes is equal or less than the data byte payload specified by SIZE minus two, the two CRC data bytes are written in the data buffer.

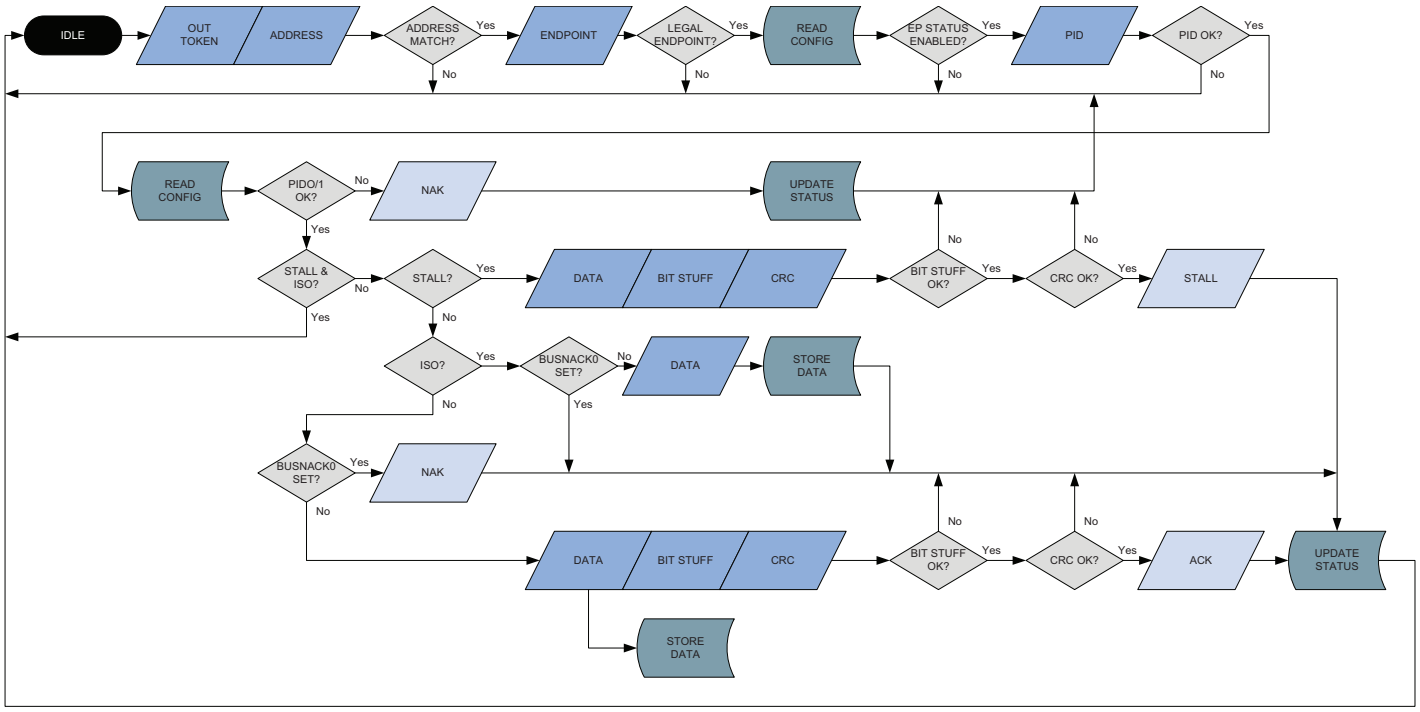
Finally, the setup transaction complete flag (SETUP), data buffer 0 not acknowledge flag (NACK0), and data toggle flag (TOGGLE) are set, while the remaining flags in the endpoint status register (STATUS) are cleared for the addressed input and output endpoints. The setup transaction complete interrupt flag (SETUPIF) in INTFLAGSBCLR/SET is set. The STALL flag in the endpoint CTRL register is cleared for the addressed input and output endpoints.

When a SETUP token is detected and the device address of the token packet does not match that of the endpoint, the packet is discarded, and the USB module returns to idle and waits for the next token packet.

20.3.3 OUT

When an OUT token is detected, the USB module fetches the endpoint CTRL and STATUS register data from the addressed output endpoint in its endpoint configuration table. If the endpoint is disabled, the USB module returns to idle and waits for the next token packet.

Figure 20-4. OUT transaction.



The USB module then fetches the endpoint DATAPTR register and waits for a DATA0 or DATA1 packet. If a PID error or any other PID than DATA0 or DATA1 is detected, the USB module returns to idle and waits for the next token packet.

If the STALL flag in the endpoint CTRL register is set, the incoming data are discarded. If the endpoint is not isochronous, and the bit stuffing and CRC of the received data are OK, a STALL handshake is returned to the host, and the STALL interrupt flag is set.

For isochronous endpoints, data from both a DATA0 and DATA1 packet will be accepted. For other endpoint types, the PID is checked against TOGGLE. If they don't match, the incoming data are discarded and a NAK handshake is returned to the host. If BUSNACK0 is set, the incoming data are discarded. The overflow flag (OVF) in the endpoint STATUS register and the overflow interrupt flag (OVFIF) in the INTFLAGSASET/CLR register are set. If the endpoint is not isochronous, a NAK handshake is returned to the host.

The incoming data are written to the data buffer pointed to by DATAPTR. If a bit-stuff error is detected in the incoming data, the USB module returns to idle and waits for the next token packet. If the number of received data bytes exceeds the maximum data payload specified by SIZE, the remaining received data bytes are discarded. The packet will still be checked for bit-stuff and CRC errors. If there was a bit-stuff or CRC error in the packet, the USB module returns to idle and waits for the next token packet.

If the endpoint is isochronous and there was a bit-stuff or CRC error in the incoming data, the number of received data bytes, excluding CRC, is written to the endpoint CNT register. Finally, CRC and BUSNACK0 in the endpoint and STATUS and CRCIF in INTFLAGSASET/CLR are set.

If data was successfully received, an ACK handshake is returned to the host if the endpoint is not isochronous, and the number of received data bytes, excluding CRC, is written to CNT. If the number of received data bytes is the maximum data payload specified by SIZE no CRC data are written in the data buffer. If the number of received data bytes is the maximum data payload specified by SIZE minus one, only the first CRC data byte is written in the data buffer If the number of received data bytes is equal or less than the data payload specified by SIZE minus two, the two CRC data bytes are written in the data buffer.

Finally, the transaction complete flag (TRNCOMPL0) and BUSNACK0 are set and TOGGLE is toggled if the endpoint is not isochronous. The transaction complete interrupt flag (TRNIF) in INTFLAGSBCLR/SET is set. The endpoint's configuration table address is written to the FIFO if the transaction complete FIFO mode is enabled.

When an OUT token is detected and the device address of the token packet does not match that of the endpoint, the packet is discarded and the USB module returns to idle and waits for the next token packet.

20.3.4 IN

If an IN token is detected the, the USB module fetches the endpoint CTRL and STATUS register data from the addressed input endpoint in the endpoint configuration table. If the endpoint is disabled, the USB module returns to idle and waits for the next token packet.

If the STALL flag in endpoint CTRL register is set, and the endpoint is not isochronous, a STALL handshake is returned to the host, the STALL flag in the endpoint STATUS register and the STALL interrupt flag (STALLIF) in INTFLAGSACLR/SET are set.

If BUSNACK0 is set, OVF in the endpoint STATUS register and OVFIF in the INTFLAGSACLR/SET register are set. If the endpoint is not isochronous, a NAK handshake is returned to the host.

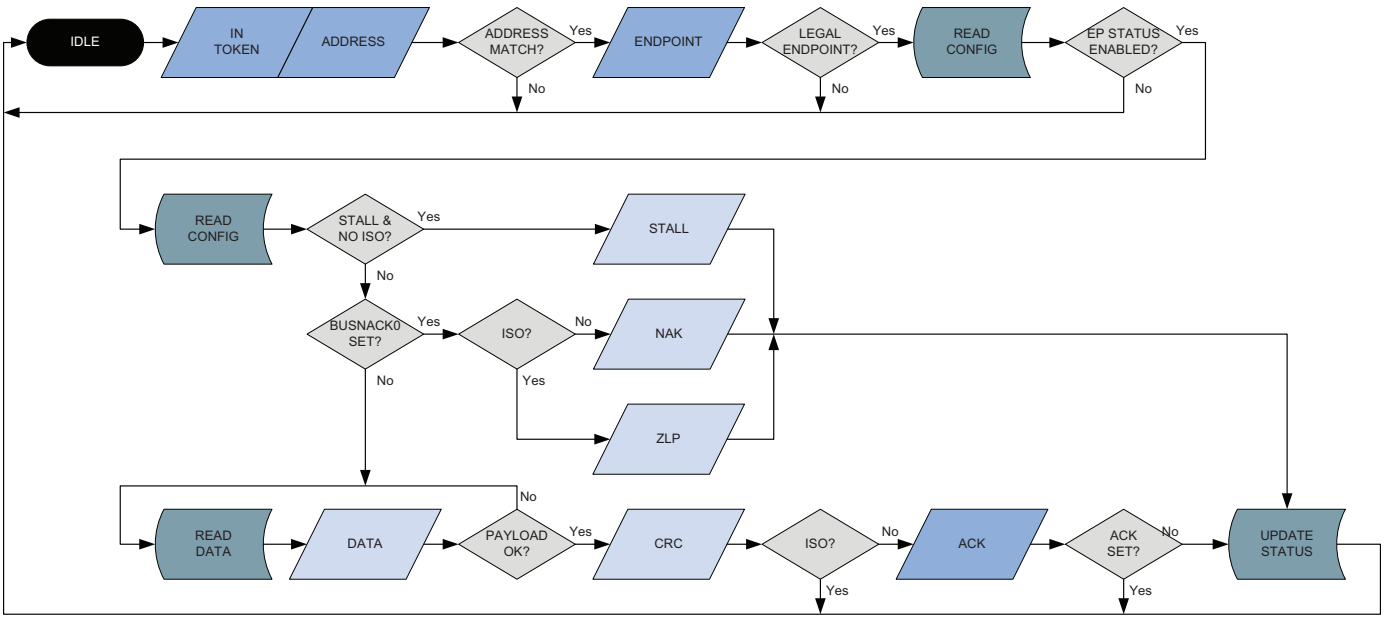
The data in the data buffer pointed to by the endpoint DATAPTR register are sent to the host in a DATA0 packet if the endpoint is isochronous; otherwise, a DATA0 or DATA1 packet according to TOGGLE is sent. When the number of data bytes specified in endpoint CNT is sent, the CRC is appended and sent to the host. If not, a ZLP handshake is returned to the host.

For isochronous endpoints, BUSNACK0 and TRNCOMPL0 in the endpoint STATUS register are set. TRNIF is set, and the endpoint's configuration table address is written to the FIFO if the transaction complete FIFO mode is enabled.

For all non-isochronous endpoints, the USB module waits for an ACK handshake from the host. If an ACK handshake is not received within 16 USB clock cycles, the USB module returns to idle and waits for the next token packet. If an ACK handshake was successfully received, BUSNACK0 and TRNCOMPL0 are set and TOGGLE is toggled. TRNIF is set and the endpoint's configuration table address is written to the FIFO if the transaction complete FIFO mode is enabled.

When an IN token is detected and the device address of the token packet does not match that of the endpoint, the packet is discarded and the USB module returns to idle and waits for the next token packet.

Figure 20-5. IN transaction.



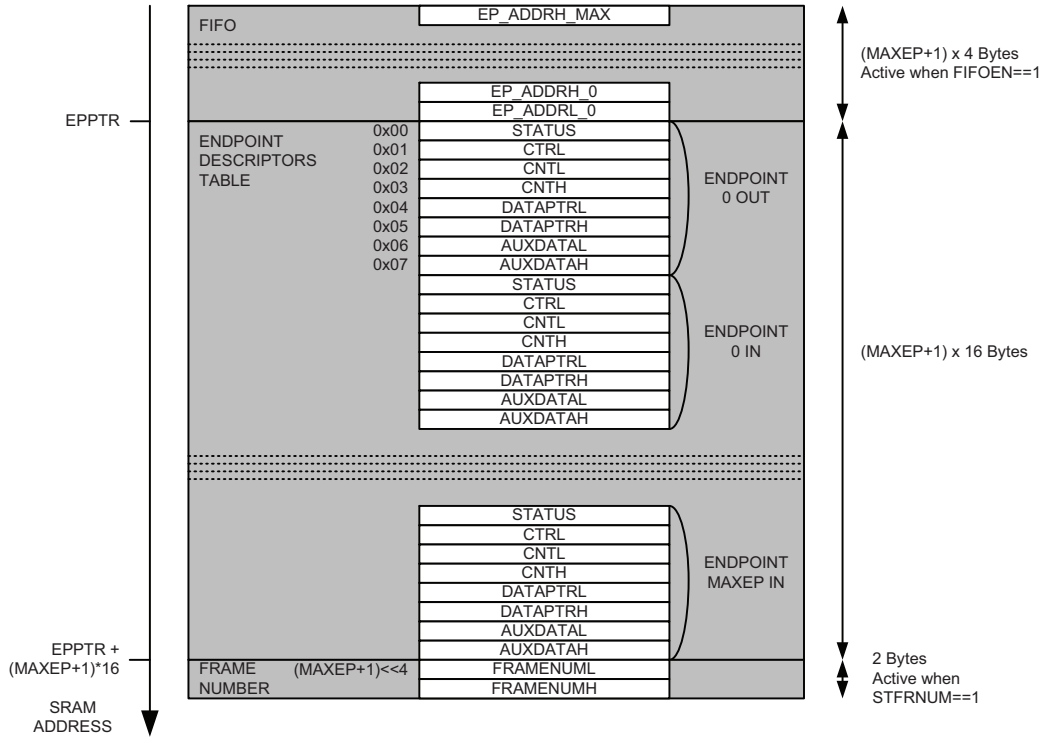
20.4 SRAM Memory Mapping

The USB module uses internal SRAM to store the:

- **Endpoint configuration table**
- **USB frame number**
- **Transaction complete FIFO**

The endpoint pointer register (EPPTR) is used to set the SRAM address for the endpoint configuration table. The USB frame number (FRAMENUM) and transaction complete FIFO (FIFO) locations are derived from this. The locations of these areas are selectable inside the internal SRAM. Figure on page 231 gives the relative memory location of each area.

Figure 20-6. SRAM memory mapping.



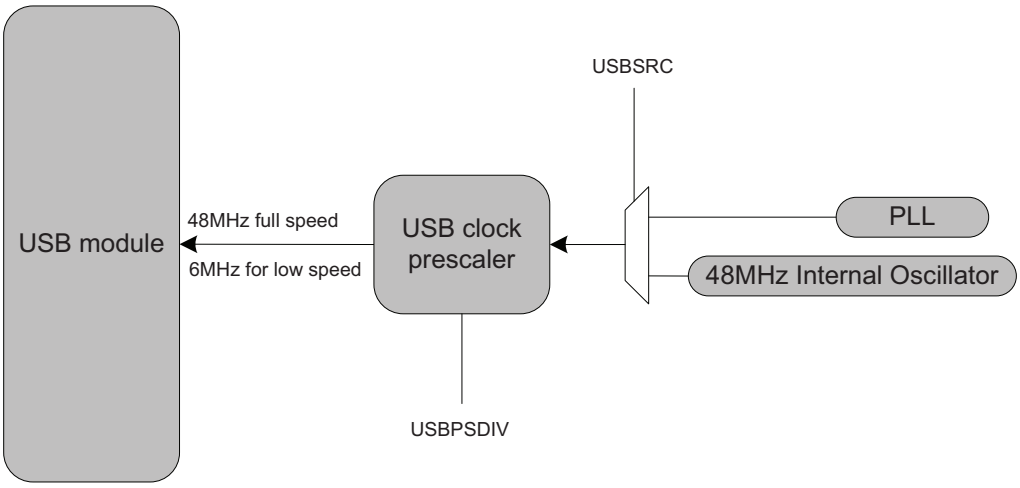
20.5 Clock Generation

The USB module requires a minimum 6MHz clock for USB low speed operation, and a minimum 48MHz clock for USB full speed operation. It can be clocked from internal or external clock sources by using the internal PLL, or directly from the 32MHz internal oscillator when it is tuned and calibrated to 48MHz. The CPU and peripherals clocks must run at a minimum of 1.5MHz for low speed operation, and a minimum of 12MHz for full speed operation.

The USB module clock selection is independent of and separate from the main system clock selection. Selection and setup are done using the main clock control settings. For details, refer to “System Clock and Clock Options” on page 82.

The Figure 20-7 on page 232 shows an overview of the USB module clock selection.

Figure 20-7. Clock generation configuration.



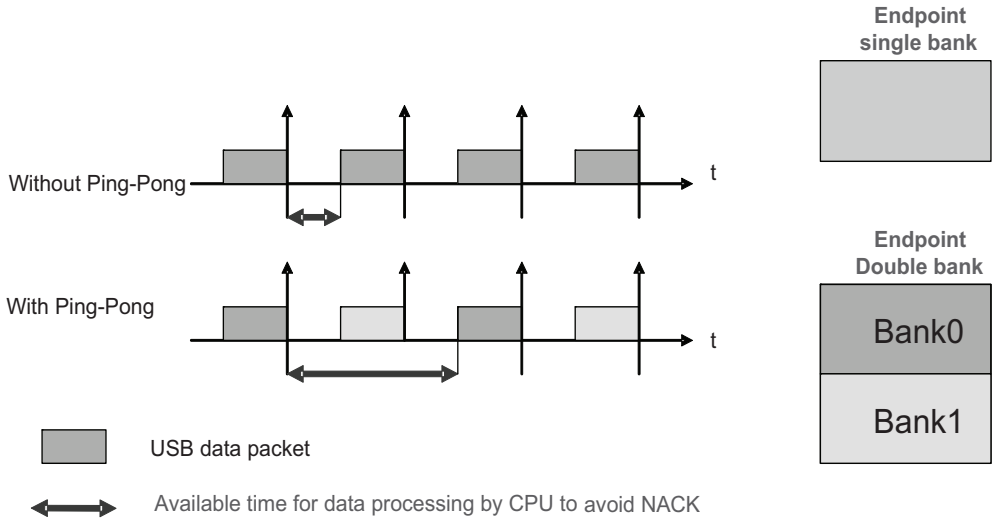
20.6 Ping-pong Operation

When an endpoint is configured for ping-pong operation, it uses the input and output data buffers to create a single, double-buffered endpoint that can be set to input or output direction. This provides double-buffered communication, as the CPU or DMA controller can access one of the buffers, while the other buffer is processing an ongoing transfer. Ping-pong operation is identical to the IN and OUT transactions described above, unless otherwise noted in this section. Ping-pong operation is not possible for control endpoints.

When ping-pong operation is enabled for an endpoint, the endpoint in the opposite direction must be disabled. The data buffer, data pointer, byte counter, and auxiliary data from the enabled endpoint are used as bank 0, and, correspondingly, bank 1 for the opposite endpoint direction.

The bank select (BANK) flag in the endpoint STATUS register indicates which data bank will be used in the next transaction. It is updated after each transaction. The TRNCOMPL0/TRNCOMPL1, underflow/overflow (UDF/OVF), and CRC flags in the STATUS register are set for either the enabled or the opposite endpoint direction according to the BANK flag. The data toggle (TOGGLE), data buffer 0/1 not acknowledge (BUSNACK0 and BUSNACK1), and BANK flags are updated for the enabled endpoint direction only.

Figure 20-8. Ping-pong operation overview.

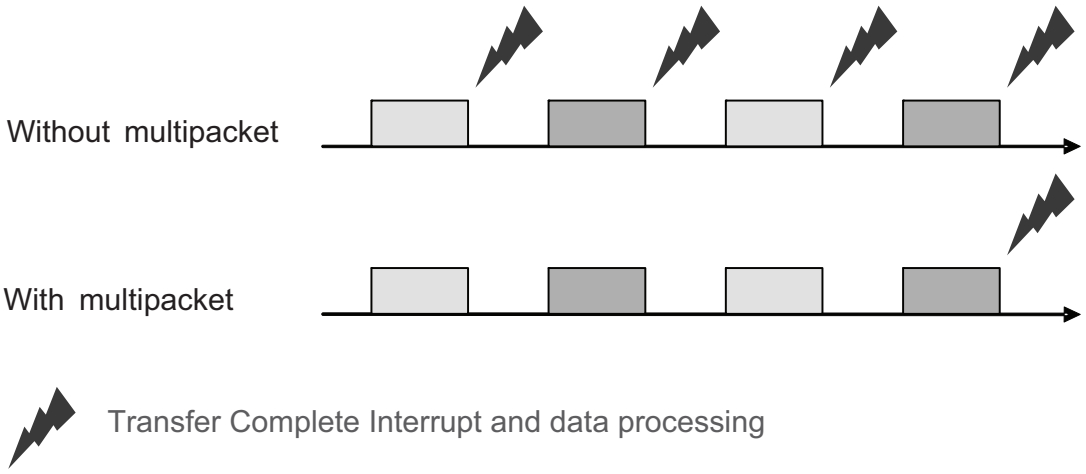


20.7 Multipacket Transfers

Multipacket transfer enables a data payload exceeding the maximum data payload size of an endpoint to be transferred as multiple packets without any software intervention. This reduces interrupts and software intervention to the higher level USB transfer, and frees up significant CPU time. Multipacket transfer is identical to the IN and OUT transactions described above, unless otherwise noted in this section.

The application software provides the size and address of the SRAM buffer to be processed by the USB module for a specific endpoint, and the USB module will then split the buffer in the required USB data transfer.

Figure 20-9. Multipacket overview.



20.7.1 For Input Endpoints

The total number of data bytes to be sent is written to CNT, as for normal operation. The auxiliary data register (AUXDATA) is used to store the number of bytes that will be sent, and must be written to zero for a new transfer.

When an IN token is received, the endpoint's CNT and AUXDATA are fetched. If CNT minus AUXDATA is less than the endpoint SIZE, endpoint CNT minus endpoint AUXDATA number bytes are transmitted; otherwise, SIZE number of bytes are transmitted. If endpoint CNT is a multiple of SIZE and auto zero length packet (AZLP) is enabled, the last packet sent will be zero length.

If a maximum payload size packet was sent (i.e., not the last transaction), AUXDATA is incremented by SIZE. TOGGLE will be toggled after the transaction has completed if the endpoint is not isochronous. If a short packet was sent (i.e., the last transaction), AUXDATA is incremented by the data payload. TOGGLE will be toggled if the endpoint is not isochronous, and BUSNACK, TRNIF, and TRNCOMPL0 will be set.

20.7.2 For Output Endpoints

The number of data bytes received is stored in the endpoint's CNT register, as for normal operation. Since the endpoint's CNT is updated after each transaction, it must be set to zero when setting up a new transfer. The total number of bytes to be received must be written to AUXDATA. This value must be a multiple of SIZE, except for ISO 1023 bytes endpoints; otherwise, excess data may be written to SRAM locations used by other parts of the application.

TOGGLE management is as for non-isochronous packets, and BUSNACK0/BUSNACK1 management is as for normal operation.

If a maximum payload size packet is received, CNT is incremented by SIZE after the transaction has completed, and TOGGLE toggles if the endpoint is not isochronous. If the updated endpoint CNT is equal to AUXDATA, then BUSNACK0/BUSNACK1, TRNIF, and TRNCOMPL0/TRNCOMPL1 will be set.

If a short or oversized packet is received, the endpoint's CNT register will be incremented by the data payload after the transaction has completed. TOGGLE will be toggled if the endpoint is not isochronous, and BUSNACK0/BUSNACK1, TRNIF, and TRNCOMPL0/TRNCOMPL1 will be set.

20.8 Auto Zero Length Packet

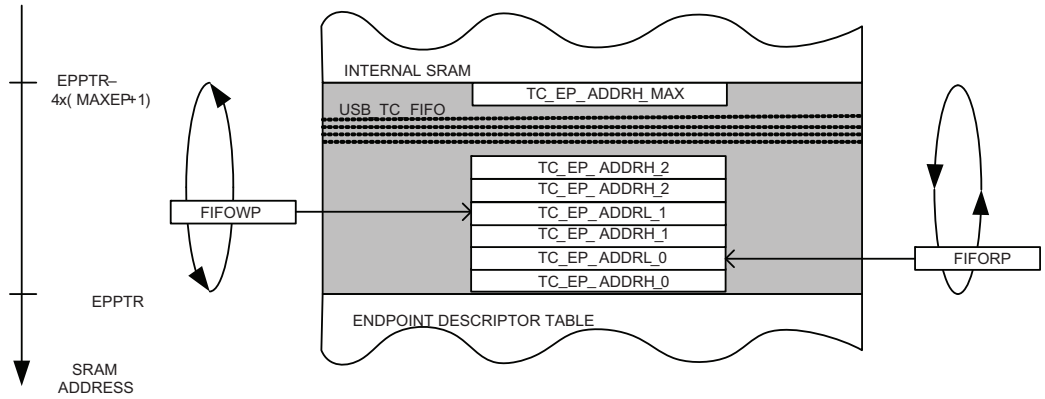
Some IN transfer requires a zero length packet to be generated in order to signal end of transfer to the host. The auto zero length packet (AZLP) function can be enabled to perform this generation automatically, thus removing the need for application software or CPU intervention to perform this task.

20.9 Transaction Complete FIFO

The transaction complete FIFO provides a convenient way to keep track of the endpoints that have completed IN or OUT transactions and need firmware intervention. It creates a first-come, first-served work queue for the application software.

The FIFO size is (MAXEP[3:0] + 1) × 4 bytes, and grows downward, starting from EPPTR - 1. This SRAM memory is allocated only when the FIFO is enabled.

Figure 20-10.Transfer complete FIFO.

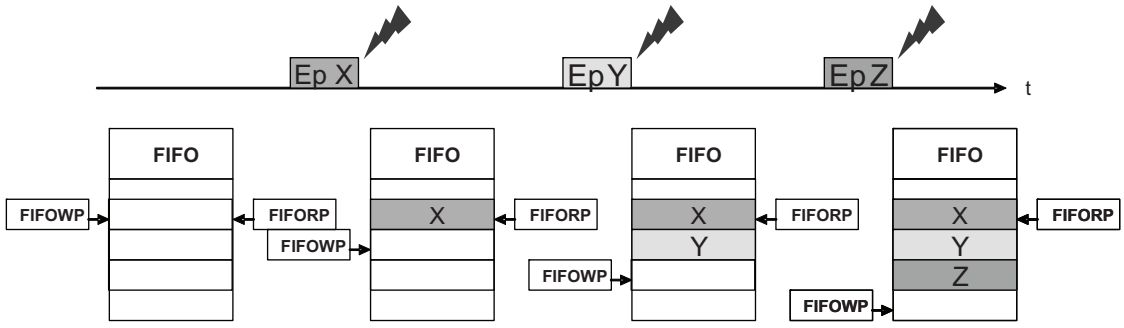


To manage the FIFO, a five-bit write pointer (FIFOWP) and five-bit read pointer (FIFORP) are used by the USB module and application software, respectively. FIFORP and FIFOWP are one's complemented, and thus hold negative values. The SRAM location of the data is the sum of EPPTR and the read or write pointer. The number of items in the FIFO is the difference between FIFOWP and FIFORP. For the programmer, the FIFORP and FIFOWP values have to be cast to a signed 8-bit integer, and then the offset into the FIFO from this signed integer must be deducted.

The transaction complete interrupt flag (TRNIF) in the INFLAGSB[CLR,SET] register is set to indicate a non-empty FIFO when FIFORP!= FIFOWP, cleared when they are equal, and also set when the FIFO is full.

Each time an endpoint IN or OUT transaction completes successfully, its endpoint configuration table address is stored in the FIFO at the current write pointer position (i.e., EPPTR + 2 × FIFOWP) and FIFOWP is decremented. When the pointer reaches the FIFO size, it wraps to zero. When application software reads FIFORP, this is decremented in the same way. Reading the write pointer has no effect. The endpoint configuration table address can then be read directly from (EPPTR + 2 × FIFORP).

Figure 20-11.USB transaction complete FIFO example.

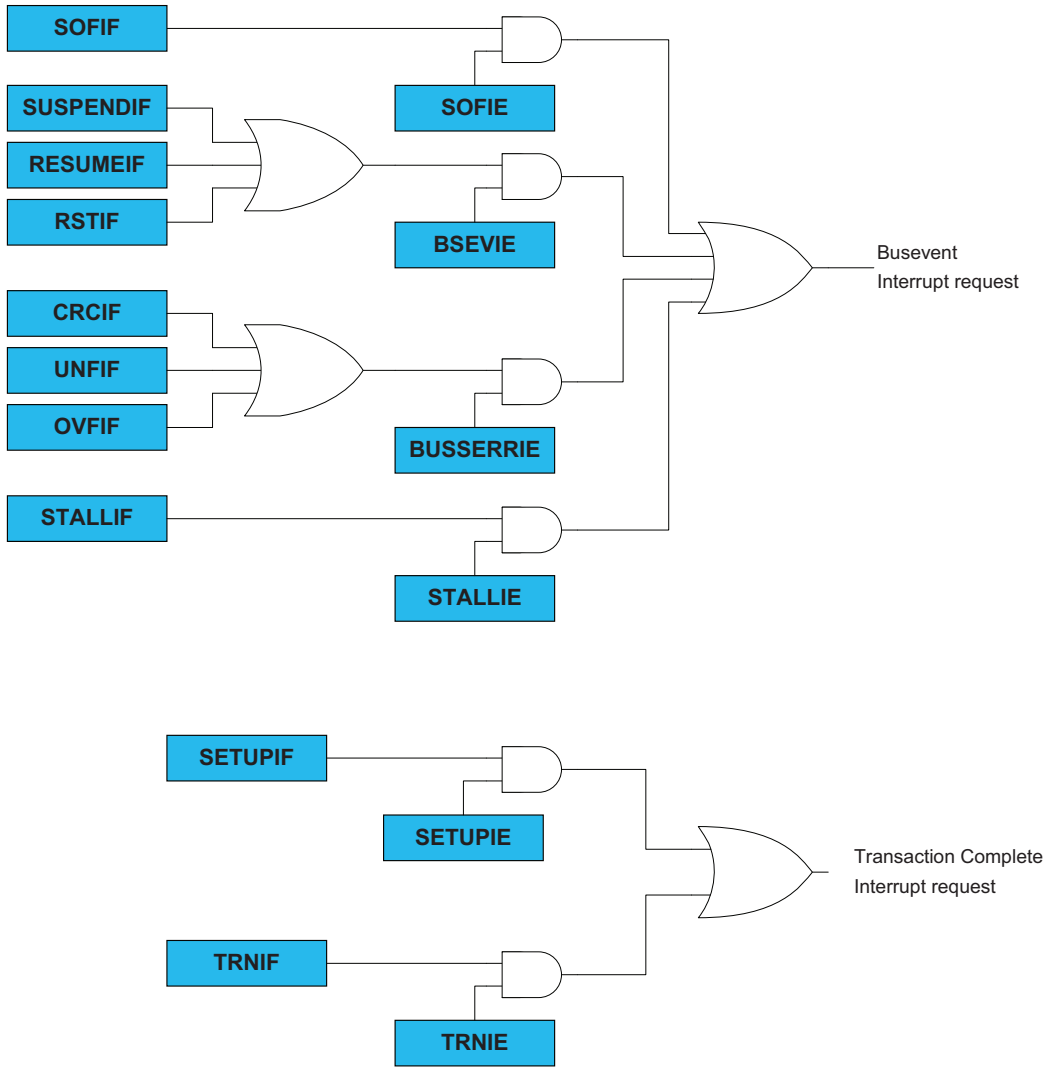


20.10 Interrupts and Events

The USB module can generate interrupts and events. The module has 10 interrupt sources. These are split between two interrupt vectors, the transaction complete (TRNCOMPL) interrupt and the bus event (BUSEVENT) interrupt. An interrupt group is enabled by setting its interrupt level (INTLVL), while different interrupt sources are enabled individually or in groups.

Figure 20-12 on page 235 summarizes the interrupts and event sources for the USB module, and shows how they are enabled.

Figure 20-12.Interrupts and events scheme summary.



20.10.1 Transaction Complete Interrupt

The transaction complete interrupt is generated per endpoint. When an interrupt occurs, the associated endpoint number is registered and optionally added to the FIFO. The following two interrupt sources use the interrupt vector:

Table 20-1. Transaction complete interrupt sources.

Interrupt source	Description
Transfer complete (TRNIF)	An IN or OUT transaction is completed
Setup complete (SETUPIF)	A SETUP transaction is completed

20.10.2 Bus Event Interrupt

The bus event (BUSEVENT) interrupt is used for all interrupts that signal various types of USB line events or error conditions. These interrupts are related to the USB lines, and are generated for the USB module and per endpoint. The following eight interrupts use the interrupt vector:

Table 20-2. Bus event interrupt source.

Interrupt source	Description
Start of frame (SOFIF)	A SOF token has been received
Suspend (SUSPENDIF)	The bus has been idle for 3ms
Resume (RESUMEIF)	A non-idle state is detected when the bus is suspended. The interrupt is asynchronous and can wake the device from all sleep modes
Reset (RSTIF)	A reset condition has been detected on the bus
Isochronous CRC error (CRCIF)	A CRC or bit-stuff error has been detected in an incoming packet to an isochronous endpoint
Underflow (UNFIF)	An endpoint is unable to return data to the host
Overflow (OVFIF)	An endpoint is unable to accept data from the host
STALL (STALLIF)	A STALL handshake has been returned to the host

20.10.3 Events

The USB module can generate several events, and these are available to the event system, allowing latency-free signaling to other peripherals or performance analysis of USB operation.

Table 20-3. Event sources.

Event source	Description
SETUP	SETUPIF
Start of Frame	SOFIF
CRC error	CRCIF
Underflow/overflow	UNFIF and OVFIF

20.11 VBUS Detection

Atmel AVR XMEGA devices can use any general purpose I/O pin to implement a VBUS detection function, and do not use a dedicated VBUS detect pin.

20.12 On-chip Debug

When a break point is reached during on-chip debug (OCD) sessions, the CPU clock can be below 12MHz. If this happens, the USB module will behave as follows:

USB OCD break mode disabled: The USB module immediately acknowledges any OCD break request. The USB module will not be able to follow up on transactions received from the USB host, and its behaviour from the host point of view is not predictable.

USB OCD break mode enabled: The USB module will immediately acknowledge any OCD break request only if there are no ongoing USB transactions. If there is an ongoing USB transaction, the USB module will acknowledge any OCD break request only when the ongoing USB transaction has been completed. The USB module will NACK any further transactions received from the USB host, whether they are SETUP, IN (ISO, BULK), or OUT (ISO, BULK).

20.13 Operating voltage

In order for the USB buffers to operate correctly and be within USB specifications, the operating voltage of the device must be in the range 2.8 - 3.6 Volts.

20.14 Register Description – USB

20.14.1 CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	ENABLE	SPEED	FIFOEN	STFRNUM	MAXEP[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – ENABLE: USB Enable**
  Setting this bit enables the USB interface. Clearing this bit disables the USB interface and immediately aborts any ongoing transactions.
- **Bit 6 – SPEED: Speed Select**
  This bit selects between low and full speed operation. By default, this bit is zero, and low speed operation is selected. Setting this bit enables full speed operation.
- **Bit 5 – FIFOEN: USB FIFO Enable**
  Setting this bit enables the USB transaction complete FIFO, and the FIFO stores the endpoint configuration table address of each endpoint that generates a transaction complete interrupt. Clearing this bit disables the FIFO and frees the allocated SRAM memory.
- **Bit 4 – STFRNUM: Store Frame Number Enable**
  Setting this bit enables storing of the last SOF token frame number in the frame number (FRAMENUM) register. Clearing this bit disables the function.
- **Bit 3:0 – MAXEP[3:0]: Maximum Endpoint Address**
  These bits select the number of endpoint addresses used by the USB module. Incoming packets with a higher endpoint number than this address will be discarded. Packets with endpoint addresses lower than or equal to this address will cause the USB module to look up the addressed endpoint in the endpoint configuration table.

20.14.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	PULLRST	–	RWAKEUP	GNACK	ATTACH
Read/Write	R	R	R	R/W	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4 – PULLRST: Pull during Reset**
  Setting this bit enables the pull-up on the USB lines to also be held when the device enters reset. The bit will be cleared on a power-on reset.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2 – RWAKEUP: Remote Wake-up**

Setting this bit sends an upstream resume on the USB lines if the bus is in the suspend state for at least 5ms.

- **Bit 1 – GNACK: Global NACK**
  When this bit is set, the USB module will NACK all incoming transactions. Expect for a SETUP packet, this prevents the USB module from performing any on-chip SRAM access, giving all SRAM bandwidth to the CPU and/or DMA controller.

- **Bit 0 – ATTACH: Attach**
  Setting this bit enables the internal D+ or D- pull-up (depending on the USB speed selection), and attaches the device to the USB lines. Clearing this bit disconnects the device from the USB lines.

20.14.3 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	**URESUME**	**RESUME**	**SUSPEND**	**BUSRST**
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 3 – URESUME: Upstream Resume**
  This flag is set when an upstream resume is sent.

- **Bit 2 – RESUME: Resume**
  This flag is set when a downstream resume is received.

- **Bit 1 – SUSPEND: Bus Suspended**
  This flag is set when the USB lines are in the suspended state (the bus has been idle for at least 3ms).

- **Bit 0 – BUSRST: Bus Reset**
  This flag is set when a reset condition has been detected (the bus has been driven to SE0 for at least 2.5µs).

20.14.4 ADDR – Address register

Bit	7	6	5	4	3	2	1	0
+0x03	–	**ADDR[6:0]**						
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

- **Bit 6:0 – ADDR[6:0]: Device Address**
  These bits contain the USB address the device will respond to.

20.14.5 FIFOWP – FIFO Write Pointer register

When the FIFO is enabled:

The TCIF interrupt flag is cleared:

- by writing to FIFORP or FIFOWP any value.

- by reading one or several times to FIFORP depending on the size of the fifo

When the FIFO is disabled:

The TCIF interrupt flag is cleared:

by writing to FIFORP or FIFOWP any value.

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	FIFOWP[4:0]				
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4:0 – FIFOWP[4:0]: FIFO Write Pointer**
  These bits contain the transaction complete FIFO write pointer. This register must be read only by the CPU or DMA controller. Writing this register will flush the FIFO write and read pointers.

20.14.6 FIFORP – FIFO Read Pointer register

When the FIFO is enabled:

The TCIF interrupt flag is cleared:

- by writing to FIFORP or FIFOWP any value.
- by reading one or several times to FIFORP depending on the size of the fifo

When the FIFO is disabled:

The TCIF interrupt flag is cleared:

by writing to FIFORP or FIFOWP any value.

Bit	7	6	5	4	3	2	1	0
+0x05	–	–	–	FIFORP[4:0]				
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4:0 – FIFORP[4:0]: FIFO Read Pointer**
  These bits contain the transaction complete FIFO read pointer. This register must only be read by the CPU or DMA controller. Writing this register will flush the FIFO write and read pointer.

20.14.7 EPPTRL – Endpoint Configuration Table Pointer Low

The EPPTRL and EPPTRH registers represent the 16-bit value, EPPTR, that contains the address to the endpoint configuration table. The pointer to the endpoint configuration table must be aligned to a 16-bit word; i.e., EPPTR[0] must be zero. Only the number of bits required to address the available internal SRAM memory is implemented for each device. Unused bits will always be read as zero.

Bit	7	6	5	4	3	2	1	0
+0x06	EPPTR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – EPPTR[7:0]: Endpoint Configuration Table Pointer**

This register contains the eight lsbs of the endpoint configuration table pointer (EPPTR).

20.14.8 EPPTRH – Endpoint Configuration Table Pointer High

Bit	7	6	5	4	3	2	1	0
+0x07	EPPTR[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – EPPTR[15:8]: Endpoint Configuration Table Pointer**
  This register contains the eight msbs of the endpoint configuration table pointer (EPPTR).

20.14.9 INTCTRLA – Interrupt Control register A

Bit	7	6	5	4	3	2	1	0
+0x06	SOFIE	BUSEVIE	BUSERRIE	STALLIE	–	–	INTLVL[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – SOFIE: Start Of Frame Interrupt Enable**

  Setting this bit enables the start of frame (SOF) interrupt for the conditions that set the start of frame interrupt flag (SOFIF) in the INTFLAGSACLR/ INTFLAGSASET register. The INTLVL bits must be nonzero for the interrupts to be generated.

- **Bit 6 – BUSEVIE: Bus Event Interrupt Enable**

Setting this bit will enable the interrupt for the following three bus events:

1. *Suspend*: An interrupt will be generated for the conditions that set the suspend interrupt flag (SUSPENDIF) in the INTFLAGSACLR/SET register.
2. *Resume*: An interrupt will be generated for the conditions that set the resume interrupt flag (RESUMEIF) in the INTFLAGSACLR/SET register.
3. *Reset*: An interrupt will be generated for the conditions that set the reset interrupt flag (RESETIF) in the INTFLAGSACLR/SET register.

The INTLVL bits must be nonzero for the interrupts to be generated.

- **Bit 5 – BUSERRIE: Bus Error Interrupt Enable**

Setting this bit will enable the interrupt for the following three bus error events:

1. *Isochronous CRC Error:* An interrupt will be generated for the conditions that set the CRC interrupt flag (CRCIF) in the INTFLAGSACLR/SET register during isochronous transfers.
2. *Underflow*: An interrupt will be generated for the conditions that set the underflow interrupt flag (UNFIF) in the INTFLAGSACLR/SET register.
3. *Overflow*: An interrupt will be generated for the conditions that set the overflow interrupt flag (OVFIF) in the INTFLAGSACLR/SET register.

The INTLVL bits must be nonzero for the interrupts to be generated.

- **Bit 4 – STALLIE: STALL Interrupt Enable**
  Setting this bit enables the STALL interrupt for the conditions that set the stall interrupt flag (STALLIF) in the INTFLAGSACLR/SET register. The INTLVL bits must be nonzero for the interrupts to be generated.

- **Bit 3:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1:0 – INTLVL[1:0]: Interrupt Level**
  These bits enable the USB interrupts and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 131. In addition, each USB interrupt source must be separately enabled.

20.14.10 INTCTRLB – Interrupt Control register B

Bit	7	6	5	4	3	2	1	0
+0x07	–	–	–	–	–	–	**TRNIE**	**SETUPIE**
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1 – TRNIE: Transaction Complete Interrupt Enable**
  Setting this bit enables the transaction complete interrupt for IN and OUT transactions. The INTLVL bits must be nonzero for interrupts to be generated.

- **Bit 0 – SETUPIE: SETUP Transaction Complete Interrupt Enable**
  Setting this bit enables the SETUP Transaction Complete Interrupt for SETUP transactions. The INTLVL bits must be non-zero for the interrupts to be generated.

20.14.11 INTFLAGSACLR/ INTFLAGSASET – Clear/ Set Interrupt Flag register A

This register is mapped into two I/O memory locations, one for clearing (INTFLAGSACLR) and one for setting (INTFLAGSASET) the flags. The individual flags can be set by writing a one to their bit locations in INFLAGSASET, and cleared by writing a one to their bit locations in INT-FLAGSACLR. Both memory locations will provide the same result when read, and writing zero to any bit location has no effect.

Bit	7	6	5	4	3	2	1	0
+0x0A/ +0x0B	**SOFIF**	**SUSPENDIF**	**RESUMEIF**	**RESETIF**	**CRCIF**	**UNFIF**	**OVFIF**	**STALLIF**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – SOFIF: Start Of Frame Interrupt Flag**
  This flag is set when a start of frame packet has been received.

- **Bit 6 – SUSPENDIF: Suspend Interrupt Flag**
  This flag is set when the bus has been idle for 3ms.

- **Bit 5 – RESUMEIF: Resume Interrupt Flag**
  This flag is set when a non-idle state has been detected on the bus while the USB module is in the suspend state. This interrupt is asynchronous, and is able to wake the CPU from sleep modes where the system clock is stopped, such as power-down and power-save sleep modes.

- **Bit 4 – RSTIF: Reset Interrupt Flag**
  This flag is set when a reset condition has been detected on the bus.

- **Bit 3 – CRCIF: Isochronous CRC Error Interrupt Flag**
  This flag is set when a CRC error has been detected in an incoming data packet to an isochronous endpoint.

- **Bit 2 – UNFIF: Underflow Interrupt Flag**
  This flag is set when the addressed endpoint in an IN transaction does not have data to send to the host.
- **Bit 1 – OVFIF: Overflow Interrupt Flag**
  This flag is set when the addressed endpoint in an OUT transaction is not ready to accept data from the host.
- **Bit 0 – STALLIF: STALL Interrupt Flag**
  This flag is set when the USB module has responded with a STALL handshake to either an IN or an OUT transaction.

20.14.12 INTFLAGSBCLR/INTFLAGSBSET – Clear/Set Interrupt Flag register B

This register is mapped into two I/O memory locations, one for clearing (INTFLAGSBCLR) and one for setting (INTFLAGSBSET) the flags. The individual flags can be set by writing a one to their bit locations in INFLAGSBSET, and cleared by writing a one to their bit locations in INTFLAGSBCLR. Both memory locations will provide the same result when read, and writing zero to any bit location has no effect.

When the FIFO is enabled:

The TCIF interrupt flag is cleared:
- by writing to FIFORP or FIFOWP any value.
- by reading one or several times to FIFORP depending on the size of the fifo

When the FIFO is disabled:

The TCIF interrupt flag is cleared:
- by writing to FIFORP or FIFOWP any value.

Bit	7	6	5	4	3	2	1	0
+0x0C/ +0x0D	–	–	–	–	–	–	**TRNIF**	**SETUPIF**
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1 – TRNIF: Transaction Complete Interrupt Flag**
  This flag is when there is a pending packet interrupt in the FIFO.
- **Bit 0 – SETUPIF: SETUP Transaction Complete Interrupt Flag**
  This flag is set when a SETUP transaction has completed successfully.

20.14.13 CAL0 – Calibration Low

CALL and CALH hold the 16-bit value, CAL. The USB PADs (D- and D+) are calibrated during production to enable operation without requiring external components on the USB lines. The calibration value is stored in the signature row of the device, and must be read from there and written to the CAL registers from software.

Bit	7	6	5	4	3	2	1	0
++0x3A	**CAL[7:0]**							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CAL[7:0]: PAD Calibration Low**

This byte holds the eight lsbs of CAL.

20.14.14 CAL1 – Calibration High

Bit	7	6	5	4	3	2	1	0
+0x3B	CAL[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CAL[15:8]: PAD Calibration High**
  This byte holds the eight msbs of CAL.

20.15 Register Description – USB Endpoint

Each of the 16 endpoint addresses have one input and one output endpoint. Each endpoint has eight bytes of configuration/status data located in internal SRAM.

The address to the first configuration byte is (EPPTR[15:0] + 16 × endpoint address) for output endpoints and (EPPTR[15:0] + 16 × endpoint address + 8) for input endpoints.

Some bit locations have different functions, depending on endpoint configuration type or direction, and this is reflected by using two different names for the bit locations.

20.15.1 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x00	STALL CRC[1]	UNF/ OVF	TRNCOMPL0	SETUP TRNCOMPL1	BANK	BUSNACK1	BUSNACK0	TOGGLE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Note: 1. For isochronous endpoints.

- **Bit 7 – STALL: STALL Flag**
  This flag is set when an IN or OUT transaction has been responded to with a STALL handshake. This flag is cleared by writing a one to its bit location.

- **Bit 7 – CRC: CRC Error Flag**
  This flag is set for isochronous output endpoints when a CRC error has been detected in an incoming data packet. This flag is cleared by writing a one to its bit location.

- **Bit 6 – UNF/OVF: Underflow/Overflow Flag**
  UNF: For input endpoints, the UNF flag is set when an input endpoint is not ready to send data to the host in response of an IN token.
  OVF: For output endpoints, the OVF flag is set when an output endpoint is not ready to accept data from the host following an OUT token.

- **Bit 5 – TRNCOMPL0: Transaction Complete Flag**
  This flag is set when an IN or OUT transaction has completed successfully. This flag is cleared by writing logical 0 to its bit location.

- **Bit 4 – SETUP: SETUP Transaction Complete Flag**
  This flag is set when a SETUP, IN, or OUT transaction has completed successfully. This flag is cleared by writing logical 0 to its bit location.

- **Bit 4 – TRNCOMPL1: Transaction Complete Flag**

This flag is set when a SETUP, IN, or OUT transaction has completed successfully. This flag is cleared by writing logical 0 to its bit location.

- **Bit 3 – BANK: Bank Select Flag**
  When ping-pong mode is enabled, this bit indicates which bank will be used for the next transaction. BANK is toggled each time a transaction has completed successfully. This bit is not sed when ping-pong is disabled. This flag is cleared by writing a one to its bit location.

- **Bit 2 – BUSNACK1: Data Buffer 1 Not Acknowledge Flag**
  When this flag is set, the USB module will discard incoming data to data buffer 1 in an OUT transaction, and will not return any data from data buffer 1 in an IN transaction. For control, bulk, and interrupt endpoints, a NAK handshake is returned. This flag is cleared by writing a one to its bit location.

- **Bit 1 – BUSNACK0: Data Buffer 0 Not Acknowledge Flag**
  When this flag is set, the USB module will discard incoming data to data buffer 0 in an OUT transaction, and will not return any data from data buffer 0 in an IN transaction. For control, bulk, and interrupt endpoints, a NAK handshake is returned. This flag is cleared by writing logical 0 to its bit location.

- **Bit 0 – TOGGLE: Data Toggle Flag**
  This indicates if a DATA0 or DATA1 PID is expected in the next data packet for an output endpoint, and if a DATA0 or DATA1 PID will be sent in the next transaction for an input endpoint. This bit has no effect for isochronous endpoints, where both DATA0 and DATA1 PIDs are accepted for output endpoint, and only DATA0 PIDs are sent for input endpoints.

20.15.2 CTRL – Control

Bit	7	6	5	4	3	2	1	0
+0x01	TYPE[1:0]		MULTIPKT	PINGPONG	INTDSBL	STALL	SIZE[1:0]	
						SIZE[2:0](1)		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Note: 1. For isochronous endpoints.

- **Bit 7:6 – TYPE[1:0]: Endpoint Type**
  These bits are used to enable and select the endpoint type. If the endpoint is disabled, the remaining seven endpoint configuration bytes are never read or written by the USB module, and their SRAM locations are free to use for other application data.

Table 20-4. Endpoint type.

TYPE[1:0]	Group configuration	Description
00	DISABLE	Endpoint enabled
01	CONTROL	Control
10	BULK	Bulk/interrupt
11	ISOCHRONOUS	Isochronous

- **Bit 5 – MULTIPKT: Multipacket Transfer Enable**
  Setting this bit enables multipacket transfers. Multipacket transfer enables a data payload exceeding the maximum packet size of an endpoint to be transferred as multiple packets without interrupts or software intervention. See “Multipacket Transfers” on page 233 for details on multipacket transfers.

- **Bit 4 – PINGPONG: Ping-pong Enable**
  Setting this bit enables ping-pong operation. Ping-pong operation enables both endpoints (IN and OUT) with same address to be used in the same direction to allow double buffering and maximize throughput. The endpoint in the opposite direction must be disabled when ping-pong operation is enabled. Ping-pong operation is not possible for control endpoints. See “Ping-pong Operation” on page 232 for details.

- **Bit 3 – INTDSBL: Interrupt Disable**
  Setting this bit disables all enabled interrupts from the endpoint. Hence, only the interrupt flags in the STATUS register are updated when interrupt conditions occur. The FIFO does not store this endpoint configuration table address upon transaction complete for the endpoint when interrupts are disabled for an endpoint. Clearing this bit enables all previously enables interrupts again.

- **Bit 2 – STALL: Endpoint STALL**
  This bit controls the STALL behavior if the endpoint.

- **Bit 1:0 – BUFSIZE[1:0]: Data Size**
  These bits configure the maximum data payload size for the endpoint. Incoming data bytes exceeding the maximum data payload size are discarded.

- **Bit 2:0 – BUFSIZE[2:0]: Data Size**
  These bits configure the maximum data payload size for the endpoint when configured for isochronous operation.

Table 20-5. BUFSIZE configuration.

BUFSIZE[2:0]	Group configuration	Description
000	8	8-byte buffer size
001	16	16-byte buffer size
010	32	32-byte buffer size
011	64	64-byte buffer size
100(1)	128	128-byte buffer size
101(1)	256	256-byte buffer size
110(1)	512	512-byte buffer size
111(1)	1023	1023-byte buffer size

Note: 1. Setting only available for isochronous endpoints.

20.15.3 CNTL – Counter Low

The CNTL and CNTH registers represent the 10-bit value, CNT, that contains the number of bytes received in the last OUT or SETUP transaction for an OUT endpoint, or the number of bytes to be sent in the next IN transaction for an IN endpoint.

Bit	7	6	5	4	3	2	1	0
+0x02	CNT[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 7:0 – CNT[7:0]: Endpoint Byte Counter**
  This byte contains the eight lsbs of the USB endpoint counter (CNT).

20.15.4 CNTH – Counter High

Bit	7	6	5	4	3	2	1	0
+0x03	AZLP	–	–	–	–	–	CNT[9:8]	
Read/Write	R/W	R	R	R	R	R	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 6 – AZLP: Automatic Zero Length Packet**
  When this bit is set, the USB module will manage the ZLP handshake by hardware. This applies to IN endpoints only. When this bit is zero, the ZLP handshake must be managed by firmware.
- **Bit 6:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 1:0 – CNT[9:8]: Endpoint Byte Counter**
  These bits contain the two msbs of the USB endpoint counter (CNT).

20.15.5 DATAPTRL – Data Pointer register Low

The DATAPTRL and DATAPTRH registers represent the 16-bit value, DATAPTR, that contains the SRAM address to the endpoint data buffer.

Bit	7	6	5	4	3	2	1	0
+0x04	DATAPTR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 7:0 – DATAPTR[7:0]: Endpoint Data Pointer Low**
  This byte contains the eight lsbs of the endpoint data pointer (DATAPTR).

20.15.6 DATAPTRH – Data Pointer register High

Bit	7	6	5	4	3	2	1	0
+0x05	DATAPTR[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 15:0 - DPTR[15:8]: Endpoint Data Pointer High**
  This byte contains the eight msbs of the endpoint data pointer (DATAPTR).

20.15.7 AUXDATAL – Auxiliary Data register Low

The AUXDATAL and AUXDATAH registers represent the 16-bit value, AUXDATA, that is used for multipacket transfers.

For IN endpoints, AUXDATA holds the total number of bytes sent. AUXDATA should be written to zero when setting up a new transfer. For OUT endpoints, AUXDATA holds the total data size for the complete transfer. This value must be a multiple of the maximum packet size, except for ISO 1023-byte endpoints.

See “Multipacket Transfers” on page 233 for more details on setting up and using multipacket transfers.

Bit	7	6	5	4	3	2	1	0
+0x06	AUXDATA[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 7:0 – AUXDATA[7:0]: Auxiliary Data Low**
  This byte contains the eight lsbs of the auxiliary data (AUXDATA). When multipacket transfer is not used, this SRAM location is free to use for other application data.

20.15.8 AUXDATAH – Auxiliary Data register High

Bit	7	6	5	4	3	2	1	0
+0x07	AUXDATA[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

- **Bit 7:0 – AUXDATA[15:8]: Auxiliary Data High**
  This byte contains the eight msbs of the auxiliary data (AUXDATA). When multipacket transfer is not used, this SRAM location is free to use for other application data.

20.16 Register Description – Frame

20.16.1 FRAMENUML – Frame Number register Low

The FRAMENUML and FRAMENUMH registers represent the 11-bit value, FRAMENUM, that holds the frame number from the most recently received start of frame packet.

Bit	7	6	5	4	3	2	1	0
+0x00	FRAMENUM[7:0]							
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – FRAMENUM[7:0]: Frame Number**
  This byte contains the eight lsbs of the frame number (FRAMENUM).

20.16.2 FRAMENUMH – Frame Number register High

Bit	7	6	5	4	3	2	1	0
+0x01	FRAMEERR	–	–	–	–	FRAMENUM[10:8]		
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – FRAMEERR: Frame Error**
  This flag is set if a CRC or bit-stuffing error was detected in the most recently received start of frame packet.
- **Bit 6:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2:0 – FRAMENUM[10:8]: Frame Number**
  This byte contains the three msbs of the frame number (FRAMENUM).

20.17 Register summary – USB module

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	ENABLE	SPEED	FIFOEN	STFRNUM	MAXEP[3:0]				238
+0x01	CTRLB	–	–	–	PULLRST	–	RWAKEUP	GNACK	ATTACH	238
+0x02	STATUS	–	–	–	–	URESUME	RESUME	SUSPEND	BUSRST	239
+0x03	ADDR	–	ADDR[6:0]							239
+0x04	FIFOWP	–	–	–	FIFOWP[4:0]					239
+0x05	FIFORP	–	–	–	FIFORP[4:0]					240
+0x06	EPPTRL	EPPTR[7:0]								240
+0x07	EPPTRH	EPPTR[15:8]								241
+0x08	INTCTRLA	SOFIE	BUSEVIE	BUSERRIE	STALLIE	–	–	INTLVL[1:0]		241
+0x09	INTCTRLB	–	–	–	–	–	–	TRNIE	SETUPIE	242
+0x0A	INFLAGSACLR	SOFIF	SUSPENDIF	RESUMEIF	RSTIF	CRCIF	UNFIF	OVFIF	STALLIF	242
+0x0B	INFLAGSASET	SOFIF	SUSPENDIF	RESUMEIF	RSTIF	CRCIF	UNFIF	OVFIF	STALLIF	242
+0x0C	INFLAGSBCLR	–	–	–	–	–	–	TRNIF	SETUPIF	243
+0x0D	INFLAGSBSET	–	–	–	–	–	–	TRNIF	SETUPIF	243
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10-0X39	Reserved	–	–	–	–	–	–	–	–	
+0x3A	CAL0	CAL[7:0]								243
+0x3B	CAL1	CAL[15:8]								244

20.18 Register summary – USB endpoint

The address to the first configuration byte is (EPPTR[15:0] + 16 × endpoint address) for OUT endpoints and (EPPTR[15:0] + 16 × endpoint address + 8) for IN endpoints.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	STALL	OVF/UNF	TRNCOMPL0	SETUP	BANK	BUSNACK1	BUSNACK0	TOGGLE	244
		CRC			TRNCOMPL1					Isochronous
+0x01	CTRL	TYPE[1:0]		MULTIPKT	PINGPONG	INTDSBL	STALL	BUFSIZE[1:0]		245
							BUFSIZE[2:0]			Isochronous
+0x02	CNTL	CNT[7:0]								246
+0x03	CNTH	AZLP	–	–	–	–	–	CNT[9:8]		247
+0x04	DATAPTRL	DATAPTR[7:0]								247
+0x05	DATAPTRH	DATAPTR[15:8]								247
+0x06	AUXDATAL	AUXDATA[7:0]								247
+0x07	AUXDATAH	AUXDATA[15:8]								248

20.19 Register summary – Frame

The address to the frame configuration byte is (MAXEP + 1) << 4. For instance with MAXEP = 3, the first address would be located at offset address 0x40.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	FRAMENUML	FRAMENUM[7:0]								248
+0x01	FRAMENUMH	FRAMEERR	–	–	–	–	FRAMENUM[10:8]			248

20.20 USB Interrupt vector summary

Table 20-6. USB interrupt vectors and their word offset addresses.

Offset	Source	Interrupt Description
0x00	BUSEVENT_vect	SOF, suspend, resume, bus reset, CRC, underflow, overflow, and stall error interrupts
0x02	TRNCOMPL_vect	Transaction complete interrupt