USB 2.0 OTG Controller 18

2014.08.18

a10_54018

The hard processor system (HPS) provides two instances of a USB On-The-Go (OTG) controller that supports both device and host functions. The controller supports all high-speed, full-speed, and low-speed transfers in both device and host modes. The controller is fully compliant with the *On The Go and Embedded Host Supplement to the USB Revision 2.0 Specification*. The controller can be programmed for both device and host functions to support data movement over the USB protocol.

The controllers are operationally independent of each other. Each USB OTG controller supports a single USB port connected through a USB 2.0 Transceiver Macrocell Interface Plus (UTMI+) Low Pin Interface (ULPI) compliant PHY. The USB OTG controllers are instances of the Synopsys® (†)DesignWare® Cores USB 2.0 Hi-Speed On-The-Go (DWC_otg) controller.

The USB OTG controller is optimized for the following applications and systems: †

- Portable electronic devices †
- Point-to-point applications (no hub, direct connection to HS, FS, or LS device) †
- Multi-point applications (as an embedded USB host) to devices (hub and split support) †

Each of the two USB OTG ports supports both host and device modes, as described in the *On The Go and Embedded Host Supplement to the USB Revision 2.0 Specification*. The USB OTG ports support connections for all types of USB peripherals, including the following peripherals:

- Mouse
- Keyboard
- Digital cameras
- Network adapters
- Hard drives
- Generic hubs

(†) Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non infringement, and any warranties arising out of a course of dealing or usage of trade.

† Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

Features of the USB OTG Controller

The USB OTG controller has the following USB-specific features:

- Complies with both Revision 1.3 and Revision 2.0 of the *On The Go and Embedded Host Supplement to the USB Revision 2.0 Specification*
- Supports software-configurable modes of operation between OTG 1.3 and OTG 2.0
- Supports all USB 2.0 speeds:
  - High speed (HS, 480-Mbps)
  - Full speed (FS, 12-Mbps)
  - Low speed (LS, 1.5-Mbps)

  Note: In host mode, all speeds are supported. However, in device mode, only high speed and full speed are supported.

- Supports USB 2.0 in ULPI mode
- Supports all USB transaction types:
  - Control transfers
  - Bulk transfers
  - Isochronous transfers
  - Interrupts
- Supports automatic ping capability
- Supports Session Request Protocol (SRP) and Host Negotiation Protocol (HNP)
- Supports suspend, resume, and remote wake
- Supports up to 16 host channels

  Note: In host mode, when the number of device endpoints is greater than the number of host channels, software can reprogram the channels to support up to 127 devices, each having 32 endpoints (IN + OUT), for a maximum of 4,064 endpoints.

- Supports up to 16 bidirectional endpoints, including control endpoint 0

  Note: Only seven periodic device IN endpoints are supported.

- Supports a generic root hub
- Performs transaction scheduling in hardware

On the USB PHY layer, the USB OTG controller supports the following features:

- A single USB port connected to each OTG instance
- A ULPI connection to an off-chip USB transceiver
- Software-controlled access, supporting vendor-specific or optional PHY registers access to ease debug
- The OTG 2.0 support for Attach Detection Protocol (ADP) only through an external (off-chip) ADP controller

On the integration side, the USB OTG controller supports the following features:

- Different clocks for system and PHY interfaces

- Dedicated TX FIFO buffer for each device IN endpoint in direct memory access (DMA) mode
- Packet-based, dynamic FIFO memory allocation for endpoints for small FIFO buffers and flexible, efficient use of RAM that can be dynamically sized by software
- Ability to change an endpoint's FIFO memory size during transfers
- Clock gating support during USB suspend and session-off modes
  - PHY clock gating support
  - System clock gating support
- Data FIFO RAM clock gating support
- Local buffering with error correction code (ECC) support

Note: The USB OTG controller does not support the following protocols:

- Enhanced Host Controller Interface (EHCI)
- Open Host Controller Interface (OHCI)
- Universal Host Controller Interface (UHCI)

Supported PHYS

The USB OTG controller only supports USB 2.0 ULPI PHYs. Only the single data rate (SDR) mode is supported.

Refer to the *Device Datasheet* for timing characteristics on the USB PHY interface.

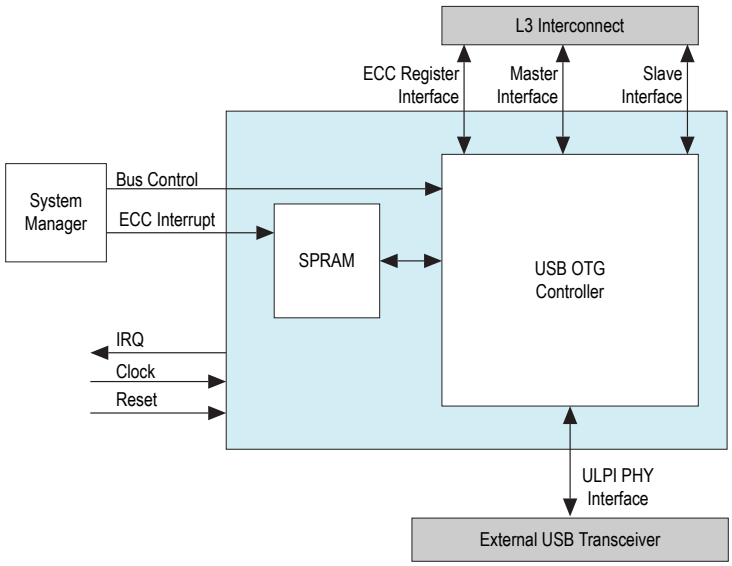
Related Information

Arria 10 Device Datasheet

USB OTG Controller Block Diagram and System Integration

Figure 18-1: USB OTG Controller System Integration

Two subsystems are included in the HPS.



The USB OTG controller connects to the level 3 (L3) interconnect through a slave interface, allowing other masters to access the control and status registers (CSRs) in the controller. The controller also connects to the L3 interconnect through a master interface, allowing the DMA engine in the controller to move data between external memory and the controller.

A single-port RAM (SPRAM) connected to the USB OTG controller is used to store USB data packets for both host and device modes. It is configured as FIFO buffers for receive and transmit data packets on the USB link.

Through the system manager, the USB OTG controller has control to use and test error correction codes (ECCs) in the SPRAM. Through the system manager, the USB OTG controller can also control the behavior of the master interface to the L3 interconnect.

The USB OTG controller connects to the external USB transceiver through a ULPI PHY interface. This interface also connects through pin multiplexers within the HPS. The pin multiplexers are controlled by the system manager.

Additional connections on the USB OTG controller include:

- Clock input from the clock manager to the USB OTG controller
- Reset input from the reset manager to the USB OTG controller
- Interrupt line from the USB OTG controller to the microprocessor unit (MPU) global interrupt controller (GIC).

The USB controller will only use Direct Shared IO 48.

Related Information

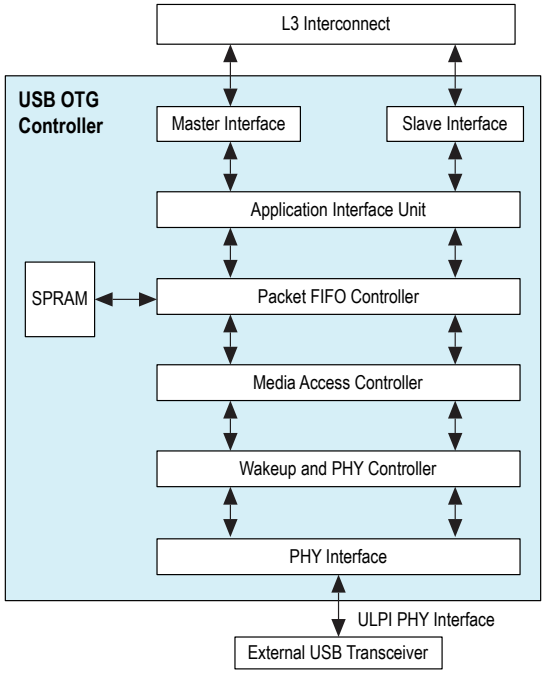
- **System Manager**
  For more information, refer to the *System Manager* chapter in the *Hard Processor System Technical Reference Manual*.
- **General Purpose I/O Interface**
  For more information, refer to the *General Purpose I/O Interface* chapter in the *Hard Processor System Technical Reference Manual*.

Functional Description of the USB OTG Controller

USB OTG Controller Block Description

Figure 18-2: USB OTG Controller Block Description

Details about each of the units that comprise the USB OTG controller are shown below.



Master Interface

The master interface includes a built-in DMA controller. The DMA controller moves data between external memory and the media access controller (MAC).

Properties of the master interface are controlled through the USB L3 Master HPROT Register (`l3master`) in the system manager. These bits provide access information to the L3 interconnect, including whether or not transactions are cacheable, bufferable, or privileged.

Note: Bits in the `l3master` register can be updated only when the master interface is guaranteed to be in an inactive state.

Slave Interface

The slave interface allows other masters in the system to access the USB OTG controller's CSRs. For testing purposes, other masters can also access the SPRAM.

Slave Interface CSR Unit

The slave interface can read from and write to all the CSRs in the USB OTG controllers. All register accesses are 32 bits.

The CSR is divided into the following groups of registers:

- Global
- Host
- Device
- Power and clock gating

Some registers are shared between host and device modes, because the controller can only be in one mode at a time. The controller generates a mode mismatch interrupt if a master attempts to access device registers when the controller is in host mode, or attempts to access host registers when the controller is in device mode. Writing to unimplemented registers is ignored. Reading from unimplemented registers returns indeterminate values.

Application Interface Unit

The application interface unit (AIU) generates DMA requests based on programmable FIFO buffer thresholds. The AIU generates interrupts to the GIC for both host and device modes. A DMA scheduler is included in the AIU to arbitrate and control the data transfer between packets in system memory and their respective USB endpoints.

Packet FIFO Controller

The Packet FIFO Controller (PFC) connects the AIU with the MAC through data FIFO buffers located in the SPRAM. In device mode, one FIFO buffer is implemented for each IN endpoint. In host mode, a single FIFO buffer stores data for all periodic (isochronous and interrupt) OUT endpoints, and a single FIFO buffer is used for nonperiodic (control and bulk) OUT endpoints. Host and device mode share a single receive data FIFO buffer.

SPRAM

An SPRAM implements the data FIFO buffers for host and device modes. The size of the FIFO buffers can be programmed dynamically.

The SPRAM supports ECCs.

MAC

The MAC module implements the following functionality:

- USB transaction support
- Host protocol support
- Device protocol support
- OTG protocol support
- Link power management (LPM) functions

USB Transactions

In device mode, the MAC decodes and checks the integrity of all token packets. For valid OUT or SETUP tokens, the following DATA packet is also checked. If the data packet is valid, the MAC performs the following steps:

1. Writes the data to the receive FIFO buffer
2. Sends the appropriate handshake when required to the USB host

If a receive FIFO buffer is not available, the MAC sends a NAK response to the host. The MAC also supports ping protocol.

For IN tokens, if data is available in the transmit FIFO buffer, the MAC performs the following steps:

1. Reads the data from the FIFO buffer
2. Forms the data packet
3. Transmits the packet to the host
4. Receives the response from the host
5. Sends the updated status to the PFC

In host mode, the MAC receives a token request from the AIU. The MAC performs the following steps:

1. Builds the token packet
2. Sends the packet to the device

For OUT or SETUP transactions, the MAC also performs the following steps:

1. Reads the data from the transmit FIFO buffer
2. Assembles the data packet
3. Sends the packet to the device
4. Waits for a response

The response from the device causes the MAC to send a status update to the AIU.

For IN or PING transactions, the MAC waits for the data or handshake response from the device. For data responses, the MAC performs the following steps:

1. Validates the data
2. Writes the data to the receive FIFO buffer
3. Sends a status update to the AIU
4. Sends a handshake to the device, if appropriate

Host Protocol

In host mode, the MAC performs the following functions:

- Detects connect, disconnect, and remote wakeup events on the USB link
- Initiates reset
- Initiates speed enumeration processes
- Generates Start of Frame (SOF) packets.

Device Protocol

In device mode, the MAC performs the following functions:

- Handles USB reset sequence
- Handles speed enumeration
- Detects USB suspend and resume activity on the USB link
- Initiates remote wakeup
- Decodes SOF packets

OTG Protocol

The MAC handles HNP and SRP for OTG operation. HNP provides a mechanism for swapping host and device roles. SRP provides mechanisms for the host to turn off V_{BUS} to save power, and for a device to request a new USB session.

LPM Functions

The USB OTG controller supports LPM in both host and device modes. With this feature, the USB OTG controller can enter a sleep state when a successful LPM transaction occurs on the USB link.

Wakeup and Power Control

To reduce power, the USB OTG controller supports a power-down mode. In power-down mode, the controller and the PHY can shut down their clocks. The controller supports wakeup on the detection of the following events:

- Resume
- Remote wakeup
- Session request protocol
- New session start

PHY Interface Unit

The USB OTG controller supports synchronous SDR data transmission to a ULPI PHY. The SDR mode implements an eight-bit data bus.

ULPI PHY Interface

Table 18-1: ULPI PHY Interfaces

The ULPI PHY interface is synchronous to the `ulpi_clk` signal coming from the PHY.

Port Name	Bit Width	Direction	Description
`ulpi_clk`	1	Input	ULPI Clock Receives the 60-MHz clock supplied by the high-speed ULPI PHY. All signals are synchronous to the positive edge of the clock.
`ulpi_dir`	1	Input	ULPI Data Bus Control 1—The PHY has data to transfer to the USB OTG controller. 0—The PHY does not have data to transfer.
`ulpi_nxt`	1	Input	ULPI Next Data Control Indicates that the PHY has accepted the current byte from the USB OTG controller. When the PHY is transmitting, this signal indicates that a new byte is available for the controller.

Port Name	Bit Width	Direction	Description
ulpi_stp	1	Output	ULPI Stop Data Control The controller drives this signal high to indicate the end of its data stream. The controller can also drive this signal high to request data from the PHY.
ulpi_data[7:0]	8	Bidirectional	Bidirectional data bus. Driven low by the controller during idle.

Local Memory Buffer

The NAND flash controller has three local SRAM memory buffers.

- The write FIFO buffer is a 128 × 32-bit memory (512 total bytes)
- The read FIFO buffer is a 32 × 32-bit memory (128 total bytes)
- The ECC buffer is a 96 × 16-bit memory (1536 total bytes)

The SPRAM is a 8192 x 35-bit (32 data bits and 3 control bits) memory and includes support for ECC (Error Checking and Correction). The ECC block is integrated around a memory wrapper. It provides outputs to notify the system manager when single-bit correctable errors are detected (and corrected) and when double-bit uncorrectable errors are detected. The ECC logic also allows the injection of single- and double-bit errors for test purposes. The ECC feature is disabled by default. It must be initialized to enable the ECC function.

Clocks

Table 18-2: USB OTG Controller Clock Inputs

All clocks must be operational when reset is released. No special handling is required on the clocks.

Clock Signal	Frequency	Functional Usage
usb_mp_clk	60 – 200 MHz	Drives the master and slave interfaces, DMA controller, and internal FIFO buffers
usb0_ulpi_clk	60 MHz	ULPI reference clock for usb0 from external ULPI PHY I/O pin
usb1_ulpi_clk	60 MHz	ULPI reference clock for usb1 from external ULPI PHY I/O pin

Resets

The USB OTG controller can be reset either through the hardware reset input or through software.

Reset Requirements

There must be a minimum of 12 cycles on the ulpi_clk clock before the controller is taken out of reset. During reset, the USB OTG controller asserts the ulpi_stp signal. The PHY outputs a clock when it sees the ulpi_stp signal asserted. However, if the pin multiplexers are not programmed, the PHY does not see the ulpi_stp signal. As a result, the ulpi_clk clock signal does not arrive at the USB OTG controller.

Software must ensure that the reset is active for a minimum of two usb_mp_c lk cycles. There is no maximum assertion time.

Hardware Reset

Each of the USB OTG controllers has one reset input from the reset manager. The reset signal is asserted during a cold or warm reset event. The reset manager holds the controllers in reset until software releases the resets. Software releases resets by clearing the appropriate USB bits in the Peripheral Module Reset Register (`permodrst`) in the HPS reset manager.

The reset input resets the following blocks:

- The master and slave interface logic
- The integrated DMA controller
- The internal FIFO buffers
- The CSR

The reset input is synchronized to the `usb_mp_clk` domain. The reset input is also synchronized to the ULPI clock within the USB OTG controller and is used to reset the ULPI PHY domain logic.

Software Reset

Software can reset the controller by setting the Core Soft Reset (`csftrst`) bit in the Reset Register (`grstctl`) in the Global Registers (`globgrp`) group of the USB OTG controller.

Software resets are useful in the following situations:

- A PHY selection bit is changed by software. Resetting the USB OTG controller is part of clean-up to ensure that the PHY can operate with the new configuration or clock.
- During software development and debugging.

Interrupts

Table 18-3: USB OTG Interrupt Conditions

Each USB OTG controller has a single interrupt output. Interrupts are asserted on the conditions shown in the following table.

Condition	Mode
Device-initiated remote wakeup is detected.	Host mode
Session request is detected from the device.	Host mode
Device disconnect is detected.	Host mode
Host LPM entry retry has expired or LPM transaction(s) are complete.	Host mode
Host periodic TX FIFO buffer is empty (can be further programmed to indicate half-empty).	Host mode
Host channels interrupt received.	Host mode
Incomplete periodic transfer is pending at the end of the microframe.	Host mode
Host port status interrupt received.	Host mode
External host initiated resume is detected.	Device mode
LPM handshake is sent.	Device mode

Condition	Mode
Reset is detected when in suspend or normal mode.	Device mode
USB suspend mode is detected.	Device mode
Data fetch is suspended due to TX FIFO buffer full or request queue full.	Device mode
At least one isochronous OUT endpoint is pending at the end of the microframe.	Device mode
At least one isochronous IN endpoint is pending at the end of the microframe.	Device mode
At least one IN or OUT endpoint interrupt is pending at the end of the microframe.	Device mode
The end of the periodic frame is reached.	Device mode
Failure to write an isochronous OUT packet to the RX FIFO buffer. The RX FIFO buffer does not have enough space to accommodate the maximum packet size for the isochronous OUT endpoint.	Device mode
Enumeration has completed.	Device mode
Connector ID change.	Common modes
Mode mismatch. Software accesses registers belonging to an incorrect mode.	Common modes
Nonperiodic TX FIFO buffer is empty.	Common modes
RX FIFO buffer is not empty.	Common modes
Start of microframe.	Common modes
Device connection debounce is complete in host mode.	OTG interrupts
A-Device timeout while waiting for B-Device connection.	OTG interrupts
Host negotiation is complete.	OTG interrupts
Session request is complete.	OTG interrupts
Session end is detected in device mode.	OTG interrupts

USB OTG Controller Programming Model

For detailed information about using the USB OTG controller, consult your operating system (OS) driver documentation. The OS vendor provides application programming interfaces (APIs) to control USB host, device and OTG operation. This section provides a brief overview of the following software operations:

- Enabling SPRAM ECCs
- Host operation

- Device operation

Enabling SPRAM ECCs

The L3 interconnect has access to the SPRAM and is accessible through the USB OTG L3 slave interface. Software accesses the SPRAM through the `directfifo` memory space, in the USB OTG controller address space.

To enable the ECC feature, refer to the ECC chapter in the *Arria 10 Hard Processor System Technical Reference Manual*.

Note: Software cannot access the SPRAM beyond the 32-KB range. Out-of-range read transactions return indeterminate data. Out-of-range write transactions are ignored.

Related Information

Error Correction Controller

For more information, refer to the *Error Correction Controller* chapter in the *Hard Processor System Technical Reference Manual*.

Host Operation

Host Initialization

After power up, the USB port is in its default mode. No VBUS is applied to the USB cable. The following process sets up the USB OTG controller as a USB host.

1. To enable power to the USB port, the software driver sets the Port Power (`prtpwr`) bit to 1 in the Host Port Control and Status Register (`hprt`) of the Host Mode Registers (`hostgrp`) group. This action drives the V_{BUS} signal on the USB link.

   The controller waits for a connection to be detected on the USB link.
2. When a USB device connects, an interrupt is generated. The Port Connect Detected (`PrtConnDe t`) bit in `hprt` is set to 1.
3. Upon detecting a port connection, the software driver initiates a port reset by setting the Port Reset (`prtrst`) bit to 1 in `hprt`.
4. The software driver must wait a minimum of 10 ms so that speed enumeration can complete on the USB link.
5. After the 10 ms, the software driver sets `prtrst` back to 0 to release the port reset.
6. The USB OTG controller generates an interrupt. The Port Enable Disable Change (`prtenchng`) and Port Speed (`prtspd`) bits, in `hprt`, are set to reflect the enumerated speed of the device that attached.

   At this point the port is enabled for communication. Keep alive or SOF packets are sent on the port. If a USB 2.0-capable device fails to initialize correctly, it is reported as a USB 1.1 device.

   The Host Frame Interval Register (`hfir`) is updated with the corresponding PHY clock settings. The `hfir`, used for sending SOF packets, is in the Host Mode Registers (`host grp`) group.
7. The software driver must program the following registers in the Global Registers (`globgrp`) group, in the order listed:

a. Receive FIFO Size Register (grxfsiz)—selects the size of the receive FIFO buffer
b. Non-periodic Transmit FIFO Size Register (gnptxfsiz)—selects the size and the start address of the non-periodic transmit FIFO buffer for nonperiodic transactions
c. Host Periodic Transmit FIFO Size Register (hptxfsiz)—selects the size and start address of the periodic transmit FIFO buffer for periodic transactions

8. System software initializes and enables at least one channel to communicate with the USB device.

Host Transaction

When configured as a host, the USB OTG controller pipes the USB transactions through one of two request queues (one for periodic transactions and one for nonperiodic transactions). Each entry in the request queue holds the SETUP, IN, or OUT channel number along with other information required to perform a transaction on the USB link. The sequence in which the requests are written to the queue determines the sequence of transactions on the USB link.

The host processes the requests in the following order at the beginning of each frame or microframe:

1. Periodic request queue, including isochronous and interrupt transactions
2. Nonperiodic request queue (bulk or control transfers)

The host schedules transactions for each enabled channel in round-robin fashion. When the host controller completes the transfer for a channel, the controller updates the DMA descriptor status in the system memory.

For OUT transactions, the host controller uses two transmit FIFO buffers to hold the packet payload to be transmitted. One transmit FIFO buffer is used for all nonperiodic OUT transactions and the other is used for all periodic OUT transactions.

For IN transactions, the USB host controller uses one receive FIFO buffer for all periodic and nonperiodic transactions. The controller holds the packet payload from the USB device in the receive FIFO buffer until the packet is transferred to the system memory. The receive FIFO buffer also holds the status of each packet received. The status entry holds the IN channel number along with other information, including received byte count and validity status.

For generic hub operations, the USB OTG controller uses SPLIT transfers to communicate with slower-speed devices downstream of the hub. For these transfers, the transaction accumulation or buffering is performed in the generic hub, and is scheduled accordingly. The USB OTG controller ensures that enough transmit and receive buffers are allocated when the downstream transactions are completed or when accumulated data is ready to be sent upstream.

Device Operation

Device Initialization

The following process sets up the USB OTG controller as a USB device:

1. After power up, the USB OTG controller must be set to the desired device speed by writing to the Device Speed (devspd) bits in the Device Configuration Register (dcfg) in the Device Mode Registers (devgrp) group. After the device speed is set, the controller waits for a USB host to detect the USB port as a device port.
2. When an external host detects the USB port, the host performs a port reset, which generates an interrupt to the USB device software. The USB Reset (usbrst) bit in the Interrupt (port reset) register in the

Global Registers (`globgrp`) group is set. The device software then sets up the data FIFO buffer to receive a SETUP packet from the external host. Endpoint 0 is not enabled yet.

3. After completion of the port reset, the operation speed required by the external host is known. Software reads the device speed status and sets up all the remaining required transaction fields to enable control endpoint 0.

After completion of this process, the device is receiving SOF packets, and is ready for the USB host to set up the device's control endpoint.

Device Transaction

When configured as a device, the USB OTG controller uses a single FIFO buffer to receive the data for all the OUT endpoints. The receive FIFO buffer holds the status of the received data packet, including the byte count, the data packet ID (PID), and the validity of the received data. The DMA controller reads the data out of the FIFO buffer as the data are received. If a FIFO buffer overflow condition occurs, the controller responds to the OUT packet with a NAK, and internally rewinds the pointers.

For IN endpoints, the controller uses dedicated transmit buffers for each endpoint. The application does not need to predict the order in which the USB host will access the nonperiodic endpoints. If a FIFO buffer underrun condition occurs during transmit, the controller inverts the cyclic redundancy code (CRC) to mark the packet as corrupt on the USB link.

The application handles one data packet at a time per endpoint in transaction-level operations. The software receives an interrupt on completion of every packet. Based on the handshake response received on the USB link, the application determines whether to retry the transaction or proceed with the next transaction, until all packets in the transfer are completed.

IN Transactions

For an IN transaction, the application performs the following steps:

1. Enables the endpoint
2. Triggers the DMA engine to write the associated data packet to the corresponding transmit FIFO buffer
3. Waits for the packet completion interrupt from the controller

When an IN token is received on an endpoint when the associated transmit FIFO buffer does not contain sufficient data, the controller performs the following steps:

1. Generates an interrupt
2. Returns a NAK handshake to the USB host

If sufficient data is available, the controller transmits the data to the USB host.

OUT Transactions

For an OUT transaction, the application performs the following steps:

1. Enables the endpoint
2. Waits for the packet received interrupt from the USB OTG controller
3. Retrieves the packet from the receive FIFO buffer

When an OUT token or PING token is received on an endpoint where the receive FIFO buffer does not have sufficient space, the controller performs the following steps:

1. Generates an interrupt
2. Returns a NAK handshake to USB host

If sufficient space is available, the controller stores the data in the receive FIFO buffer and returns an ACK handshake to the USB link.

Control Transfers

For control transfers, the application performs the following steps:

1. Waits for the packet received interrupt from the controller
2. Retrieves the packet from the receive buffer

Because the control transfer is governed by USB protocol, the controller always responds with an ACK handshake.

Address Map and Register Definitions

The address map and register definitions for this module will be available in a future document revision.

Document Revision History

Table 18-4: Document Revision History

Date	Version	Changes
August 2014	2014.08.18	Initial release.